App.tsx

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

import useLocalStorage from './hooks/useLocalStorage';

import { Department, StaffMember, View, SkillCategory, Assessment, Hospital, AppScreen, NamedChecklistTemplate, ExamTemplate, ExamSubmission, LoggedInUser, UserRole, TrainingMaterial, MonthlyTraining, NewsBanner, MonthlyWorkLog, Patient, ChatMessage, AdminMessage } from './types';

import WelcomeScreen from './components/WelcomeScreen';

import HospitalList from './components/HospitalList';

import DepartmentList from './components/DepartmentList';

import DepartmentView from './components/DepartmentView';

import StaffMemberView from './components/StaffMemberView';

import ChecklistManager from './components/ChecklistManager';

import ExamManager from './components/ExamManager';

import TrainingManager from './components/TrainingManager';

import AccreditationManager from './components/AccreditationManager';

import NewsBannerManager from './components/NewsBannerManager';

import PatientEducationManager from './components/PatientEducationManager';

import PatientPortalView from './components/PatientPortalView';

import AboutModal from './components/AboutModal';

import LoginModal from './components/LoginModal';

import { SaveIcon } from './components/icons/SaveIcon';

import { UploadIcon } from './components/icons/UploadIcon';

import { InfoIcon } from './components/icons/InfoIcon';

import { LogoutIcon } from './components/icons/LogoutIcon';

import { BackIcon } from './components/icons/BackIcon';

import \* as db from './services/db';

import AdminCommunicationView from './components/AdminCommunicationView';

import HospitalCommunicationView from './components/HospitalCommunicationView';

const PERSIAN\_MONTHS = [

"فروردین", "اردیبهشت", "خرداد",

"تیر", "مرداد", "شهریور",

"مهر", "آبان", "آذر",

"دی", "بهمن", "اسفند"

];

type MessageContent = { text?: string; file?: { id: string; name: string; type: string } };

const App: React.FC = () => {

const [hospitals, setHospitals] = useLocalStorage<Hospital[]>('skill-assessment-data-v4', []);

const [appScreen, setAppScreen] = useState<AppScreen>(AppScreen.Welcome);

const [currentView, setCurrentView] = useState<View>(View.DepartmentList);

const [selectedHospitalId, setSelectedHospitalId] = useState<string | null>(null);

const [selectedDepartmentId, setSelectedDepartmentId] = useState<string | null>(null);

const [selectedStaffId, setSelectedStaffId] = useState<string | null>(null);

const [isAboutModalOpen, setIsAboutModalOpen] = useState(false);

const [isLoginModalOpen, setIsLoginModalOpen] = useState(false);

const [loginError, setLoginError] = useState<string | null>(null);

const [loggedInUser, setLoggedInUser] = useState<LoggedInUser | null>(null);

// Initialize IndexedDB

useEffect(() => {

db.initDB().then(success => {

if (!success) {

alert('Failed to initialize the database. The app might not work correctly for file uploads.');

}

});

}, []);

const handleGoToWelcome = () => {

setAppScreen(AppScreen.Welcome);

setSelectedHospitalId(null);

setSelectedDepartmentId(null);

setSelectedStaffId(null);

setCurrentView(View.DepartmentList);

setLoggedInUser(null);

}

const findHospital = (hospitalId: string | null) => hospitals.find(h => h.id === hospitalId);

const findDepartment = (hospital: Hospital | undefined, departmentId: string | null) => hospital?.departments.find(d => d.id === departmentId);

const findStaffMember = (department: Department | undefined, staffId: string | null) => department?.staff.find(s => s.id === staffId);

// --- Data Handlers ---

const handleSaveData = async () => {

try {

const hospital = findHospital(selectedHospitalId);

const department = findDepartment(hospital, selectedDepartmentId);

let dataToSave: any;

let fileName = `skill\_assessment\_data\_${new Date().toISOString().split('T')[0]}.json`;

// Scope: Department (inside DepartmentView or StaffMemberView)

if (department && hospital && (currentView === View.DepartmentView || currentView === View.StaffMemberView || currentView === View.PatientEducationManager)) {

dataToSave = {

type: 'department',

data: department,

context: {

hospitalId: hospital.id,

hospitalName: hospital.name,

}

};

fileName = `skill\_assessment\_DEPT\_${department.name.replace(/\s/g, '\_')}\_${new Date().toISOString().split('T')[0]}.json`;

}

// Scope: Hospital (inside DepartmentList, ChecklistManager, etc.)

else if (hospital) {

const materialIds = new Set<string>();

hospital.trainingMaterials?.forEach(monthly => {

monthly.materials.forEach(material => materialIds.add(material.id));

});

hospital.accreditationMaterials?.forEach(material => materialIds.add(material.id));

hospital.departments.forEach(dept => {

dept.patientEducationMaterials?.forEach(material => materialIds.add(material.id));

dept.patients?.forEach(p => {

p.chatHistory?.forEach(msg => {

if (msg.file) materialIds.add(msg.file.id);

});

});

});

hospital.newsBanners?.forEach(banner => materialIds.add(banner.imageId));

hospital.adminMessages?.forEach(msg => {

if (msg.file) materialIds.add(msg.file.id);

});

const allDbMaterials = await db.getAllMaterials();

const hospitalDbData = allDbMaterials.filter(m => materialIds.has(m.id));

dataToSave = {

type: 'hospital',

data: hospital,

dbData: hospitalDbData

};

fileName = `skill\_assessment\_HOSPITAL\_${hospital.name.replace(/\s/g, '\_')}\_${new Date().toISOString().split('T')[0]}.json`;

}

// Scope: All data

else {

const allDbMaterials = await db.getAllMaterials();

dataToSave = {

type: 'all\_hospitals',

data: hospitals,

dbData: allDbMaterials

};

fileName = `skill\_assessment\_ALL\_${new Date().toISOString().split('T')[0]}.json`;

}

const jsonString = `data:text/json;charset=utf-8,${encodeURIComponent(JSON.stringify(dataToSave, null, 2))}`;

const link = document.createElement('a');

link.href = jsonString;

link.download = fileName;

link.click();

} catch (error) {

console.error('Failed to save data:', error);

alert('خطا در ذخیره سازی داده ها.');

}

};

const handleLoadData = (event: React.ChangeEvent<HTMLInputElement>) => {

const file = event.target.files?.[0];

if (file) {

const reader = new FileReader();

reader.onload = async (e) => {

try {

const text = e.target?.result as string;

const loadedData = JSON.parse(text);

const { type, data, dbData, context } = loadedData;

const scopeInfo = getScope();

const currentHospital = findHospital(selectedHospitalId);

const currentDepartment = findDepartment(currentHospital, selectedDepartmentId);

const legacyFormat = !type && Array.isArray(loadedData);

const effectiveType = legacyFormat ? 'all\_hospitals' : type;

if (effectiveType === 'department') {

if (scopeInfo.scope !== 'department' || !currentHospital || !currentDepartment) {

alert('خطا: فایل بارگذاری شده مربوط به یک بخش است. لطفاً ابتدا وارد بخش مورد نظر خود شوید تا بتوانید اطلاعات آن را جایگزین کنید.');

return;

}

if (context?.hospitalId !== currentHospital.id) {

alert(`خطا: این فایل متعلق به بیمارستان دیگری است ("${context?.hospitalName}"). شما نمی‌توانید آن را در بیمارستان فعلی بارگذاری کنید.`);

return;

}

const loadedDept = data as Department;

if (window.confirm(`آیا مطمئن هستید که می‌خواهید اطلاعات بخش فعلی ("${currentDepartment.name}") را با اطلاعات بخش "${loadedDept.name}" از فایل جایگزین کنید؟ این عمل غیرقابل بازگشت است.`)) {

setHospitals(hospitals.map(h =>

h.id === currentHospital.id

? { ...h, departments: h.departments.map(d => d.id === currentDepartment.id ? loadedDept : d) }

: h

));

// FIX: Update selected department ID to prevent errors after re-render

setSelectedDepartmentId(loadedDept.id);

alert(`داده های بخش "${loadedDept.name}" با موفقیت بارگذاری و جایگزین بخش فعلی شد.`);

}

} else if (effectiveType === 'hospital') {

const loadedHosp = data as Hospital;

const loadedDbData = dbData;

const overwriteHospital = async (hospitalToOverwrite: Hospital, loadedHospitalData: Hospital, loadedDb: any[]) => {

const oldMaterialIds = new Set<string>();

hospitalToOverwrite.trainingMaterials?.forEach(m => m.materials.forEach(mat => oldMaterialIds.add(mat.id)));

hospitalToOverwrite.accreditationMaterials?.forEach(mat => oldMaterialIds.add(mat.id));

hospitalToOverwrite.departments.forEach(d => {

d.patientEducationMaterials?.forEach(mat => oldMaterialIds.add(mat.id));

d.patients?.forEach(p => {

p.chatHistory?.forEach(msg => {

if (msg.file) oldMaterialIds.add(msg.file.id);

});

});

});

hospitalToOverwrite.newsBanners?.forEach(b => oldMaterialIds.add(b.imageId));

hospitalToOverwrite.adminMessages?.forEach(msg => {

if (msg.file) oldMaterialIds.add(msg.file.id);

});

for (const id of oldMaterialIds) { await db.deleteMaterial(id); }

if (loadedDb && Array.isArray(loadedDb)) {

for (const material of loadedDb) { await db.addMaterial(material); }

}

const wasSelected = selectedHospitalId === hospitalToOverwrite.id;

setHospitals(hospitals.map(h => h.id === hospitalToOverwrite.id ? loadedHospitalData : h));

// FIX: If the overwritten hospital was selected, update the ID to the new one

if (wasSelected) {

setSelectedHospitalId(loadedHospitalData.id);

}

alert(`داده های بیمارستان "${loadedHospitalData.name}" با موفقیت جایگزین شد.`);

};

if (scopeInfo.scope === 'hospital' && currentHospital) {

if (window.confirm(`آیا مطمئن هستید که می‌خواهید تمام اطلاعات بیمارستان فعلی ("${currentHospital.name}") را با اطلاعات بیمارستان "${loadedHosp.name}" از فایل جایگزین کنید؟ این عمل غیرقابل بازگشت است.`)) {

await overwriteHospital(currentHospital, loadedHosp, loadedDbData);

}

} else if (scopeInfo.scope === 'all') {

const existingHospital = hospitals.find(h => h.id === loadedHosp.id);

if (existingHospital) {

if (window.confirm(`بیمارستانی با نام "${loadedHosp.name}" از قبل وجود دارد. آیا می‌خواهید اطلاعات آن را با اطلاعات فایل جایگزین کنید؟`)) {

await overwriteHospital(existingHospital, loadedHosp, loadedDbData);

}

} else {

if (window.confirm(`بیمارستان "${loadedHosp.name}" در لیست شما وجود ندارد. آیا می‌خواهید آن را به عنوان بیمارستان جدید اضافه کنید؟`)) {

if (loadedDbData && Array.isArray(loadedDbData)) {

for (const material of loadedDbData) { await db.addMaterial(material); }

}

setHospitals([...hospitals, loadedHosp]);

alert(`بیمارستان "${loadedHosp.name}" با موفقیت اضافه شد.`);

}

}

} else {

alert('برای بارگذاری فایل بیمارستان، باید در صفحه لیست بیمارستان‌ها یا داخل یک بیمارستان باشید.');

return;

}

} else if (effectiveType === 'all\_hospitals') {

if (scopeInfo.scope !== 'all') {

alert('خطا: فایل بارگذاری شده شامل اطلاعات تمام بیمارستان‌ها است. برای بارگذاری آن، لطفاً به صفحه اصلی (لیست بیمارستان‌ها) بازگردید.');

return;

}

if (window.confirm('آیا مطمئن هستید که می‌خواهید کل داده‌های برنامه را با اطلاعات این فایل جایگزین کنید؟ تمام اطلاعات فعلی حذف خواهد شد.')) {

const hospitalsToLoad = data || loadedData;

const dbDataToLoad = dbData;

if (!Array.isArray(hospitalsToLoad)) throw new Error("Invalid format for hospitals data.");

await db.clearAllMaterials();

if (dbDataToLoad) { // New format with separate dbData

for(const material of dbDataToLoad) { await db.addMaterial(material); }

setHospitals(hospitalsToLoad);

alert('کل داده‌های برنامه با موفقیت بارگذاری شد!');

} else { // Old format, migration needed

const migrationPromises: Promise<any>[] = [];

for (const h of hospitalsToLoad) {

if (h.trainingMaterials) {

for (const monthly of h.trainingMaterials) {

for (const material of monthly.materials) {

if (material.data) {

migrationPromises.push(db.addMaterial({id: material.id, data: material.data}));

delete material.data;

}

}

}

}

if (h.accreditationMaterials) {

for (const material of h.accreditationMaterials) {

if (material.data) {

migrationPromises.push(db.addMaterial({id: material.id, data: material.data}));

delete material.data;

}

}

}

}

await Promise.all(migrationPromises);

setHospitals(hospitalsToLoad);

alert('داده‌های نسخه قدیمی با موفقیت بارگذاری و منتقل شد!');

}

}

} else {

alert('فرمت فایل نامعتبر است یا در جای اشتباهی در حال بارگذاری آن هستید.');

}

} catch (error) {

console.error('Failed to load data:', error);

alert('خطا در بارگذاری داده. فایل ممکن است خراب یا با فرمت نامعتبر باشد.');

} finally {

if (event.target) event.target.value = '';

}

};

reader.readAsText(file);

}

};

// --- Navigation Handlers ---

const handleSelectHospital = (id: string) => {

setSelectedHospitalId(id);

setAppScreen(AppScreen.MainApp);

setCurrentView(View.DepartmentList);

};

const handleSelectDepartment = (id: string) => {

setSelectedDepartmentId(id);

setCurrentView(View.DepartmentView);

};

const handleSelectStaff = (id: string) => {

setSelectedStaffId(id);

setCurrentView(View.StaffMemberView);

};

const handleBack = () => {

switch (currentView) {

case View.StaffMemberView:

setSelectedStaffId(null);

setCurrentView(View.DepartmentView);

break;

case View.DepartmentView:

case View.ChecklistManager:

case View.ExamManager:

case View.TrainingManager:

case View.AccreditationManager:

case View.NewsBannerManager:

case View.PatientEducationManager:

case View.HospitalCommunication:

setSelectedDepartmentId(null);

setCurrentView(View.DepartmentList);

break;

case View.AdminCommunication:

setCurrentView(View.DepartmentList);

break;

case View.DepartmentList:

setSelectedHospitalId(null);

setAppScreen(AppScreen.HospitalList);

break;

}

};

// --- Hospital Handlers ---

const handleAddHospital = (name: string, province: string, city: string, supervisorName: string, supervisorNationalId: string, supervisorPassword: string) => {

const newHospital: Hospital = {

id: Date.now().toString(),

name,

province,

city,

supervisorName,

supervisorNationalId,

supervisorPassword,

departments: [],

checklistTemplates: [],

examTemplates: [],

trainingMaterials: [],

accreditationMaterials: [],

newsBanners: [],

};

setHospitals([...hospitals, newHospital]);

};

const handleUpdateHospital = (id: string, updatedData: Partial<Omit<Hospital, 'id' | 'departments' | 'checklistTemplates' | 'examTemplates' | 'trainingMaterials' | 'newsBanners'>>) => {

setHospitals(hospitals.map(h => h.id === id ? { ...h, ...updatedData } : h));

}

const handleDeleteHospital = (id: string) => {

setHospitals(hospitals.filter(h => h.id !== id));

};

// --- Department Handlers ---

const handleAddDepartment = (name: string, managerName: string, managerNationalId: string, managerPassword: string, staffCount: number, bedCount: number) => {

const newDepartment: Department = { id: Date.now().toString(), name, managerName, managerNationalId, managerPassword, staffCount, bedCount, staff: [] };

setHospitals(

hospitals.map(h =>

h.id === selectedHospitalId

? { ...h, departments: [...h.departments, newDepartment] }

: h

)

);

};

const handleUpdateDepartment = (id: string, updatedData: Partial<Omit<Department, 'id' | 'staff'>>) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => d.id === id ? { ...d, ...updatedData } : d)

};

}

return h;

}));

}

const handleDeleteDepartment = (id: string) => {

setHospitals(

hospitals.map(h =>

h.id === selectedHospitalId

? { ...h, departments: h.departments.filter(d => d.id !== id) }

: h

)

);

};

const handleResetHospital = (supervisorNationalId: string, supervisorPassword: string): boolean => {

const hospital = findHospital(selectedHospitalId);

if (!hospital) {

return false;

}

if (hospital.supervisorNationalId === supervisorNationalId && hospital.supervisorPassword === supervisorPassword) {

setHospitals(hospitals.map(h =>

h.id === selectedHospitalId

? { ...h, departments: [] }

: h

));

return true;

}

return false;

};

// --- Staff Handlers ---

const handleAddStaff = (departmentId: string, name: string, title: string, nationalId: string, password?: string) => {

const newStaff: StaffMember = { id: Date.now().toString(), name, title, nationalId, password, assessments: [] };

setHospitals(

hospitals.map(h =>

h.id === selectedHospitalId

? {

...h,

departments: h.departments.map(d =>

d.id === departmentId ? { ...d, staff: [...d.staff, newStaff] } : d

)

}

: h

)

);

};

const handleUpdateStaff = (departmentId: string, staffId: string, updatedData: Partial<Omit<StaffMember, 'id' | 'assessments'>>) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === departmentId) {

return { ...d, staff: d.staff.map(s => s.id === staffId ? { ...s, ...updatedData } : s)};

}

return d;

})

}

}

return h;

}));

};

const handleDeleteStaff = (departmentId: string, staffId: string) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d =>

d.id === departmentId

? { ...d, staff: d.staff.filter(s => s.id !== staffId) }

: d

)

};

}

return h;

}));

};

const handleAddOrUpdateAssessment = (departmentId: string, staffId: string, month: string, skills: SkillCategory[], template?: NamedChecklistTemplate) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === departmentId) {

return {

...d,

staff: d.staff.map(s => {

if (s.id === staffId) {

const existingAssessmentIndex = s.assessments.findIndex(a => a.month === month);

const newAssessment: Assessment = {

id: existingAssessmentIndex > -1 ? s.assessments[existingAssessmentIndex].id : Date.now().toString(),

month,

skillCategories: skills,

supervisorMessage: existingAssessmentIndex > -1 ? s.assessments[existingAssessmentIndex].supervisorMessage : '',

managerMessage: existingAssessmentIndex > -1 ? s.assessments[existingAssessmentIndex].managerMessage : '',

templateId: template?.id,

minScore: template?.minScore,

maxScore: template?.maxScore,

examSubmissions: existingAssessmentIndex > -1 ? s.assessments[existingAssessmentIndex].examSubmissions : [],

};

if (existingAssessmentIndex > -1) {

const updatedAssessments = [...s.assessments];

updatedAssessments[existingAssessmentIndex] = newAssessment;

return { ...s, assessments: updatedAssessments };

} else {

return { ...s, assessments: [...s.assessments, newAssessment] };

}

}

return s;

})

};

}

return d;

})

};

}

return h;

}));

};

const handleUpdateAssessmentMessages = (departmentId: string, staffId: string, month: string, messages: { supervisorMessage: string; managerMessage: string; }) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === departmentId) {

return {

...d,

staff: d.staff.map(s => {

if (s.id === staffId) {

return {

...s,

assessments: s.assessments.map(a =>

a.month === month ? { ...a, ...messages } : a

)

};

}

return s;

})

};

}

return d;

})

};

}

return h;

}));

};

const handleComprehensiveImport = (departmentId: string, data: { [staffName: string]: Map<string, SkillCategory[]> }) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === departmentId) {

let updatedStaff = [...d.staff];

const existingStaffNames = new Set(updatedStaff.map(s => s.name));

for (const staffName in data) {

const assessmentsMap = data[staffName];

const newAssessments: Assessment[] = Array.from(assessmentsMap.entries()).map(([month, categories]) => ({

id: `${staffName}-${month}-${Date.now()}`,

month,

skillCategories: categories,

maxScore: 4, // Default from Excel parser

minScore: 0,

}));

const staffIndex = updatedStaff.findIndex(s => s.name === staffName);

if (staffIndex > -1) {

// Update existing staff

const existingStaff = updatedStaff[staffIndex];

const assessmentsToUpdate = new Map(existingStaff.assessments.map(a => [a.month, a]));

newAssessments.forEach(newA => assessmentsToUpdate.set(newA.month, newA));

updatedStaff[staffIndex] = { ...existingStaff, assessments: Array.from(assessmentsToUpdate.values()) };

} else {

// Add new staff

const newStaff: StaffMember = {

id: Date.now().toString() + staffName,

name: staffName,

title: 'پرسنل',

assessments: newAssessments,

};

updatedStaff.push(newStaff);

}

}

return { ...d, staff: updatedStaff };

}

return d;

})

};

}

return h;

}));

alert("داده‌های جامع با موفقیت بارگذاری شد.");

};

const handleAddOrUpdateChecklistTemplate = (template: NamedChecklistTemplate) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const templates = h.checklistTemplates || [];

const existingIndex = templates.findIndex(t => t.id === template.id);

if (existingIndex > -1) {

templates[existingIndex] = template;

} else {

templates.push(template);

}

return { ...h, checklistTemplates: templates };

}

return h;

}));

};

const handleDeleteChecklistTemplate = (templateId: string) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const templates = (h.checklistTemplates || []).filter(t => t.id !== templateId);

return { ...h, checklistTemplates: templates };

}

return h;

}));

};

const handleAddOrUpdateExamTemplate = (template: ExamTemplate) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const templates = h.examTemplates || [];

const existingIndex = templates.findIndex(t => t.id === template.id);

if (existingIndex > -1) {

templates[existingIndex] = template;

} else {

templates.push(template);

}

return { ...h, examTemplates: templates };

}

return h;

}));

};

const handleDeleteExamTemplate = (templateId: string) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const templates = (h.examTemplates || []).filter(t => t.id !== templateId);

return { ...h, examTemplates: templates };

}

return h;

}));

};

const handleSubmitExam = (departmentId: string, staffId: string, month: string, submission: ExamSubmission) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === departmentId) {

return {

...d,

staff: d.staff.map(s => {

if (s.id === staffId) {

return {

...s,

assessments: s.assessments.map(a => {

if (a.month === month) {

const submissions = a.examSubmissions || [];

return { ...a, examSubmissions: [...submissions, submission] };

}

return a;

})

};

}

return s;

})

};

}

return d;

})

};

}

return h;

}));

}

const handleAddOrUpdateWorkLog = (departmentId: string, staffId: string, workLog: MonthlyWorkLog) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === departmentId) {

return {

...d,

staff: d.staff.map(s => {

if (s.id === staffId) {

const workLogs = s.workLogs ? [...s.workLogs] : [];

const existingLogIndex = workLogs.findIndex(l => l.month === workLog.month);

if (existingLogIndex > -1) {

workLogs[existingLogIndex] = workLog;

} else {

workLogs.push(workLog);

}

return { ...s, workLogs };

}

return s;

})

};

}

return d;

})

};

}

return h;

}));

};

// --- Training Handlers ---

const handleAddTrainingMaterial = async (month: string, material: TrainingMaterial) => {

if (!material.data) {

alert("Error: Material data is missing.");

return;

}

try {

await db.addMaterial({ id: material.id, data: material.data });

// Create a version of the material to store in localStorage without the large data part

const materialMetadata: TrainingMaterial = { ...material };

delete materialMetadata.data;

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const materialsByMonth = h.trainingMaterials ? [...h.trainingMaterials] : [];

const monthIndex = materialsByMonth.findIndex(t => t.month === month);

if (monthIndex > -1) {

materialsByMonth[monthIndex].materials.push(materialMetadata);

} else {

materialsByMonth.push({ month, materials: [materialMetadata] });

// Sort by Persian month order

materialsByMonth.sort((a,b) => PERSIAN\_MONTHS.indexOf(a.month) - PERSIAN\_MONTHS.indexOf(b.month));

}

return { ...h, trainingMaterials: materialsByMonth };

}

return h;

}));

} catch (error) {

console.error("Failed to add training material:", error);

alert("Failed to save material to the database.");

}

};

const handleDeleteTrainingMaterial = async (month: string, materialId: string) => {

try {

await db.deleteMaterial(materialId);

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const materialsByMonth = (h.trainingMaterials || []).map(monthly => {

if (monthly.month === month) {

return {

...monthly,

materials: monthly.materials.filter(m => m.id !== materialId)

};

}

return monthly;

}).filter(monthly => monthly.materials.length > 0); // Remove month if it becomes empty

return { ...h, trainingMaterials: materialsByMonth };

}

return h;

}));

} catch (error) {

console.error("Failed to delete training material:", error);

alert("Failed to delete material from the database.");

}

};

const handleUpdateMaterialDescription = (month: string, materialId: string, description: string) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const materialsByMonth = (h.trainingMaterials || []).map(monthly => {

if (monthly.month === month) {

return {

...monthly,

materials: monthly.materials.map(m => m.id === materialId ? { ...m, description } : m)

};

}

return monthly;

});

return { ...h, trainingMaterials: materialsByMonth };

}

return h;

}));

};

// --- Accreditation Handlers ---

const handleAddAccreditationMaterial = async (material: TrainingMaterial) => {

if (!material.data || !selectedHospitalId) {

alert("Error: Material data or hospital is missing.");

return;

}

try {

await db.addMaterial({ id: material.id, data: material.data });

const materialMetadata: TrainingMaterial = { ...material };

delete materialMetadata.data;

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const materials = h.accreditationMaterials ? [...h.accreditationMaterials] : [];

materials.push(materialMetadata);

return { ...h, accreditationMaterials: materials };

}

return h;

}));

} catch (error) {

console.error("Failed to add accreditation material:", error);

alert("Failed to save material to the database.");

}

};

const handleDeleteAccreditationMaterial = async (materialId: string) => {

if (!selectedHospitalId) return;

try {

await db.deleteMaterial(materialId);

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const materials = (h.accreditationMaterials || []).filter(m => m.id !== materialId);

return { ...h, accreditationMaterials: materials };

}

return h;

}));

} catch (error) {

console.error("Failed to delete accreditation material:", error);

alert("Failed to delete material from the database.");

}

};

const handleUpdateAccreditationMaterialDescription = (materialId: string, description: string) => {

if (!selectedHospitalId) return;

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const materials = (h.accreditationMaterials || []).map(m => m.id === materialId ? { ...m, description } : m);

return { ...h, accreditationMaterials: materials };

}

return h;

}));

};

// --- News Banner Handlers ---

const handleAddNewsBanner = async (banner: Omit<NewsBanner, 'id' | 'imageId'>, imageData: string) => {

if (!selectedHospitalId) return;

const imageId = `banner-img-${Date.now()}`;

const newBanner: NewsBanner = {

id: `banner-meta-${Date.now()}`,

...banner,

imageId,

};

try {

await db.addMaterial({ id: imageId, data: imageData });

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const banners = h.newsBanners ? [...h.newsBanners, newBanner] : [newBanner];

return { ...h, newsBanners: banners };

}

return h;

}));

} catch (error) {

console.error("Failed to add news banner:", error);

alert("Failed to save banner image to the database.");

}

};

const handleDeleteNewsBanner = async (bannerId: string) => {

if (!selectedHospitalId) return;

let imageIdToDelete: string | null = null;

const updatedHospitals = hospitals.map(h => {

if (h.id === selectedHospitalId) {

const banner = (h.newsBanners || []).find(b => b.id === bannerId);

if (banner) {

imageIdToDelete = banner.imageId;

}

const banners = (h.newsBanners || []).filter(b => b.id !== bannerId);

return { ...h, newsBanners: banners };

}

return h;

});

if (imageIdToDelete) {

try {

await db.deleteMaterial(imageIdToDelete);

setHospitals(updatedHospitals);

} catch (error) {

console.error("Failed to delete banner image:", error);

alert("Failed to delete banner image from the database.");

}

} else {

setHospitals(updatedHospitals);

}

};

const handleUpdateNewsBanner = (bannerId: string, title: string, description: string) => {

if (!selectedHospitalId) return;

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

const banners = (h.newsBanners || []).map(b =>

b.id === bannerId ? { ...b, title, description } : b

);

return { ...h, newsBanners: banners };

}

return h;

}));

};

// --- Patient Education & Patient Handlers ---

const handleAddPatient = (departmentId: string, name: string, nationalId: string, password?: string) => {

const newPatient: Patient = { id: Date.now().toString(), name, nationalId, password, chatHistory: [] };

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h, departments: h.departments.map(d => {

if (d.id === departmentId) {

const patients = d.patients ? [...d.patients, newPatient] : [newPatient];

return { ...d, patients };

}

return d;

})

};

}

return h;

}));

};

const handleDeletePatient = (departmentId: string, patientId: string) => {

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h, departments: h.departments.map(d => {

if (d.id === departmentId) {

const patients = (d.patients || []).filter(p => p.id !== patientId);

return { ...d, patients };

}

return d;

})

};

}

return h;

}));

};

const handleSendPatientChatMessage = (departmentId: string, patientId: string, content: MessageContent, sender: 'patient' | 'manager') => {

const newMessage: ChatMessage = {

id: Date.now().toString(),

sender,

timestamp: new Date().toISOString(),

...content

};

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === departmentId) {

return {

...d,

patients: (d.patients || []).map(p => {

if (p.id === patientId) {

const chatHistory = p.chatHistory ? [...p.chatHistory, newMessage] : [newMessage];

return { ...p, chatHistory };

}

return p;

})

};

}

return d;

})

};

}

return h;

}));

};

const handleAddPatientEducationMaterial = async (material: TrainingMaterial) => {

if (!material.data || !selectedHospitalId || !selectedDepartmentId) {

alert("Error: Material data, hospital, or department is missing.");

return;

}

try {

await db.addMaterial({ id: material.id, data: material.data });

const materialMetadata: TrainingMaterial = { ...material };

delete materialMetadata.data;

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === selectedDepartmentId) {

const materials = d.patientEducationMaterials ? [...d.patientEducationMaterials, materialMetadata] : [materialMetadata];

return { ...d, patientEducationMaterials: materials };

}

return d;

})

};

}

return h;

}));

} catch (error) {

console.error("Failed to add patient education material:", error);

alert("Failed to save material to the database.");

}

};

const handleDeletePatientEducationMaterial = async (materialId: string) => {

if (!selectedHospitalId || !selectedDepartmentId) return;

try {

await db.deleteMaterial(materialId);

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === selectedDepartmentId) {

const materials = (d.patientEducationMaterials || []).filter(m => m.id !== materialId);

return { ...d, patientEducationMaterials: materials };

}

return d;

})

};

}

return h;

}));

} catch (error) {

console.error("Failed to delete patient education material:", error);

alert("Failed to delete material from the database.");

}

};

const handleUpdatePatientEducationMaterialDescription = (materialId: string, description: string) => {

if (!selectedHospitalId || !selectedDepartmentId) return;

setHospitals(hospitals.map(h => {

if (h.id === selectedHospitalId) {

return {

...h,

departments: h.departments.map(d => {

if (d.id === selectedDepartmentId) {

const materials = (d.patientEducationMaterials || []).map(m => m.id === materialId ? { ...m, description } : m);

return { ...d, patientEducationMaterials: materials };

}

return d;

})

};

}

return h;

}));

};

// --- Communication Handlers ---

const handleSendMessage = (hospitalId: string, content: MessageContent, sender: 'hospital' | 'admin') => {

const newMessage: AdminMessage = {

id: Date.now().toString(),

sender,

timestamp: new Date().toISOString(),

...content

};

setHospitals(hospitals.map(h => {

if (h.id === hospitalId) {

const messages = h.adminMessages ? [...h.adminMessages, newMessage] : [newMessage];

return { ...h, adminMessages: messages };

}

return h;

}));

};

// --- Auth Handlers ---

const handleLogin = (nationalId: string, password: string) => {

setLoginError(null);

if (!nationalId || !password) {

setLoginError('کد ملی و رمز عبور الزامی است.');

return;

}

// Admin Login

if (nationalId === "5850008985" && password === "64546") {

setLoggedInUser({ role: UserRole.Admin, name: 'ادمین کل' });

setIsLoginModalOpen(false);

if (appScreen === AppScreen.Welcome) {

setAppScreen(AppScreen.HospitalList);

}

return;

}

// Supervisor, Manager, or Staff Login

for(const hospital of hospitals) {

// Supervisor

if (hospital.supervisorNationalId === nationalId && hospital.supervisorPassword === password) {

setLoggedInUser({ role: UserRole.Supervisor, name: hospital.supervisorName || 'سوپروایزر', hospitalId: hospital.id });

handleSelectHospital(hospital.id);

setIsLoginModalOpen(false);

return;

}

for(const department of hospital.departments) {

// Manager

if (department.managerNationalId === nationalId && department.managerPassword === password) {

setLoggedInUser({ role: UserRole.Manager, name: department.managerName, hospitalId: hospital.id, departmentId: department.id });

handleSelectHospital(hospital.id);

handleSelectDepartment(department.id);

setIsLoginModalOpen(false);

return;

}

// Staff

for (const staff of department.staff) {

if (staff.nationalId === nationalId && staff.password === password) {

setLoggedInUser({ role: UserRole.Staff, name: staff.name, hospitalId: hospital.id, departmentId: department.id, staffId: staff.id });

handleSelectHospital(hospital.id);

handleSelectDepartment(department.id);

handleSelectStaff(staff.id);

setIsLoginModalOpen(false);

return;

}

}

// Patient

for (const patient of department.patients || []) {

if (patient.nationalId === nationalId && patient.password === password) {

setLoggedInUser({ role: UserRole.Patient, name: patient.name, hospitalId: hospital.id, departmentId: department.id, patientId: patient.id });

setAppScreen(AppScreen.MainApp);

setCurrentView(View.PatientPortal);

setIsLoginModalOpen(false);

return;

}

}

}

}

setLoginError('کد ملی یا رمز عبور نامعتبر است.');

}

const handleLogout = () => {

setLoggedInUser(null);

setSelectedHospitalId(null);

setSelectedDepartmentId(null);

setSelectedStaffId(null);

setCurrentView(View.DepartmentList);

setAppScreen(AppScreen.Welcome);

}

// --- Render Logic ---

const getScope = (): { scope: 'department' | 'hospital' | 'all', name?: string } => {

const hospital = findHospital(selectedHospitalId);

const department = findDepartment(hospital, selectedDepartmentId);

if (department && hospital && (currentView === View.DepartmentView || currentView === View.StaffMemberView || currentView === View.PatientEducationManager)) {

return { scope: 'department', name: department.name };

}

if (hospital && (currentView === View.DepartmentList || currentView === View.ChecklistManager || currentView === View.ExamManager || currentView === View.TrainingManager || currentView === View.AccreditationManager || currentView === View.NewsBannerManager || currentView === View.HospitalCommunication)) {

return { scope: 'hospital', name: hospital.name };

}

return { scope: 'all' };

};

const getSaveLoadLabels = () => {

const scopeInfo = getScope();

switch (scopeInfo.scope) {

case 'department':

return { save: 'ذخیره بخش', load: 'بارگذاری بخش' };

case 'hospital':

return { save: 'ذخیره بیمارستان', load: 'بارگذاری بیمارستان' };

case 'all':

default:

return { save: 'ذخیره کل', load: 'بارگذاری کل' };

}

};

const renderContent = () => {

// Non-admin users are locked to their scope

if (loggedInUser && loggedInUser.role !== UserRole.Admin) {

const hospital = findHospital(loggedInUser.hospitalId!);

if (!hospital) return <p>Error: Hospital not found for user.</p>;

if (loggedInUser.role === UserRole.Patient) {

const department = findDepartment(hospital, loggedInUser.departmentId!);

const patient = department?.patients?.find(p => p.id === loggedInUser.patientId);

if (!department || !patient) return <p>Error: Patient data not found.</p>;

return <PatientPortalView

department={department}

patient={patient}

onSendMessage={(content) => handleSendPatientChatMessage(department.id, patient.id, content, 'patient')}

/>

}

if (loggedInUser.role === UserRole.Staff) {

const department = findDepartment(hospital, loggedInUser.departmentId!);

const staffMember = findStaffMember(department, loggedInUser.staffId!);

if (!department || !staffMember) return <p>Error: Staff member not found.</p>;

return <StaffMemberView

department={department}

staffMember={staffMember}

onBack={() => {}} // Staff cannot go back

onAddOrUpdateAssessment={handleAddOrUpdateAssessment}

onUpdateAssessmentMessages={handleUpdateAssessmentMessages}

onSubmitExam={handleSubmitExam}

checklistTemplates={hospital.checklistTemplates || []}

examTemplates={hospital.examTemplates || []}

trainingMaterials={hospital.trainingMaterials || []}

accreditationMaterials={hospital.accreditationMaterials || []}

newsBanners={hospital.newsBanners || []}

userRole={loggedInUser.role}

/>

}

if (loggedInUser.role === UserRole.Manager) {

const department = findDepartment(hospital, loggedInUser.departmentId!);

if (!department) return <p>Error: Department not found for manager.</p>;

if (currentView === View.StaffMemberView) {

const staffMember = findStaffMember(department, selectedStaffId);

if (!staffMember) return <p>Selected staff member not found.</p>;

return <StaffMemberView

department={department}

staffMember={staffMember}

onBack={handleBack}

onAddOrUpdateAssessment={handleAddOrUpdateAssessment}

onUpdateAssessmentMessages={handleUpdateAssessmentMessages}

onSubmitExam={handleSubmitExam}

checklistTemplates={hospital.checklistTemplates || []}

examTemplates={hospital.examTemplates || []}

trainingMaterials={hospital.trainingMaterials || []}

accreditationMaterials={hospital.accreditationMaterials || []}

newsBanners={hospital.newsBanners || []}

userRole={loggedInUser.role}

/>

}

if (currentView === View.PatientEducationManager) {

return <PatientEducationManager

department={department}

onAddMaterial={handleAddPatientEducationMaterial}

onDeleteMaterial={handleDeletePatientEducationMaterial}

onUpdateMaterialDescription={handleUpdatePatientEducationMaterialDescription}

onBack={handleBack}

onAddPatient={(name, nationalId, password) => handleAddPatient(department.id, name, nationalId, password)}

onDeletePatient={(patientId) => handleDeletePatient(department.id, patientId)}

onSendMessage={(patientId, content, sender) => handleSendPatientChatMessage(department.id, patientId, content, sender)}

/>;

}

return <DepartmentView

department={department}

onBack={() => {}} // Manager cannot go back from their department

onAddStaff={handleAddStaff}

onUpdateStaff={handleUpdateStaff}

onDeleteStaff={handleDeleteStaff}

onSelectStaff={handleSelectStaff}

onComprehensiveImport={handleComprehensiveImport}

onManageChecklists={() => setCurrentView(View.ChecklistManager)}

onManageExams={() => setCurrentView(View.ExamManager)}

onManageTraining={() => setCurrentView(View.TrainingManager)}

onManagePatientEducation={() => setCurrentView(View.PatientEducationManager)}

onAddOrUpdateWorkLog={handleAddOrUpdateWorkLog}

userRole={loggedInUser.role}

newsBanners={hospital.newsBanners || []}

/>

}

// Supervisor role

if (loggedInUser.role === UserRole.Supervisor) {

const department = findDepartment(hospital, selectedDepartmentId);

const staffMember = findStaffMember(department, selectedStaffId);

switch (currentView) {

case View.DepartmentView:

if (!department) return <p>Selected department not found.</p>;

return <DepartmentView department={department} onBack={handleBack} onAddStaff={handleAddStaff} onUpdateStaff={handleUpdateStaff} onDeleteStaff={handleDeleteStaff} onSelectStaff={handleSelectStaff} onComprehensiveImport={handleComprehensiveImport} onManageChecklists={() => setCurrentView(View.ChecklistManager)} onManageExams={() => setCurrentView(View.ExamManager)} onManageTraining={() => setCurrentView(View.TrainingManager)} onManagePatientEducation={() => setCurrentView(View.PatientEducationManager)} onAddOrUpdateWorkLog={handleAddOrUpdateWorkLog} userRole={loggedInUser.role} newsBanners={hospital.newsBanners || []} />;

case View.StaffMemberView:

if (!department || !staffMember) return <p>Selected staff member not found.</p>;

return <StaffMemberView department={department} staffMember={staffMember} onBack={handleBack} onAddOrUpdateAssessment={handleAddOrUpdateAssessment} onUpdateAssessmentMessages={handleUpdateAssessmentMessages} onSubmitExam={handleSubmitExam} checklistTemplates={hospital.checklistTemplates || []} examTemplates={hospital.examTemplates || []} trainingMaterials={hospital.trainingMaterials || []} accreditationMaterials={hospital.accreditationMaterials || []} newsBanners={hospital.newsBanners || []} userRole={loggedInUser.role} />;

case View.ChecklistManager:

return <ChecklistManager templates={hospital.checklistTemplates || []} onAddOrUpdate={handleAddOrUpdateChecklistTemplate} onDelete={handleDeleteChecklistTemplate} onBack={handleBack} />;

case View.ExamManager:

return <ExamManager templates={hospital.examTemplates || []} onAddOrUpdate={handleAddOrUpdateExamTemplate} onDelete={handleDeleteExamTemplate} onBack={handleBack} />;

case View.TrainingManager:

return <TrainingManager monthlyTrainings={hospital.trainingMaterials || []} onAddMaterial={handleAddTrainingMaterial} onDeleteMaterial={handleDeleteTrainingMaterial} onUpdateMaterialDescription={handleUpdateMaterialDescription} onBack={handleBack} />

case View.AccreditationManager:

return <AccreditationManager materials={hospital.accreditationMaterials || []} onAddMaterial={handleAddAccreditationMaterial} onDeleteMaterial={handleDeleteAccreditationMaterial} onUpdateMaterialDescription={handleUpdateAccreditationMaterialDescription} onBack={handleBack} />

case View.NewsBannerManager:

return <NewsBannerManager banners={hospital.newsBanners || []} onAddBanner={handleAddNewsBanner} onUpdateBanner={handleUpdateNewsBanner} onDeleteBanner={handleDeleteNewsBanner} onBack={handleBack} />;

case View.PatientEducationManager:

if (!department) return <p>Selected department not found.</p>;

return <PatientEducationManager

department={department}

onAddMaterial={handleAddPatientEducationMaterial}

onDeleteMaterial={handleDeletePatientEducationMaterial}

onUpdateMaterialDescription={handleUpdatePatientEducationMaterialDescription}

onBack={handleBack}

onAddPatient={(name, nationalId, password) => handleAddPatient(department.id, name, nationalId, password)}

onDeletePatient={(patientId) => handleDeletePatient(department.id, patientId)}

onSendMessage={(patientId, content, sender) => handleSendPatientChatMessage(department.id, patientId, content, sender)}

/>;

case View.HospitalCommunication:

return <HospitalCommunicationView hospital={hospital} onSendMessage={(content) => handleSendMessage(hospital.id, content, 'hospital')} onBack={handleBack} />;

case View.DepartmentList:

default:

return <DepartmentList

departments={hospital.departments}

hospitalName={hospital.name}

onAddDepartment={handleAddDepartment}

onUpdateDepartment={handleUpdateDepartment}

onDeleteDepartment={handleDeleteDepartment}

onSelectDepartment={handleSelectDepartment}

onBack={() => {}} // Supervisor in DepartmentList for their hospital doesn't go back further

onManageAccreditation={() => setCurrentView(View.AccreditationManager)}

onManageNewsBanners={() => setCurrentView(View.NewsBannerManager)}

onResetHospital={handleResetHospital}

onContactAdmin={() => setCurrentView(View.HospitalCommunication)}

userRole={loggedInUser.role}

/>;

}

}

}

// Admin user flow

if (appScreen === AppScreen.HospitalList || !selectedHospitalId) {

if (currentView === View.AdminCommunication) {

return <AdminCommunicationView hospitals={hospitals} onSendMessage={(hospitalId, content) => handleSendMessage(hospitalId, content, 'admin')} onBack={() => setCurrentView(View.DepartmentList)} />;

}

return <HospitalList hospitals={hospitals} onAddHospital={handleAddHospital} onUpdateHospital={handleUpdateHospital} onDeleteHospital={handleDeleteHospital} onSelectHospital={handleSelectHospital} onGoToWelcome={handleGoToWelcome} userRole={loggedInUser?.role ?? UserRole.Admin} onContactAdmin={() => setCurrentView(View.AdminCommunication)} />;

}

const hospital = findHospital(selectedHospitalId);

if (!hospital) return <p>Selected hospital not found.</p>;

const department = findDepartment(hospital, selectedDepartmentId);

const staffMember = findStaffMember(department, selectedStaffId);

switch (currentView) {

case View.DepartmentView:

if (!department) return <p>Selected department not found.</p>;

return <DepartmentView department={department} onBack={handleBack} onAddStaff={handleAddStaff} onUpdateStaff={handleUpdateStaff} onDeleteStaff={handleDeleteStaff} onSelectStaff={handleSelectStaff} onComprehensiveImport={handleComprehensiveImport} onManageChecklists={() => setCurrentView(View.ChecklistManager)} onManageExams={() => setCurrentView(View.ExamManager)} onManageTraining={() => setCurrentView(View.TrainingManager)} onManagePatientEducation={() => setCurrentView(View.PatientEducationManager)} onAddOrUpdateWorkLog={handleAddOrUpdateWorkLog} userRole={loggedInUser?.role ?? UserRole.Admin} newsBanners={hospital.newsBanners || []} />;

case View.StaffMemberView:

if (!department || !staffMember) return <p>Selected staff member not found.</p>;

return <StaffMemberView department={department} staffMember={staffMember} onBack={handleBack} onAddOrUpdateAssessment={handleAddOrUpdateAssessment} onUpdateAssessmentMessages={handleUpdateAssessmentMessages} onSubmitExam={handleSubmitExam} checklistTemplates={hospital.checklistTemplates || []} examTemplates={hospital.examTemplates || []} trainingMaterials={hospital.trainingMaterials || []} accreditationMaterials={hospital.accreditationMaterials || []} newsBanners={hospital.newsBanners || []} userRole={loggedInUser?.role ?? UserRole.Admin} />;

case View.ChecklistManager:

return <ChecklistManager templates={hospital.checklistTemplates || []} onAddOrUpdate={handleAddOrUpdateChecklistTemplate} onDelete={handleDeleteChecklistTemplate} onBack={handleBack} />;

case View.ExamManager:

return <ExamManager templates={hospital.examTemplates || []} onAddOrUpdate={handleAddOrUpdateExamTemplate} onDelete={handleDeleteExamTemplate} onBack={handleBack} />;

case View.TrainingManager:

return <TrainingManager monthlyTrainings={hospital.trainingMaterials || []} onAddMaterial={handleAddTrainingMaterial} onDeleteMaterial={handleDeleteTrainingMaterial} onUpdateMaterialDescription={handleUpdateMaterialDescription} onBack={handleBack} />

case View.AccreditationManager:

return <AccreditationManager materials={hospital.accreditationMaterials || []} onAddMaterial={handleAddAccreditationMaterial} onDeleteMaterial={handleDeleteAccreditationMaterial} onUpdateMaterialDescription={handleUpdateAccreditationMaterialDescription} onBack={handleBack} />

case View.NewsBannerManager:

return <NewsBannerManager banners={hospital.newsBanners || []} onAddBanner={handleAddNewsBanner} onUpdateBanner={handleUpdateNewsBanner} onDeleteBanner={handleDeleteNewsBanner} onBack={handleBack} />;

case View.PatientEducationManager:

if (!department) return <p>Selected department not found.</p>;

return <PatientEducationManager

department={department}

onAddMaterial={handleAddPatientEducationMaterial}

onDeleteMaterial={handleDeletePatientEducationMaterial}

onUpdateMaterialDescription={handleUpdatePatientEducationMaterialDescription}

onBack={handleBack}

onAddPatient={(name, nationalId, password) => handleAddPatient(department.id, name, nationalId, password)}

onDeletePatient={(patientId) => handleDeletePatient(department.id, patientId)}

onSendMessage={(patientId, content, sender) => handleSendPatientChatMessage(department.id, patientId, content, sender)}

/>;

case View.HospitalCommunication:

return <HospitalCommunicationView hospital={hospital} onSendMessage={(content) => handleSendMessage(hospital.id, content, 'hospital')} onBack={handleBack} />;

case View.DepartmentList:

default:

return <DepartmentList

departments={hospital.departments}

hospitalName={hospital.name}

onAddDepartment={handleAddDepartment}

onUpdateDepartment={handleUpdateDepartment}

onDeleteDepartment={handleDeleteDepartment}

onSelectDepartment={handleSelectDepartment}

onBack={handleBack}

onManageAccreditation={() => setCurrentView(View.AccreditationManager)}

onManageNewsBanners={() => setCurrentView(View.NewsBannerManager)}

onResetHospital={handleResetHospital}

onContactAdmin={() => setCurrentView(View.HospitalCommunication)}

userRole={loggedInUser?.role ?? UserRole.Admin}

/>;

}

};

const showSaveLoad = loggedInUser && (appScreen === AppScreen.HospitalList || currentView === View.DepartmentList || currentView === View.DepartmentView);

const showHeaderBackButton = loggedInUser && appScreen === AppScreen.MainApp &&

!(currentView === View.DepartmentList && (loggedInUser.role === UserRole.Supervisor || loggedInUser.role === UserRole.Manager)) &&

!(currentView === View.StaffMemberView && loggedInUser.role === UserRole.Staff) &&

!(currentView === View.PatientPortal && loggedInUser.role === UserRole.Patient);

const { save: saveLabel, load: loadLabel } = getSaveLoadLabels();

const renderApp = () => {

if (appScreen === AppScreen.Welcome) {

return <WelcomeScreen onEnter={() => setIsLoginModalOpen(true)} />;

}

return (

<div className="bg-slate-100 dark:bg-slate-900 min-h-screen flex flex-col">

<header className="bg-gradient-to-r from-indigo-600 to-purple-600 text-white p-4 shadow-md flex justify-between items-center">

<div className="flex items-center gap-4">

{showHeaderBackButton && (

<button onClick={handleBack} className="flex items-center gap-2 px-3 py-1.5 text-sm font-semibold rounded-md bg-white/10 text-white hover:bg-white/20 transition-colors">

<BackIcon className="w-5 h-5" />

<span className="hidden sm:inline">بازگشت</span>

</button>

)}

<h1 className="text-2xl font-bold flex items-center gap-2">

سامانه بیمارستان من

{loggedInUser && loggedInUser.role !== UserRole.Admin && (

<span className="text-sm font-normal mr-2">

({loggedInUser.role === UserRole.Supervisor ? 'سوپروایزر' : loggedInUser.role === UserRole.Manager ? 'مدیر بخش' : loggedInUser.role === UserRole.Staff ? 'پرسنل' : 'بیمار'})

</span>

)}

</h1>

</div>

<div className="flex items-center space-x-2 rtl:space-x-reverse">

{loggedInUser && (

<span className="text-sm font-medium hidden sm:block">

خوش آمدید، {loggedInUser.name}

{getScope().name && ` (${getScope().name})`}

</span>

)}

{showSaveLoad && (

<>

<button onClick={handleSaveData} className="inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold rounded-md text-white bg-green-600 hover:bg-green-700 transition-colors">

<SaveIcon className="w-5 h-5" />

<span className="hidden sm:inline">{saveLabel}</span>

</button>

<label htmlFor="file-upload" className="cursor-pointer inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold rounded-md text-white bg-cyan-500 hover:bg-cyan-600 transition-colors">

<UploadIcon className="w-5 h-5" />

<span className="hidden sm:inline">{loadLabel}</span>

</label>

<input id="file-upload" type="file" accept=".json" onChange={handleLoadData} className="hidden" />

</>

)}

<div className="w-px h-6 bg-white/30 mx-1"></div>

<button onClick={() => setIsAboutModalOpen(true)} className="inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold rounded-md text-white bg-yellow-500 hover:bg-yellow-600 transition-colors" title="درباره برنامه">

<InfoIcon className="w-5 h-5" />

<span className="hidden sm:inline">درباره</span>

</button>

{loggedInUser ? (

<button onClick={handleLogout} className="inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold rounded-md text-white bg-red-600 hover:bg-red-700 transition-colors" title="خروج">

<LogoutIcon className="w-5 h-5" />

<span className="hidden sm:inline">خروج</span>

</button>

) : (

<button onClick={() => setIsLoginModalOpen(true)} className="inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold rounded-md text-white bg-purple-600 hover:bg-purple-700 transition-colors" title="ورود">

<svg className="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M11 16l-4-4m0 0l4-4m-4 4h14m-5 4v1a3 3 0 01-3 3H6a3 3 0 01-3-3V7a3 3 0 013-3h7a3 3 0 013 3v1"></path></svg>

<span className="hidden sm:inline">ورود</span>

</button>

)}

</div>

</header>

<main className="flex-grow p-4 sm:p-6 lg:p-8">

{renderContent()}

</main>

</div>

);

};

return (

<>

{renderApp()}

<AboutModal isOpen={isAboutModalOpen} onClose={() => setIsAboutModalOpen(false)} />

<LoginModal isOpen={isLoginModalOpen} onClose={() => setIsLoginModalOpen(false)} onLogin={handleLogin} loginError={loginError} />

</>

);

};

export default App;

index.html

<!DOCTYPE html>

<html lang="fa" dir="rtl">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="/vite.svg" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Hospital Staff Skill Assessment</title>

<script src="https://cdn.tailwindcss.com"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

<script>

tailwind.config = {

theme: {

extend: {

fontFamily: {

sans: ['Vazirmatn', 'sans-serif'],

},

keyframes: {

gradient: {

'0%': { backgroundPosition: '0% 50%' },

'50%': { backgroundPosition: '100% 50%' },

'100%': { backgroundPosition: '0% 50%' },

},

rotate: {

'0%': { transform: 'rotate(0deg)' },

'100%': { transform: 'rotate(360deg)' },

},

'fade-in-up': {

'0%': { opacity: '0', transform: 'translateY(20px)' },

'100%': { opacity: '1', transform: 'translateY(0)' },

},

float: {

'0%': { transform: 'translateY(0px)' },

'50%': { transform: 'translateY(-20px)' },

'100%': { transform: 'translateY(0px)' },

},

'heartbeat-pulse': {

'0%, 100%': { transform: 'scale(1)' },

'50%': { transform: 'scale(1.15)' },

}

},

animation: {

gradient: 'gradient 15s ease infinite',

'spin-slow': 'rotate 20s linear infinite',

'fade-in-up': 'fade-in-up 0.8s ease-out forwards',

'float': 'float 12s ease-in-out infinite',

'heartbeat-pulse': 'heartbeat-pulse 1.5s ease-in-out infinite',

},

},

},

}

</script>

<link href="https://fonts.googleapis.com/css2?family=Vazirmatn:wght@400;500;600;700&display=swap" rel="stylesheet">

<link href="https://fonts.googleapis.com/css2?family=Scheherazade+New:wght@400;700&display=swap" rel="stylesheet">

<style>

body {

font-family: 'Vazirmatn', sans-serif;

background-color: rgb(248 250 252); /\* slate-50 \*/

}

@media (prefers-color-scheme: dark) {

body {

background-color: rgb(15 23 42); /\* slate-900 \*/

}

}

.animation-delay-200 { animation-delay: 200ms; }

.animation-delay-400 { animation-delay: 400ms; }

.animation-delay-600 { animation-delay: 600ms; }

.animation-delay-800 { animation-delay: 800ms; }

.animation-delay-1000 { animation-delay: 1000ms; }

.animation-delay-3000 { animation-delay: 3s; }

.animation-delay-6000 { animation-delay: 6s; }

/\* --- Custom Scrollbar Styles --- \*/

/\* For Webkit Browsers (Chrome, Safari, Edge) \*/

::-webkit-scrollbar {

width: 12px;

height: 12px;

}

::-webkit-scrollbar-track {

background-color: rgb(226 232 250); /\* slate-200 \*/

border-radius: 10px;

}

::-webkit-scrollbar-thumb {

background-color: rgb(14 165 233); /\* sky-500 \*/

border-radius: 10px;

border: 2px solid rgb(226 232 250); /\* slate-200 \*/

}

::-webkit-scrollbar-thumb:hover {

background-color: rgb(2 132 199); /\* sky-600 \*/

}

/\* For Firefox \*/

html {

scrollbar-width: thin;

/\* thumb color, track color \*/

scrollbar-color: rgb(14 165 233) rgb(226 232 250);

}

/\* --- Celestial Glow Effect --- \*/

.celestial-glow {

position: relative;

z-index: 0;

}

.celestial-glow::before {

content: '';

position: absolute;

z-index: -1;

inset: -10px;

background: conic-gradient(

from 180deg at 50% 50%,

#0ea5e9, /\* sky-500 \*/

#a855f7, /\* violet-500 \*/

#ec4899, /\* pink-500 \*/

#f97316, /\* orange-500 \*/

#0ea5e9

);

filter: blur(40px);

animation: rotate 10s linear infinite;

}

</style>

<script type="importmap">

{

"imports": {

"recharts": "https://esm.sh/recharts@^3.1.2",

"react-dom/": "https://esm.sh/react-dom@^19.1.1/",

"@google/genai": "https://esm.sh/@google/genai@^1.14.0",

"react/": "https://esm.sh/react@^19.1.1/",

"react": "https://esm.sh/react@^19.1.1",

"jspdf": "https://esm.sh/jspdf@2.5.1",

"html2canvas": "https://esm.sh/html2canvas@1.4.1",

"qrcode.react": "https://esm.sh/qrcode.react@3.1.0"

}

}

</script>

<link rel="stylesheet" href="/index.css">

</head>

<body class="text-slate-800 dark:text-slate-200">

<div id="root"></div>

<script type="module" src="/index.tsx"></script>

</body>

</html>

Index.tsx

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const rootElement = document.getElementById('root');

if (!rootElement) {

throw new Error("Could not find root element to mount to");

}

const root = ReactDOM.createRoot(rootElement);

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

Metadata.json

{

"name": "maharat",

"description": "An application to manage and evaluate the skills of hospital staff. It allows uploading skill checklists from Excel, dynamically assesses performance based on the categories defined in the file, and provides AI-driven suggestions for improvement.",

"requestFramePermissions": []

}

Package.json

{

"name": "maharat",

"private": true,

"version": "0.0.0",

"type": "module",

"scripts": {

"dev": "vite",

"build": "vite build",

"preview": "vite preview"

},

"dependencies": {

"recharts": "^3.1.2",

"react-dom": "^19.1.1",

"@google/genai": "^1.14.0",

"react": "^19.1.1",

"jspdf": "2.5.1",

"html2canvas": "1.4.1",

"qrcode.react": "3.1.0"

},

"devDependencies": {

"@types/node": "^22.14.0",

"typescript": "~5.8.2",

"vite": "^6.2.0"

}

}

tsconfig.json

{

"compilerOptions": {

"target": "ES2022",

"experimentalDecorators": true,

"useDefineForClassFields": false,

"module": "ESNext",

"lib": [

"ES2022",

"DOM",

"DOM.Iterable"

],

"skipLibCheck": true,

"types": [

"node"

],

"moduleResolution": "bundler",

"isolatedModules": true,

"moduleDetection": "force",

"allowJs": true,

"jsx": "react-jsx",

"paths": {

"@/\*": [

"./\*"

]

},

"allowImportingTsExtensions": true,

"noEmit": true

}

}

Types.ts

export interface SkillItem {

description: string;

score: number;

}

export interface SkillCategory {

name: string;

items: SkillItem[];

}

export interface Assessment {

id: string;

month: string;

skillCategories: SkillCategory[];

supervisorMessage?: string;

managerMessage?: string;

templateId?: string; // ID of the template used

minScore?: number; // Score range snapshot at time of assessment

maxScore?: number; // Score range snapshot at time of assessment

examSubmissions?: ExamSubmission[];

}

export interface MonthlyWorkLog {

month: string;

overtimeHours: number;

requiredHours: number;

quarterlyLeaveRemaining: number;

annualLeaveRemaining: number;

}

export interface StaffMember {

id: string;

name: string;

title: string;

nationalId?: string;

password?: string;

assessments: Assessment[];

workLogs?: MonthlyWorkLog[];

}

export interface ChecklistItemTemplate {

id: string;

description: string;

}

export interface ChecklistCategoryTemplate {

id: string;

name: string;

items: ChecklistItemTemplate[];

}

export interface NamedChecklistTemplate {

id: string;

name: string;

categories: ChecklistCategoryTemplate[];

minScore?: number;

maxScore?: number;

}

export enum QuestionType {

MultipleChoice = 'multiple-choice',

Descriptive = 'descriptive',

}

export interface Question {

id: string;

text: string;

type: QuestionType;

options?: string[];

correctAnswer: string; // for MC, it's the option text. for descriptive, it's the model answer.

}

export interface ExamTemplate {

id: string;

name: string;

questions: Question[];

}

export interface ExamAnswer {

questionId: string;

answer: string;

}

export interface ExamSubmission {

id: string;

examTemplateId: string;

examName: string; // denormalize for easier display

answers: ExamAnswer[];

score: number; // Correct answers

totalCorrectableQuestions: number; // Total multiple choice questions

submissionDate: string;

questions: Question[]; // Snapshot of questions at time of submission

}

export interface ChatMessage {

id: string;

sender: 'patient' | 'manager';

timestamp: string;

text?: string;

file?: {

id: string;

name: string;

type: string;

}

}

export interface Patient {

id: string;

name: string;

nationalId: string;

password?: string;

chatHistory?: ChatMessage[];

}

export interface Department {

id: string;

name: string;

managerName: string;

managerNationalId: string;

managerPassword: string;

staffCount: number;

bedCount: number;

staff: StaffMember[];

patientEducationMaterials?: TrainingMaterial[];

patients?: Patient[];

}

export interface TrainingMaterial {

id: string;

name: string;

type: string; // Mime type

data?: string; // Base64 encoded data URL

description?: string;

}

export interface MonthlyTraining {

month: string;

materials: TrainingMaterial[];

}

export interface NewsBanner {

id: string;

title: string;

description: string;

imageId: string; // a reference to the image data stored in IndexedDB

}

export interface AdminMessage {

id: string;

sender: 'hospital' | 'admin';

timestamp: string;

text?: string;

file?: {

id: string;

name: string;

type: string;

}

}

export interface Hospital {

id: string;

name: string;

province: string;

city: string;

supervisorName?: string;

supervisorNationalId?: string;

supervisorPassword?: string;

departments: Department[];

checklistTemplates?: NamedChecklistTemplate[];

examTemplates?: ExamTemplate[];

trainingMaterials?: MonthlyTraining[];

accreditationMaterials?: TrainingMaterial[];

newsBanners?: NewsBanner[];

adminMessages?: AdminMessage[];

}

export enum AppScreen {

Welcome,

HospitalList,

MainApp,

}

export enum View {

DepartmentList,

DepartmentView,

StaffMemberView,

ChecklistManager,

ExamManager,

TrainingManager,

AccreditationManager,

NewsBannerManager,

PatientEducationManager,

PatientPortal,

HospitalCommunication,

AdminCommunication,

}

export enum UserRole {

Admin,

Supervisor,

Manager,

Staff,

Patient,

}

export interface LoggedInUser {

role: UserRole;

name: string;

hospitalId?: string;

departmentId?: string;

staffId?: string;

patientId?: string;

}

Vit.config.ts

import path from 'path';

import { defineConfig, loadEnv } from 'vite';

export default defineConfig(({ mode }) => {

const env = loadEnv(mode, '.', '');

return {

define: {

'process.env.API\_KEY': JSON.stringify(env.GEMINI\_API\_KEY),

'process.env.GEMINI\_API\_KEY': JSON.stringify(env.GEMINI\_API\_KEY)

},

resolve: {

alias: {

'@': path.resolve(\_\_dirname, '.'),

}

}

};

});

.env.local

GEMINI\_API\_KEY=PLACEHOLDER\_API\_KEY

.gitignore

# Logs

logs

\*.log

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

pnpm-debug.log\*

lerna-debug.log\*

node\_modules

dist

dist-ssr

\*.local

# Editor directories and files

.vscode/\*

!.vscode/extensions.json

.idea

.DS\_Store

\*.suo

\*.ntvs\*

\*.njsproj

\*.sln

\*.sw?

Components folder :

Aboutmodel.tsx

import React from 'react';

import Modal from './Modal';

interface AboutModalProps {

isOpen: boolean;

onClose: () => void;

}

const AboutModal: React.FC<AboutModalProps> = ({ isOpen, onClose }) => {

return (

<Modal isOpen={isOpen} onClose={onClose} title="درباره برنامه" maxWidthClass="max-w-2xl">

<div className="space-y-4 text-slate-600 dark:text-slate-300 text-right leading-relaxed">

<p>

سامانه «مهارت» یک پلتفرم پیشرفته و یکپارچه برای مدیریت هوشمند عملکرد و توانمندسازی پرسنل مراکز درمانی است. این سامانه با هدف دیجیتالی کردن فرآیندهای ارزیابی، آموزش و بهبود مستمر طراحی شده تا به مدیران در تصمیم‌گیری‌های مبتنی بر داده و به پرسنل در مسیر رشد حرفه‌ای خود کمک کند.

</p>

<h4 className="font-bold text-lg text-slate-800 dark:text-slate-100 pt-2">قابلیت‌های برجسته سامانه:</h4>

<ul className="list-disc list-inside space-y-2 pr-4">

<li><strong>مدیریت جامع:</strong> تعریف و مدیریت همزمان چندین بیمارستان، بخش و پرسنل با سطوح دسترسی مختلف (ادمین، سوپروایزر، مسئول بخش).</li>

<li><strong>ارزیابی عملکرد:</strong> امکان بارگذاری چک‌لیست‌های عملکردی از طریق فایل اکسل یا ساخت قالب‌های سفارشی‌سازی‌شده درون برنامه برای ارزیابی دقیق مهارت‌ها.</li>

<li><strong>آزمون‌های آنلاین:</strong> طراحی و برگزاری آزمون‌های تئوری (چهارگزینه‌ای و تشریحی) برای سنجش دانش پرسنل و مشاهده آنی نتایج.</li>

<li><strong>مدیریت آموزش:</strong> بارگذاری و آرشیو محتواهای آموزشی (ویدئو، PDF، تصویر و...) به تفکیک ماه برای دسترسی آسان پرسنل.</li>

<li><strong>تحلیل و گزارش‌دهی:</strong> مشاهده روند پیشرفت فردی و گروهی با نمودارهای تحلیلی و بصری.</li>

<li><strong>برنامه بهبود هوشمند:</strong> ارائه خودکار برنامه مطالعاتی ۳۰ روزه بر اساس نقاط ضعف شناسایی‌شده در ارزیابی‌ها جهت توانمندسازی هدفمند.</li>

<li><strong>ذخیره‌سازی و امنیت داده:</strong> قابلیت پشتیبان‌گیری و بازیابی اطلاعات در سطوح مختلف (بخش، بیمارستان یا کل داده‌ها) و عملکرد آفلاین برای دسترسی پایدار.</li>

</ul>

<div className="pt-4 text-center text-sm text-slate-500 dark:text-slate-400">

<p>سازنده: حسین نصاری</p>

</div>

</div>

</Modal>

);

};

export default AboutModal;

AccreditationManager.tsx

import React, { useState } from 'react';

import { TrainingMaterial } from '../types';

import { TrashIcon } from './icons/TrashIcon';

import { EditIcon } from './icons/EditIcon';

import FileUploader from './FileUploader';

import PreviewModal from './PreviewModal';

import Modal from './Modal';

import { ImageIcon } from './icons/ImageIcon';

import { VideoIcon } from './icons/VideoIcon';

import { AudioIcon } from './icons/AudioIcon';

import { PdfIcon } from './icons/PdfIcon';

import { DocumentIcon } from './icons/DocumentIcon';

interface AccreditationManagerProps {

materials: TrainingMaterial[];

onAddMaterial: (material: TrainingMaterial) => void;

onDeleteMaterial: (materialId: string) => void;

onUpdateMaterialDescription: (materialId: string, description: string) => void;

onBack: () => void;

}

const getIconForMimeType = (type: string): { icon: React.ReactNode, color: string } => {

if (type.startsWith('image/')) return { icon: <ImageIcon className="w-10 h-10" />, color: 'text-blue-500' };

if (type.startsWith('video/')) return { icon: <VideoIcon className="w-10 h-10" />, color: 'text-red-500' };

if (type.startsWith('audio/')) return { icon: <AudioIcon className="w-10 h-10" />, color: 'text-purple-500' };

if (type === 'application/pdf') return { icon: <PdfIcon className="w-10 h-10" />, color: 'text-orange-500' };

return { icon: <DocumentIcon className="w-10 h-10" />, color: 'text-slate-500' };

};

const AccreditationManager: React.FC<AccreditationManagerProps> = ({ materials, onAddMaterial, onDeleteMaterial, onUpdateMaterialDescription, onBack }) => {

const [isUploading, setIsUploading] = useState(false);

const [uploadError, setUploadError] = useState<string | null>(null);

const [previewMaterial, setPreviewMaterial] = useState<TrainingMaterial | null>(null);

// For description modal

const [descriptionModalOpen, setDescriptionModalOpen] = useState(false);

const [editingMaterial, setEditingMaterial] = useState<TrainingMaterial | null>(null);

const [pendingFile, setPendingFile] = useState<{file: File, dataUrl: string} | null>(null);

const [materialDescription, setMaterialDescription] = useState('');

const handleFileUpload = (file: File) => {

setIsUploading(true);

setUploadError(null);

const reader = new FileReader();

reader.onload = (e) => {

try {

const dataUrl = e.target?.result as string;

if (!dataUrl) throw new Error("Failed to read file.");

setPendingFile({ file, dataUrl });

setMaterialDescription(''); // Reset for new file

setDescriptionModalOpen(true);

} catch (err) {

setUploadError(err instanceof Error ? err.message : "خطای ناشناخته در پردازش فایل.");

} finally {

setIsUploading(false);

}

};

reader.onerror = () => {

setUploadError("خطا در خواندن فایل.");

setIsUploading(false);

};

reader.readAsDataURL(file);

};

const handleDeleteClick = (materialId: string, materialName: string) => {

if (window.confirm(`آیا از حذف فایل "${materialName}" مطمئن هستید؟`)) {

onDeleteMaterial(materialId);

}

};

const handleSaveMaterialWithDescription = () => {

if (pendingFile) { // Adding new

const { file, dataUrl } = pendingFile;

const newMaterial: TrainingMaterial = {

id: Date.now().toString(),

name: file.name,

type: file.type,

data: dataUrl,

description: materialDescription.trim(),

};

onAddMaterial(newMaterial);

} else if (editingMaterial) { // Editing existing

onUpdateMaterialDescription(editingMaterial.id, materialDescription.trim());

}

setDescriptionModalOpen(false);

setPendingFile(null);

setEditingMaterial(null);

setMaterialDescription('');

};

const handleOpenEditDescriptionModal = (material: TrainingMaterial) => {

setPendingFile(null);

setEditingMaterial(material);

setMaterialDescription(material.description || '');

setDescriptionModalOpen(true);

};

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">مدیریت مطالب اعتباربخشی</h1>

</div>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-xl font-bold mb-4">بارگذاری محتوای جدید</h2>

{isUploading && <p className="text-center text-indigo-500">در حال پردازش فایل...</p>}

{uploadError && <p className="text-center text-red-500 my-2">{uploadError}</p>}

<FileUploader onFileUpload={handleFileUpload} accept="\*" title="آپلود فایل، صدا، عکس یا فیلم" />

<div className="mt-8">

<h3 className="text-xl font-bold mb-4 border-t pt-6 border-slate-200 dark:border-slate-700">محتوای بارگذاری شده</h3>

{materials.length === 0 ? (

<p className="text-center py-8 text-slate-400">هیچ محتوایی بارگذاری نشده است.</p>

) : (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4">

{materials.map(material => {

const { icon, color } = getIconForMimeType(material.type);

return (

<div key={material.id} className="group relative flex flex-col text-right p-4 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-md transition-all text-slate-800 dark:text-slate-200">

<button onClick={() => setPreviewMaterial(material)} className="flex-grow flex flex-col text-right">

<div className={`mb-3 ${color}`}>{icon}</div>

<h4 className="font-bold text-sm break-all w-full truncate" title={material.name}>{material.name}</h4>

<p className="text-xs text-slate-500 dark:text-slate-400 mt-1 flex-grow h-8 overflow-hidden text-ellipsis">

{material.description || 'بدون توضیح'}

</p>

</button>

<div className="absolute top-2 left-2 flex items-center gap-1 opacity-0 group-hover:opacity-100 transition-opacity">

<button onClick={() => handleOpenEditDescriptionModal(material)} className="p-1.5 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-500 dark:text-slate-300 hover:text-indigo-500 dark:hover:text-indigo-400">

<EditIcon className="w-4 h-4"/>

</button>

<button onClick={() => handleDeleteClick(material.id, material.name)} className="p-1.5 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-500 dark:text-slate-300 hover:text-red-500 dark:hover:text-red-400">

<TrashIcon className="w-4 h-4"/>

</button>

</div>

</div>

)

})}

</div>

)}

</div>

</div>

{previewMaterial && <PreviewModal isOpen={!!previewMaterial} onClose={() => setPreviewMaterial(null)} material={previewMaterial}/>}

<Modal isOpen={descriptionModalOpen} onClose={() => setDescriptionModalOpen(false)} title={pendingFile ? "افزودن توضیحات" : "ویرایش توضیحات"}>

<div className="space-y-4">

<textarea

value={materialDescription}

onChange={(e) => setMaterialDescription(e.target.value)}

placeholder="توضیحات مربوط به فایل را اینجا وارد کنید..."

rows={4}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3">

<button onClick={() => setDescriptionModalOpen(false)} className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500">

انصراف

</button>

<button onClick={handleSaveMaterialWithDescription} className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700">

ذخیره

</button>

</div>

</div>

</Modal>

</div>

);

};

export default AccreditationManager;

AdminCommunicationView.tsx

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

import { Hospital, TrainingMaterial } from '../types';

import { BackIcon } from './icons/BackIcon';

import \* as db from '../services/db';

import { PaperClipIcon } from './icons/PaperClipIcon';

import PreviewModal from './PreviewModal';

import { ImageIcon } from './icons/ImageIcon';

import { VideoIcon } from './icons/VideoIcon';

import { AudioIcon } from './icons/AudioIcon';

import { PdfIcon } from './icons/PdfIcon';

import { DocumentIcon } from './icons/DocumentIcon';

const getIconForMimeType = (type: string): { icon: React.ReactNode, color: string } => {

if (type.startsWith('image/')) return { icon: <ImageIcon className="w-8 h-8" />, color: 'text-blue-500' };

if (type.startsWith('video/')) return { icon: <VideoIcon className="w-8 h-8" />, color: 'text-red-500' };

if (type.startsWith('audio/')) return { icon: <AudioIcon className="w-8 h-8" />, color: 'text-purple-500' };

if (type === 'application/pdf') return { icon: <PdfIcon className="w-8 h-8" />, color: 'text-orange-500' };

return { icon: <DocumentIcon className="w-8 h-8" />, color: 'text-slate-500' };

};

interface AdminCommunicationViewProps {

hospitals: Hospital[];

onSendMessage: (hospitalId: string, content: { text?: string; file?: { id: string; name: string; type: string; } }) => void;

onBack: () => void;

}

const AdminCommunicationView: React.FC<AdminCommunicationViewProps> = ({ hospitals, onSendMessage, onBack }) => {

const [selectedHospitalId, setSelectedHospitalId] = useState<string | null>(hospitals[0]?.id || null);

const [replyTexts, setReplyTexts] = useState<{ [hospitalId: string]: string }>({});

const [previewMaterial, setPreviewMaterial] = useState<TrainingMaterial | null>(null);

const messagesEndRef = useRef<HTMLDivElement>(null);

const fileInputRef = useRef<HTMLInputElement>(null);

const selectedHospital = hospitals.find(h => h.id === selectedHospitalId);

const scrollToBottom = () => {

messagesEndRef.current?.scrollIntoView({ behavior: "auto" });

};

useEffect(() => {

scrollToBottom();

}, [selectedHospitalId, selectedHospital?.adminMessages]);

const handleSendReply = (hospitalId: string) => {

const text = replyTexts[hospitalId];

if (text && text.trim()) {

onSendMessage(hospitalId, { text: text.trim() });

setReplyTexts(prev => ({ ...prev, [hospitalId]: '' }));

}

};

const handleFileSelect = async (e: React.ChangeEvent<HTMLInputElement>) => {

if (!selectedHospitalId) return;

const file = e.target.files?.[0];

if (!file) return;

try {

const dataUrl = await new Promise<string>((resolve, reject) => {

const reader = new FileReader();

reader.onload = (event) => resolve(event.target?.result as string);

reader.onerror = (error) => reject(error);

reader.readAsDataURL(file);

});

const fileId = `chat-file-${Date.now()}`;

await db.addMaterial({ id: fileId, data: dataUrl });

onSendMessage(selectedHospitalId, {

file: {

id: fileId,

name: file.name,

type: file.type,

}

});

} catch (error) {

console.error("Error sending file:", error);

alert("خطا در ارسال فایل.");

} finally {

if (fileInputRef.current) {

fileInputRef.current.value = '';

}

}

};

const handleKeyDown = (e: React.KeyboardEvent<HTMLTextAreaElement>, hospitalId: string) => {

if (e.key === 'Enter' && !e.shiftKey) {

e.preventDefault();

handleSendReply(hospitalId);

}

};

return (

<>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md flex h-[calc(100vh-12rem)]">

{/\* Left Column: Hospital List \*/}

<div className="w-1/3 border-l border-slate-200 dark:border-slate-700 flex flex-col">

<div className="p-4 border-b border-slate-200 dark:border-slate-700 flex justify-between items-center">

<h2 className="text-xl font-bold">بیمارستان‌ها</h2>

<button

onClick={onBack}

className="inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold text-slate-700 bg-slate-200 dark:bg-slate-700 dark:text-slate-200 rounded-lg hover:bg-slate-300 dark:hover:bg-slate-600 border border-slate-300 dark:border-slate-600"

>

<BackIcon className="w-5 h-5" />

بازگشت

</button>

</div>

<div className="flex-grow overflow-y-auto">

{hospitals.map(h => (

<button

key={h.id}

onClick={() => setSelectedHospitalId(h.id)}

className={`w-full text-right p-4 border-b dark:border-slate-700 transition-colors ${selectedHospitalId === h.id ? 'bg-sky-100 dark:bg-sky-900/50' : 'hover:bg-slate-50 dark:hover:bg-slate-700/50'}`}

>

<p className="font-bold text-slate-800 dark:text-slate-100">{h.name}</p>

<p className="text-sm text-slate-500 dark:text-slate-400">{(h.adminMessages || []).length} پیام</p>

</button>

))}

</div>

</div>

{/\* Right Column: Chat View \*/}

<div className="w-2/3 flex flex-col">

{selectedHospital ? (

<>

<div className="p-4 border-b border-slate-200 dark:border-slate-700">

<h3 className="text-xl font-bold text-slate-800 dark:text-slate-100">{selectedHospital.name}</h3>

</div>

<div className="flex-grow overflow-y-auto p-4 space-y-4 bg-slate-50 dark:bg-slate-900/50">

{(selectedHospital.adminMessages || []).map(msg => (

<div key={msg.id} className={`flex items-end gap-2 ${msg.sender === 'admin' ? 'justify-end' : 'justify-start'}`}>

<div className={`max-w-xl p-3 rounded-lg shadow ${msg.sender === 'admin' ? 'bg-green-500 text-white rounded-br-none' : 'bg-slate-200 dark:bg-slate-700 text-slate-800 dark:text-slate-100 rounded-bl-none'}`}>

{msg.text && <p className="whitespace-pre-wrap">{msg.text}</p>}

{msg.file && (

<button

onClick={() => setPreviewMaterial({ id: msg.file!.id, name: msg.file!.name, type: msg.file!.type })}

className="flex items-center gap-3 text-left"

>

<div className={`flex-shrink-0 ${msg.sender === 'admin' ? 'text-white' : 'text-slate-600 dark:text-slate-300'}`}>

{getIconForMimeType(msg.file.type).icon}

</div>

<div>

<p className="font-semibold break-all">{msg.file.name}</p>

<p className="text-xs opacity-80">برای مشاهده کلیک کنید</p>

</div>

</button>

)}

<p className={`text-xs mt-1 text-right ${msg.sender === 'admin' ? 'text-green-100' : 'text-slate-400'}`}>

{new Date(msg.timestamp).toLocaleString('fa-IR', { hour: '2-digit', minute: '2-digit' })}

</p>

</div>

</div>

))}

<div ref={messagesEndRef} />

</div>

<div className="p-4 border-t border-slate-200 dark:border-slate-700 flex items-center gap-2">

<input type="file" ref={fileInputRef} onChange={handleFileSelect} className="hidden" />

<button

onClick={() => fileInputRef.current?.click()}

className="p-3 text-slate-500 bg-slate-100 dark:bg-slate-700 dark:text-slate-300 rounded-lg hover:bg-slate-200 dark:hover:bg-slate-600"

aria-label="Attach file"

>

<PaperClipIcon className="w-6 h-6"/>

</button>

<textarea

value={replyTexts[selectedHospital.id] || ''}

onChange={(e) => setReplyTexts(prev => ({ ...prev, [selectedHospital.id]: e.target.value }))}

onKeyDown={(e) => handleKeyDown(e, selectedHospital.id)}

placeholder="پاسخ خود را بنویسید..."

rows={2}

className="flex-grow px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<button

onClick={() => handleSendReply(selectedHospital.id)}

className="px-6 py-3 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 disabled:opacity-50"

disabled={!(replyTexts[selectedHospital.id] || '').trim()}

>

ارسال

</button>

</div>

</>

) : (

<div className="flex items-center justify-center h-full text-slate-500">

<p>برای مشاهده مکالمات، یک بیمارستان را انتخاب کنید.</p>

</div>

)}

</div>

</div>

<PreviewModal isOpen={!!previewMaterial} onClose={() => setPreviewMaterial(null)} material={previewMaterial!}/>

</>

);

};

export default AdminCommunicationView;

ChecklistBuilder.tsx

import React, { useState } from 'react';

import { ChecklistCategoryTemplate, ChecklistItemTemplate, NamedChecklistTemplate } from '../types';

import { PlusIcon } from './icons/PlusIcon';

import { TrashIcon } from './icons/TrashIcon';

import { SaveIcon } from './icons/SaveIcon';

interface ChecklistBuilderProps {

template: NamedChecklistTemplate;

onSave: (newTemplate: NamedChecklistTemplate) => void;

onCancel: () => void;

}

const ChecklistBuilder: React.FC<ChecklistBuilderProps> = ({ template, onSave, onCancel }) => {

const [name, setName] = useState(template.name || '');

const [minScore, setMinScore] = useState<number>(template.minScore ?? 0);

const [maxScore, setMaxScore] = useState<number>(template.maxScore ?? 4);

const [categories, setCategories] = useState<ChecklistCategoryTemplate[]>(

() => JSON.parse(JSON.stringify(template.categories || [])) // Deep copy

);

const handleAddCategory = () => {

const newCategory: ChecklistCategoryTemplate = {

id: Date.now().toString(),

name: 'دسته بندی جدید',

items: [],

};

setCategories([...categories, newCategory]);

};

const handleCategoryChange = (categoryId: string, field: 'name', value: string) => {

setCategories(

categories.map(cat => (cat.id === categoryId ? { ...cat, [field]: value } : cat))

);

};

const handleDeleteCategory = (categoryId: string) => {

if (window.confirm('آیا از حذف این دسته و تمام موارد آن مطمئن هستید؟')) {

setCategories(categories.filter(cat => cat.id !== categoryId));

}

};

const handleAddItem = (categoryId: string) => {

const newItem: ChecklistItemTemplate = {

id: Date.now().toString(),

description: 'مورد مهارت جدید',

};

setCategories(

categories.map(cat =>

cat.id === categoryId ? { ...cat, items: [...cat.items, newItem] } : cat

)

);

};

const handleItemChange = (categoryId: string, itemId: string, value: string) => {

setCategories(categories.map(cat =>

cat.id === categoryId ? {

...cat,

items: cat.items.map(item =>

item.id === itemId ? {...item, description: value} : item

)

} : cat

))

}

const handleDeleteItem = (categoryId: string, itemId: string) => {

setCategories(

categories.map(cat =>

cat.id === categoryId

? { ...cat, items: cat.items.filter(item => item.id !== itemId) }

: cat

)

);

};

const handleSaveChanges = () => {

if (!name.trim()) {

alert('نام چک لیست نمی‌تواند خالی باشد.');

return;

}

const numMin = Number(minScore);

const numMax = Number(maxScore);

if (isNaN(numMin) || isNaN(numMax)) {

alert('مقادیر حداقل و حداکثر نمره باید عدد باشند.');

return;

}

if (numMin >= numMax) {

alert('حداقل نمره باید کمتر از حداکثر نمره باشد.');

return;

}

onSave({ ...template, name: name.trim(), minScore: numMin, maxScore: numMax, categories });

alert('قالب با موفقیت ذخیره شد.');

};

return (

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<div className="flex justify-between items-center mb-6 pb-4 border-b border-slate-200 dark:border-slate-700">

<div>

<h2 className="text-2xl font-bold">ساخت / ویرایش قالب چک‌لیست</h2>

<p className="text-slate-500 dark:text-slate-400">یک قالب برای ارزیابی عملکردی پرسنل ایجاد کنید.</p>

</div>

<div className="flex gap-2">

<button

onClick={onCancel}

className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500"

>

انصراف

</button>

<button

onClick={handleSaveChanges}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-green-600 rounded-lg hover:bg-green-700"

>

<SaveIcon className="w-5 h-5"/>

ذخیره قالب

</button>

</div>

</div>

<div className="grid grid-cols-1 md:grid-cols-3 gap-6 mb-6">

<div className="md:col-span-1">

<label htmlFor="template-name" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

نام قالب

</label>

<input

id="template-name"

type="text"

value={name}

onChange={e => setName(e.target.value)}

placeholder="مثال: چک لیست عمومی پرستاری"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

<div>

<label htmlFor="min-score" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

حداقل نمره

</label>

<input

id="min-score"

type="number"

value={minScore}

onChange={e => setMinScore(Number(e.target.value))}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

<div>

<label htmlFor="max-score" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

حداکثر نمره

</label>

<input

id="max-score"

type="number"

value={maxScore}

onChange={e => setMaxScore(Number(e.target.value))}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

</div>

<div className="space-y-6">

{categories.map(cat => (

<div key={cat.id} className="border border-slate-200 dark:border-slate-700 rounded-lg p-4">

<div className="flex justify-between items-center mb-4">

<input

type="text"

value={cat.name}

onChange={e => handleCategoryChange(cat.id, 'name', e.target.value)}

placeholder="نام دسته"

className="text-lg font-bold bg-transparent focus:outline-none focus:ring-0 border-none p-0 text-slate-800 dark:text-slate-100 w-full"

/>

<button

onClick={() => handleDeleteCategory(cat.id)}

className="p-2 text-slate-400 hover:text-red-500 rounded-full"

aria-label="Delete Category"

>

<TrashIcon className="w-5 h-5" />

</button>

</div>

<div className="space-y-2 pl-4">

{cat.items.map(item => (

<div key={item.id} className="flex items-center gap-2">

<input

type="text"

value={item.description}

onChange={(e) => handleItemChange(cat.id, item.id, e.target.value)}

className="flex-grow px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-indigo-500"

placeholder="شرح مهارت"

/>

<button

onClick={() => handleDeleteItem(cat.id, item.id)}

className="p-2 text-slate-400 hover:text-red-500 rounded-full"

aria-label={`Delete skill ${item.description}`}

>

<TrashIcon className="w-4 h-4" />

</button>

</div>

))}

</div>

<div className="mt-4">

<button

onClick={() => handleAddItem(cat.id)}

className="inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold text-indigo-600 bg-indigo-100 rounded-md hover:bg-indigo-200 dark:bg-indigo-900/50 dark:text-indigo-300 dark:hover:bg-indigo-900"

>

<PlusIcon className="w-4 h-4" />

افزودن مورد

</button>

</div>

</div>

))}

</div>

<div className="mt-6 border-t border-slate-200 dark:border-slate-700 pt-6">

<button

onClick={handleAddCategory}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-slate-600 rounded-lg hover:bg-slate-700"

>

<PlusIcon className="w-5 h-5" />

افزودن دسته بندی جدید

</button>

</div>

</div>

);

};

export default ChecklistBuilder;

ChecklistManager.tsx

import React, { useState } from 'react';

import { NamedChecklistTemplate } from '../types';

import ChecklistBuilder from './ChecklistBuilder';

import { PlusIcon } from './icons/PlusIcon';

import { EditIcon } from './icons/EditIcon';

import { TrashIcon } from './icons/TrashIcon';

interface ChecklistManagerProps {

templates: NamedChecklistTemplate[];

onAddOrUpdate: (template: NamedChecklistTemplate) => void;

onDelete: (templateId: string) => void;

onBack: () => void;

}

const ChecklistManager: React.FC<ChecklistManagerProps> = ({ templates, onAddOrUpdate, onDelete, onBack }) => {

const [editingTemplate, setEditingTemplate] = useState<NamedChecklistTemplate | null>(null);

const handleAddNew = () => {

setEditingTemplate({

id: Date.now().toString(),

name: '',

categories: []

});

};

const handleSave = (template: NamedChecklistTemplate) => {

onAddOrUpdate(template);

setEditingTemplate(null);

};

const handleDelete = (templateId: string, templateName: string) => {

if(window.confirm(`آیا از حذف قالب چک لیست "${templateName}" مطمئن هستید؟ این عمل قابل بازگشت نیست.`)){

onDelete(templateId);

}

}

if (editingTemplate) {

return (

<div className="p-4 sm:p-6 lg:p-8">

<ChecklistBuilder

template={editingTemplate}

onSave={handleSave}

onCancel={() => setEditingTemplate(null)}

/>

</div>

)

}

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">مدیریت قالب‌های چک‌لیست</h1>

<button

onClick={handleAddNew}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"

>

<PlusIcon className="w-5 h-5" />

افزودن قالب جدید

</button>

</div>

{templates.length === 0 ? (

<div className="text-center py-16 bg-white dark:bg-slate-800 rounded-lg shadow">

<h2 className="text-xl font-medium text-slate-500">هیچ قالبی یافت نشد.</h2>

<p className="text-slate-400 mt-2">برای شروع، یک قالب چک‌لیست جدید اضافه کنید.</p>

</div>

) : (

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md overflow-x-auto">

<table className="w-full text-sm text-right text-slate-500 dark:text-slate-400">

<thead className="text-xs text-slate-700 uppercase bg-slate-50 dark:bg-slate-700 dark:text-slate-300">

<tr>

<th scope="col" className="px-6 py-3">نام قالب</th>

<th scope="col" className="px-6 py-3">جزئیات</th>

<th scope="col" className="px-6 py-3 text-center">

<span className="sr-only">اقدامات</span>

</th>

</tr>

</thead>

<tbody>

{templates.map((template) => (

<tr key={template.id} className="group border-b dark:border-slate-700 odd:bg-white odd:dark:bg-slate-800 even:bg-slate-50 even:dark:bg-slate-700/50">

<td scope="row" className="px-6 py-4 font-medium text-slate-900 whitespace-nowrap dark:text-white">

{template.name}

</td>

<td className="px-6 py-4">

{template.categories.length} دسته, {template.categories.reduce((acc, cat) => acc + cat.items.length, 0)} مورد

</td>

<td className="px-6 py-4 text-center">

<div className="flex items-center justify-center opacity-0 group-hover:opacity-100 transition-opacity">

<button

onClick={() => setEditingTemplate(template)}

className="p-2 text-slate-400 hover:text-indigo-500 rounded-full"

aria-label="Edit Template"

>

<EditIcon className="w-5 h-5" />

</button>

<button

onClick={() => handleDelete(template.id, template.name)}

className="p-2 text-slate-400 hover:text-red-500 rounded-full"

aria-label="Delete Template"

>

<TrashIcon className="w-5 h-5" />

</button>

</div>

</td>

</tr>

))}

</tbody>

</table>

</div>

)}

</div>

);

};

export default ChecklistManager;

DepartmentList.tsx

import React, { useState } from 'react';

import { Department, UserRole } from '../types';

import Modal from './Modal';

import { PlusIcon } from './icons/PlusIcon';

import { TrashIcon } from './icons/TrashIcon';

import { EditIcon } from './icons/EditIcon';

import { ShieldCheckIcon } from './icons/ShieldCheckIcon';

import { NewspaperIcon } from './icons/NewspaperIcon';

import { RefreshIcon } from './icons/RefreshIcon';

import { ChatIcon } from './icons/ChatIcon';

interface DepartmentListProps {

departments: Department[];

hospitalName: string;

onAddDepartment: (name: string, managerName: string, managerNationalId: string, managerPassword: string, staffCount: number, bedCount: number) => void;

onUpdateDepartment: (id: string, updatedData: Partial<Omit<Department, 'id' | 'staff'>>) => void;

onDeleteDepartment: (id: string) => void;

onSelectDepartment: (id: string) => void;

onBack: () => void;

onManageAccreditation: () => void;

onManageNewsBanners: () => void;

onResetHospital: (supervisorNationalId: string, supervisorPassword: string) => boolean;

onContactAdmin: () => void;

userRole: UserRole;

}

const DepartmentList: React.FC<DepartmentListProps> = ({

departments,

hospitalName,

onAddDepartment,

onUpdateDepartment,

onDeleteDepartment,

onSelectDepartment,

onBack,

onManageAccreditation,

onManageNewsBanners,

onResetHospital,

onContactAdmin,

userRole,

}) => {

const [isModalOpen, setIsModalOpen] = useState(false);

const [editingDepartment, setEditingDepartment] = useState<Department | null>(null);

const [newDepartmentName, setNewDepartmentName] = useState('');

const [newManagerName, setNewManagerName] = useState('');

const [newManagerNationalId, setNewManagerNationalId] = useState('');

const [newManagerPassword, setNewManagerPassword] = useState('');

const [newStaffCount, setNewStaffCount] = useState('');

const [newBedCount, setNewBedCount] = useState('');

// State for reset modal

const [isResetModalOpen, setIsResetModalOpen] = useState(false);

const [resetStep, setResetStep] = useState(1); // 1: confirm, 2: credentials

const [supervisorIdInput, setSupervisorIdInput] = useState('');

const [supervisorPasswordInput, setSupervisorPasswordInput] = useState('');

const [resetError, setResetError] = useState<string | null>(null);

const resetForm = () => {

setNewDepartmentName('');

setNewManagerName('');

setNewManagerNationalId('');

setNewManagerPassword('');

setNewStaffCount('');

setNewBedCount('');

setEditingDepartment(null);

}

const handleOpenAddModal = () => {

resetForm();

setIsModalOpen(true);

};

const handleOpenEditModal = (dep: Department) => {

setEditingDepartment(dep);

setNewDepartmentName(dep.name);

setNewManagerName(dep.managerName);

setNewManagerNationalId(dep.managerNationalId);

setNewManagerPassword(dep.managerPassword);

setNewStaffCount(String(dep.staffCount));

setNewBedCount(String(dep.bedCount));

setIsModalOpen(true);

};

const handleCloseModal = () => {

resetForm();

setIsModalOpen(false);

};

const handleSaveDepartment = () => {

if (newDepartmentName.trim() && newManagerName.trim() && newManagerNationalId.trim() && newManagerPassword.trim() && newStaffCount && newBedCount) {

const staffCountNum = parseInt(newStaffCount, 10);

const bedCountNum = parseInt(newBedCount, 10);

if(editingDepartment) {

onUpdateDepartment(editingDepartment.id, {

name: newDepartmentName.trim(),

managerName: newManagerName.trim(),

managerNationalId: newManagerNationalId.trim(),

managerPassword: newManagerPassword.trim(),

staffCount: staffCountNum,

bedCount: bedCountNum,

});

} else {

onAddDepartment(

newDepartmentName.trim(),

newManagerName.trim(),

newManagerNationalId.trim(),

newManagerPassword.trim(),

staffCountNum,

bedCountNum

);

}

resetForm();

setIsModalOpen(false);

} else {

alert("لطفاً تمام فیلدها را پر کنید.")

}

};

// --- Reset Hospital Modal Handlers ---

const resetResetModal = () => {

setIsResetModalOpen(false);

setTimeout(() => {

setResetStep(1);

setSupervisorIdInput('');

setSupervisorPasswordInput('');

setResetError(null);

}, 300); // Delay reset to allow modal to close gracefully

};

const handleOpenResetModal = () => {

setResetError(null);

setResetStep(1);

setIsResetModalOpen(true);

};

const handleConfirmReset = () => {

const success = onResetHospital(supervisorIdInput, supervisorPasswordInput);

if (success) {

alert('تمام بخش‌های بیمارستان با موفقیت حذف شدند.');

resetResetModal();

} else {

setResetError('کد ملی یا رمز عبور سوپروایزر نامعتبر است.');

}

};

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">بخش های بیمارستان: <span className="text-slate-600 dark:text-slate-400">{hospitalName}</span></h1>

<div className="flex items-center gap-2">

{userRole === UserRole.Supervisor && (

<button

onClick={onContactAdmin}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-violet-600 rounded-lg hover:bg-violet-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-violet-500"

>

<ChatIcon className="w-5 h-5" />

تماس با ادمین کل

</button>

)}

<button

onClick={onManageNewsBanners}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-cyan-600 rounded-lg hover:bg-cyan-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-cyan-500"

>

<NewspaperIcon className="w-5 h-5" />

افزودن بنرهای خبری

</button>

<button

onClick={onManageAccreditation}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-emerald-600 rounded-lg hover:bg-emerald-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-emerald-500"

>

<ShieldCheckIcon className="w-5 h-5" />

مطالب اعتباربخشی

</button>

{(userRole === UserRole.Admin || userRole === UserRole.Supervisor) && (

<button

onClick={handleOpenResetModal}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-red-600 rounded-lg hover:bg-red-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-red-500"

>

<RefreshIcon className="w-5 h-5" />

ریست کردن بیمارستان

</button>

)}

<button

onClick={handleOpenAddModal}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"

>

<PlusIcon className="w-5 h-5" />

افزودن بخش جدید

</button>

</div>

</div>

{departments.length === 0 ? (

<div className="text-center py-16 bg-white dark:bg-slate-800 rounded-xl shadow">

<h2 className="text-xl font-medium text-slate-500">هیچ بخشی یافت نشد.</h2>

<p className="text-slate-400 mt-2">برای شروع، یک بخش جدید اضافه کنید.</p>

</div>

) : (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">

{departments.map((dep) => (

<div

key={dep.id}

className="relative group bg-white dark:bg-slate-800 rounded-xl shadow-lg hover:shadow-2xl hover:shadow-indigo-500/20 border-t-4 border-indigo-500 transition-all duration-300 hover:-translate-y-1"

>

<div

onClick={() => onSelectDepartment(dep.id)}

className="p-6 cursor-pointer"

>

<h2 className="text-xl font-bold text-slate-800 dark:text-slate-100 truncate">{dep.name}</h2>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-2">{dep.staff.length} نفر پرسنل</p>

</div>

<div className="absolute top-3 left-3 flex items-center gap-1 opacity-0 group-hover:opacity-100 transition-opacity">

<button

onClick={(e) => { e.stopPropagation(); handleOpenEditModal(dep); }}

className="p-2 text-slate-400 hover:text-indigo-500 bg-slate-100 dark:bg-slate-700 rounded-full"

aria-label="Edit Department"

>

<EditIcon className="w-5 h-5" />

</button>

<button

onClick={(e) => {

e.stopPropagation();

if(window.confirm(`آیا از حذف بخش "${dep.name}" مطمئن هستید؟`)) {

onDeleteDepartment(dep.id);

}

}}

className="p-2 text-slate-400 hover:text-red-500 bg-slate-100 dark:bg-slate-700 rounded-full"

aria-label="Delete Department"

>

<TrashIcon className="w-5 h-5" />

</button>

</div>

</div>

))}

</div>

)}

<Modal isOpen={isModalOpen} onClose={handleCloseModal} title={editingDepartment ? "ویرایش مشخصات بخش" : "افزودن بخش جدید"}>

<div className="space-y-4">

<input

type="text"

value={newDepartmentName}

onChange={(e) => setNewDepartmentName(e.target.value)}

placeholder="نام بخش (مثال: اورژانس)"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

value={newManagerName}

onChange={(e) => setNewManagerName(e.target.value)}

placeholder="نام مسئول بخش"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

inputMode="numeric"

value={newManagerNationalId}

onChange={(e) => setNewManagerNationalId(e.target.value.replace(/\D/g, ''))}

placeholder="کد ملی مسئول بخش"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="password"

value={newManagerPassword}

onChange={(e) => setNewManagerPassword(e.target.value)}

placeholder="رمز ورود مسئول بخش"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="number"

value={newStaffCount}

onChange={(e) => setNewStaffCount(e.target.value)}

placeholder="تعداد نیروی بخش"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="number"

value={newBedCount}

onChange={(e) => setNewBedCount(e.target.value)}

placeholder="تعداد تخت های بخش"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3">

<button

onClick={handleCloseModal}

className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500"

>

انصراف

</button>

<button

onClick={handleSaveDepartment}

className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700"

>

{editingDepartment ? 'ذخیره تغییرات' : 'افزودن'}

</button>

</div>

</div>

</Modal>

<Modal isOpen={isResetModalOpen} onClose={resetResetModal} title="ریست کردن اطلاعات بیمارستان">

{resetStep === 1 && (

<div className="text-center">

<p className="text-lg mb-6">آیا مایل به پاک کردن تمام بخش ها هستین؟ این عمل غیرقابل بازگشت است.</p>

<div className="flex justify-center gap-4">

<button

onClick={resetResetModal}

className="px-6 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500"

>

خیر

</button>

<button

onClick={() => setResetStep(2)}

className="px-6 py-2 font-semibold text-white bg-red-600 rounded-md hover:bg-red-700"

>

بله، ادامه

</button>

</div>

</div>

)}

{resetStep === 2 && (

<div className="space-y-4">

<p className="text-sm text-slate-500 dark:text-slate-400">برای تایید، لطفاً کد ملی و رمز عبور سوپروایزر آموزشی این بیمارستان را وارد کنید.</p>

<input

type="text"

inputMode="numeric"

value={supervisorIdInput}

onChange={(e) => { setSupervisorIdInput(e.target.value.replace(/\D/g, '')); setResetError(null); }}

placeholder="کد ملی سوپروایزر"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="password"

value={supervisorPasswordInput}

onChange={(e) => { setSupervisorPasswordInput(e.target.value); setResetError(null); }}

placeholder="رمز عبور سوپروایزر"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

{resetError && <p className="text-red-500 text-sm text-center">{resetError}</p>}

<div className="flex justify-end gap-3 pt-2">

<button

onClick={resetResetModal}

className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500"

>

انصراف

</button>

<button

onClick={handleConfirmReset}

className="px-4 py-2 font-semibold text-white bg-red-600 rounded-md hover:bg-red-700"

>

تایید و حذف تمام بخش‌ها

</button>

</div>

</div>

)}

</Modal>

</div>

);

};

export default DepartmentList;

DepartmentView.tsx

import React, { useState, useMemo, useEffect } from 'react';

import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from 'recharts';

import { Department, StaffMember, SkillCategory, UserRole, NewsBanner, MonthlyWorkLog } from '../types';

import Modal from './Modal';

import { PlusIcon } from './icons/PlusIcon';

import FileUploader from './FileUploader';

import { parseComprehensiveExcel } from '../services/excelParser';

import { AcademicCapIcon } from './icons/AcademicCapIcon';

import { ClipboardDocumentCheckIcon } from './icons/ClipboardDocumentCheckIcon';

import { ChecklistIcon } from './icons/ChecklistIcon';

import NewsCarousel from './NewsCarousel';

import { BookOpenIcon } from './icons/BookOpenIcon';

interface DepartmentViewProps {

department: Department;

onBack: () => void;

onAddStaff: (departmentId: string, name: string, title: string, nationalId: string, password?: string) => void;

onUpdateStaff: (departmentId: string, staffId: string, updatedData: Partial<Omit<StaffMember, 'id' | 'assessments'>>) => void;

onDeleteStaff: (departmentId: string, staffId: string) => void;

onSelectStaff: (staffId: string) => void;

onComprehensiveImport: (departmentId: string, data: { [staffName: string]: Map<string, SkillCategory[]> }) => void;

onManageChecklists: () => void;

onManageExams: () => void;

onManageTraining: () => void;

onManagePatientEducation: () => void;

onAddOrUpdateWorkLog: (departmentId: string, staffId: string, workLog: MonthlyWorkLog) => void;

userRole: UserRole;

newsBanners: NewsBanner[];

}

const PERSIAN\_MONTHS = [

"فروردین", "اردیبهشت", "خرداد",

"تیر", "مرداد", "شهریور",

"مهر", "آبان", "آذر",

"دی", "بهمن", "اسفند"

];

const CHART\_COLORS = ['#3b82f6', '#16a34a', '#f97316', '#dc2626', '#8b5cf6', '#db2777'];

const DepartmentView: React.FC<DepartmentViewProps> = ({

department,

onBack,

onAddStaff,

onUpdateStaff,

onDeleteStaff,

onSelectStaff,

onComprehensiveImport,

onManageChecklists,

onManageExams,

onManageTraining,

onManagePatientEducation,

onAddOrUpdateWorkLog,

userRole,

newsBanners,

}) => {

const [isModalOpen, setIsModalOpen] = useState(false);

const [editingStaff, setEditingStaff] = useState<StaffMember | null>(null);

// Form state for staff modal

const [staffName, setStaffName] = useState('');

const [staffTitle, setStaffTitle] = useState('');

const [staffNationalId, setStaffNationalId] = useState('');

const [staffPassword, setStaffPassword] = useState('');

const [isUploading, setIsUploading] = useState(false);

const [uploadError, setUploadError] = useState<string | null>(null);

// Work Log Modal State

const [isWorkLogModalOpen, setIsWorkLogModalOpen] = useState(false);

const [selectedStaffForWorkLog, setSelectedStaffForWorkLog] = useState<StaffMember | null>(null);

const [selectedLogMonth, setSelectedLogMonth] = useState<string>(PERSIAN\_MONTHS[0]);

const [overtimeHours, setOvertimeHours] = useState('');

const [requiredHours, setRequiredHours] = useState('');

const [quarterlyLeave, setQuarterlyLeave] = useState('');

const [annualLeave, setAnnualLeave] = useState('');

const resetForm = () => {

setStaffName('');

setStaffTitle('');

setStaffNationalId('');

setStaffPassword('');

setEditingStaff(null);

};

const handleOpenAddModal = () => {

resetForm();

setIsModalOpen(true);

};

const handleOpenEditModal = (staff: StaffMember) => {

setEditingStaff(staff);

setStaffName(staff.name);

setStaffTitle(staff.title);

setStaffNationalId(staff.nationalId || '');

setStaffPassword(staff.password || '');

setIsModalOpen(true);

};

const handleSaveStaff = () => {

if (!staffName.trim() || !staffTitle.trim()) {

alert("لطفا نام و تخصص شغلی را وارد کنید.");

return;

}

if (editingStaff) {

onUpdateStaff(department.id, editingStaff.id, {

name: staffName.trim(),

title: staffTitle.trim(),

nationalId: staffNationalId.trim(),

password: staffPassword.trim(),

});

} else {

onAddStaff(department.id, staffName.trim(), staffTitle.trim(), staffNationalId.trim(), staffPassword.trim());

}

setIsModalOpen(false);

resetForm();

};

const handleComprehensiveUpload = async (file: File) => {

setUploadError(null);

setIsUploading(true);

try {

const allStaffData = await parseComprehensiveExcel(file);

onComprehensiveImport(department.id, allStaffData);

} catch (err) {

setUploadError(err instanceof Error ? err.message : "خطای ناشناخته در پردازش فایل.");

} finally {

setIsUploading(false);

}

};

// --- Work Log Handlers ---

const resetWorkLogForm = () => {

setSelectedLogMonth(PERSIAN\_MONTHS[0]);

setOvertimeHours('');

setRequiredHours('');

setQuarterlyLeave('');

setAnnualLeave('');

};

const handleOpenWorkLogModal = (staff: StaffMember) => {

setSelectedStaffForWorkLog(staff);

resetWorkLogForm();

setIsWorkLogModalOpen(true);

};

const handleCloseWorkLogModal = () => {

setIsWorkLogModalOpen(false);

setSelectedStaffForWorkLog(null);

resetWorkLogForm();

};

useEffect(() => {

if (isWorkLogModalOpen && selectedStaffForWorkLog) {

const log = selectedStaffForWorkLog.workLogs?.find(l => l.month === selectedLogMonth);

if (log) {

setOvertimeHours(String(log.overtimeHours));

setRequiredHours(String(log.requiredHours));

setQuarterlyLeave(String(log.quarterlyLeaveRemaining));

setAnnualLeave(String(log.annualLeaveRemaining));

} else {

setOvertimeHours('');

setRequiredHours('');

setQuarterlyLeave('');

setAnnualLeave('');

}

}

}, [isWorkLogModalOpen, selectedStaffForWorkLog, selectedLogMonth]);

const handleSaveWorkLog = () => {

if (!selectedStaffForWorkLog) return;

const overtime = parseFloat(overtimeHours);

const required = parseFloat(requiredHours);

const quarterly = parseFloat(quarterlyLeave);

const annual = parseFloat(annualLeave);

if (isNaN(overtime) || isNaN(required) || isNaN(quarterly) || isNaN(annual)) {

alert('لطفاً تمام فیلدها را با مقادیر عددی معتبر پر کنید.');

return;

}

const workLog: MonthlyWorkLog = {

month: selectedLogMonth,

overtimeHours: overtime,

requiredHours: required,

quarterlyLeaveRemaining: quarterly,

annualLeaveRemaining: annual,

};

onAddOrUpdateWorkLog(department.id, selectedStaffForWorkLog.id, workLog);

handleCloseWorkLogModal();

alert('اطلاعات با موفقیت ذخیره شد.');

};

const chartInfo = useMemo(() => {

const allCategoryNames = new Set<string>();

department.staff.forEach(staff => {

(staff.assessments || []).forEach(assessment => {

(assessment.skillCategories || []).forEach(cat => allCategoryNames.add(cat.name));

});

});

const categoryNames = Array.from(allCategoryNames);

const monthlyData: { [month: string]: { [category: string]: number[] } } = {};

department.staff.forEach(staff => {

(staff.assessments || []).forEach(assessment => {

if (!monthlyData[assessment.month]) monthlyData[assessment.month] = {};

const assessmentHasItems = (assessment.skillCategories || []).some(cat => (cat.items || []).length > 0);

if (!assessmentHasItems) return; // Skip empty assessments

(assessment.skillCategories || []).forEach(cat => {

if (!monthlyData[assessment.month][cat.name]) monthlyData[assessment.month][cat.name] = [];

const items = cat.items || [];

const totalScore = items.reduce((sum, item) => sum + (item?.score || 0), 0);

const maxScore = items.length \* (assessment.maxScore ?? 4);

const percentage = maxScore > 0 ? (totalScore / maxScore) \* 100 : 0;

monthlyData[assessment.month][cat.name].push(percentage);

});

});

});

const avg = (arr: number[]) => arr.length > 0 ? arr.reduce((a, b) => a + b, 0) / arr.length : null;

const formattedData = PERSIAN\_MONTHS.map(month => {

const monthScores = monthlyData[month];

const monthAvgs: { [key:string]: any } = { name: month };

categoryNames.forEach(catName => {

const avgResult = avg(monthScores?.[catName] || []);

monthAvgs[catName] = avgResult !== null ? parseFloat(avgResult.toFixed(1)) : null;

});

return monthAvgs;

});

return { data: formattedData, categoryNames };

}, [department.staff]);

const staffProgressChartData = useMemo(() => {

const monthlyData: { [month: string]: { name: string; [staffName: string]: number | string | null } } = {};

PERSIAN\_MONTHS.forEach(month => {

monthlyData[month] = { name: month };

});

department.staff.forEach(staff => {

(staff.assessments || []).forEach(assessment => {

const categories = assessment.skillCategories || [];

const totalScore = categories.reduce((catSum, cat) =>

(cat?.items || []).reduce((itemSum, item) => itemSum + (item?.score || 0), 0) + catSum, 0);

const maxScore = categories.reduce((catSum, cat) =>

((cat?.items || []).length \* (assessment.maxScore ?? 4)) + catSum, 0);

const percentage = maxScore > 0 ? parseFloat(((totalScore / maxScore) \* 100).toFixed(1)) : null;

if (monthlyData[assessment.month]) {

monthlyData[assessment.month][staff.name] = percentage;

}

});

});

// Sort and return

return PERSIAN\_MONTHS

.map(month => monthlyData[month])

.filter(monthData => Object.keys(monthData).length > 1); // Only include months with at least one assessment

}, [department.staff]);

const sortedStaff = useMemo(() => {

if (!department.staff) {

return [];

}

// Sort staff alphabetically by name

return [...department.staff].sort((a, b) => a.name.localeCompare(b.name, 'fa'));

}, [department.staff]);

const renderStaffListView = () => (

<>

<div className="bg-white dark:bg-slate-800 rounded-xl shadow-lg p-6 mb-8 grid grid-cols-2 md:grid-cols-4 gap-4 text-center">

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">مسئول بخش</p>

<p className="text-lg font-semibold text-slate-800 dark:text-slate-100">{department.managerName}</p>

</div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">کد ملی مسئول</p>

<p className="text-lg font-semibold text-slate-800 dark:text-slate-100">{department.managerNationalId}</p>

</div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">تعداد نیرو</p>

<p className="text-lg font-semibold text-slate-800 dark:text-slate-100">{department.staffCount}</p>

</div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">تعداد تخت</p>

<p className="text-lg font-semibold text-slate-800 dark:text-slate-100">{department.bedCount}</p>

</div>

</div>

{chartInfo.data.length > 0 && (

<div className="bg-white dark:bg-slate-800 rounded-xl shadow-lg p-6 mb-8">

<h2 className="text-xl font-bold mb-4">روند میانگین امتیازات مهارت بخش</h2>

<div className="bg-slate-100 dark:bg-slate-900 p-4 rounded-lg border border-slate-200 dark:border-slate-700">

<ResponsiveContainer width="100%" height={300}>

<LineChart data={chartInfo.data} margin={{ top: 5, right: 20, left: -10, bottom: 5 }}>

<CartesianGrid strokeDasharray="5 5" stroke="rgba(100, 116, 139, 0.3)" />

<XAxis dataKey="name" tick={{ fill: 'currentColor', fontSize: 12 }} className="text-slate-500 dark:text-slate-400" />

<YAxis unit="%" domain={[0, 100]} tick={{ fill: 'currentColor', fontSize: 12 }} className="text-slate-500 dark:text-slate-400" />

<Tooltip

cursor={{ stroke: '#94a3b8', strokeWidth: 1, strokeDasharray: '5 5' }}

contentStyle={{

backgroundColor: 'rgba(15, 23, 42, 0.9)',

borderColor: '#334155',

borderRadius: '0.5rem',

}}

labelStyle={{ color: '#f1f5f9' }}

formatter={(value: number | null, name: string) => {

if (value === null || value === undefined) {

return ["ثبت نشده", name];

}

return [`${value}%`, name];

}}

/>

<Legend wrapperStyle={{ fontSize: '0.8rem' }} />

{chartInfo.categoryNames.map((catName, index) => {

return <Line key={catName} type="monotone" dataKey={catName} name={catName} stroke={CHART\_COLORS[index % CHART\_COLORS.length]} strokeWidth={2} activeDot={{ r: 8 }} />

})}

</LineChart>

</ResponsiveContainer>

</div>

</div>

)}

{staffProgressChartData.length > 0 && (

<div className="bg-white dark:bg-slate-800 rounded-xl shadow-lg p-6 mb-8">

<h2 className="text-xl font-bold mb-4">روند پیشرفت فردی پرسنل</h2>

<div className="bg-slate-100 dark:bg-slate-900 p-4 rounded-lg border border-slate-200 dark:border-slate-700">

<ResponsiveContainer width="100%" height={300}>

<LineChart data={staffProgressChartData} margin={{ top: 5, right: 20, left: -10, bottom: 5 }}>

<CartesianGrid strokeDasharray="5 5" stroke="rgba(100, 116, 139, 0.3)" />

<XAxis dataKey="name" tick={{ fill: 'currentColor', fontSize: 12 }} className="text-slate-500 dark:text-slate-400" />

<YAxis unit="%" domain={[0, 100]} tick={{ fill: 'currentColor', fontSize: 12 }} className="text-slate-500 dark:text-slate-400" />

<Tooltip

cursor={{ stroke: '#94a3b8', strokeWidth: 1, strokeDasharray: '5 5' }}

contentStyle={{

backgroundColor: 'rgba(15, 23, 42, 0.9)',

borderColor: '#334155',

borderRadius: '0.5rem',

}}

labelStyle={{ color: '#f1f5f9' }}

formatter={(value: number | null, name: string) => {

if (value === null || value === undefined) {

return ["ثبت نشده", name];

}

return [`${value}%`, name];

}}

/>

<Legend wrapperStyle={{ fontSize: '0.8rem' }} />

{department.staff.map((staff, index) => {

return <Line key={staff.id} type="monotone" dataKey={staff.name} name={staff.name} stroke={CHART\_COLORS[index % CHART\_COLORS.length]} strokeWidth={2} connectNulls activeDot={{ r: 8 }} />

})}

</LineChart>

</ResponsiveContainer>

</div>

</div>

)}

<div className="bg-white dark:bg-slate-800 rounded-xl shadow-lg p-6 mb-8">

<h2 className="text-xl font-bold mb-4">بارگذاری اکسل عملکردی جامع</h2>

<p className="text-sm text-slate-500 dark:text-slate-400 mb-4">

یک فایل اکسل با شیت‌های متعدد آپلود کنید. نام هر شیت باید نام یکی از پرسنل باشد. این فایل، ارزیابی‌های تمام پرسنل موجود در آن را به صورت یکجا ثبت یا به‌روزرسانی می‌کند. اگر پرسنلی در فایل اکسل وجود داشته باشد که در لیست زیر نیست، به طور خودکار اضافه خواهد شد.

</p>

{isUploading && <p className="text-center text-slate-500">در حال پردازش فایل...</p>}

{uploadError && <p className="text-center text-red-500 my-2">{uploadError}</p>}

<FileUploader

onFileUpload={handleComprehensiveUpload}

accept=".xlsx"

title="آپلود فایل اکسل جامع بخش"

/>

</div>

<div className="flex justify-between items-center mb-6">

<h2 className="text-2xl font-bold">لیست پرسنل</h2>

<button

onClick={handleOpenAddModal}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"

>

<PlusIcon className="w-5 h-5" />

افزودن پرسنل

</button>

</div>

<div className="bg-white dark:bg-slate-800 rounded-xl shadow-lg overflow-x-auto">

<table className="w-full text-sm text-right text-slate-500 dark:text-slate-400">

<thead className="text-xs text-slate-700 uppercase bg-slate-50 dark:bg-slate-700 dark:text-slate-300">

<tr>

<th scope="col" className="px-6 py-3">نام پرسنل</th>

<th scope="col" className="px-6 py-3">تخصص شغلی</th>

<th scope="col" className="px-6 py-3 text-center">اقدامات</th>

</tr>

</thead>

<tbody>

{sortedStaff.map((staff) => (

<tr key={staff.id} className="border-b dark:border-slate-700 odd:bg-white odd:dark:bg-slate-800 even:bg-slate-50 even:dark:bg-slate-700/50">

<td scope="row" className="px-6 py-4 font-medium text-slate-900 whitespace-nowrap dark:text-white cursor-pointer" onClick={() => onSelectStaff(staff.id)}>

{staff.name}

</td>

<td className="px-6 py-4 cursor-pointer" onClick={() => onSelectStaff(staff.id)}>

{staff.title}

</td>

<td className="px-6 py-4 text-center">

<div className="flex items-center justify-center gap-2">

<button

onClick={() => handleOpenWorkLogModal(staff)}

className="px-3 py-1 text-xs font-semibold text-white bg-teal-500 rounded-md shadow-sm hover:bg-teal-600 transition-transform transform hover:scale-105"

title="ثبت کارکرد ماهانه"

>

کارکرد

</button>

<button

onClick={() => handleOpenEditModal(staff)}

className="px-3 py-1 text-xs font-semibold text-white bg-indigo-600 rounded-md shadow-sm hover:bg-indigo-700 transition-transform transform hover:scale-105"

title="ویرایش پرسنل"

>

ویرایش

</button>

<button

onClick={(e) => {

e.stopPropagation();

if(window.confirm(`آیا از حذف "${staff.name}" مطمئن هستید؟`)) {

onDeleteStaff(department.id, staff.id);

}

}}

className="px-3 py-1 text-xs font-semibold text-white bg-red-500 rounded-md shadow-sm hover:bg-red-600 transition-transform transform hover:scale-105"

title="حذف پرسنل"

>

حذف

</button>

</div>

</td>

</tr>

))}

{sortedStaff.length === 0 && (

<tr>

<td colSpan={3} className="text-center py-8 text-slate-400">هیچ پرسنلی در این بخش ثبت نشده است.</td>

</tr>

)}

</tbody>

</table>

</div>

<Modal isOpen={isModalOpen} onClose={() => setIsModalOpen(false)} title={editingStaff ? 'ویرایش اطلاعات پرسنل' : 'افزودن پرسنل جدید'}>

<div className="space-y-4">

<input

type="text"

value={staffName}

onChange={(e) => setStaffName(e.target.value)}

placeholder="نام و نام خانوادگی"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

value={staffTitle}

onChange={(e) => setStaffTitle(e.target.value)}

placeholder="تخصص شغلی (مثال: پرستار)"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

inputMode="numeric"

value={staffNationalId}

onChange={(e) => setStaffNationalId(e.target.value.replace(/\D/g, ''))}

placeholder="کد ملی (برای ورود پرسنل)"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="password"

value={staffPassword}

onChange={(e) => setStaffPassword(e.target.value)}

placeholder="رمز عبور (برای ورود پرسنل)"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3">

<button onClick={() => setIsModalOpen(false)} className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500">انصراف</button>

<button onClick={handleSaveStaff} className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700">{editingStaff ? 'ذخیره تغییرات' : 'افزودن'}</button>

</div>

</div>

</Modal>

<Modal isOpen={isWorkLogModalOpen} onClose={handleCloseWorkLogModal} title={`ثبت کارکرد ماهانه برای ${selectedStaffForWorkLog?.name}`}>

<div className="space-y-4">

<div>

<label htmlFor="log-month" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

انتخاب ماه

</label>

<select

id="log-month"

value={selectedLogMonth}

onChange={(e) => setSelectedLogMonth(e.target.value)}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

>

{PERSIAN\_MONTHS.map(month => <option key={month} value={month}>{month}</option>)}

</select>

</div>

<div>

<label htmlFor="required-hours" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

موظفی پرسنل در این ماه

</label>

<input

id="required-hours"

type="number"

value={requiredHours}

onChange={(e) => setRequiredHours(e.target.value)}

placeholder="مثال: 180"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

<div>

<label htmlFor="overtime-hours" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

ساعت اضافه کار

</label>

<input

id="overtime-hours"

type="number"

value={overtimeHours}

onChange={(e) => setOvertimeHours(e.target.value)}

placeholder="مثال: 25.5"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

<div>

<label htmlFor="quarterly-leave" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

میزان مرخصی مانده در فصل

</label>

<input

id="quarterly-leave"

type="number"

value={quarterlyLeave}

onChange={(e) => setQuarterlyLeave(e.target.value)}

placeholder="مثال: 5"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

<div>

<label htmlFor="annual-leave" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

میزان مرخصی مانده در سال

</label>

<input

id="annual-leave"

type="number"

value={annualLeave}

onChange={(e) => setAnnualLeave(e.target.value)}

placeholder="مثال: 18"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

<div className="flex justify-end gap-3 pt-2">

<button onClick={handleCloseWorkLogModal} className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500">

انصراف

</button>

<button onClick={handleSaveWorkLog} className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700">

ذخیره

</button>

</div>

</div>

</Modal>

</>

);

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">{department.name}</h1>

{(userRole === UserRole.Admin || userRole === UserRole.Supervisor || userRole === UserRole.Manager) && (

<div className="flex items-center gap-2">

<button

onClick={onManagePatientEducation}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-orange-600 rounded-lg hover:bg-orange-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-orange-500"

>

<BookOpenIcon className="w-5 h-5" />

آموزش به بیمار

</button>

<button

onClick={onManageTraining}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-blue-600 rounded-lg hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500"

>

<AcademicCapIcon className="w-5 h-5" />

آموزش به پرسنل

</button>

<button

onClick={onManageExams}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-purple-600 rounded-lg hover:bg-purple-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-purple-500"

>

<ClipboardDocumentCheckIcon className="w-5 h-5" />

مدیریت آزمون‌ها

</button>

<button

onClick={onManageChecklists}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-teal-600 rounded-lg hover:bg-teal-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-teal-500"

>

<ChecklistIcon className="w-5 h-5" />

مدیریت قالب‌های چک‌لیست

</button>

</div>

)}

</div>

{newsBanners && newsBanners.length > 0 && (

<div className="mb-8 max-w-5xl mx-auto">

<NewsCarousel banners={newsBanners} />

</div>

)}

{renderStaffListView()}

</div>

);

};

export default DepartmentView;

ExamBuilder.tsx

import React, { useState } from 'react';

import { ExamTemplate, Question, QuestionType } from '../types';

import { PlusIcon } from './icons/PlusIcon';

import { TrashIcon } from './icons/TrashIcon';

import { SaveIcon } from './icons/SaveIcon';

interface ExamBuilderProps {

template: ExamTemplate;

onSave: (newTemplate: ExamTemplate) => void;

onCancel: () => void;

}

const ExamBuilder: React.FC<ExamBuilderProps> = ({ template, onSave, onCancel }) => {

const [name, setName] = useState(template.name || '');

const [questions, setQuestions] = useState<Question[]>(

() => JSON.parse(JSON.stringify(template.questions || [])) // Deep copy

);

const handleAddQuestion = (type: QuestionType) => {

const newQuestion: Question = {

id: Date.now().toString(),

text: '',

type: type,

correctAnswer: '',

...(type === QuestionType.MultipleChoice && { options: ['', '', '', ''] })

};

setQuestions([...questions, newQuestion]);

};

const handleQuestionChange = (qId: string, field: keyof Question, value: any) => {

setQuestions(

questions.map(q => (q.id === qId ? { ...q, [field]: value } : q))

);

};

const handleOptionChange = (qId: string, optionIndex: number, value: string) => {

setQuestions(questions.map(q => {

if (q.id === qId && q.options) {

const newOptions = [...q.options];

newOptions[optionIndex] = value;

// If this option was the correct answer, update the correct answer value as well

if (q.correctAnswer === q.options[optionIndex]) {

return { ...q, options: newOptions, correctAnswer: value };

}

return { ...q, options: newOptions };

}

return q;

}));

};

const handleDeleteQuestion = (qId: string) => {

if (window.confirm('آیا از حذف این سوال مطمئن هستید؟')) {

setQuestions(questions.filter(q => q.id !== qId));

}

};

const handleSaveChanges = () => {

if (!name.trim()) {

alert('نام آزمون نمی‌تواند خالی باشد.');

return;

}

if (questions.length === 0) {

alert('آزمون باید حداقل یک سوال داشته باشد.');

return;

}

for (const q of questions) {

if (!q.text.trim()) {

alert('متن سوال نمی‌تواند خالی باشد.');

return;

}

if (q.type === QuestionType.MultipleChoice) {

if (q.options?.some(opt => !opt.trim())) {

alert('متن گزینه‌ها نمی‌تواند خالی باشد.');

return;

}

if (!q.correctAnswer.trim()) {

alert('باید یک گزینه صحیح برای سوالات چهارگزینه‌ای انتخاب کنید.');

return;

}

} else {

if (!q.correctAnswer.trim()) {

alert('پاسخ نمونه برای سوال تشریحی نمی‌تواند خالی باشد.');

return;

}

}

}

onSave({ ...template, name, questions });

alert('آزمون با موفقیت ذخیره شد.');

}

const renderQuestionForm = (q: Question) => {

return (

<div key={q.id} className="border border-slate-200 dark:border-slate-700 rounded-lg p-4">

<div className="flex justify-between items-center mb-4">

<select

value={q.type}

onChange={(e) => handleQuestionChange(q.id, 'type', e.target.value)}

className="p-1 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-indigo-500"

>

<option value={QuestionType.MultipleChoice}>چهار گزینه‌ای</option>

<option value={QuestionType.Descriptive}>تشریحی</option>

</select>

<button

onClick={() => handleDeleteQuestion(q.id)}

className="p-2 text-slate-400 hover:text-red-500 rounded-full"

aria-label="Delete Question"

>

<TrashIcon className="w-5 h-5" />

</button>

</div>

<textarea

value={q.text}

onChange={e => handleQuestionChange(q.id, 'text', e.target.value)}

placeholder="متن سوال را اینجا بنویسید..."

rows={3}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-indigo-500 mb-3"

/>

{q.type === QuestionType.MultipleChoice ? (

<div className="space-y-2">

<p className="text-sm text-slate-500">گزینه‌ها را وارد کرده و پاسخ صحیح را انتخاب کنید:</p>

{q.options?.map((opt, index) => (

<div key={index} className="flex items-center gap-2">

<input

type="radio"

name={`correct-answer-${q.id}`}

checked={q.correctAnswer === opt}

onChange={() => handleQuestionChange(q.id, 'correctAnswer', opt)}

className="w-4 h-4 text-indigo-600 focus:ring-indigo-500"

/>

<input

type="text"

value={opt}

onChange={e => handleOptionChange(q.id, index, e.target.value)}

placeholder={`گزینه ${index + 1}`}

className="flex-grow px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-indigo-500"

/>

</div>

))}

</div>

) : (

<div>

<p className="text-sm text-slate-500 mb-1">پاسخ نمونه یا کلید تصحیح را وارد کنید:</p>

<textarea

value={q.correctAnswer}

onChange={e => handleQuestionChange(q.id, 'correctAnswer', e.target.value)}

placeholder="پاسخ تشریحی..."

rows={4}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-indigo-500"

/>

</div>

)}

</div>

);

};

return (

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<div className="flex justify-between items-center mb-6 pb-4 border-b border-slate-200 dark:border-slate-700">

<div>

<h2 className="text-2xl font-bold">ساخت / ویرایش آزمون</h2>

<p className="text-slate-500 dark:text-slate-400">یک آزمون برای ارزیابی دانش تئوری پرسنل ایجاد کنید.</p>

</div>

<div className="flex gap-2">

<button

onClick={onCancel}

className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500"

>

انصراف

</button>

<button

onClick={handleSaveChanges}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-green-600 rounded-lg hover:bg-green-700"

>

<SaveIcon className="w-5 h-5"/>

ذخیره آزمون

</button>

</div>

</div>

<div className="mb-6">

<label htmlFor="exam-name" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

نام آزمون

</label>

<input

id="exam-name"

type="text"

value={name}

onChange={e => setName(e.target.value)}

placeholder="مثال: آزمون ایمنی بیمار"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

</div>

<div className="space-y-6">

{questions.map(q => renderQuestionForm(q))}

</div>

<div className="mt-6 border-t border-slate-200 dark:border-slate-700 pt-6 flex items-center gap-4">

<p className="font-semibold">افزودن سوال جدید:</p>

<button

onClick={() => handleAddQuestion(QuestionType.MultipleChoice)}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700"

>

<PlusIcon className="w-5 h-5" />

چهار گزینه‌ای

</button>

<button

onClick={() => handleAddQuestion(QuestionType.Descriptive)}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-teal-600 rounded-lg hover:bg-teal-700"

>

<PlusIcon className="w-5 h-5" />

تشریحی

</button>

</div>

</div>

);

};

export default ExamBuilder;

ExamManager.tsx

import React, { useState } from 'react';

import { ExamTemplate } from '../types';

import ExamBuilder from './ExamBuilder';

import { PlusIcon } from './icons/PlusIcon';

import { EditIcon } from './icons/EditIcon';

import { TrashIcon } from './icons/TrashIcon';

interface ExamManagerProps {

templates: ExamTemplate[];

onAddOrUpdate: (template: ExamTemplate) => void;

onDelete: (templateId: string) => void;

onBack: () => void;

}

const ExamManager: React.FC<ExamManagerProps> = ({ templates, onAddOrUpdate, onDelete, onBack }) => {

const [editingTemplate, setEditingTemplate] = useState<ExamTemplate | null>(null);

const handleAddNew = () => {

setEditingTemplate({

id: Date.now().toString(),

name: '',

questions: []

});

};

const handleSave = (template: ExamTemplate) => {

onAddOrUpdate(template);

setEditingTemplate(null);

};

const handleDelete = (templateId: string, templateName: string) => {

if(window.confirm(`آیا از حذف آزمون "${templateName}" مطمئن هستید؟ این عمل قابل بازگشت نیست.`)){

onDelete(templateId);

}

}

if (editingTemplate) {

return (

<div className="p-4 sm:p-6 lg:p-8">

<ExamBuilder

template={editingTemplate}

onSave={handleSave}

onCancel={() => setEditingTemplate(null)}

/>

</div>

)

}

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">مدیریت آزمون‌ها</h1>

<button

onClick={handleAddNew}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-purple-600 rounded-lg hover:bg-purple-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-purple-500"

>

<PlusIcon className="w-5 h-5" />

افزودن آزمون جدید

</button>

</div>

{templates.length === 0 ? (

<div className="text-center py-16 bg-white dark:bg-slate-800 rounded-lg shadow">

<h2 className="text-xl font-medium text-slate-500">هیچ آزمونی یافت نشد.</h2>

<p className="text-slate-400 mt-2">برای شروع، یک آزمون جدید اضافه کنید.</p>

</div>

) : (

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md overflow-x-auto">

<table className="w-full text-sm text-right text-slate-500 dark:text-slate-400">

<thead className="text-xs text-slate-700 uppercase bg-slate-50 dark:bg-slate-700 dark:text-slate-300">

<tr>

<th scope="col" className="px-6 py-3">نام آزمون</th>

<th scope="col" className="px-6 py-3 text-center">تعداد سوالات</th>

<th scope="col" className="px-6 py-3 text-center">

<span className="sr-only">اقدامات</span>

</th>

</tr>

</thead>

<tbody>

{templates.map((template) => (

<tr key={template.id} className="group border-b dark:border-slate-700 odd:bg-white odd:dark:bg-slate-800 even:bg-slate-50 even:dark:bg-slate-700/50">

<td scope="row" className="px-6 py-4 font-medium text-slate-900 whitespace-nowrap dark:text-white">

{template.name}

</td>

<td className="px-6 py-4 text-center">

{template.questions.length}

</td>

<td className="px-6 py-4 text-center">

<div className="flex items-center justify-center opacity-0 group-hover:opacity-100 transition-opacity">

<button

onClick={() => setEditingTemplate(template)}

className="p-2 text-slate-400 hover:text-indigo-500 rounded-full"

aria-label="Edit Template"

>

<EditIcon className="w-5 h-5" />

</button>

<button

onClick={() => handleDelete(template.id, template.name)}

className="p-2 text-slate-400 hover:text-red-500 rounded-full"

aria-label="Delete Template"

>

<TrashIcon className="w-5 h-5" />

</button>

</div>

</td>

</tr>

))}

</tbody>

</table>

</div>

)}

</div>

);

};

export default ExamManager;

FileUploader.tsx

import React, { useState, useCallback } from 'react';

import { UploadIcon } from './icons/UploadIcon';

interface FileUploaderProps {

onFileUpload: (file: File) => void;

accept: string;

title: string;

}

const FileUploader: React.FC<FileUploaderProps> = ({ onFileUpload, accept, title }) => {

const [isDragging, setIsDragging] = useState(false);

const handleDrag = useCallback((e: React.DragEvent<HTMLDivElement>) => {

e.preventDefault();

e.stopPropagation();

if (e.type === 'dragenter' || e.type === 'dragover') {

setIsDragging(true);

} else if (e.type === 'dragleave') {

setIsDragging(false);

}

}, []);

const handleDrop = useCallback((e: React.DragEvent<HTMLDivElement>) => {

e.preventDefault();

e.stopPropagation();

setIsDragging(false);

if (e.dataTransfer.files && e.dataTransfer.files[0]) {

onFileUpload(e.dataTransfer.files[0]);

}

}, [onFileUpload]);

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {

if (e.target.files && e.target.files[0]) {

onFileUpload(e.target.files[0]);

}

};

return (

<div

className={`border-2 border-dashed rounded-lg p-8 text-center cursor-pointer transition-colors duration-200

${isDragging ? 'border-sky-500 bg-sky-50 dark:bg-sky-900/20' : 'border-slate-300 dark:border-slate-600 hover:border-sky-400 dark:hover:border-sky-500'}`}

onDragEnter={handleDrag}

onDragLeave={handleDrag}

onDragOver={handleDrag}

onDrop={handleDrop}

onClick={() => document.getElementById('file-upload-input')?.click()}

>

<input

type="file"

id="file-upload-input"

className="hidden"

accept={accept}

onChange={handleChange}

/>

<div className="flex flex-col items-center justify-center space-y-2 text-slate-500 dark:text-slate-400">

<UploadIcon className="w-12 h-12" />

<p className="font-semibold">{title}</p>

<p className="text-sm">فایل را بکشید و رها کنید یا کلیک کنید</p>

</div>

</div>

);

};

export default FileUploader;

HospitalCommunicationView.tsx

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

import { Hospital, TrainingMaterial } from '../types';

import { BackIcon } from './icons/BackIcon';

import \* as db from '../services/db';

import { PaperClipIcon } from './icons/PaperClipIcon';

import PreviewModal from './PreviewModal';

import { ImageIcon } from './icons/ImageIcon';

import { VideoIcon } from './icons/VideoIcon';

import { AudioIcon } from './icons/AudioIcon';

import { PdfIcon } from './icons/PdfIcon';

import { DocumentIcon } from './icons/DocumentIcon';

const getIconForMimeType = (type: string): { icon: React.ReactNode, color: string } => {

if (type.startsWith('image/')) return { icon: <ImageIcon className="w-8 h-8" />, color: 'text-blue-500' };

if (type.startsWith('video/')) return { icon: <VideoIcon className="w-8 h-8" />, color: 'text-red-500' };

if (type.startsWith('audio/')) return { icon: <AudioIcon className="w-8 h-8" />, color: 'text-purple-500' };

if (type === 'application/pdf') return { icon: <PdfIcon className="w-8 h-8" />, color: 'text-orange-500' };

return { icon: <DocumentIcon className="w-8 h-8" />, color: 'text-slate-500' };

};

interface HospitalCommunicationViewProps {

hospital: Hospital;

onSendMessage: (content: { text?: string; file?: { id: string; name: string; type: string; } }) => void;

onBack: () => void;

}

const HospitalCommunicationView: React.FC<HospitalCommunicationViewProps> = ({ hospital, onSendMessage, onBack }) => {

const [newMessage, setNewMessage] = useState('');

const [previewMaterial, setPreviewMaterial] = useState<TrainingMaterial | null>(null);

const messagesEndRef = useRef<HTMLDivElement>(null);

const fileInputRef = useRef<HTMLInputElement>(null);

const scrollToBottom = () => {

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

};

useEffect(() => {

scrollToBottom();

}, [hospital.adminMessages]);

const handleSend = () => {

if (newMessage.trim()) {

onSendMessage({ text: newMessage.trim() });

setNewMessage('');

}

};

const handleFileSelect = async (e: React.ChangeEvent<HTMLInputElement>) => {

const file = e.target.files?.[0];

if (!file) return;

try {

const dataUrl = await new Promise<string>((resolve, reject) => {

const reader = new FileReader();

reader.onload = (event) => resolve(event.target?.result as string);

reader.onerror = (error) => reject(error);

reader.readAsDataURL(file);

});

const fileId = `chat-file-${Date.now()}`;

await db.addMaterial({ id: fileId, data: dataUrl });

onSendMessage({

file: {

id: fileId,

name: file.name,

type: file.type,

}

});

} catch (error) {

console.error("Error sending file:", error);

alert("خطا در ارسال فایل.");

} finally {

if (fileInputRef.current) {

fileInputRef.current.value = '';

}

}

};

const handleKeyDown = (e: React.KeyboardEvent<HTMLTextAreaElement>) => {

if (e.key === 'Enter' && !e.shiftKey) {

e.preventDefault();

handleSend();

}

};

return (

<>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md flex flex-col h-[calc(100vh-12rem)]">

<div className="flex justify-between items-center p-4 border-b border-slate-200 dark:border-slate-700">

<h2 className="text-xl font-bold">تماس با ادمین کل - {hospital.name}</h2>

</div>

<div className="flex-grow overflow-y-auto p-4 space-y-4 bg-slate-50 dark:bg-slate-900/50">

{(hospital.adminMessages || []).map(msg => (

<div key={msg.id} className={`flex items-end gap-2 ${msg.sender === 'hospital' ? 'justify-end' : 'justify-start'}`}>

<div className={`max-w-xl p-3 rounded-lg shadow ${msg.sender === 'hospital' ? 'bg-sky-500 text-white rounded-br-none' : 'bg-slate-200 dark:bg-slate-700 text-slate-800 dark:text-slate-100 rounded-bl-none'}`}>

{msg.text && <p className="whitespace-pre-wrap">{msg.text}</p>}

{msg.file && (

<button

onClick={() => setPreviewMaterial({ id: msg.file!.id, name: msg.file!.name, type: msg.file!.type })}

className="flex items-center gap-3 text-left"

>

<div className={`flex-shrink-0 ${msg.sender === 'hospital' ? 'text-white' : 'text-slate-600 dark:text-slate-300'}`}>

{getIconForMimeType(msg.file.type).icon}

</div>

<div>

<p className="font-semibold break-all">{msg.file.name}</p>

<p className="text-xs opacity-80">برای مشاهده کلیک کنید</p>

</div>

</button>

)}

<p className={`text-xs mt-1 text-right ${msg.sender === 'hospital' ? 'text-sky-100' : 'text-slate-400'}`}>

{new Date(msg.timestamp).toLocaleString('fa-IR', { hour: '2-digit', minute: '2-digit' })}

</p>

</div>

</div>

))}

<div ref={messagesEndRef} />

</div>

<div className="p-4 border-t border-slate-200 dark:border-slate-700 flex items-center gap-2">

<input type="file" ref={fileInputRef} onChange={handleFileSelect} className="hidden" />

<button

onClick={() => fileInputRef.current?.click()}

className="p-3 text-slate-500 bg-slate-100 dark:bg-slate-700 dark:text-slate-300 rounded-lg hover:bg-slate-200 dark:hover:bg-slate-600"

aria-label="Attach file"

>

<PaperClipIcon className="w-6 h-6"/>

</button>

<textarea

value={newMessage}

onChange={(e) => setNewMessage(e.target.value)}

onKeyDown={handleKeyDown}

placeholder="پیام خود را بنویسید..."

rows={2}

className="flex-grow px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<button

onClick={handleSend}

className="px-6 py-3 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 disabled:opacity-50"

disabled={!newMessage.trim()}

>

ارسال

</button>

</div>

</div>

<PreviewModal isOpen={!!previewMaterial} onClose={() => setPreviewMaterial(null)} material={previewMaterial!}/>

</>

);

};

export default HospitalCommunicationView;

HospitalList.tsx

import React, { useState } from 'react';

import { Hospital, UserRole } from '../types';

import Modal from './Modal';

import { PlusIcon } from './icons/PlusIcon';

import { TrashIcon } from './icons/TrashIcon';

import { HomeIcon } from './icons/HomeIcon';

import { EditIcon } from './icons/EditIcon';

import { ChatIcon } from './icons/ChatIcon';

interface HospitalListProps {

hospitals: Hospital[];

onAddHospital: (name: string, province: string, city: string, supervisorName: string, supervisorNationalId: string, supervisorPassword: string) => void;

onUpdateHospital: (id: string, updatedData: Partial<Omit<Hospital, 'id' | 'departments' | 'checklistTemplates' | 'examTemplates'>>) => void;

onDeleteHospital: (id: string) => void;

onSelectHospital: (id: string) => void;

onGoToWelcome: () => void;

userRole: UserRole;

onContactAdmin: () => void;

}

const HospitalList: React.FC<HospitalListProps> = ({

hospitals,

onAddHospital,

onUpdateHospital,

onDeleteHospital,

onSelectHospital,

onGoToWelcome,

userRole,

onContactAdmin

}) => {

const [isModalOpen, setIsModalOpen] = useState(false);

const [editingHospital, setEditingHospital] = useState<Hospital | null>(null);

const [hospitalName, setHospitalName] = useState('');

const [province, setProvince] = useState('');

const [city, setCity] = useState('');

const [supervisorName, setSupervisorName] = useState('');

const [supervisorNationalId, setSupervisorNationalId] = useState('');

const [supervisorPassword, setSupervisorPassword] = useState('');

const resetForm = () => {

setHospitalName('');

setProvince('');

setCity('');

setSupervisorName('');

setSupervisorNationalId('');

setSupervisorPassword('');

setEditingHospital(null);

}

const handleOpenAddModal = () => {

resetForm();

setIsModalOpen(true);

};

const handleOpenEditModal = (hospital: Hospital) => {

setEditingHospital(hospital);

setHospitalName(hospital.name);

setProvince(hospital.province);

setCity(hospital.city);

setSupervisorName(hospital.supervisorName || '');

setSupervisorNationalId(hospital.supervisorNationalId || '');

setSupervisorPassword(hospital.supervisorPassword || '');

setIsModalOpen(true);

};

const handleSave = () => {

if (hospitalName.trim() && province.trim() && city.trim() && supervisorName.trim() && supervisorNationalId.trim() && supervisorPassword.trim()) {

if (editingHospital) {

onUpdateHospital(editingHospital.id, {

name: hospitalName.trim(),

province: province.trim(),

city: city.trim(),

supervisorName: supervisorName.trim(),

supervisorNationalId: supervisorNationalId.trim(),

supervisorPassword: supervisorPassword.trim(),

});

} else {

onAddHospital(

hospitalName.trim(),

province.trim(),

city.trim(),

supervisorName.trim(),

supervisorNationalId.trim(),

supervisorPassword.trim()

);

}

resetForm();

setIsModalOpen(false);

} else {

alert("لطفاً تمام فیلدها را پر کنید.")

}

};

const handleCloseModal = () => {

resetForm();

setIsModalOpen(false);

}

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<div className="flex items-center gap-4">

<button

onClick={onGoToWelcome}

className="inline-flex items-center gap-2 px-4 py-2 text-sm font-semibold text-slate-700 bg-slate-200 dark:bg-slate-700 dark:text-slate-200 rounded-lg hover:bg-slate-300 dark:hover:bg-slate-600 border border-slate-300 dark:border-slate-600"

aria-label="Go to Welcome Screen"

>

<HomeIcon className="w-5 h-5" />

صفحه اصلی

</button>

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">انتخاب بیمارستان</h1>

</div>

<div className="flex items-center gap-2">

{userRole === UserRole.Admin && (

<button

onClick={onContactAdmin}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-violet-600 rounded-lg hover:bg-violet-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-violet-500"

>

<ChatIcon className="w-5 h-5" />

تماس با بیمارستان ها

</button>

)}

<button

onClick={handleOpenAddModal}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"

>

<PlusIcon className="w-5 h-5" />

افزودن بیمارستان

</button>

</div>

</div>

{hospitals.length === 0 ? (

<div className="text-center py-16 bg-white dark:bg-slate-800 rounded-xl shadow">

<h2 className="text-xl font-medium text-slate-500">هیچ بیمارستانی یافت نشد.</h2>

<p className="text-slate-400 mt-2">برای شروع، یک بیمارستان جدید اضافه کنید.</p>

</div>

) : (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">

{hospitals.map((h) => (

<div

key={h.id}

className="relative group bg-white dark:bg-slate-800 rounded-xl shadow-lg hover:shadow-2xl hover:shadow-indigo-500/20 border-t-4 border-indigo-500 transition-all duration-300 hover:-translate-y-1"

>

<div

onClick={() => onSelectHospital(h.id)}

className="p-6 cursor-pointer"

>

<h2 className="text-xl font-bold text-slate-800 dark:text-slate-100 truncate">{h.name}</h2>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-2">{h.province}, {h.city}</p>

{h.supervisorName && <p className="text-sm text-slate-500 dark:text-slate-400 mt-1">سوپروایزر: {h.supervisorName}</p>}

<p className="text-sm text-slate-500 dark:text-slate-400 mt-1">{h.departments.length} بخش</p>

</div>

<div className="absolute top-3 left-3 flex items-center gap-1 opacity-0 group-hover:opacity-100 transition-opacity">

<button

onClick={(e) => { e.stopPropagation(); handleOpenEditModal(h); }}

className="p-2 text-slate-400 hover:text-indigo-500 bg-slate-100 dark:bg-slate-700 rounded-full"

aria-label="Edit Hospital"

>

<EditIcon className="w-5 h-5" />

</button>

<button

onClick={(e) => {

e.stopPropagation();

if(window.confirm(`آیا از حذف بیمارستان "${h.name}" مطمئن هستید؟`)) {

onDeleteHospital(h.id);

}

}}

className="p-2 text-slate-400 hover:text-red-500 bg-slate-100 dark:bg-slate-700 rounded-full"

aria-label="Delete Hospital"

>

<TrashIcon className="w-5 h-5" />

</button>

</div>

</div>

))}

</div>

)}

<Modal isOpen={isModalOpen} onClose={handleCloseModal} title={editingHospital ? "ویرایش مشخصات بیمارستان" : "افزودن بیمارستان جدید"}>

<div className="space-y-4">

<input

type="text"

value={hospitalName}

onChange={(e) => setHospitalName(e.target.value)}

placeholder="نام بیمارستان"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

value={province}

onChange={(e) => setProvince(e.target.value)}

placeholder="استان"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

value={city}

onChange={(e) => setCity(e.target.value)}

placeholder="شهر"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

value={supervisorName}

onChange={(e) => setSupervisorName(e.target.value)}

placeholder="نام و نام خانوادگی سوپروایزر آموزشی"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

inputMode="numeric"

value={supervisorNationalId}

onChange={(e) => setSupervisorNationalId(e.target.value.replace(/\D/g, ''))}

placeholder="کد ملی سوپروایزر آموزشی"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="password"

value={supervisorPassword}

onChange={(e) => setSupervisorPassword(e.target.value)}

placeholder="رمز عبور سوپروایزر آموزشی"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3">

<button

onClick={handleCloseModal}

className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500"

>

انصراف

</button>

<button

onClick={handleSave}

className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700"

>

{editingHospital ? 'ذخیره تغییرات' : 'افزودن'}

</button>

</div>

</div>

</Modal>

</div>

);

};

export default HospitalList;

LoginModal.tsx

import React, { useState } from 'react';

interface LoginModalProps {

isOpen: boolean;

onClose: () => void;

onLogin: (nationalId: string, password: string) => void;

loginError: string | null;

}

const LoginModal: React.FC<LoginModalProps> = ({ isOpen, onClose, onLogin, loginError }) => {

const [nationalId, setNationalId] = useState('');

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

if (!isOpen) return null;

const handleLoginClick = () => {

onLogin(nationalId.trim(), password.trim());

};

const handleKeyDown = (e: React.KeyboardEvent<HTMLInputElement>) => {

if (e.key === 'Enter') {

handleLoginClick();

}

}

return (

<div className="fixed inset-0 bg-black bg-opacity-60 z-50 flex justify-center items-center p-4" onClick={onClose}>

<div

className="bg-white dark:bg-slate-800 rounded-lg shadow-xl w-full max-w-sm p-8 transform transition-all"

onClick={(e) => e.stopPropagation()}

>

<h3 className="text-2xl font-bold text-center text-slate-900 dark:text-slate-100 mb-6">ورود به سامانه</h3>

<div className="space-y-4">

<input

type="text"

inputMode="numeric"

value={nationalId}

onChange={(e) => setNationalId(e.target.value.replace(/\D/g, ''))}

onKeyDown={handleKeyDown}

placeholder="کد ملی"

className="w-full px-4 py-3 border border-slate-300 rounded-lg dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500 text-center"

/>

<input

type="password"

value={password}

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

onKeyDown={handleKeyDown}

placeholder="رمز عبور"

className="w-full px-4 py-3 border border-slate-300 rounded-lg dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500 text-center"

/>

</div>

{loginError && <p className="text-red-500 text-sm text-center mt-4">{loginError}</p>}

<div className="mt-6">

<button

onClick={handleLoginClick}

className="w-full px-4 py-3 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"

>

ورود

</button>

</div>

<div className="mt-4 text-center">

<button

onClick={onClose}

className="text-sm text-slate-500 hover:text-slate-700 dark:hover:text-slate-300"

>

انصراف

</button>

</div>

</div>

</div>

);

};

export default LoginModal;

Modal.tsx

import React from 'react';

interface ModalProps {

isOpen: boolean;

onClose: () => void;

title: string;

children: React.ReactNode;

maxWidthClass?: string;

}

const Modal: React.FC<ModalProps> = ({ isOpen, onClose, title, children, maxWidthClass = 'max-w-md' }) => {

if (!isOpen) return null;

return (

<div className="fixed inset-0 bg-black bg-opacity-50 z-50 flex justify-center items-center p-4" onClick={onClose}>

<div

className={`bg-white dark:bg-slate-800 rounded-lg shadow-xl w-full ${maxWidthClass} p-6 transform transition-all`}

onClick={(e) => e.stopPropagation()}

>

<div className="flex justify-between items-center border-b border-slate-200 dark:border-slate-700 pb-3">

<h3 className="text-xl font-semibold text-slate-900 dark:text-slate-100">{title}</h3>

<button

onClick={onClose}

className="text-slate-400 hover:text-slate-600 dark:hover:text-slate-200"

>

<svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor">

<path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M6 18L18 6M6 6l12 12" />

</svg>

</button>

</div>

<div className="mt-4">{children}</div>

</div>

</div>

);

};

export default Modal;

NewsBannerManager.tsx

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

import { NewsBanner } from '../types';

import { TrashIcon } from './icons/TrashIcon';

import { EditIcon } from './icons/EditIcon';

import FileUploader from './FileUploader';

import Modal from './Modal';

import \* as db from '../services/db';

interface NewsBannerManagerProps {

banners: NewsBanner[];

onAddBanner: (banner: Omit<NewsBanner, 'id' | 'imageId'>, imageData: string) => void;

onUpdateBanner: (bannerId: string, title: string, description: string) => void;

onDeleteBanner: (bannerId: string) => void;

onBack: () => void;

}

const NewsBannerManager: React.FC<NewsBannerManagerProps> = ({ banners, onAddBanner, onUpdateBanner, onDeleteBanner, onBack }) => {

const [isUploading, setIsUploading] = useState(false);

const [uploadError, setUploadError] = useState<string | null>(null);

const [modalOpen, setModalOpen] = useState(false);

const [editingBanner, setEditingBanner] = useState<NewsBanner | null>(null);

const [pendingFile, setPendingFile] = useState<{file: File, dataUrl: string} | null>(null);

const [bannerTitle, setBannerTitle] = useState('');

const [bannerDescription, setBannerDescription] = useState('');

const [bannerImages, setBannerImages] = useState<{[key: string]: string}>({});

useEffect(() => {

const fetchImages = async () => {

const images: {[key: string]: string} = {};

for (const banner of banners) {

const data = await db.getMaterialData(banner.imageId);

if (data) {

images[banner.imageId] = data;

}

}

setBannerImages(images);

};

fetchImages();

}, [banners]);

const handleFileUpload = (file: File) => {

if (!file.type.startsWith('image/')) {

alert('لطفا فقط فایل تصویری آپلود کنید.');

return;

}

setIsUploading(true);

setUploadError(null);

const reader = new FileReader();

reader.onload = (e) => {

try {

const dataUrl = e.target?.result as string;

if (!dataUrl) throw new Error("Failed to read file.");

setPendingFile({ file, dataUrl });

setEditingBanner(null);

setBannerTitle('');

setBannerDescription('');

setModalOpen(true);

} catch (err) {

setUploadError(err instanceof Error ? err.message : "خطای ناشناخته در پردازش فایل.");

} finally {

setIsUploading(false);

}

};

reader.onerror = () => {

setUploadError("خطا در خواندن فایل.");

setIsUploading(false);

};

reader.readAsDataURL(file);

};

const handleDeleteClick = (bannerId: string, bannerTitle: string) => {

if (window.confirm(`آیا از حذف بنر خبری "${bannerTitle}" مطمئن هستید؟`)) {

onDeleteBanner(bannerId);

}

};

const handleSave = () => {

if (!bannerTitle.trim()) {

alert("عنوان بنر نمی‌تواند خالی باشد.");

return;

}

if (pendingFile) { // Adding new

onAddBanner({ title: bannerTitle, description: bannerDescription }, pendingFile.dataUrl);

} else if (editingBanner) { // Editing existing

onUpdateBanner(editingBanner.id, bannerTitle, bannerDescription);

}

handleCloseModal();

};

const handleOpenEditModal = (banner: NewsBanner) => {

setPendingFile(null);

setEditingBanner(banner);

setBannerTitle(banner.title);

setBannerDescription(banner.description);

setModalOpen(true);

};

const handleCloseModal = () => {

setModalOpen(false);

setPendingFile(null);

setEditingBanner(null);

setBannerTitle('');

setBannerDescription('');

};

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">مدیریت بنرهای خبری</h1>

</div>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-xl font-bold mb-2">بارگذاری بنر جدید</h2>

<p className="text-sm text-slate-500 dark:text-slate-400 mb-4">

برای بهترین نتیجه، از تصاویر با نسبت ابعاد 16:9 استفاده کنید (مانند 1280x720 پیکسل).

</p>

{isUploading && <p className="text-center text-indigo-500">در حال پردازش فایل...</p>}

{uploadError && <p className="text-center text-red-500 my-2">{uploadError}</p>}

<FileUploader onFileUpload={handleFileUpload} accept="image/\*" title="آپلود تصویر بنر" />

<div className="mt-8">

<h3 className="text-xl font-bold mb-4 border-t pt-6 border-slate-200 dark:border-slate-700">بنرهای موجود</h3>

{banners.length === 0 ? (

<p className="text-center py-8 text-slate-400">هیچ بنری بارگذاری نشده است.</p>

) : (

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

{banners.map(banner => (

<div key={banner.id} className="group relative bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm overflow-hidden">

<img src={bannerImages[banner.imageId] || ''} alt={banner.title} className="w-full aspect-video object-cover bg-slate-200 dark:bg-slate-600" />

<div className="p-4">

<h4 className="font-bold text-slate-800 dark:text-slate-100 truncate">{banner.title}</h4>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-1 h-10 overflow-hidden text-ellipsis">

{banner.description || 'بدون توضیح'}

</p>

</div>

<div className="absolute top-2 left-2 flex items-center gap-2 opacity-0 group-hover:opacity-100 transition-opacity">

<button onClick={() => handleOpenEditModal(banner)} className="p-2 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-600 dark:text-slate-200 hover:text-indigo-500 dark:hover:text-indigo-400">

<EditIcon className="w-5 h-5"/>

</button>

<button onClick={() => handleDeleteClick(banner.id, banner.title)} className="p-2 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-600 dark:text-slate-200 hover:text-red-500 dark:hover:text-red-400">

<TrashIcon className="w-5 h-5"/>

</button>

</div>

</div>

))}

</div>

)}

</div>

</div>

<Modal isOpen={modalOpen} onClose={handleCloseModal} title={editingBanner ? "ویرایش بنر خبری" : "افزودن بنر خبری"}>

<div className="space-y-4">

<input

type="text"

value={bannerTitle}

onChange={e => setBannerTitle(e.target.value)}

placeholder="عنوان بنر"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<textarea

value={bannerDescription}

onChange={(e) => setBannerDescription(e.target.value)}

placeholder="توضیحات کوتاه (اختیاری)"

rows={3}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3">

<button onClick={handleCloseModal} className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500">

انصراف

</button>

<button onClick={handleSave} className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700">

ذخیره

</button>

</div>

</div>

</Modal>

</div>

);

};

export default NewsBannerManager;

NewsCarousel.tsx

import React, { useState, useEffect, useCallback } from 'react';

import { NewsBanner } from '../types';

import \* as db from '../services/db';

interface NewsCarouselProps {

banners: NewsBanner[];

}

const NewsCarousel: React.FC<NewsCarouselProps> = ({ banners }) => {

const [currentIndex, setCurrentIndex] = useState(0);

const [images, setImages] = useState<{ [key: string]: string }>({});

const [isLoading, setIsLoading] = useState(true);

useEffect(() => {

const fetchImages = async () => {

setIsLoading(true);

const imageMap: { [key: string]: string } = {};

for (const banner of banners) {

try {

const data = await db.getMaterialData(banner.imageId);

if (data) {

imageMap[banner.imageId] = data;

}

} catch (error) {

console.error(`Failed to load image for banner ${banner.id}`, error);

}

}

setImages(imageMap);

setIsLoading(false);

};

if (banners.length > 0) {

fetchImages();

} else {

setIsLoading(false);

}

}, [banners]);

const goToPrevious = useCallback(() => {

const isFirstSlide = currentIndex === 0;

const newIndex = isFirstSlide ? banners.length - 1 : currentIndex - 1;

setCurrentIndex(newIndex);

}, [currentIndex, banners.length]);

const goToNext = useCallback(() => {

const isLastSlide = currentIndex === banners.length - 1;

const newIndex = isLastSlide ? 0 : currentIndex + 1;

setCurrentIndex(newIndex);

}, [currentIndex, banners.length]);

const goToSlide = (slideIndex: number) => {

setCurrentIndex(slideIndex);

};

useEffect(() => {

if (banners.length > 1) {

const timer = setTimeout(goToNext, 7000);

return () => clearTimeout(timer);

}

}, [currentIndex, banners.length, goToNext]);

if (isLoading) {

return (

<div className="w-full aspect-video bg-slate-200 dark:bg-slate-700 rounded-2xl animate-pulse flex items-center justify-center">

<p className="text-slate-500">در حال بارگذاری بنرها...</p>

</div>

);

}

if (banners.length === 0) {

return null;

}

const currentBanner = banners[currentIndex];

const currentImage = images[currentBanner.imageId];

return (

<div className="rounded-2xl overflow-hidden shadow-lg bg-white dark:bg-slate-800">

{/\* Image Section \*/}

<div className="relative w-full aspect-video group bg-slate-100 dark:bg-slate-700 flex items-center justify-center">

{/\* Image \*/}

{currentImage ? (

<img

key={currentIndex}

src={currentImage}

alt={currentBanner.title}

className="w-full h-full object-cover"

/>

) : (

<p className="text-slate-500">تصویر در دسترس نیست</p>

)}

{/\* Navigation Buttons \*/}

{banners.length > 1 && (

<>

<button

onClick={goToPrevious}

className="absolute top-1/2 -translate-y-1/2 left-4 bg-black/30 text-white p-2 rounded-full opacity-0 group-hover:opacity-100 transition-all duration-300 hover:bg-black/50 hover:scale-110 focus:outline-none focus:ring-2 focus:ring-white/50"

aria-label="Previous Slide"

>

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={2.5} stroke="currentColor" className="w-6 h-6">

<path strokeLinecap="round" strokeLinejoin="round" d="M15.75 19.5 8.25 12l7.5-7.5" />

</svg>

</button>

<button

onClick={goToNext}

className="absolute top-1/2 -translate-y-1/2 right-4 bg-black/30 text-white p-2 rounded-full opacity-0 group-hover:opacity-100 transition-all duration-300 hover:bg-black/50 hover:scale-110 focus:outline-none focus:ring-2 focus:ring-white/50"

aria-label="Next Slide"

>

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={2.5} stroke="currentColor" className="w-6 h-6">

<path strokeLinecap="round" strokeLinejoin="round" d="m8.25 4.5 7.5 7.5-7.5 7.5" />

</svg>

</button>

</>

)}

{/\* Dot Indicators \*/}

{banners.length > 1 && (

<div className="absolute bottom-4 right-4 flex items-center gap-2">

{banners.map((\_, slideIndex) => (

<button

key={slideIndex}

onClick={() => goToSlide(slideIndex)}

className={`w-2.5 h-2.5 rounded-full transition-all duration-300 transform ${currentIndex === slideIndex ? 'bg-cyan-400 scale-125' : 'bg-white/50 hover:bg-white/75'}`}

aria-label={`Go to slide ${slideIndex + 1}`}

></button>

))}

</div>

)}

</div>

{/\* Text Content Section \*/}

<div className="p-4 bg-slate-50 dark:bg-slate-800/50">

<h3 className="text-xl font-bold text-slate-900 dark:text-slate-100">{currentBanner.title}</h3>

{currentBanner.description && <p className="mt-1 text-sm text-slate-600 dark:text-slate-400">{currentBanner.description}</p>}

</div>

</div>

);

};

export default NewsCarousel;

PatientEducationManager.tsx

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

import { Department, Patient, TrainingMaterial } from '../types';

import { TrashIcon } from './icons/TrashIcon';

import { PlusIcon } from './icons/PlusIcon';

import FileUploader from './FileUploader';

import PreviewModal from './PreviewModal';

import Modal from './Modal';

import { ImageIcon } from './icons/ImageIcon';

import { VideoIcon } from './icons/VideoIcon';

import { AudioIcon } from './icons/AudioIcon';

import { PdfIcon } from './icons/PdfIcon';

import { DocumentIcon } from './icons/DocumentIcon';

import \* as db from '../services/db';

import { PaperClipIcon } from './icons/PaperClipIcon';

import { EditIcon } from './icons/EditIcon';

interface PatientEducationManagerProps {

department: Department;

onAddMaterial: (material: TrainingMaterial) => void;

onDeleteMaterial: (materialId: string) => void;

onUpdateMaterialDescription: (materialId: string, description: string) => void;

onAddPatient: (name: string, nationalId: string, password?: string) => void;

onDeletePatient: (patientId: string) => void;

onSendMessage: (patientId: string, content: { text?: string; file?: { id: string; name: string; type: string; } }, sender: 'patient' | 'manager') => void;

onBack: () => void;

}

const getIconForMimeType = (type: string): { icon: React.ReactNode, color: string } => {

if (type.startsWith('image/')) return { icon: <ImageIcon className="w-8 h-8" />, color: 'text-blue-500' };

if (type.startsWith('video/')) return { icon: <VideoIcon className="w-8 h-8" />, color: 'text-red-500' };

if (type.startsWith('audio/')) return { icon: <AudioIcon className="w-8 h-8" />, color: 'text-purple-500' };

if (type === 'application/pdf') return { icon: <PdfIcon className="w-8 h-8" />, color: 'text-orange-500' };

return { icon: <DocumentIcon className="w-8 h-8" />, color: 'text-slate-500' };

};

const PatientEducationManager: React.FC<PatientEducationManagerProps> = ({ department, onAddMaterial, onDeleteMaterial, onUpdateMaterialDescription, onAddPatient, onDeletePatient, onSendMessage, onBack }) => {

const [selectedPatient, setSelectedPatient] = useState<Patient | null>(null);

const [replyText, setReplyText] = useState('');

// Modals state

const [isPatientModalOpen, setIsPatientModalOpen] = useState(false);

const [isContentModalOpen, setIsContentModalOpen] = useState(false);

const [previewMaterial, setPreviewMaterial] = useState<TrainingMaterial | null>(null);

// Patient Modal form state

const [newPatientName, setNewPatientName] = useState('');

const [newPatientNationalId, setNewPatientNationalId] = useState('');

const [newPatientPassword, setNewPatientPassword] = useState('');

const messagesEndRef = useRef<HTMLDivElement>(null);

const fileInputRef = useRef<HTMLInputElement>(null);

useEffect(() => {

messagesEndRef.current?.scrollIntoView({ behavior: "auto" });

}, [selectedPatient?.chatHistory]);

// This effect synchronizes the selected patient's data with the main state from props.

// It prevents displaying stale chat history after a new message is sent.

useEffect(() => {

if (selectedPatient) {

const updatedPatient = department.patients?.find(p => p.id === selectedPatient.id);

if (updatedPatient) {

// Prevent re-setting state if the object is identical

if (JSON.stringify(updatedPatient) !== JSON.stringify(selectedPatient)) {

setSelectedPatient(updatedPatient);

}

} else {

// The patient was deleted, so clear the view

setSelectedPatient(null);

}

}

}, [department, selectedPatient]);

const handleSendReply = () => {

if (selectedPatient && replyText.trim()) {

onSendMessage(selectedPatient.id, { text: replyText.trim() }, 'manager');

setReplyText('');

}

};

const handleFileSelect = async (e: React.ChangeEvent<HTMLInputElement>) => {

if (!selectedPatient) return;

const file = e.target.files?.[0];

if (!file) return;

try {

const dataUrl = await new Promise<string>((resolve, reject) => {

const reader = new FileReader();

reader.onload = (event) => resolve(event.target?.result as string);

reader.onerror = (error) => reject(error);

reader.readAsDataURL(file);

});

const fileId = `chat-file-${Date.now()}`;

await db.addMaterial({ id: fileId, data: dataUrl });

onSendMessage(selectedPatient.id, {

file: { id: fileId, name: file.name, type: file.type }

}, 'manager');

} catch (error) {

console.error("Error sending file:", error);

alert("خطا در ارسال فایل.");

} finally {

if (fileInputRef.current) fileInputRef.current.value = '';

}

};

const handleKeyDown = (e: React.KeyboardEvent<HTMLTextAreaElement>) => {

if (e.key === 'Enter' && !e.shiftKey) {

e.preventDefault();

handleSendReply();

}

};

const handleOpenAddPatientModal = () => {

setNewPatientName('');

setNewPatientNationalId('');

setNewPatientPassword('');

setIsPatientModalOpen(true);

};

const handleSavePatient = () => {

if (!newPatientName.trim() || !newPatientNationalId.trim() || !newPatientPassword.trim()) {

alert('لطفا تمام فیلدها را پر کنید.');

return;

}

onAddPatient(newPatientName.trim(), newPatientNationalId.trim(), newPatientPassword.trim());

setIsPatientModalOpen(false);

};

const handleDeletePatientClick = (patient: Patient) => {

if (window.confirm(`آیا از حذف بیمار "${patient.name}" با کد ملی ${patient.nationalId} مطمئن هستید؟`)) {

if (selectedPatient?.id === patient.id) {

setSelectedPatient(null);

}

onDeletePatient(patient.id);

}

};

return (

<>

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">آموزش به بیمار: <span className="text-orange-500">{department.name}</span></h1>

<div className="flex items-center gap-2">

<button

onClick={() => setIsContentModalOpen(true)}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700"

>

مدیریت محتوای آموزشی

</button>

<button

onClick={handleOpenAddPatientModal}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-teal-600 rounded-lg hover:bg-teal-700"

>

<PlusIcon className="w-5 h-5"/>

افزودن بیمار جدید

</button>

</div>

</div>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md flex h-[calc(100vh-14rem)]">

{/\* Right Column: Patient List \*/}

<div className="w-1/3 border-l border-slate-200 dark:border-slate-700 flex flex-col">

<div className="p-4 border-b border-slate-200 dark:border-slate-700">

<h2 className="text-xl font-bold">بیماران بخش</h2>

</div>

<div className="flex-grow overflow-y-auto">

{(department.patients || []).map(p => (

<div key={p.id} className={`w-full text-right p-4 border-b dark:border-slate-700 group flex justify-between items-center ${selectedPatient?.id === p.id ? 'bg-indigo-100 dark:bg-indigo-900/50' : 'hover:bg-slate-50 dark:hover:bg-slate-700/50'}`}>

<button onClick={() => setSelectedPatient(p)} className="flex-grow text-right">

<p className="font-bold text-slate-800 dark:text-slate-100">{p.name}</p>

<p className="text-sm text-slate-500 dark:text-slate-400">{(p.chatHistory || []).length} پیام</p>

</button>

<button onClick={() => handleDeletePatientClick(p)} className="p-2 text-slate-400 hover:text-red-500 rounded-full opacity-0 group-hover:opacity-100" aria-label="Delete Patient">

<TrashIcon className="w-5 h-5"/>

</button>

</div>

))}

</div>

</div>

{/\* Left Column: Chat and Content View \*/}

<div className="w-2/3 flex flex-col">

{selectedPatient ? (

<div className="flex flex-col h-full">

{/\* Chat Container \*/}

<div className="flex flex-col flex-grow min-h-0">

<div className="p-4 border-b border-slate-200 dark:border-slate-700">

<h3 className="text-xl font-bold text-slate-800 dark:text-slate-100">چت با {selectedPatient.name}</h3>

</div>

<div className="flex-grow overflow-y-auto p-4 space-y-4 bg-slate-50 dark:bg-slate-900/50">

{(selectedPatient.chatHistory || []).map(msg => (

<div key={msg.id} className={`flex items-end gap-2 ${msg.sender === 'manager' ? 'justify-end' : 'justify-start'}`}>

<div className={`max-w-xl p-3 rounded-lg shadow ${msg.sender === 'manager' ? 'bg-green-500 text-white rounded-br-none' : 'bg-slate-200 dark:bg-slate-700 text-slate-800 dark:text-slate-100 rounded-bl-none'}`}>

{msg.text && <p className="whitespace-pre-wrap">{msg.text}</p>}

{msg.file && (

<button

onClick={() => setPreviewMaterial({ id: msg.file!.id, name: msg.file!.name, type: msg.file!.type })}

className="flex items-center gap-3 text-left"

>

<div className={`flex-shrink-0 ${msg.sender === 'manager' ? 'text-white' : 'text-slate-600 dark:text-slate-300'}`}>

{getIconForMimeType(msg.file.type).icon}

</div>

<div>

<p className="font-semibold break-all">{msg.file.name}</p>

<p className="text-xs opacity-80">برای مشاهده کلیک کنید</p>

</div>

</button>

)}

<p className={`text-xs mt-1 text-right ${msg.sender === 'manager' ? 'text-green-100' : 'text-slate-400'}`}>

{new Date(msg.timestamp).toLocaleString('fa-IR', { hour: '2-digit', minute: '2-digit' })}

</p>

</div>

</div>

))}

<div ref={messagesEndRef} />

</div>

<div className="p-4 border-t border-slate-200 dark:border-slate-700 flex items-center gap-2">

<input type="file" ref={fileInputRef} onChange={handleFileSelect} className="hidden" />

<button

onClick={() => fileInputRef.current?.click()}

className="p-3 text-slate-500 bg-slate-100 dark:bg-slate-700 dark:text-slate-300 rounded-lg hover:bg-slate-200 dark:hover:bg-slate-600"

aria-label="Attach file"

>

<PaperClipIcon className="w-6 h-6"/>

</button>

<textarea

value={replyText}

onChange={(e) => setReplyText(e.target.value)}

onKeyDown={handleKeyDown}

placeholder="پاسخ خود را بنویسید..."

rows={2}

className="flex-grow px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<button

onClick={handleSendReply}

className="px-6 py-3 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700 disabled:opacity-50"

disabled={!replyText.trim()}

>

ارسال

</button>

</div>

</div>

{/\* Educational Content Viewer \*/}

<div className="flex-shrink-0 border-t border-slate-200 dark:border-slate-700 bg-slate-50 dark:bg-slate-800">

<div className="p-4">

<h4 className="text-lg font-bold">محتوای آموزشی در دسترس بیمار</h4>

</div>

<div className="px-4 pb-4 max-h-48 overflow-y-auto">

{department.patientEducationMaterials && department.patientEducationMaterials.length > 0 ? (

<div className="grid grid-cols-2 md:grid-cols-3 gap-3">

{department.patientEducationMaterials.map(material => {

const { icon, color } = getIconForMimeType(material.type);

return (

<button

key={material.id}

onClick={() => setPreviewMaterial(material)}

className="group flex flex-col text-right p-3 bg-slate-100 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-md hover:bg-slate-200 dark:hover:bg-slate-700 transition-all text-slate-800 dark:text-slate-200"

>

<div className={`mb-2 ${color}`}>

{icon}

</div>

<h5 className="font-semibold text-xs break-all w-full truncate" title={material.name}>{material.name}</h5>

<p className="text-xs text-slate-500 dark:text-slate-400 mt-1 h-8 overflow-hidden text-ellipsis">

{material.description || 'برای مشاهده کلیک کنید'}

</p>

</button>

);

})}

</div>

) : (

<p className="text-center text-sm text-slate-400 py-4">هیچ محتوای آموزشی برای این بخش تعریف نشده است.</p>

)}

</div>

</div>

</div>

) : (

<div className="flex items-center justify-center h-full text-slate-500">

<p>برای مشاهده مکالمات، یک بیمار را انتخاب کنید.</p>

</div>

)}

</div>

</div>

{/\* Modals \*/}

<Modal isOpen={isContentModalOpen} onClose={() => setIsContentModalOpen(false)} title="مدیریت محتوای آموزشی بیماران" maxWidthClass="max-w-4xl">

<ContentManager

materials={department.patientEducationMaterials || []}

onAddMaterial={onAddMaterial}

onDeleteMaterial={onDeleteMaterial}

onUpdateMaterialDescription={onUpdateMaterialDescription}

setPreviewMaterial={setPreviewMaterial}

/>

</Modal>

<Modal isOpen={isPatientModalOpen} onClose={() => setIsPatientModalOpen(false)} title="افزودن بیمار جدید">

<div className="space-y-4">

<input

type="text"

value={newPatientName}

onChange={(e) => setNewPatientName(e.target.value)}

placeholder="نام و نام خانوادگی بیمار"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="text"

inputMode="numeric"

value={newPatientNationalId}

onChange={(e) => setNewPatientNationalId(e.target.value.replace(/\D/g, ''))}

placeholder="کد ملی بیمار (برای ورود)"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<input

type="password"

value={newPatientPassword}

onChange={(e) => setNewPatientPassword(e.target.value)}

placeholder="رمز عبور"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3 pt-2">

<button onClick={() => setIsPatientModalOpen(false)} className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500">

انصراف

</button>

<button onClick={handleSavePatient} className="px-4 py-2 font-semibold text-white bg-teal-600 rounded-md hover:bg-teal-700">

ذخیره بیمار

</button>

</div>

</div>

</Modal>

{previewMaterial && <PreviewModal isOpen={!!previewMaterial} onClose={() => setPreviewMaterial(null)} material={previewMaterial}/>}

</>

);

};

// Sub-component for Content Management inside the modal

const ContentManager: React.FC<{

materials: TrainingMaterial[];

onAddMaterial: (material: TrainingMaterial) => void;

onDeleteMaterial: (materialId: string) => void;

onUpdateMaterialDescription: (materialId: string, description: string) => void;

setPreviewMaterial: (material: TrainingMaterial | null) => void;

}> = ({ materials, onAddMaterial, onDeleteMaterial, onUpdateMaterialDescription, setPreviewMaterial }) => {

const [isUploading, setIsUploading] = useState(false);

const [uploadError, setUploadError] = useState<string | null>(null);

const [descriptionModalOpen, setDescriptionModalOpen] = useState(false);

const [editingMaterial, setEditingMaterial] = useState<TrainingMaterial | null>(null);

const [pendingFile, setPendingFile] = useState<{file: File, dataUrl: string} | null>(null);

const [materialDescription, setMaterialDescription] = useState('');

const handleFileUpload = (file: File) => {

setIsUploading(true);

setUploadError(null);

const reader = new FileReader();

reader.onload = (e) => {

try {

const dataUrl = e.target?.result as string;

if (!dataUrl) throw new Error("Failed to read file.");

setPendingFile({ file, dataUrl });

setMaterialDescription('');

setDescriptionModalOpen(true);

} catch (err) {

setUploadError(err instanceof Error ? err.message : "خطای ناشناخته در پردازش فایل.");

} finally {

setIsUploading(false);

}

};

reader.onerror = () => {

setUploadError("خطا در خواندن فایل.");

setIsUploading(false);

};

reader.readAsDataURL(file);

};

const handleSaveMaterialWithDescription = () => {

if (pendingFile) {

const { file, dataUrl } = pendingFile;

const newMaterial: TrainingMaterial = {

id: Date.now().toString(),

name: file.name,

type: file.type,

data: dataUrl,

description: materialDescription.trim(),

};

onAddMaterial(newMaterial);

} else if (editingMaterial) {

onUpdateMaterialDescription(editingMaterial.id, materialDescription.trim());

}

setDescriptionModalOpen(false);

setPendingFile(null);

setEditingMaterial(null);

setMaterialDescription('');

};

const handleOpenEditDescriptionModal = (material: TrainingMaterial) => {

setPendingFile(null);

setEditingMaterial(material);

setMaterialDescription(material.description || '');

setDescriptionModalOpen(true);

};

const handleDeleteClick = (materialId: string, materialName: string) => {

if (window.confirm(`آیا از حذف فایل "${materialName}" مطمئن هستید؟`)) {

onDeleteMaterial(materialId);

}

};

return (

<div className="space-y-6">

<div>

<h3 className="text-xl font-bold mb-4">بارگذاری محتوای جدید</h3>

{isUploading && <p className="text-center text-indigo-500">در حال پردازش فایل...</p>}

{uploadError && <p className="text-center text-red-500 my-2">{uploadError}</p>}

<FileUploader onFileUpload={handleFileUpload} accept="\*" title="آپلود فایل، صدا، عکس یا فیلم" />

</div>

<div>

<h3 className="text-xl font-bold mb-4 border-t pt-6 border-slate-200 dark:border-slate-700">محتوای موجود</h3>

{materials.length === 0 ? (

<p className="text-center py-8 text-slate-400">هیچ محتوایی بارگذاری نشده است.</p>

) : (

<div className="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 gap-4">

{materials.map(material => {

const { icon, color } = getIconForMimeType(material.type);

return (

<div key={material.id} className="group relative flex flex-col text-right p-4 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-md transition-all text-slate-800 dark:text-slate-200">

<button onClick={() => setPreviewMaterial(material)} className="flex-grow flex flex-col text-right">

<div className={`mb-3 ${color}`}>{icon}</div>

<h4 className="font-bold text-sm break-all w-full truncate" title={material.name}>{material.name}</h4>

<p className="text-xs text-slate-500 dark:text-slate-400 mt-1 flex-grow h-8 overflow-hidden text-ellipsis">

{material.description || 'بدون توضیح'}

</p>

</button>

<div className="absolute top-2 left-2 flex items-center gap-1 opacity-0 group-hover:opacity-100 transition-opacity">

<button onClick={() => handleOpenEditDescriptionModal(material)} className="p-1.5 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-500 dark:text-slate-300 hover:text-indigo-500 dark:hover:text-indigo-400">

<EditIcon className="w-4 h-4"/>

</button>

<button onClick={() => handleDeleteClick(material.id, material.name)} className="p-1.5 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-500 dark:text-slate-300 hover:text-red-500 dark:hover:text-red-400">

<TrashIcon className="w-4 h-4"/>

</button>

</div>

</div>

);

})}

</div>

)}

</div>

<Modal isOpen={descriptionModalOpen} onClose={() => setDescriptionModalOpen(false)} title={pendingFile ? "افزودن توضیحات" : "ویرایش توضیحات"}>

<div className="space-y-4">

<textarea

value={materialDescription}

onChange={(e) => setMaterialDescription(e.target.value)}

placeholder="توضیحات مربوط به فایل را اینجا وارد کنید..."

rows={4}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3">

<button onClick={() => setDescriptionModalOpen(false)} className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500">

انصراف

</button>

<button onClick={handleSaveMaterialWithDescription} className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700">

ذخیره

</button>

</div>

</div>

</Modal>

</div>

);

};

export default PatientEducationManager;

PatientPortalView.tsx

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

import { Department, Patient, TrainingMaterial } from '../types';

import PreviewModal from './PreviewModal';

import { ImageIcon } from './icons/ImageIcon';

import { VideoIcon } from './icons/VideoIcon';

import { AudioIcon } from './icons/AudioIcon';

import { PdfIcon } from './icons/PdfIcon';

import { DocumentIcon } from './icons/DocumentIcon';

import { PaperClipIcon } from './icons/PaperClipIcon';

import \* as db from '../services/db';

import { SunIcon } from './icons/SunIcon';

interface PatientPortalViewProps {

department: Department;

patient: Patient;

onSendMessage: (content: { text?: string; file?: { id: string; name: string; type: string; } }) => void;

}

const getIconForMimeType = (type: string, size: 'large' | 'small' = 'large'): { icon: React.ReactNode, color: string } => {

const className = size === 'large' ? "w-12 h-12" : "w-8 h-8";

if (type.startsWith('image/')) return { icon: <ImageIcon className={className} />, color: 'text-blue-500' };

if (type.startsWith('video/')) return { icon: <VideoIcon className={className} />, color: 'text-red-500' };

if (type.startsWith('audio/')) return { icon: <AudioIcon className={className} />, color: 'text-purple-500' };

if (type === 'application/pdf') return { icon: <PdfIcon className={className} />, color: 'text-orange-500' };

return { icon: <DocumentIcon className={className} />, color: 'text-slate-500' };

};

const PatientPortalView: React.FC<PatientPortalViewProps> = ({ department, patient, onSendMessage }) => {

const [previewMaterial, setPreviewMaterial] = useState<TrainingMaterial | null>(null);

const [newMessage, setNewMessage] = useState('');

const messagesEndRef = useRef<HTMLDivElement>(null);

const fileInputRef = useRef<HTMLInputElement>(null);

const materials = department.patientEducationMaterials || [];

const chatHistory = patient.chatHistory || [];

useEffect(() => {

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

}, [chatHistory]);

const handleSend = () => {

if (newMessage.trim()) {

onSendMessage({ text: newMessage.trim() });

setNewMessage('');

}

};

const handleFileSelect = async (e: React.ChangeEvent<HTMLInputElement>) => {

const file = e.target.files?.[0];

if (!file) return;

try {

const dataUrl = await new Promise<string>((resolve, reject) => {

const reader = new FileReader();

reader.onload = (event) => resolve(event.target?.result as string);

reader.onerror = (error) => reject(error);

reader.readAsDataURL(file);

});

const fileId = `chat-file-${Date.now()}`;

await db.addMaterial({ id: fileId, data: dataUrl });

onSendMessage({

file: { id: fileId, name: file.name, type: file.type }

});

} catch (error) {

console.error("Error sending file:", error);

alert("خطا در ارسال فایل.");

} finally {

if (fileInputRef.current) fileInputRef.current.value = '';

}

};

const handleKeyDown = (e: React.KeyboardEvent<HTMLTextAreaElement>) => {

if (e.key === 'Enter' && !e.shiftKey) {

e.preventDefault();

handleSend();

}

};

return (

<div className="max-w-6xl mx-auto">

{/\* New Banner \*/}

<div className="bg-gradient-to-br from-sky-100 to-blue-200 dark:from-sky-900 dark:to-blue-900/70 p-8 rounded-2xl shadow-lg mb-10 flex items-center gap-6">

<SunIcon className="w-20 h-20 text-yellow-500 dark:text-yellow-400 flex-shrink-0"/>

<div>

<h1 className="text-4xl font-bold text-slate-800 dark:text-slate-100">سلام، {patient.name} عزیز</h1>

<p className="text-lg mt-2 text-slate-600 dark:text-slate-300">

به پورتال آموزشی خود خوش آمدید. ما اینجا هستیم تا در مسیر بهبودی همراه شما باشیم.

</p>

</div>

</div>

{/\* Educational Content Section - MOVED UP \*/}

<div className="mb-10">

<h2 className="text-3xl font-bold text-center mb-6 text-slate-800 dark:text-slate-100">مطالب آموزشی بخش {department.name}</h2>

{materials.length === 0 ? (

<div className="text-center py-16 bg-white dark:bg-slate-800 rounded-lg shadow">

<h3 className="text-xl font-medium text-slate-500">محتوای آموزشی برای این بخش وجود ندارد.</h3>

</div>

) : (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">

{materials.map(material => {

const { icon, color } = getIconForMimeType(material.type, 'large');

return (

<button

key={material.id}

onClick={() => setPreviewMaterial(material)}

className="group flex flex-col text-right p-5 bg-white dark:bg-slate-800 rounded-xl shadow-md hover:shadow-xl hover:border-indigo-500 border-2 border-transparent transition-all transform hover:-translate-y-1.5 text-slate-800 dark:text-slate-200"

>

<div className={`mb-4 ${color}`}>

{icon}

</div>

<h4 className="font-bold text-base break-all w-full truncate" title={material.name}>{material.name}</h4>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-1 flex-grow">

{material.description || 'برای مشاهده کلیک کنید'}

</p>

</button>

);

})}

</div>

)}

</div>

{/\* Chat Section \*/}

<div className="bg-white dark:bg-slate-800 rounded-2xl shadow-lg flex flex-col h-[70vh] mb-8 overflow-hidden border border-slate-200 dark:border-slate-700">

<div className="p-4 border-b border-slate-200 dark:border-slate-700 flex-shrink-0">

<h3 className="text-2xl font-bold text-center">سوالی راجب بیماری و درمانت داری همین جا بپرس</h3>

</div>

<div className="flex-grow overflow-y-auto p-4 space-y-4 bg-slate-50 dark:bg-slate-900/50">

{chatHistory.map(msg => (

<div key={msg.id} className={`flex items-end gap-2 ${msg.sender === 'patient' ? 'justify-end' : 'justify-start'}`}>

<div className={`max-w-xl p-3 rounded-2xl shadow ${msg.sender === 'patient' ? 'bg-indigo-600 text-white rounded-br-none' : 'bg-slate-200 dark:bg-slate-700 text-slate-800 dark:text-slate-100 rounded-bl-none'}`}>

{msg.text && <p className="whitespace-pre-wrap">{msg.text}</p>}

{msg.file && (

<button

onClick={() => setPreviewMaterial({ id: msg.file!.id, name: msg.file!.name, type: msg.file!.type })}

className="flex items-center gap-3 text-left p-2 -m-2 rounded-lg hover:bg-black/10"

>

<div className={`flex-shrink-0 ${msg.sender === 'patient' ? 'text-white' : 'text-slate-600 dark:text-slate-300'}`}>

{getIconForMimeType(msg.file.type, 'small').icon}

</div>

<div>

<p className="font-semibold break-all">{msg.file.name}</p>

<p className="text-xs opacity-80">برای مشاهده کلیک کنید</p>

</div>

</button>

)}

<p className={`text-xs mt-1 text-right ${msg.sender === 'patient' ? 'text-indigo-200' : 'text-slate-400'}`}>

{new Date(msg.timestamp).toLocaleString('fa-IR', { hour: '2-digit', minute: '2-digit' })}

</p>

</div>

</div>

))}

<div ref={messagesEndRef} />

</div>

<div className="p-4 border-t border-slate-200 dark:border-slate-700 flex items-center gap-2 flex-shrink-0 bg-white dark:bg-slate-800">

<input type="file" ref={fileInputRef} onChange={handleFileSelect} className="hidden" />

<button

onClick={() => fileInputRef.current?.click()}

className="p-3 text-slate-500 bg-slate-100 dark:bg-slate-700 dark:text-slate-300 rounded-full hover:bg-slate-200 dark:hover:bg-slate-600 transition-colors"

aria-label="Attach file"

>

<PaperClipIcon className="w-6 h-6"/>

</button>

<textarea

value={newMessage}

onChange={(e) => setNewMessage(e.target.value)}

onKeyDown={handleKeyDown}

placeholder="پیام خود را بنویسید..."

rows={1}

className="flex-grow px-4 py-3 border border-slate-300 rounded-full dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500 resize-none"

/>

<button

onClick={handleSend}

className="px-6 py-3 font-semibold text-white bg-indigo-600 rounded-full hover:bg-indigo-700 disabled:opacity-50 transition-colors"

disabled={!newMessage.trim()}

>

ارسال

</button>

</div>

</div>

<footer className="text-center mt-12 py-4 text-sm text-slate-400">

<p>این محتوا صرفاً جهت اطلاع‌رسانی است و جایگزین مشاوره پزشکی نیست.</p>

</footer>

{previewMaterial && <PreviewModal isOpen={!!previewMaterial} onClose={() => setPreviewMaterial(null)} material={previewMaterial}/>}

</div>

);

};

export default PatientPortalView;

PreviewModal.tsx

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

import { TrainingMaterial } from '../types';

import { DocumentIcon } from './icons/DocumentIcon';

import { SaveIcon } from './icons/SaveIcon';

import \* as db from '../services/db';

interface PreviewModalProps {

isOpen: boolean;

onClose: () => void;

material: TrainingMaterial;

}

const PreviewModal: React.FC<PreviewModalProps> = ({ isOpen, onClose, material }) => {

const [materialData, setMaterialData] = useState<string | null>(null);

const [isLoading, setIsLoading] = useState(true);

const [error, setError] = useState<string | null>(null);

useEffect(() => {

if (isOpen && material) {

setIsLoading(true);

setError(null);

setMaterialData(null);

db.getMaterialData(material.id)

.then(data => {

if (data) {

setMaterialData(data);

} else {

setError('فایل یافت نشد. ممکن است حذف شده باشد.');

}

})

.catch(err => {

console.error(err);

setError('خطا در بارگذاری فایل.');

})

.finally(() => {

setIsLoading(false);

});

}

}, [isOpen, material]);

if (!isOpen) return null;

const renderContent = () => {

if (isLoading) {

return (

<div className="flex flex-col items-center justify-center h-full">

<svg className="animate-spin h-10 w-10 text-indigo-500" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

<circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>

<path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

</svg>

<p className="mt-4 text-slate-500">در حال بارگذاری پیش نمایش...</p>

</div>

);

}

if (error) {

return <p className="text-center text-red-500">{error}</p>

}

if (!materialData) {

return <p className="text-center text-slate-500">محتوایی برای نمایش وجود ندارد.</p>

}

const { type, name } = material;

if (type.startsWith('image/')) {

return <img src={materialData} alt={name} className="max-w-full max-h-full object-contain" />;

}

if (type.startsWith('video/')) {

return (

<video controls className="w-full h-auto max-h-full" autoPlay>

<source src={materialData} type={type} />

مرورگر شما از تگ ویدئو پشتیبانی نمی‌کند.

</video>

);

}

if (type.startsWith('audio/')) {

return (

<div className="p-8">

<audio controls className="w-full" autoPlay>

<source src={materialData} type={type} />

مرورگر شما از تگ صوتی پشتیبانی نمی‌کند.

</audio>

</div>

);

}

// Fallback for other file types

return (

<div className="flex flex-col items-center justify-center p-8 text-center">

<DocumentIcon className="w-24 h-24 text-slate-400 mb-4" />

<p className="font-semibold text-lg text-slate-800 dark:text-slate-100 break-all">{name}</p>

<p className="text-sm text-slate-500 dark:text-slate-400">پیش‌نمایش برای این نوع فایل در دسترس نیست.</p>

</div>

);

};

return (

<div className="fixed inset-0 bg-black bg-opacity-75 z-50 flex justify-center items-center p-4" onClick={onClose}>

<div

className="bg-white dark:bg-slate-800 rounded-lg shadow-xl w-full max-w-4xl h-[90vh] flex flex-col"

onClick={(e) => e.stopPropagation()}

>

<div className="flex justify-between items-center border-b border-slate-200 dark:border-slate-700 p-4 sticky top-0 bg-white dark:bg-slate-800 z-10">

<h3 className="text-lg font-semibold text-slate-900 dark:text-slate-100 truncate pr-4" title={material.name}>

{material.name}

</h3>

<div className="flex items-center gap-2">

<a

href={materialData || '#'}

download={material.name}

className={`inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-700 dark:text-slate-200 dark:hover:bg-slate-600 ${!materialData || isLoading ? 'opacity-50 pointer-events-none' : ''}`}

>

<SaveIcon className="w-4 h-4"/>

دانلود فایل

</a>

<button

onClick={onClose}

className="text-slate-400 hover:text-slate-600 dark:hover:text-slate-200"

aria-label="Close"

>

<svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor">

<path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M6 18L18 6M6 6l12 12" />

</svg>

</button>

</div>

</div>

<div className="p-4 overflow-auto flex-grow flex justify-center items-center bg-slate-50 dark:bg-slate-900/50">

{renderContent()}

</div>

</div>

</div>

);

};

export default PreviewModal;

SkillCategoryDisplay.tsx

import React, { useMemo } from 'react';

import { SkillCategory, SkillItem } from '../types';

import { TrashIcon } from './icons/TrashIcon';

import { PlusIcon } from './icons/PlusIcon';

interface SkillCategoryDisplayProps {

category: SkillCategory;

isEditing?: boolean;

onCategoryChange?: (updatedCategory: SkillCategory) => void;

onDeleteCategory?: () => void;

maxPossibleScore?: number;

}

const MIN\_SCORE\_PERCENTAGE = 60;

const SkillCategoryDisplay: React.FC<SkillCategoryDisplayProps> = ({ category, isEditing = false, onCategoryChange, onDeleteCategory, maxPossibleScore = 4 }) => {

const { totalScore, maxScore, percentage } = useMemo(() => {

if (!category || !category.items) return { totalScore: 0, maxScore: 0, percentage: 0 };

const total = category.items.reduce((sum, item) => sum + item.score, 0);

const max = category.items.length \* maxPossibleScore;

const perc = max > 0 ? (total / max) \* 100 : 0;

return { totalScore: total, maxScore: max, percentage: perc };

}, [category, maxPossibleScore]);

const handleNameChange = (e: React.ChangeEvent<HTMLInputElement>) => {

if (onCategoryChange) {

onCategoryChange({ ...category, name: e.target.value });

}

};

const handleItemChange = (index: number, field: keyof SkillItem, value: string | number) => {

if (onCategoryChange) {

const newItems = [...category.items];

const currentItem = { ...newItems[index] };

if (field === 'score') {

const numValue = typeof value === 'string' ? parseFloat(value) : value;

currentItem[field] = isNaN(numValue) ? 0 : Math.max(0, Math.min(maxPossibleScore, numValue));

} else {

currentItem[field] = value as string;

}

newItems[index] = currentItem;

onCategoryChange({ ...category, items: newItems });

}

};

const handleAddItem = () => {

if (onCategoryChange) {

const newItems = [...category.items, { description: 'مهارت جدید', score: 0 }];

onCategoryChange({ ...category, items: newItems });

}

}

const handleDeleteItem = (index: number) => {

if (onCategoryChange) {

const newItems = category.items.filter((\_, i) => i !== index);

onCategoryChange({ ...category, items: newItems });

}

}

if (isEditing) {

return (

<div className="bg-white dark:bg-slate-800 rounded-xl shadow-lg p-6 mb-6 border-2 border-dashed border-slate-400 dark:border-slate-600">

<div className="flex justify-between items-center mb-4 pb-3 border-b border-slate-200 dark:border-slate-700">

<input

type="text"

value={category.name}

onChange={handleNameChange}

placeholder="نام دسته"

className="text-2xl font-bold bg-transparent focus:outline-none focus:ring-0 border-none p-0 text-slate-800 dark:text-slate-100 w-full"

/>

<button

onClick={onDeleteCategory}

className="p-2 text-slate-400 hover:text-red-500 rounded-full"

aria-label="Delete Category"

>

<TrashIcon className="w-5 h-5" />

</button>

</div>

<div className="overflow-x-auto">

<table className="w-full text-sm text-right">

<thead className="text-xs text-slate-700 uppercase bg-slate-50 dark:bg-slate-700 dark:text-slate-300">

<tr>

<th className="px-4 py-2">شرح مهارت</th>

<th className="px-4 py-2 w-32 text-center">نمره (از {maxPossibleScore})</th>

<th className="px-4 py-2 w-16"></th>

</tr>

</thead>

<tbody>

{category.items.map((item, index) => (

<tr key={index} className="border-b dark:border-slate-700 odd:bg-white odd:dark:bg-slate-800 even:bg-slate-50 even:dark:bg-slate-700/50">

<td className="p-2">

<input

type="text"

value={item.description}

onChange={(e) => handleItemChange(index, 'description', e.target.value)}

className="w-full px-2 py-1 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-slate-500"

/>

</td>

<td className="p-2">

<input

type="number"

value={item.score}

onChange={(e) => handleItemChange(index, 'score', e.target.value)}

min="0"

max={maxPossibleScore}

step="0.1"

className="w-full px-2 py-1 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-slate-500 text-center"

/>

</td>

<td className="p-2 text-center">

<button

onClick={() => handleDeleteItem(index)}

className="p-2 text-slate-400 hover:text-red-500 rounded-full"

aria-label={`Delete skill ${item.description}`}

>

<TrashIcon className="w-5 h-5" />

</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

<div className="mt-4">

<button

onClick={handleAddItem}

className="inline-flex items-center gap-2 px-3 py-1.5 text-sm font-semibold text-slate-600 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-900/50 dark:text-slate-300 dark:hover:bg-slate-900"

>

<PlusIcon className="w-4 h-4" />

افزودن مهارت

</button>

</div>

</div>

);

}

// --- Display View ---

const isQualified = percentage >= MIN\_SCORE\_PERCENTAGE;

const scoreColor = isQualified ? (percentage >= 80 ? 'bg-teal-500' : 'bg-amber-400') : 'bg-rose-500';

const scoreTextColor = isQualified ? (percentage >= 80 ? 'text-teal-600 dark:text-teal-400' : 'text-amber-600 dark:text-amber-400') : 'text-rose-600 dark:text-rose-400';

return (

<div className="bg-white dark:bg-slate-800 rounded-xl shadow-lg p-6 mb-6">

<div className="flex justify-between items-start mb-4">

<h3 className="text-2xl font-bold text-slate-800 dark:text-slate-100">{category.name}</h3>

{category?.items?.length > 0 && (

<span className={`px-3 py-1 text-sm font-semibold rounded-full ${isQualified ? 'bg-teal-100 text-teal-800 dark:bg-teal-900 dark:text-teal-200' : 'bg-rose-100 text-rose-800 dark:bg-rose-900 dark:text-rose-200'}`}>

{isQualified ? 'احراز صلاحیت' : 'عدم احراز صلاحیت'}

</span>

)}

</div>

{category?.items?.length > 0 ? (

<>

<div className="mb-4">

<div className="flex justify-between items-center mb-1">

<span className="text-sm font-medium text-slate-600 dark:text-slate-300">امتیاز کل: {totalScore.toFixed(1)} / {maxScore}</span>

<span className={`text-lg font-bold ${scoreTextColor}`}>{percentage.toFixed(1)}%</span>

</div>

<div className="w-full bg-slate-200 dark:bg-slate-700 rounded-full h-4">

<div

className={`h-4 rounded-full transition-all duration-500 ${scoreColor}`}

style={{ width: `${percentage}%` }}

></div>

</div>

</div>

<div className="max-h-96 overflow-y-auto border border-slate-200 dark:border-slate-700 rounded-md">

<table className="w-full text-sm text-right text-slate-500 dark:text-slate-400">

<thead className="text-xs text-slate-700 uppercase bg-slate-50 dark:bg-slate-700 dark:text-slate-300 sticky top-0">

<tr>

<th scope="col" className="px-6 py-3">شرح مهارت</th>

<th scope="col" className="px-6 py-3 w-24 text-center">نمره (از {maxPossibleScore})</th>

</tr>

</thead>

<tbody>

{category.items.map((item, index) => (

<tr key={index} className="border-b dark:border-slate-700 odd:bg-white odd:dark:bg-slate-800 even:bg-slate-50 even:dark:bg-slate-700/50">

<td className="px-6 py-4 font-medium text-slate-900 dark:text-white">{item.description}</td>

<td className="px-6 py-4 text-center">{item.score}</td>

</tr>

))}

</tbody>

</table>

</div>

</>

) : (

<p className="text-center py-8 text-slate-400">هیچ موردی برای نمایش در این دسته وجود ندارد.</p>

)}

</div>

);

};

export default SkillCategoryDisplay;

StaffMemberView.tsx

import React, { useState, useMemo, useEffect } from 'react';

import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from 'recharts';

import { Department, StaffMember, SkillCategory, Assessment, NamedChecklistTemplate, ExamTemplate, Question, QuestionType, ExamSubmission, ExamAnswer, UserRole, MonthlyTraining, TrainingMaterial, NewsBanner } from '../types';

import { generateImprovementPlan } from '../services/geminiService';

import SkillCategoryDisplay from './SkillCategoryDisplay';

import SuggestionModal from './SuggestionModal';

import Modal from './Modal';

import { BackIcon } from './icons/BackIcon';

import { AiIcon } from './icons/AiIcon';

import { SaveIcon } from './icons/SaveIcon';

import { EditIcon } from './icons/EditIcon';

import { ChartBarIcon } from './icons/ChartBarIcon';

import { ClipboardDocumentCheckIcon } from './icons/ClipboardDocumentCheckIcon';

import { AcademicCapIcon } from './icons/AcademicCapIcon';

import { DocumentIcon } from './icons/DocumentIcon';

import PreviewModal from './PreviewModal';

import { ImageIcon } from './icons/ImageIcon';

import { VideoIcon } from './icons/VideoIcon';

import { AudioIcon } from './icons/AudioIcon';

import { PdfIcon } from './icons/PdfIcon';

import { ShieldCheckIcon } from './icons/ShieldCheckIcon';

import NewsCarousel from './NewsCarousel';

interface StaffMemberViewProps {

department: Department;

staffMember: StaffMember;

onBack: () => void;

onAddOrUpdateAssessment: (departmentId: string, staffId: string, month: string, skills: SkillCategory[], template?: NamedChecklistTemplate) => void;

onUpdateAssessmentMessages: (departmentId: string, staffId: string, month: string, messages: { supervisorMessage: string; managerMessage: string; }) => void;

onSubmitExam: (departmentId: string, staffId: string, month: string, submission: ExamSubmission) => void;

checklistTemplates: NamedChecklistTemplate[];

examTemplates: ExamTemplate[];

trainingMaterials: MonthlyTraining[];

accreditationMaterials: TrainingMaterial[];

newsBanners: NewsBanner[];

userRole: UserRole;

}

const PERSIAN\_MONTHS = [

"فروردین", "اردیبهشت", "خرداد",

"تیر", "مرداد", "شهریور",

"مهر", "آبان", "آذر",

"دی", "بهمن", "اسفند"

];

const CHART\_COLORS = ['#3b82f6', '#16a34a', '#f97316', '#dc2626', '#8b5cf6', '#db2777'];

type ViewMode = 'month\_selection' | 'assessment\_form' | 'assessment\_result';

type AssessmentResultView = 'details' | 'summary' | 'exam\_list' | 'exam\_taking' | 'exam\_result' | 'training\_materials' | 'accreditation\_materials';

const getIconForMimeType = (type: string): { icon: React.ReactNode, color: string } => {

if (type.startsWith('image/')) return { icon: <ImageIcon className="w-10 h-10" />, color: 'text-blue-500' };

if (type.startsWith('video/')) return { icon: <VideoIcon className="w-10 h-10" />, color: 'text-red-500' };

if (type.startsWith('audio/')) return { icon: <AudioIcon className="w-10 h-10" />, color: 'text-purple-500' };

if (type === 'application/pdf') return { icon: <PdfIcon className="w-10 h-10" />, color: 'text-orange-500' };

return { icon: <DocumentIcon className="w-10 h-10" />, color: 'text-slate-500' };

};

const StaffMemberView: React.FC<StaffMemberViewProps> = ({

department,

staffMember,

onBack,

onAddOrUpdateAssessment,

onUpdateAssessmentMessages,

onSubmitExam,

checklistTemplates,

examTemplates,

trainingMaterials,

accreditationMaterials,

newsBanners,

userRole,

}) => {

const [selectedMonth, setSelectedMonth] = useState<string | null>(null);

const [isSuggestionModalOpen, setIsSuggestionModalOpen] = useState(false);

const [suggestionContent, setSuggestionContent] = useState<string | null>(null);

const [isSuggestionLoading, setIsSuggestionLoading] = useState(false);

const [supervisorMessage, setSupervisorMessage] = useState('');

const [managerMessage, setManagerMessage] = useState('');

const [isChecklistModalOpen, setIsChecklistModalOpen] = useState(false);

const [viewMode, setViewMode] = useState<ViewMode>('month\_selection');

const [assessmentSubView, setAssessmentSubView] = useState<AssessmentResultView>('details');

const [assessmentFormData, setAssessmentFormData] = useState<SkillCategory[]>([]);

const [activeAssessmentTemplate, setActiveAssessmentTemplate] = useState<NamedChecklistTemplate | null>(null);

// Exam state

const [currentExam, setCurrentExam] = useState<ExamTemplate | null>(null);

const [examAnswers, setExamAnswers] = useState<Map<string, string>>(new Map());

const [currentSubmissionResult, setCurrentSubmissionResult] = useState<ExamSubmission | null>(null);

// Preview State

const [previewMaterial, setPreviewMaterial] = useState<TrainingMaterial | null>(null);

const assessmentsByMonth = useMemo(() => {

const map = new Map<string, Assessment>();

staffMember.assessments.forEach(a => map.set(a.month, a));

return map;

}, [staffMember.assessments]);

const handleGetComprehensiveSuggestions = () => {

if (!selectedMonth) return;

const assessment = assessmentsByMonth.get(selectedMonth);

if (!assessment) return;

const maxScore = assessment.maxScore ?? 4;

const weakSkillsByCateogry = assessment.skillCategories

.map(cat => ({

categoryName: cat.name,

skills: cat.items.filter(item => item.score < maxScore)

}))

.filter(cat => cat.skills.length > 0);

if (weakSkillsByCateogry.length === 0) {

alert("این پرسنل در تمام مهارت‌ها نمره کامل کسب کرده است.");

return;

}

setIsSuggestionModalOpen(true);

setIsSuggestionLoading(true);

setSuggestionContent(null);

setTimeout(() => {

const result = generateImprovementPlan(

staffMember,

weakSkillsByCateogry,

assessment.supervisorMessage,

assessment.managerMessage

);

setSuggestionContent(result);

setIsSuggestionLoading(false);

}, 50);

};

const hasWeakSkillsInSelectedMonth = useMemo(() => {

if (!selectedMonth) return false;

const assessment = assessmentsByMonth.get(selectedMonth);

if (!assessment) return false;

const maxScore = assessment.maxScore ?? 4;

return assessment.skillCategories.some(cat => cat.items.some(item => item.score < maxScore));

}, [selectedMonth, assessmentsByMonth]);

const progressChartInfo = useMemo(() => {

const allCategoryNames = new Set<string>();

staffMember.assessments.forEach(ass => ass.skillCategories.forEach(cat => allCategoryNames.add(cat.name)));

const categoryNames = Array.from(allCategoryNames);

const data = staffMember.assessments

.filter(assessment => PERSIAN\_MONTHS.includes(assessment.month))

.map(assessment => {

const scores: { [key: string]: any } = { name: assessment.month };

const maxPossibleScore = assessment.maxScore ?? 4;

assessment.skillCategories.forEach(cat => {

const totalScore = cat.items.reduce((sum, item) => sum + item.score, 0);

const maxScore = cat.items.length \* maxPossibleScore;

scores[cat.name] = maxScore > 0 ? parseFloat(((totalScore / maxScore) \* 100).toFixed(1)) : null;

});

// Ensure all categories are present for this month, even if with a null score

categoryNames.forEach(name => {

if (!(name in scores)) {

scores[name] = null;

}

});

return scores;

}).sort((a, b) => PERSIAN\_MONTHS.indexOf(a.name as string) - PERSIAN\_MONTHS.indexOf(b.name as string));

return { data, categoryNames };

}, [staffMember.assessments]);

const handleSaveMessages = () => {

if (!selectedMonth) return;

onUpdateAssessmentMessages(department.id, staffMember.id, selectedMonth, {

supervisorMessage,

managerMessage

});

alert('پیام ها با موفقیت ذخیره شدند.');

};

const handleMonthSelect = (month: string) => {

setSelectedMonth(month);

const assessment = assessmentsByMonth.get(month);

// Allow staff to view the result page even without an assessment,

// to access the training materials archive.

if (userRole === UserRole.Staff) {

setViewMode('assessment\_result');

setAssessmentSubView('summary');

setSupervisorMessage(assessment?.supervisorMessage || '');

setManagerMessage(assessment?.managerMessage || '');

return;

}

// For other roles (Admin, Supervisor, Manager)

if (assessment) {

setViewMode('assessment\_result');

setAssessmentSubView('details');

setSupervisorMessage(assessment.supervisorMessage || '');

setManagerMessage(assessment.managerMessage || '');

} else {

// No assessment exists, prompt to create one.

if (!checklistTemplates || checklistTemplates.length === 0) {

alert("هیچ قالب چک لیستی برای این بیمارستان تعریف نشده است. لطفا ابتدا از صفحه بخش‌ها، یک قالب چک لیست بسازید.");

setSelectedMonth(null);

return;

}

setIsChecklistModalOpen(true);

}

};

const handleStartAssessmentWithTemplate = (template: NamedChecklistTemplate) => {

const newCategoriesFromTemplate: SkillCategory[] = template.categories.map(catTemplate => ({

name: catTemplate.name,

items: catTemplate.items.map(itemTemplate => ({

description: itemTemplate.description,

score: template.minScore ?? 0,

})),

}));

setAssessmentFormData(newCategoriesFromTemplate);

setActiveAssessmentTemplate(template);

setIsChecklistModalOpen(false);

setViewMode('assessment\_form');

};

const handleSaveAssessment = () => {

if(!selectedMonth) return;

onAddOrUpdateAssessment(department.id, staffMember.id, selectedMonth, assessmentFormData, activeAssessmentTemplate || undefined);

setViewMode('assessment\_result');

setAssessmentSubView('details');

};

const handleEditAssessment = () => {

if (!selectedMonth) return;

const assessment = assessmentsByMonth.get(selectedMonth);

if (!assessment) return;

// Reconstruct a temporary template to hold the scoring rules for the form

const templateForEditing: NamedChecklistTemplate = {

id: assessment.templateId || 'imported-template',

name: 'Editing Assessment',

categories: [], // Not needed for the form logic

minScore: assessment.minScore ?? 0,

maxScore: assessment.maxScore ?? 4,

};

setActiveAssessmentTemplate(templateForEditing);

const categoriesToEdit = JSON.parse(JSON.stringify(assessment.skillCategories));

setAssessmentFormData(categoriesToEdit);

setViewMode('assessment\_form');

};

const handleScoreChange = (catIndex: number, itemIndex: number, score: number) => {

const newCategories = [...assessmentFormData];

const category = { ...newCategories[catIndex] };

const items = [...category.items];

const min = activeAssessmentTemplate?.minScore ?? 0;

const max = activeAssessmentTemplate?.maxScore ?? 4;

const finalScore = Math.max(min, Math.min(max, isNaN(score) ? 0 : score));

items[itemIndex] = { ...items[itemIndex], score: finalScore };

category.items = items;

newCategories[catIndex] = category;

setAssessmentFormData(newCategories);

};

const handleStartExam = (exam: ExamTemplate) => {

setCurrentExam(exam);

setExamAnswers(new Map());

setAssessmentSubView('exam\_taking');

};

const handleExamAnswerChange = (questionId: string, answer: string) => {

setExamAnswers(new Map(examAnswers.set(questionId, answer)));

};

const handleSubmitExamClick = () => {

if (!currentExam) return;

if (examAnswers.size !== currentExam.questions.length) {

if(!window.confirm('شما به تمام سوالات پاسخ نداده‌اید. آیا مایل به ثبت آزمون هستید؟')) {

return;

}

}

const answers: ExamAnswer[] = Array.from(examAnswers.entries()).map(([questionId, answer]) => ({

questionId,

answer,

}));

let score = 0;

const correctableQuestions = currentExam.questions.filter(q => q.type === QuestionType.MultipleChoice);

correctableQuestions.forEach(q => {

const userAnswer = examAnswers.get(q.id);

if (userAnswer === q.correctAnswer) {

score++;

}

});

const submission: ExamSubmission = {

id: Date.now().toString(),

examTemplateId: currentExam.id,

examName: currentExam.name,

answers,

score,

totalCorrectableQuestions: correctableQuestions.length,

submissionDate: new Date().toISOString(),

questions: currentExam.questions, // Snapshot

};

onSubmitExam(department.id, staffMember.id, selectedMonth!, submission);

setCurrentSubmissionResult(submission);

setCurrentExam(null);

setAssessmentSubView('exam\_result');

};

const handleInternalBack = () => {

if (viewMode === 'assessment\_form') {

setViewMode('month\_selection');

setSelectedMonth(null);

setActiveAssessmentTemplate(null);

return;

}

if (viewMode === 'assessment\_result') {

switch (assessmentSubView) {

case 'summary':

if (userRole === UserRole.Staff) {

setViewMode('month\_selection');

setSelectedMonth(null);

} else {

setAssessmentSubView('details');

}

break;

case 'exam\_list':

case 'training\_materials':

case 'accreditation\_materials':

setAssessmentSubView('summary');

break;

case 'exam\_taking':

case 'exam\_result':

setAssessmentSubView('exam\_list');

setCurrentExam(null);

setCurrentSubmissionResult(null);

break;

case 'details':

default:

setViewMode('month\_selection');

setSelectedMonth(null);

break;

}

}

};

const renderAssessmentForm = () => {

const minScore = activeAssessmentTemplate?.minScore ?? 0;

const maxScore = activeAssessmentTemplate?.maxScore ?? 4;

return (

<div>

<div className="flex justify-between items-center mb-6 bg-slate-100 dark:bg-slate-800 p-4 rounded-lg">

<h2 className="text-xl font-bold">

ثبت ارزیابی ماه <span className="text-indigo-600 dark:text-indigo-400">{selectedMonth}</span>

</h2>

<div className="flex gap-2">

<button

onClick={handleSaveAssessment}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-green-600 rounded-lg hover:bg-green-700"

>

<SaveIcon className="w-5 h-5" />

تایید و ذخیره نمرات

</button>

</div>

</div>

<div className="space-y-6">

{assessmentFormData.map((category, catIndex) => (

<div key={catIndex} className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h3 className="text-2xl font-bold text-slate-800 dark:text-slate-100 mb-4 pb-3 border-b border-slate-200 dark:border-slate-700">{category.name}</h3>

<div className="space-y-4">

{category.items.map((item, itemIndex) => (

<div key={itemIndex} className="grid grid-cols-1 md:grid-cols-3 gap-4 items-center">

<label className="md:col-span-2 text-slate-700 dark:text-slate-300" htmlFor={`score-${catIndex}-${itemIndex}`}>

{item.description}

</label>

<input

id={`score-${catIndex}-${itemIndex}`}

type="number"

value={item.score}

onChange={(e) => handleScoreChange(catIndex, itemIndex, parseFloat(e.target.value))}

min={minScore}

max={maxScore}

step="0.1"

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-1 focus:ring-indigo-500 text-center"

/>

</div>

))}

</div>

</div>

))}

</div>

</div>

);

};

const renderAssessmentDetails = (assessment: Assessment) => {

return (

<div>

<div className="flex justify-end items-start mb-6">

<div className="flex items-center gap-2">

{userRole !== UserRole.Staff && (

<button

onClick={handleEditAssessment}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-indigo-600 rounded-lg hover:bg-indigo-700"

>

<EditIcon className="w-5 h-5" />

ویرایش نمرات

</button>

)}

<button onClick={() => setAssessmentSubView('summary')} className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-teal-600 rounded-lg hover:bg-teal-700">

<ChartBarIcon className="w-5 h-5" />

مشاهده نتایج و برنامه پیشنهادی

</button>

</div>

</div>

<h2 className="text-2xl font-bold mb-4">جزئیات ارزیابی در <span className="text-indigo-600 dark:text-indigo-400">{selectedMonth}</span></h2>

{assessment.skillCategories.map((category) => (

<SkillCategoryDisplay

key={category.name}

category={category}

maxPossibleScore={assessment.maxScore}

/>

))}

{userRole !== UserRole.Staff && (

<div className="mt-8 bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h3 className="text-xl font-bold mb-4">پیام‌های مدیریتی</h3>

<p className="text-slate-500 dark:text-slate-400 mb-6">در این بخش می‌توانید بازخورد خود را برای این ارزیابی ثبت کنید. این پیام‌ها در برنامه بهبود پیشنهادی نیز نمایش داده خواهند شد.</p>

<div className="space-y-4">

<div>

<label htmlFor="supervisorMessage" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

پیام سوپروایزر آموزشی

</label>

<textarea

id="supervisorMessage"

rows={4}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

value={supervisorMessage}

onChange={(e) => setSupervisorMessage(e.target.value)}

placeholder="یک متن چند خطی بنویسید..."

/>

</div>

<div>

<label htmlFor="managerMessage" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-1">

پیام مسئول بخش

</label>

<textarea

id="managerMessage"

rows={4}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

value={managerMessage}

onChange={(e) => setManagerMessage(e.target.value)}

placeholder="یک متن چند خطی بنویسید..."

/>

</div>

<div className="flex justify-end">

<button

onClick={handleSaveMessages}

className="inline-flex items-center gap-2 px-4 py-2 font-semibold text-white bg-green-600 rounded-lg hover:bg-green-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-green-500"

>

<SaveIcon className="w-5 h-5" />

ذخیره پیام ها

</button>

</div>

</div>

</div>

)}

</div>

);

};

const renderAssessmentSummary = () => {

const assessmentExists = selectedMonth ? assessmentsByMonth.has(selectedMonth) : false;

const workLog = staffMember.workLogs?.find(log => log.month === selectedMonth);

return (

<div>

{workLog && (

<div className="mb-8 bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h3 className="text-xl font-bold text-slate-800 dark:text-slate-100 mb-4">گزارش کارکرد ماهانه</h3>

<div className="grid grid-cols-2 md:grid-cols-4 gap-4 text-center">

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">موظفی ماه</p>

<p className="text-2xl font-semibold text-slate-600 dark:text-slate-400">{workLog.requiredHours} <span className="text-sm">ساعت</span></p>

</div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">اضافه کار</p>

<p className="text-2xl font-semibold text-green-600 dark:text-green-400">{workLog.overtimeHours} <span className="text-sm">ساعت</span></p>

</div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">مانده مرخصی فصل</p>

<p className="text-2xl font-semibold text-yellow-600 dark:text-yellow-400">{workLog.quarterlyLeaveRemaining} <span className="text-sm">روز</span></p>

</div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">مانده مرخصی سال</p>

<p className="text-2xl font-semibold text-red-600 dark:text-red-400">{workLog.annualLeaveRemaining} <span className="text-sm">روز</span></p>

</div>

</div>

</div>

)}

{newsBanners && newsBanners.length > 0 && (

<div className="mb-8 max-w-5xl mx-auto">

<NewsCarousel banners={newsBanners} />

</div>

)}

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-2xl font-bold mb-4">صفحه پرسنلی - <span className="text-indigo-600 dark:text-indigo-400">{selectedMonth}</span></h2>

<p className="text-slate-500 dark:text-slate-400 mb-8">از گزینه‌های زیر برای مشاهده برنامه بهبود، برگزاری آزمون یا دسترسی به محتوای آموزشی استفاده کنید.</p>

{!assessmentExists && (

<div className="bg-yellow-50 dark:bg-yellow-900/30 border-r-4 border-yellow-400 p-4 mb-6 rounded-md">

<p className="text-yellow-800 dark:text-yellow-200">ارزیابی برای این ماه ثبت نشده است. برخی گزینه‌ها ممکن است غیرفعال باشند.</p>

</div>

)}

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">

{/\* Card 1: Suggestions \*/}

<button

onClick={handleGetComprehensiveSuggestions}

disabled={isSuggestionLoading || !hasWeakSkillsInSelectedMonth}

className="group relative flex flex-col items-center justify-center text-center p-6 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-lg hover:bg-white dark:hover:bg-slate-700 transition-all duration-300 transform hover:-translate-y-1 disabled:opacity-50 disabled:cursor-not-allowed disabled:transform-none disabled:shadow-sm"

>

<AiIcon className="w-16 h-16 mb-4 text-indigo-500 transition-transform duration-300 group-hover:scale-110" />

<h3 className="text-lg font-semibold text-slate-800 dark:text-slate-100">برنامه جامع پیشنهادی</h3>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-1">

دریافت برنامه مطالعه ۳۰ روزه بر اساس نقاط ضعف.

</p>

{assessmentExists && !hasWeakSkillsInSelectedMonth && (

<span className="absolute top-2 right-2 text-xs font-semibold bg-green-100 text-green-800 dark:bg-green-900 dark:text-green-200 px-2 py-0.5 rounded-full">تکمیل</span>

)}

</button>

{/\* Card 2: Exam \*/}

<button

onClick={() => setAssessmentSubView('exam\_list')}

disabled={!assessmentExists}

className="group relative flex flex-col items-center justify-center text-center p-6 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-lg hover:bg-white dark:hover:bg-slate-700 transition-all duration-300 transform hover:-translate-y-1 disabled:opacity-50 disabled:cursor-not-allowed disabled:transform-none disabled:shadow-sm"

>

<ClipboardDocumentCheckIcon className="w-16 h-16 mb-4 text-purple-500 transition-transform duration-300 group-hover:scale-110" />

<h3 className="text-lg font-semibold text-slate-800 dark:text-slate-100">صفحه آزمون</h3>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-1">

برگزاری آزمون‌های تئوری مرتبط با مهارت‌ها.

</p>

</button>

{/\* Card 3: Training \*/}

<button

onClick={() => setAssessmentSubView('training\_materials')}

className="group relative flex flex-col items-center justify-center text-center p-6 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-lg hover:bg-white dark:hover:bg-slate-700 transition-all duration-300 transform hover:-translate-y-1"

>

<AcademicCapIcon className="w-16 h-16 mb-4 text-blue-500 transition-transform duration-300 group-hover:scale-110" />

<h3 className="text-lg font-semibold text-slate-800 dark:text-slate-100">صفحه آموزشی</h3>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-1">

دسترسی به منابع و محتواهای آموزشی تکمیلی.

</p>

</button>

{/\* Card 4: Accreditation \*/}

<button

onClick={() => setAssessmentSubView('accreditation\_materials')}

className="group relative flex flex-col items-center justify-center text-center p-6 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-lg hover:bg-white dark:hover:bg-slate-700 transition-all duration-300 transform hover:-translate-y-1"

>

<ShieldCheckIcon className="w-16 h-16 mb-4 text-emerald-500 transition-transform duration-300 group-hover:scale-110" />

<h3 className="text-lg font-semibold text-slate-800 dark:text-slate-100">مطالب اعتباربخشی</h3>

<p className="text-sm text-slate-500 dark:text-slate-400 mt-1">

دسترسی به دستورالعمل‌ها و مستندات اعتباربخشی.

</p>

</button>

</div>

</div>

</div>

);

};

const renderExamList = (assessment: Assessment) => {

const previousSubmissions = assessment.examSubmissions || [];

return (

<div>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-2xl font-bold mb-4">لیست آزمون‌ها</h2>

{examTemplates.length === 0 ? (

<p className="text-center py-8 text-slate-400">آزمونی برای این بیمارستان تعریف نشده است.</p>

) : (

<div className="space-y-3">

{examTemplates.map(exam => (

<button key={exam.id} onClick={() => handleStartExam(exam)} className="w-full text-right p-4 bg-slate-50 dark:bg-slate-700/50 rounded-lg hover:bg-slate-100 dark:hover:bg-slate-700 border border-slate-200 dark:border-slate-700 transition-colors">

<h3 className="font-semibold text-lg text-indigo-600 dark:text-indigo-400">{exam.name}</h3>

<p className="text-sm text-slate-500 dark:text-slate-400">{exam.questions.length} سوال</p>

</button>

))}

</div>

)}

{previousSubmissions.length > 0 && (

<div className="mt-8">

<h3 className="text-xl font-bold mb-4 border-t pt-6 border-slate-200 dark:border-slate-700">آزمون‌های انجام شده</h3>

<div className="overflow-x-auto">

<table className="w-full text-sm text-right text-slate-500 dark:text-slate-400">

<thead className="text-xs text-slate-700 uppercase bg-slate-50 dark:bg-slate-700 dark:text-slate-300">

<tr>

<th scope="col" className="px-6 py-3">نام آزمون</th>

<th scope="col" className="px-6 py-3">تاریخ انجام</th>

<th scope="col" className="px-6 py-3 text-center">نمره</th>

<th scope="col" className="px-6 py-3">

<span className="sr-only">مشاهده نتایج</span>

</th>

</tr>

</thead>

<tbody>

{previousSubmissions.map(sub => (

<tr key={sub.id} className="border-b dark:border-slate-700 odd:bg-white odd:dark:bg-slate-800 even:bg-slate-50 even:dark:bg-slate-700/50">

<td scope="row" className="px-6 py-4 font-medium text-slate-900 whitespace-nowrap dark:text-white">

{sub.examName}

</td>

<td className="px-6 py-4">

{new Date(sub.submissionDate).toLocaleString('fa-IR', { year: 'numeric', month: 'long', day: 'numeric', hour: '2-digit', minute: '2-digit' })}

</td>

<td className="px-6 py-4 text-center font-bold text-lg">

{sub.score} / {sub.totalCorrectableQuestions}

</td>

<td className="px-6 py-4 text-left">

<button onClick={() => { setCurrentSubmissionResult(sub); setAssessmentSubView('exam\_result'); }} className="font-medium text-indigo-600 dark:text-indigo-500 hover:underline">مشاهده نتایج</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

</div>

)}

</div>

</div>

);

};

const renderExamTaking = () => {

if (!currentExam) return null;

return (

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-2xl font-bold mb-1">آزمون: {currentExam.name}</h2>

<p className="text-slate-500 dark:text-slate-400 mb-6">به سوالات زیر پاسخ دهید.</p>

<div className="space-y-8">

{currentExam.questions.map((q, index) => (

<div key={q.id} className="border-t border-slate-200 dark:border-slate-700 pt-6">

<p className="font-semibold mb-3">سوال {index + 1}: {q.text}</p>

{q.type === QuestionType.MultipleChoice ? (

<div className="space-y-2">

{q.options?.map((opt, optIndex) => (

<label key={optIndex} className="flex items-center p-3 rounded-md hover:bg-slate-100 dark:hover:bg-slate-700 cursor-pointer">

<input

type="radio"

name={q.id}

value={opt}

checked={examAnswers.get(q.id) === opt}

onChange={(e) => handleExamAnswerChange(q.id, e.target.value)}

className="w-4 h-4 text-indigo-600 focus:ring-indigo-500"

/>

<span className="mr-3">{opt}</span>

</label>

))}

</div>

) : (

<textarea

rows={5}

value={examAnswers.get(q.id) || ''}

onChange={(e) => handleExamAnswerChange(q.id, e.target.value)}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

placeholder="پاسخ خود را اینجا بنویسید..."

/>

)}

</div>

))}

</div>

<div className="flex justify-end items-center mt-8 border-t border-slate-200 dark:border-slate-700 pt-6">

<button onClick={handleSubmitExamClick} className="px-6 py-2 font-semibold text-white bg-green-600 rounded-lg hover:bg-green-700">

ثبت نهایی و مشاهده نتیجه

</button>

</div>

</div>

)

};

const renderExamResult = () => {

if(!currentSubmissionResult) return null;

const { questions, answers, score, totalCorrectableQuestions, examName } = currentSubmissionResult;

const answersMap = new Map(answers.map(a => [a.questionId, a.answer]));

const percentage = totalCorrectableQuestions > 0 ? (score / totalCorrectableQuestions) \* 100 : 0;

const PASS\_THRESHOLD = 70;

const isPass = percentage >= PASS\_THRESHOLD;

return (

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<div className="flex justify-between items-start mb-6 pb-4 border-b border-slate-200 dark:border-slate-700">

<div>

<h2 className="text-2xl font-bold">نتیجه آزمون: {examName}</h2>

<p className="text-slate-500 dark:text-slate-400">نتیجه سوالات شما در زیر آمده است.</p>

</div>

</div>

{totalCorrectableQuestions > 0 && (

<div className="bg-slate-50 dark:bg-slate-700/50 rounded-lg p-6 mb-6 flex flex-col sm:flex-row justify-around items-center text-center gap-4">

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">نمره کسب شده</p>

<p className="text-4xl font-bold text-slate-800 dark:text-slate-100">{score} <span className="text-2xl">/ {totalCorrectableQuestions}</span></p>

</div>

<div className="w-px h-16 bg-slate-200 dark:bg-slate-600 hidden sm:block"></div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">درصد</p>

<p className="text-4xl font-bold text-slate-800 dark:text-slate-100">{percentage.toFixed(1)}<span className="text-2xl">%</span></p>

</div>

<div className="w-px h-16 bg-slate-200 dark:bg-slate-600 hidden sm:block"></div>

<div>

<p className="text-sm text-slate-500 dark:text-slate-400">وضعیت</p>

<p className={`text-4xl font-bold ${isPass ? 'text-green-600 dark:text-green-400' : 'text-red-600 dark:text-red-400'}`}>

{isPass ? 'قبول' : 'مردود'}

</p>

</div>

</div>

)}

<h3 className="text-xl font-bold mb-4">پاسخنامه</h3>

<div className="space-y-6">

{questions.map((q, index) => {

const userAnswer = answersMap.get(q.id) || 'پاسخ داده نشده';

const isCorrect = q.type === QuestionType.MultipleChoice && userAnswer === q.correctAnswer;

const isMC = q.type === QuestionType.MultipleChoice;

return (

<div key={q.id} className="p-4 rounded-lg bg-slate-50 dark:bg-slate-700/50 border border-slate-200 dark:border-slate-700">

<p className="font-semibold mb-2">سوال {index+1}: {q.text}</p>

<p className={`p-2 rounded w-full ${isMC ? (isCorrect ? 'bg-green-100 dark:bg-green-900/50 text-green-800 dark:text-green-300' : 'bg-red-100 dark:bg-red-900/50 text-red-800 dark:text-red-300') : 'bg-blue-100 dark:bg-blue-900/50 text-slate-800 dark:text-slate-200'}`}>

<strong>پاسخ شما:</strong> {userAnswer}

</p>

{isMC && !isCorrect && (

<p className="mt-2 p-2 rounded w-full bg-slate-200 dark:bg-slate-600 text-slate-800 dark:text-slate-200">

<strong>پاسخ صحیح:</strong> {q.correctAnswer}

</p>

)}

{!isMC && (

<p className="mt-2 p-2 rounded w-full bg-slate-200 dark:bg-slate-600 text-slate-800 dark:text-slate-200">

<strong>پاسخ نمونه:</strong> {q.correctAnswer}

</p>

)}

</div>

);

})}

</div>

</div>

);

};

const renderTrainingMaterials = () => {

const allMaterialsByMonth = trainingMaterials;

return (

<div>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-2xl font-bold mb-4">آرشیو محتوای آموزشی</h2>

{allMaterialsByMonth.length === 0 ? (

<p className="text-center py-8 text-slate-400">هیچ محتوای آموزشی برای این بیمارستان بارگذاری نشده است.</p>

) : (

<div className="space-y-8">

{allMaterialsByMonth.map(monthlyTraining => (

<div key={monthlyTraining.month}>

<h3 className="text-xl font-semibold mb-4 pb-2 border-b-2 border-slate-200 dark:border-slate-800 text-slate-700 dark:text-slate-300">

{monthlyTraining.month}

</h3>

{monthlyTraining.materials.length === 0 ? (

<p className="text-slate-400">محتوایی برای این ماه وجود ندارد.</p>

) : (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4">

{monthlyTraining.materials.map(material => {

const { icon, color } = getIconForMimeType(material.type);

return (

<button

key={material.id}

onClick={() => setPreviewMaterial(material)}

className="group relative flex flex-col text-right p-4 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-md hover:bg-slate-100 dark:hover:bg-slate-700 transition-all text-slate-800 dark:text-slate-200"

>

<div className={`mb-3 ${color}`}>

{icon}

</div>

<h4 className="font-bold text-sm break-all w-full truncate" title={material.name}>{material.name}</h4>

<p className="text-xs text-slate-500 dark:text-slate-400 mt-1 h-8 overflow-hidden text-ellipsis">

{material.description || 'بدون توضیح'}

</p>

</button>

);

})}

</div>

)}

</div>

))}

</div>

)}

</div>

</div>

);

};

const renderAccreditationMaterials = () => {

return (

<div>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-2xl font-bold mb-4">آرشیو مطالب اعتباربخشی</h2>

{accreditationMaterials.length === 0 ? (

<p className="text-center py-8 text-slate-400">هیچ محتوای اعتباربخشی برای این بیمارستان بارگذاری نشده است.</p>

) : (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4">

{accreditationMaterials.map(material => {

const { icon, color } = getIconForMimeType(material.type);

return (

<button

key={material.id}

onClick={() => setPreviewMaterial(material)}

className="group relative flex flex-col text-right p-4 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-md hover:bg-slate-100 dark:hover:bg-slate-700 transition-all text-slate-800 dark:text-slate-200"

>

<div className={`mb-3 ${color}`}>

{icon}

</div>

<h4 className="font-bold text-sm break-all w-full truncate" title={material.name}>{material.name}</h4>

<p className="text-xs text-slate-500 dark:text-slate-400 mt-1 h-8 overflow-hidden text-ellipsis">

{material.description || 'بدون توضیح'}

</p>

</button>

);

})}

</div>

)}

</div>

</div>

);

};

const renderAssessmentResult = () => {

const assessment = selectedMonth ? assessmentsByMonth.get(selectedMonth) : undefined;

if (!assessment) {

if (assessmentSubView === 'training\_materials') {

return renderTrainingMaterials();

}

if (assessmentSubView === 'accreditation\_materials') {

return renderAccreditationMaterials();

}

// For staff or anyone else without an assessment for the month,

// show the summary page, which handles this case gracefully.

// Or if trying to access another page, redirect to summary.

return renderAssessmentSummary();

}

switch(assessmentSubView) {

case 'summary': return renderAssessmentSummary();

case 'exam\_list': return renderExamList(assessment);

case 'exam\_taking': return renderExamTaking();

case 'exam\_result': return renderExamResult();

case 'training\_materials': return renderTrainingMaterials();

case 'accreditation\_materials': return renderAccreditationMaterials();

case 'details':

default:

return renderAssessmentDetails(assessment);

}

};

const renderMonthSelector = () => {

const MONTH\_DECORATIONS = [

{ base: 'border-t-teal-500 hover:bg-teal-50 dark:hover:bg-teal-900/50', text: 'text-teal-800 dark:text-teal-300', status: 'text-teal-500 dark:text-teal-400' },

{ base: 'border-t-sky-500 hover:bg-sky-50 dark:hover:bg-sky-900/50', text: 'text-sky-800 dark:text-sky-300', status: 'text-sky-500 dark:text-sky-400' },

{ base: 'border-t-amber-500 hover:bg-amber-50 dark:hover:bg-amber-900/50', text: 'text-amber-800 dark:text-amber-300', status: 'text-amber-500 dark:text-amber-400' },

{ base: 'border-t-rose-500 hover:bg-rose-50 dark:hover:bg-rose-900/50', text: 'text-rose-800 dark:text-rose-300', status: 'text-rose-500 dark:text-rose-400' },

{ base: 'border-t-violet-500 hover:bg-violet-50 dark:hover:bg-violet-900/50', text: 'text-violet-800 dark:text-violet-300', status: 'text-violet-500 dark:text-violet-400' },

{ base: 'border-t-lime-500 hover:bg-lime-50 dark:hover:bg-lime-900/50', text: 'text-lime-800 dark:text-lime-300', status: 'text-lime-500 dark:text-lime-400' },

];

return (

<div>

{progressChartInfo.data.length > 0 && (

<div className="mb-8">

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<h2 className="text-xl font-bold mb-4">نمودار پیشرفت مهارت‌ها</h2>

<div className="bg-slate-100 dark:bg-slate-900 p-4 rounded-lg border border-slate-200 dark:border-slate-700">

<ResponsiveContainer width="100%" height={300}>

<LineChart data={progressChartInfo.data} margin={{ top: 5, right: 20, left: -10, bottom: 5 }}>

<CartesianGrid strokeDasharray="5 5" stroke="rgba(100, 116, 139, 0.3)" />

<XAxis dataKey="name" tick={{ fill: 'currentColor', fontSize: 12 }} className="text-slate-500 dark:text-slate-400" />

<YAxis unit="%" domain={[0, 100]} tick={{ fill: 'currentColor', fontSize: 12 }} className="text-slate-500 dark:text-slate-400" />

<Tooltip

cursor={{ stroke: '#94a3b8', strokeWidth: 1, strokeDasharray: '5 5' }}

contentStyle={{

backgroundColor: 'rgba(15, 23, 42, 0.9)',

borderColor: '#334155',

borderRadius: '0.5rem',

}}

labelStyle={{ color: '#f1f5f9' }}

formatter={(value: number | null, name: string) => {

if (value === null || value === undefined) {

return ["ثبت نشده", name];

}

return [`${value}%`, name];

}}

/>

<Legend wrapperStyle={{ fontSize: '0.8rem' }} />

{progressChartInfo.categoryNames.map((catName, index) => {

return <Line key={catName} type="monotone" dataKey={catName} name={catName} stroke={CHART\_COLORS[index % CHART\_COLORS.length]} strokeWidth={2} activeDot={{ r: 8 }} connectNulls/>

})}

</LineChart>

</ResponsiveContainer>

</div>

</div>

</div>

)}

<h2 className="text-2xl font-bold mb-4">انتخاب ماه برای مشاهده یا ثبت ارزیابی</h2>

<div className="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 lg:grid-cols-6 gap-4">

{PERSIAN\_MONTHS.map((month, index) => {

const hasAssessment = assessmentsByMonth.has(month);

const isSelected = month === selectedMonth;

const decoration = MONTH\_DECORATIONS[index % MONTH\_DECORATIONS.length];

return (

<button

key={month}

onClick={() => handleMonthSelect(month)}

className={`p-6 rounded-lg text-center font-semibold transition-all duration-200 transform focus:outline-none focus:ring-2 focus:ring-offset-2 dark:focus:ring-offset-slate-900

${isSelected

? 'bg-indigo-600 text-white shadow-lg scale-105 ring-indigo-500'

: hasAssessment

? `bg-white dark:bg-slate-800 shadow-md hover:-translate-y-1 border-t-4 ${decoration.base} ${decoration.text}`

: 'bg-slate-100 dark:bg-slate-700/50 text-slate-500 dark:text-slate-400 hover:bg-slate-200 dark:hover:bg-slate-700 border-2 border-dashed border-slate-300 dark:border-slate-600'}`}

>

{month}

{hasAssessment && !isSelected && <span className={`block text-xs font-normal mt-1 ${decoration.status}`}>ثبت شده</span>}

</button>

);

})}

</div>

</div>

);

};

const renderView = () => {

switch(viewMode) {

case 'assessment\_form':

return renderAssessmentForm();

case 'assessment\_result':

return renderAssessmentResult();

case 'month\_selection':

default:

return renderMonthSelector();

}

};

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<div className="flex items-center gap-4">

{viewMode !== 'month\_selection' && (

<button

onClick={handleInternalBack}

className="inline-flex items-center gap-2 px-4 py-2 text-sm font-semibold text-slate-700 bg-slate-200 dark:bg-slate-700 dark:text-slate-200 rounded-lg hover:bg-slate-300 dark:hover:bg-slate-600 border border-slate-300 dark:border-slate-600"

aria-label="Back to previous step"

>

<BackIcon className="w-5 h-5" />

بازگشت

</button>

)}

<div>

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">{staffMember.name}</h1>

<p className="text-slate-500 dark:text-slate-400">{staffMember.title} - بخش {department.name}</p>

</div>

</div>

</div>

{renderView()}

<SuggestionModal

isOpen={isSuggestionModalOpen}

onClose={() => setIsSuggestionModalOpen(false)}

title={`برنامه پیشنهادی برای ${staffMember.name}`}

content={suggestionContent}

isLoading={isSuggestionLoading}

/>

<Modal isOpen={isChecklistModalOpen} onClose={() => setIsChecklistModalOpen(false)} title="انتخاب قالب چک لیست">

<div className="space-y-3">

<p className="text-sm text-slate-500 dark:text-slate-400">یک قالب برای شروع ارزیابی این ماه انتخاب کنید.</p>

{checklistTemplates.map(template => (

<button

key={template.id}

onClick={() => handleStartAssessmentWithTemplate(template)}

className="w-full text-right p-3 bg-slate-50 dark:bg-slate-700 rounded-md hover:bg-slate-100 dark:hover:bg-slate-900/50 transition-colors"

>

<h4 className="font-semibold text-indigo-700 dark:text-indigo-400">{template.name}</h4>

<p className="text-xs text-slate-400">{template.categories.length} دسته, {template.categories.reduce((acc, cat) => acc + cat.items.length, 0)} مورد</p>

</button>

))}

</div>

</Modal>

<PreviewModal isOpen={!!previewMaterial} onClose={() => setPreviewMaterial(null)} material={previewMaterial!}/>

</div>

);

};

export default StaffMemberView;

SuggestionModal.tsx

import React, { useRef, useState } from 'react';

import jspdf from 'jspdf';

import html2canvas from 'html2canvas';

import { AiIcon } from './icons/AiIcon';

import { DocumentIcon } from './icons/DocumentIcon';

interface SuggestionModalProps {

isOpen: boolean;

onClose: () => void;

title: string;

content: string | null;

isLoading: boolean;

}

const SuggestionModal: React.FC<SuggestionModalProps> = ({ isOpen, onClose, title, content, isLoading }) => {

const contentRef = useRef<HTMLDivElement>(null);

const [isPdfDownloading, setIsPdfDownloading] = useState(false);

if (!isOpen) return null;

const handleDownloadPdf = async () => {

if (!contentRef.current) return;

setIsPdfDownloading(true);

const staffName = title.replace('برنامه پیشنهادی برای ', '');

const filename = `برنامه-بهبود-${staffName.replace(/ /g, '\_')}.pdf`;

try {

const contentElement = contentRef.current;

// The key fix is to tell html2canvas to render the entire scrollable height, not just the visible part.

const canvas = await html2canvas(contentElement, {

scale: 2,

useCORS: true,

backgroundColor: window.getComputedStyle(contentElement).getPropertyValue('background-color'),

height: contentElement.scrollHeight,

windowHeight: contentElement.scrollHeight,

});

const imgData = canvas.toDataURL('image/png');

const pdf = new jspdf('p', 'pt', 'a4');

const pdfSize = {

width: pdf.internal.pageSize.getWidth(),

height: pdf.internal.pageSize.getHeight(),

};

const imgProps = {

width: canvas.width,

height: canvas.height,

};

const ratio = imgProps.width / imgProps.height;

const pdfImgWidth = pdfSize.width;

const pdfImgHeight = pdfSize.width / ratio;

let position = 0;

let heightLeft = pdfImgHeight;

pdf.addImage(imgData, 'PNG', 0, position, pdfImgWidth, pdfImgHeight);

heightLeft -= pdfSize.height;

while (heightLeft > 0) {

position -= pdfSize.height;

pdf.addPage();

pdf.addImage(imgData, 'PNG', 0, position, pdfImgWidth, pdfImgHeight);

heightLeft -= pdfSize.height;

}

pdf.save(filename);

} catch (error) {

console.error("Error generating PDF:", error);

alert("خطایی در ساخت فایل PDF رخ داد.");

} finally {

setIsPdfDownloading(false);

}

};

const handleDownloadWord = () => {

if (!contentRef.current) return;

const staffName = title.replace('برنامه پیشنهادی برای ', '');

const filename = `برنامه-بهبود-${staffName.replace(/ /g, '\_')}.doc`;

// The contentRef contains the formatted prose div, let's get its innerHTML

const contentHtml = contentRef.current.querySelector('.prose')?.innerHTML || '';

const header = `<html xmlns:o='urn:schemas-microsoft-com:office:office' xmlns:w='urn:schemas-microsoft-com:office:word' xmlns='http://www.w3.org/TR/REC-html40'><head><meta charset='utf-8'><title>Improvement Plan</title><style>body { font-family: 'Vazirmatn', 'Times New Roman', serif; direction: rtl; }</style></head><body>`;

const footer = "</body></html>";

const sourceHTML = header + `<h1>${title}</h1>` + contentHtml + footer;

const source = 'data:application/vnd.ms-word;charset=utf-8,' + encodeURIComponent(sourceHTML);

const fileDownload = document.createElement("a");

document.body.appendChild(fileDownload);

fileDownload.href = source;

fileDownload.download = filename;

fileDownload.click();

document.body.removeChild(fileDownload);

};

const formatContent = (text: string | null): string => {

if (!text) return "";

return text

.replace(/\\*\\*(.\*?)\\*\\*/g, '<strong>$1</strong>')

.replace(/---/g, '<hr class="my-4 border-slate-300 dark:border-slate-600" />')

.replace(/<QUOTE>(.\*?)<\/QUOTE>/gs, (match, p1) => `<blockquote class="border-r-4 border-slate-300 dark:border-slate-600 pr-4 italic text-slate-500 dark:text-slate-400 my-4">${p1.trim().replace(/\n/g, '<br />')}</blockquote>`)

.replace(/\n/g, '<br />');

};

return (

<div className="fixed inset-0 bg-black bg-opacity-75 z-50 flex justify-center items-center p-4" onClick={onClose}>

<div

className="bg-white dark:bg-slate-800 rounded-lg shadow-xl w-full max-w-3xl h-[90vh] flex flex-col"

onClick={(e) => e.stopPropagation()}

>

<div className="flex justify-between items-center border-b border-slate-200 dark:border-slate-700 p-4 sticky top-0 bg-white dark:bg-slate-800">

<h3 className="text-xl font-semibold text-slate-900 dark:text-slate-100 flex items-center gap-2">

<AiIcon className="w-6 h-6 text-indigo-500" />

{title}

</h3>

<button

onClick={onClose}

className="text-slate-400 hover:text-slate-600 dark:hover:text-slate-200"

aria-label="Close"

>

<svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor">

<path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M6 18L18 6M6 6l12 12" />

</svg>

</button>

</div>

<div ref={contentRef} className="p-6 overflow-y-auto flex-grow bg-white dark:bg-slate-800">

{isLoading ? (

<div className="flex flex-col items-center justify-center h-full">

<svg className="animate-spin h-12 w-12 text-indigo-500" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

<circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>

<path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

</svg>

<p className="mt-4 text-slate-500">در حال آماده سازی برنامه پیشنهادی...</p>

</div>

) : (

<div className="prose prose-slate dark:prose-invert max-w-none" dangerouslySetInnerHTML={{ \_\_html: formatContent(content) }}></div>

)}

</div>

{!isLoading && content && (

<div className="border-t border-slate-200 dark:border-slate-700 p-4 flex justify-end items-center gap-3 bg-slate-50 dark:bg-slate-800/50">

<button

onClick={handleDownloadWord}

className="inline-flex items-center gap-2 px-4 py-2 text-sm font-semibold text-slate-700 bg-white dark:bg-slate-700 dark:text-slate-200 border border-slate-300 dark:border-slate-600 rounded-lg hover:bg-slate-100 dark:hover:bg-slate-600"

>

<DocumentIcon className="w-5 h-5" />

دانلود Word

</button>

<button

onClick={handleDownloadPdf}

disabled={isPdfDownloading}

className="inline-flex items-center gap-2 px-4 py-2 text-sm font-semibold text-slate-700 bg-white dark:bg-slate-700 dark:text-slate-200 border border-slate-300 dark:border-slate-600 rounded-lg hover:bg-slate-100 dark:hover:bg-slate-600 disabled:opacity-50 disabled:cursor-wait"

>

<DocumentIcon className="w-5 h-5" />

{isPdfDownloading ? 'در حال ساخت...' : 'دانلود PDF'}

</button>

</div>

)}

</div>

</div>

);

};

export default SuggestionModal;

TrainingManager.tsx

import React, { useState, useMemo } from 'react';

import { MonthlyTraining, TrainingMaterial } from '../types';

import { TrashIcon } from './icons/TrashIcon';

import { EditIcon } from './icons/EditIcon';

import FileUploader from './FileUploader';

import PreviewModal from './PreviewModal';

import Modal from './Modal';

import { ImageIcon } from './icons/ImageIcon';

import { VideoIcon } from './icons/VideoIcon';

import { AudioIcon } from './icons/AudioIcon';

import { PdfIcon } from './icons/PdfIcon';

import { DocumentIcon } from './icons/DocumentIcon';

interface TrainingManagerProps {

monthlyTrainings: MonthlyTraining[];

onAddMaterial: (month: string, material: TrainingMaterial) => void;

onDeleteMaterial: (month: string, materialId: string) => void;

onUpdateMaterialDescription: (month: string, materialId: string, description: string) => void;

onBack: () => void;

}

const PERSIAN\_MONTHS = [

"فروردین", "اردیبهشت", "خرداد", "تیر", "مرداد", "شهریور", "مهر", "آبان", "آذر", "دی", "بهمن", "اسفند"

];

const getIconForMimeType = (type: string): { icon: React.ReactNode, color: string } => {

if (type.startsWith('image/')) return { icon: <ImageIcon className="w-10 h-10" />, color: 'text-blue-500' };

if (type.startsWith('video/')) return { icon: <VideoIcon className="w-10 h-10" />, color: 'text-red-500' };

if (type.startsWith('audio/')) return { icon: <AudioIcon className="w-10 h-10" />, color: 'text-purple-500' };

if (type === 'application/pdf') return { icon: <PdfIcon className="w-10 h-10" />, color: 'text-orange-500' };

return { icon: <DocumentIcon className="w-10 h-10" />, color: 'text-slate-500' };

};

const TrainingManager: React.FC<TrainingManagerProps> = ({ monthlyTrainings, onAddMaterial, onDeleteMaterial, onUpdateMaterialDescription, onBack }) => {

const [selectedMonth, setSelectedMonth] = useState<string>(PERSIAN\_MONTHS[0]);

const [isUploading, setIsUploading] = useState(false);

const [uploadError, setUploadError] = useState<string | null>(null);

const [previewMaterial, setPreviewMaterial] = useState<TrainingMaterial | null>(null);

// For description modal

const [descriptionModalOpen, setDescriptionModalOpen] = useState(false);

const [editingMaterial, setEditingMaterial] = useState<TrainingMaterial | null>(null);

const [pendingFile, setPendingFile] = useState<{file: File, dataUrl: string} | null>(null);

const [materialDescription, setMaterialDescription] = useState('');

const materialsForSelectedMonth = useMemo(() => {

return monthlyTrainings.find(t => t.month === selectedMonth)?.materials || [];

}, [monthlyTrainings, selectedMonth]);

const handleFileUpload = (file: File) => {

setIsUploading(true);

setUploadError(null);

const reader = new FileReader();

reader.onload = (e) => {

try {

const dataUrl = e.target?.result as string;

if (!dataUrl) throw new Error("Failed to read file.");

setPendingFile({ file, dataUrl });

setMaterialDescription(''); // Reset for new file

setDescriptionModalOpen(true);

} catch (err) {

setUploadError(err instanceof Error ? err.message : "خطای ناشناخته در پردازش فایل.");

} finally {

setIsUploading(false);

}

};

reader.onerror = () => {

setUploadError("خطا در خواندن فایل.");

setIsUploading(false);

};

reader.readAsDataURL(file);

};

const handleDeleteClick = (month: string, materialId: string, materialName: string) => {

if (window.confirm(`آیا از حذف فایل "${materialName}" مطمئن هستید؟`)) {

onDeleteMaterial(month, materialId);

}

};

const handleSaveMaterialWithDescription = () => {

if (pendingFile) { // Adding new

const { file, dataUrl } = pendingFile;

const newMaterial: TrainingMaterial = {

id: Date.now().toString(),

name: file.name,

type: file.type,

data: dataUrl,

description: materialDescription.trim(),

};

onAddMaterial(selectedMonth, newMaterial);

} else if (editingMaterial) { // Editing existing

onUpdateMaterialDescription(selectedMonth, editingMaterial.id, materialDescription.trim());

}

// Reset state

setDescriptionModalOpen(false);

setPendingFile(null);

setEditingMaterial(null);

setMaterialDescription('');

};

const handleOpenEditDescriptionModal = (material: TrainingMaterial) => {

setPendingFile(null); // Ensure we're in edit mode

setEditingMaterial(material);

setMaterialDescription(material.description || '');

setDescriptionModalOpen(true);

};

return (

<div className="p-4 sm:p-6 lg:p-8">

<div className="flex justify-between items-center mb-6">

<h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100">مدیریت آموزش پرسنل</h1>

</div>

<div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">

<div className="mb-6">

<label htmlFor="month-select" className="block text-sm font-medium text-slate-700 dark:text-slate-300 mb-2">

انتخاب ماه برای بارگذاری محتوای آموزشی:

</label>

<select

id="month-select" value={selectedMonth} onChange={e => setSelectedMonth(e.target.value)}

className="w-full max-w-xs px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

>

{PERSIAN\_MONTHS.map(month => (<option key={month} value={month}>{month}</option>))}

</select>

</div>

<h2 className="text-xl font-bold mb-4">بارگذاری محتوای جدید برای <span className="text-indigo-500">{selectedMonth}</span></h2>

{isUploading && <p className="text-center text-indigo-500">در حال پردازش فایل...</p>}

{uploadError && <p className="text-center text-red-500 my-2">{uploadError}</p>}

<FileUploader onFileUpload={handleFileUpload} accept="\*" title="آپلود فایل، صدا، عکس یا فیلم آموزشی" />

<div className="mt-8">

<h3 className="text-xl font-bold mb-4 border-t pt-6 border-slate-200 dark:border-slate-700">محتوای بارگذاری شده برای {selectedMonth}</h3>

{materialsForSelectedMonth.length === 0 ? (

<p className="text-center py-8 text-slate-400">هیچ محتوایی برای این ماه بارگذاری نشده است.</p>

) : (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4">

{materialsForSelectedMonth.map(material => {

const { icon, color } = getIconForMimeType(material.type);

return (

<div key={material.id} className="group relative flex flex-col text-right p-4 bg-slate-50 dark:bg-slate-700/50 rounded-lg shadow-sm hover:shadow-md transition-all text-slate-800 dark:text-slate-200">

<button onClick={() => setPreviewMaterial(material)} className="flex-grow flex flex-col text-right">

<div className={`mb-3 ${color}`}>{icon}</div>

<h4 className="font-bold text-sm break-all w-full truncate" title={material.name}>{material.name}</h4>

<p className="text-xs text-slate-500 dark:text-slate-400 mt-1 flex-grow h-8 overflow-hidden text-ellipsis">

{material.description || 'بدون توضیح'}

</p>

</button>

<div className="absolute top-2 left-2 flex items-center gap-1 opacity-0 group-hover:opacity-100 transition-opacity">

<button onClick={() => handleOpenEditDescriptionModal(material)} className="p-1.5 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-500 dark:text-slate-300 hover:text-indigo-500 dark:hover:text-indigo-400">

<EditIcon className="w-4 h-4"/>

</button>

<button onClick={() => handleDeleteClick(selectedMonth, material.id, material.name)} className="p-1.5 bg-slate-200 dark:bg-slate-600 rounded-full text-slate-500 dark:text-slate-300 hover:text-red-500 dark:hover:text-red-400">

<TrashIcon className="w-4 h-4"/>

</button>

</div>

</div>

)

})}

</div>

)}

</div>

</div>

{previewMaterial && <PreviewModal isOpen={!!previewMaterial} onClose={() => setPreviewMaterial(null)} material={previewMaterial}/>}

<Modal isOpen={descriptionModalOpen} onClose={() => setDescriptionModalOpen(false)} title={pendingFile ? "افزودن توضیحات" : "ویرایش توضیحات"}>

<div className="space-y-4">

<textarea

value={materialDescription}

onChange={(e) => setMaterialDescription(e.target.value)}

placeholder="توضیحات مربوط به فایل آموزشی را اینجا وارد کنید..."

rows={4}

className="w-full px-3 py-2 border border-slate-300 rounded-md dark:bg-slate-700 dark:border-slate-600 focus:outline-none focus:ring-2 focus:ring-indigo-500"

/>

<div className="flex justify-end gap-3">

<button onClick={() => setDescriptionModalOpen(false)} className="px-4 py-2 font-semibold text-slate-700 bg-slate-100 rounded-md hover:bg-slate-200 dark:bg-slate-600 dark:text-slate-200 dark:hover:bg-slate-500">

انصراف

</button>

<button onClick={handleSaveMaterialWithDescription} className="px-4 py-2 font-semibold text-white bg-indigo-600 rounded-md hover:bg-indigo-700">

ذخیره

</button>

</div>

</div>

</Modal>

</div>

);

};

export default TrainingManager;

WelcomeScreen.tsx

import React from 'react';

import { AparatIcon } from './icons/AparatIcon';

import { TelegramIcon } from './icons/TelegramIcon';

import { HeartbeatIcon } from './icons/HeartbeatIcon';

interface WelcomeScreenProps {

onEnter: () => void;

}

const WelcomeScreen: React.FC<WelcomeScreenProps> = ({ onEnter }) => {

return (

<div className="h-screen w-screen flex items-center justify-center p-4 bg-gradient-to-br from-gray-50 via-sky-50 to-indigo-100 dark:from-slate-800 dark:via-slate-900 dark:to-indigo-900 transition-colors duration-500">

<div className="relative celestial-glow rounded-2xl">

<div className="w-full max-w-2xl text-center bg-white dark:bg-slate-800 rounded-2xl shadow-xl p-8 md:p-12 border border-slate-200 dark:border-slate-700">

<div className="mx-auto mb-6 w-20 h-20 rounded-full flex items-center justify-center bg-indigo-100 dark:bg-indigo-900/50 border border-indigo-200 dark:border-indigo-800 shadow-lg opacity-0 animate-fade-in-up">

<HeartbeatIcon className="w-10 h-10 text-indigo-500 dark:text-indigo-400 animate-heartbeat-pulse" />

</div>

<h1 className="text-5xl md:text-6xl font-bold text-slate-800 dark:text-slate-100 mb-4 opacity-0 animate-fade-in-up animation-delay-200" style={{ fontFamily: "'Scheherazade New', serif" }}>

بسم الله الرحمن الرحیم

</h1>

<h2 className="text-3xl md:text-4xl font-semibold text-slate-700 dark:text-slate-200 mt-6 mb-3 opacity-0 animate-fade-in-up animation-delay-400">

به سامانه بیمارستان من خوش آمدید

</h2>

<p className="text-xl md:text-2xl text-slate-500 dark:text-slate-400 mb-8 opacity-0 animate-fade-in-up animation-delay-600">

سازنده: حسین نصاری

</p>

<button

onClick={onEnter}

className="px-8 py-3 font-bold text-white bg-indigo-600 rounded-lg shadow-lg hover:shadow-indigo-500/30 hover:bg-indigo-700 focus:outline-none focus:ring-4 focus:ring-offset-2 focus:ring-offset-gray-50 dark:focus:ring-offset-slate-900 focus:ring-indigo-500 transform hover:scale-105 transition-all duration-300 opacity-0 animate-fade-in-up animation-delay-800"

>

ورود

</button>

<div className="mt-12 flex items-center justify-center gap-8 opacity-0 animate-fade-in-up animation-delay-1000">

<a href="https://www.aparat.com/Amazing.Nurse/" target="\_blank" rel="noopener noreferrer" aria-label="Aparat" className="text-slate-400 hover:text-red-500 transition-colors duration-300">

<AparatIcon className="w-8 h-8"/>

</a>

<a href="https://t.me/ho3in925/" target="\_blank" rel="noopener noreferrer" aria-label="Telegram" className="text-slate-400 hover:text-sky-400 transition-colors duration-300">

<TelegramIcon className="w-8 h-8"/>

</a>

</div>

</div>

</div>

</div>

);

};

export default WelcomeScreen;

components\icons folder files :

AcademicCapIcon.tsx

import React from 'react';

export const AcademicCapIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M4.26 10.147a60.438 60.438 0 0 0-.491 6.347A48.627 48.627 0 0 1 12 20.904a48.627 48.627 0 0 1 8.232-4.41 60.46 60.46 0 0 0-.491-6.347m-15.482 0a50.57 50.57 0 0 0-2.658-.813A59.906 59.906 0 0 1 12 3.493a59.906 59.906 0 0 1 10.399 5.84c-.896.248-1.783.52-2.658.814m-15.482 0A50.697 50.697 0 0 1 12 13.489a50.702 50.702 0 0 1 7.74-3.342M6.75 15a.75.75 0 1 0 0-1.5.75.75 0 0 0 0 1.5Zm0 0v-3.675A55.378 55.378 0 0 1 12 8.443m-7.007 11.55A5.981 5.981 0 0 0 6.75 15.75v-1.5" />

</svg>

);

AiIcon.tsx

import React from 'react';

export const AiIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M9.813 15.904 9 18.75l-.813-2.846a4.5 4.5 0 0 0-3.09-3.09L2.25 12l2.846-.813a4.5 4.5 0 0 0 3.09-3.09L9 5.25l.813 2.846a4.5 4.5 0 0 0 3.09 3.09L15.75 12l-2.846.813a4.5 4.5 0 0 0-3.09 3.09ZM18.259 8.715 18 9.75l-.259-1.035a3.375 3.375 0 0 0-2.455-2.456L14.25 6l1.036-.259a3.375 3.375 0 0 0 2.455-2.456L18 2.25l.259 1.035a3.375 3.375 0 0 0 2.456 2.456L21.75 6l-1.035.259a3.375 3.375 0 0 0-2.456 2.456Z" />

</svg>

);

AparatIcon.tsx

import React from 'react';

export const AparatIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="currentColor" {...props}>

<path d="M5.15 2c-1.74 0-3.15 1.41-3.15 3.15v13.7C2 20.59 3.41 22 5.15 22h13.7c1.74 0 3.15-1.41 3.15-3.15V5.15C22 3.41 20.59 2 18.85 2H5.15zM9.5 16.95V7.05l7.5 4.95-7.5 4.95z" />

</svg>

);

AudioIcon.tsx

import React from 'react';

export const AudioIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M9 9l10.5-3m0 6.553v3.75a2.25 2.25 0 01-1.632 2.163l-1.32.377a1.803 1.803 0 11-.99-3.467l2.31-.66a2.25 2.25 0 001.632-2.163zm0 0V7.5A2.25 2.25 0 0013.5 3h-3a2.25 2.25 0 00-2.25 2.25v.75M9 9h3.75M9 15h3.75M9 15a2.25 2.25 0 01-2.25-2.25v-1.5A2.25 2.25 0 019 9" />

</svg>

);

BackIcon.tsx

import React from 'react';

export const BackIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M10.5 19.5 3 12m0 0 7.5-7.5M3 12h18" />

</svg>

);

BookOpenIcon.tsx

import React from 'react';

export const BookOpenIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M12 6.042A8.967 8.967 0 0 0 6 3.75c-1.052 0-2.062.18-3 .512v14.25A8.987 8.987 0 0 1 6 18c2.305 0 4.408.867 6 2.292m0-14.25a8.966 8.966 0 0 1 6-2.292c1.052 0 2.062.18 3 .512v14.25A8.987 8.987 0 0 0 18 18a8.967 8.967 0 0 0-6 2.292m0-14.25v14.25" />

</svg>

);

ChartBarIcon.tsx

import React from 'react';

export const ChartBarIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M3 13.125C3 12.504 3.504 12 4.125 12h2.25c.621 0 1.125.504 1.125 1.125v6.75C7.5 20.496 6.996 21 6.375 21h-2.25A1.125 1.125 0 0 1 3 19.875v-6.75ZM9.75 8.625c0-.621.504-1.125 1.125-1.125h2.25c.621 0 1.125.504 1.125 1.125v11.25c0 .621-.504 1.125-1.125 1.125h-2.25a1.125 1.125 0 0 1-1.125-1.125V8.625ZM16.5 4.125c0-.621.504-1.125 1.125-1.125h2.25C20.496 3 21 3.504 21 4.125v15.75c0 .621-.504 1.125-1.125 1.125h-2.25a1.125 1.125 0 0 1-1.125-1.125V4.125Z" />

</svg>

);

ChatIcon.tsx

import React from 'react';

export const ChatIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M8.625 12a.375.375 0 1 1-.75 0 .375.375 0 0 1 .75 0Zm0 0H8.25m4.125 0a.375.375 0 1 1-.75 0 .375.375 0 0 1 .75 0Zm0 0H12m4.125 0a.375.375 0 1 1-.75 0 .375.375 0 0 1 .75 0Zm0 0h-.375M21 12c0 4.556-4.03 8.25-9 8.25a9.76 9.76 0 0 1-2.53-.388m-5.168-4.47a2.25 2.25 0 0 1-.64-1.618V7.64c0-1.657 1.343-3 3-3h12.156c1.026 0 1.943.52 2.47 1.332a9.125 9.125 0 0 1 .844 5.046c0 4.136-2.986 7.55-6.812 8.156" />

</svg>

);

ChecklistIcon.tsx

import React from 'react';

export const ChecklistIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M9 12.75 11.25 15 15 9.75M21 12a9 9 0 1 1-18 0 9 9 0 0 1 18 0Z" />

</svg>

);

ClipboardDocumentCheckIcon.tsx

import React from 'react';

export const ClipboardDocumentCheckIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M10.125 2.25h-4.5c-.621 0-1.125.504-1.125 1.125v17.25c0 .621.504 1.125 1.125 1.125h12.75c.621 0 1.125-.504 1.125-1.125v-9M10.125 2.25h.375a9 9 0 0 1 9 9v.375M10.125 2.25A3.375 3.375 0 0 1 13.5 5.625v1.5c0 .621.504 1.125 1.125 1.125h1.5a3.375 3.375 0 0 1 3.375 3.375M9 15l2.25 2.25L15 12" />

</svg>

);

ClipboardDocumentListIcon.tsx

import React from 'react';

export const ClipboardDocumentListIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M19.5 14.25v2.25a3.375 3.375 0 0 1-3.375 3.375h-9.375a3.375 3.375 0 0 1-3.375-3.375v-9.375a3.375 3.375 0 0 1 3.375-3.375h1.5A3.375 3.375 0 0 1 12 6h1.5a3.375 3.375 0 0 1 3.375 3.375v1.5m-3.375-6.375v.625a2.625 2.625 0 0 0 2.625 2.625h.625m-6.375 0V3.375c0-1.036.84-1.875 1.875-1.875h3.375c1.036 0 1.875.84 1.875 1.875v3.375m-6.375 0h6.375m-6.375 3.375h6.375m-6.375 3.375h6.375m0 3.375h6.375m-6.375-3.375v3.375m-6.375-3.375v3.375" />

</svg>

);

DocumentIcon.tsx

import React from 'react';

export const DocumentIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M19.5 14.25v-2.625a3.375 3.375 0 0 0-3.375-3.375h-1.5A1.125 1.125 0 0 1 13.5 7.125v-1.5a3.375 3.375 0 0 0-3.375-3.375H8.25m2.25 0H5.625c-.621 0-1.125.504-1.125 1.125v17.25c0 .621.504 1.125 1.125 1.125h12.75c.621 0 1.125-.504 1.125-1.125V11.25a9 9 0 0 0-9-9Z" />

</svg>

);

EditExcelIcon.tsx

import React from 'react';

export const EditExcelIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M16.862 4.487l1.687-1.688a1.875 1.875 0 112.652 2.652L10.582 16.07a4.5 4.5 0 01-1.897 1.13L6 18l.8-2.685a4.5 4.5 0 011.13-1.897l8.932-8.931zm0 0L19.5 7.125M5.625 21a2.25 2.25 0 01-2.25-2.25V7.5c0-1.242.984-2.25 2.208-2.25h1.092a48.532 48.532 0 011.88-1.5H5.625A2.25 2.25 0 003.375 6v12.75c0 1.242 1.008 2.25 2.25 2.25h12.75c1.242 0 2.25-1.008 2.25-2.25V13.5a48.532 48.532 0 01-1.5 1.88V18.75a2.25 2.25 0 01-2.25 2.25H5.625z" />

</svg>

);

EditIcon.tsx

import React from 'react';

export const EditIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="m16.862 4.487 1.687-1.688a1.875 1.875 0 1 1 2.652 2.652L10.582 16.07a4.5 4.5 0 0 1-1.897 1.13L6 18l.8-2.685a4.5 4.5 0 0 1 1.13-1.897l8.932-8.931Zm0 0L19.5 7.125M18 14v4.75A2.25 2.25 0 0 1 15.75 21H5.25A2.25 2.25 0 0 1 3 18.75V8.25A2.25 2.25 0 0 1 5.25 6H10" />

</svg>

);

ExcelIcon.tsx

import React from 'react';

export const ExcelIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M19.5 14.25v-2.625a3.375 3.375 0 0 0-3.375-3.375h-1.5A1.125 1.125 0 0 1 13.5 7.125v-1.5a3.375 3.375 0 0 0-3.375-3.375H8.25m0 12.75h7.5m-7.5 3H12M10.5 2.25H5.625c-.621 0-1.125.504-1.125 1.125v17.25c0 .621.504 1.125 1.125 1.125h12.75c.621 0 1.125-.504 1.125-1.125V11.25a9 9 0 0 0-9-9Z" />

</svg>

);

HeartbeatIcon.tsx

import React from 'react';

export const HeartbeatIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M2.25 12.75h3.86l1.28-5.122 2.035 10.174 2.896-14.48 2.036 10.179h3.86" />

</svg>

);

HomeIcon.tsx

import React from 'react';

export const HomeIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="m2.25 12 8.954-8.955c.44-.439 1.152-.439 1.591 0L21.75 12M4.5 9.75v10.125c0 .621.504 1.125 1.125 1.125H9.75v-4.875c0-.621.504-1.125 1.125-1.125h2.25c.621 0 1.125.504 1.125 1.125V21h4.125c.621 0 1.125-.504 1.125-1.125V9.75M8.25 21h7.5" />

</svg>

);

ImageIcon.tsx

import React from 'react';

export const ImageIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M2.25 15.75l5.159-5.159a2.25 2.25 0 013.182 0l5.159 5.159m-1.5-1.5l1.409-1.409a2.25 2.25 0 013.182 0l2.909 2.909m-18 3.75h16.5a1.5 1.5 0 001.5-1.5V6a1.5 1.5 0 00-1.5-1.5H3.75A1.5 1.5 0 002.25 6v12a1.5 1.5 0 001.5 1.5zm16.5-1.5H3.75" />

</svg>

);

InfoIcon.tsx

import React from 'react';

export const InfoIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="m11.25 11.25.041-.02a.75.75 0 0 1 1.063.852l-.708 2.836a.75.75 0 0 0 1.063.853l.041-.021M21 12a9 9 0 1 1-18 0 9 9 0 0 1 18 0Zm-9-3.75h.008v.008H12V8.25Z" />

</svg>

);

LogoutIcon.tsx

import React from 'react';

export const LogoutIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M15.75 9V5.25A2.25 2.25 0 0 0 13.5 3h-6a2.25 2.25 0 0 0-2.25 2.25v13.5A2.25 2.25 0 0 0 7.5 21h6a2.25 2.25 0 0 0 2.25-2.25V15m3 0 3-3m0 0-3-3m3 3H9" />

</svg>

);

NewspaperIcon.tsx

import React from 'react';

export const NewspaperIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M12 7.5h1.5m-1.5 3h1.5m-7.5 3h7.5m-7.5 3h7.5m3-9h3.375c.621 0 1.125.504 1.125 1.125V18a2.25 2.25 0 0 1-2.25 2.25M16.5 7.5V18a2.25 2.25 0 0 0 2.25 2.25M16.5 7.5V4.875c0-.621-.504-1.125-1.125-1.125H4.125C3.504 3.75 3 4.254 3 4.875V18a2.25 2.25 0 0 0 2.25 2.25h13.5M6 7.5h3v3H6v-3Z" />

</svg>

);

PaperClipIcon.tsx

import React from 'react';

export const PaperClipIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="m18.375 12.739-7.693 7.693a4.5 4.5 0 0 1-6.364-6.364l10.94-10.94A3 3 0 1 1 19.5 7.372L8.552 18.32m.009-.01-.01.01m5.699-9.941-7.81 7.81a1.5 1.5 0 0 0 2.122 2.122l7.81-7.81" />

</svg>

);

PdfIcon.tsx

import React from 'react';

export const PdfIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M19.5 14.25v-2.625a3.375 3.375 0 0 0-3.375-3.375h-1.5A1.125 1.125 0 0 1 13.5 7.125v-1.5a3.375 3.375 0 0 0-3.375-3.375H8.25m.75 12V9A1.5 1.5 0 0 0 8.25 7.5h-1.5a1.5 1.5 0 0 0-1.5 1.5v7.5m1.5-7.5H5.625c-.621 0-1.125.504-1.125 1.125v17.25c0 .621.504 1.125 1.125 1.125h12.75c.621 0 1.125-.504 1.125-1.125V11.25a9 9 0 0 0-9-9Z" />

</svg>

);

PlusIcon.tsx

import React from 'react';

export const PlusIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M12 4.5v15m7.5-7.5h-15" />

</svg>

);

RefreshIcon.tsx

import React from 'react';

export const RefreshIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M16.023 9.348h4.992v-.001M2.985 19.644v-4.992m0 0h4.992m-4.993 0 3.181 3.183a8.25 8.25 0 0 0 11.664 0l3.181-3.183m-4.991-2.691V5.25a2.25 2.25 0 0 0-2.25-2.25h-6.75a2.25 2.25 0 0 0-2.25 2.25v6.75a2.25 2.25 0 0 0 2.25 2.25h6.75a2.25 2.25 0 0 0 2.25-2.25Z" />

</svg>

);

SaveIcon.tsx

import React from 'react';

export const SaveIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M3 16.5v2.25A2.25 2.25 0 0 0 5.25 21h13.5A2.25 2.25 0 0 0 21 18.75V16.5M16.5 12 12 16.5m0 0L7.5 12m4.5 4.5V3" />

</svg>

);

ShieldCheckIcon.tsx

import React from 'react';

export const ShieldCheckIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M9 12.75 11.25 15 15 9.75m-3-7.036A11.959 11.959 0 0 1 3.598 6 11.99 11.99 0 0 0 3 9.749c0 5.592 3.824 10.29 9 11.622 5.176-1.332 9-6.03 9-11.622 0-1.31-.21-2.571-.598-3.751h-.152c-3.196 0-6.1-1.248-8.25-3.286Z" />

</svg>

);

SunIcon.tsx

import React from 'react';

export const SunIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M12 3v2.25m6.364.386-1.591 1.591M21 12h-2.25m-.386 6.364-1.591-1.591M12 18.75V21m-4.773-4.227-1.591 1.591M5.25 12H3m4.227-4.773L5.636 5.636M15.75 12a3.75 3.75 0 1 1-7.5 0 3.75 3.75 0 0 1 7.5 0Z" />

</svg>

);

TelegramIcon.tsx

import React from 'react';

export const TelegramIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="currentColor" {...props}>

<path d="M9.78 18.65l.28-4.23 7.68-6.92c.34-.31-.07-.46-.52-.19L7.74 13.3 3.64 12c-.88-.25-.89-.86.2-1.3l15.97-6.16c.73-.28 1.4.24 1.15.99l-3.37 15.61c-.23.95-1.05 1.17-1.7.73l-4.54-3.32-2.14 2.05c-.23.22-.42.41-.8.41z"></path>

</svg>

);

ThemeIcon.tsx

import React from 'react';

export const ThemeIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M15.59 14.37a6 6 0 0 1-5.84 7.38v.01l-.001.001-.001.001a6 6 0 0 1-5.84-7.38m5.85-7.38a6 6 0 0 0-5.84 7.38m5.84-7.38a6 6 0 0 1 5.84 7.38m0-7.38L15.59 7" />

<path strokeLinecap="round" strokeLinejoin="round" d="M12 3a9 9 0 1 0 0 18 9 9 0 0 0 0-18Z" />

</svg>

);

TrashIcon.tsx

import React from 'react';

export const TrashIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="m14.74 9-.346 9m-4.788 0L9.26 9m9.968-3.21c.342.052.682.107 1.022.166m-1.022-.165L18.16 19.673a2.25 2.25 0 0 1-2.244 2.077H8.084a2.25 2.25 0 0 1-2.244-2.077L4.772 5.79m14.456 0a48.108 48.108 0 0 0-3.478-.397m-12 .562c.34-.059.68-.114 1.022-.165m0 0a48.11 48.11 0 0 1 3.478-.397m7.5 0v-.916c0-1.18-.91-2.164-2.09-2.201a51.964 51.964 0 0 0-3.32 0c-1.18.037-2.09 1.022-2.09 2.201v.916m7.5 0a48.667 48.667 0 0 0-7.5 0" />

</svg>

);

UploadIcon.tsx

import React from 'react';

export const UploadIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M3 16.5v2.25A2.25 2.25 0 0 0 5.25 21h13.5A2.25 2.25 0 0 0 21 18.75V16.5m-13.5-9L12 3m0 0 4.5 4.5M12 3v13.5" />

</svg>

);

VideoIcon.tsx

import React from 'react';

export const VideoIcon: React.FC<React.SVGProps<SVGSVGElement>> = (props) => (

<svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" {...props}>

<path strokeLinecap="round" strokeLinejoin="round" d="M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />

<path strokeLinecap="round" strokeLinejoin="round" d="M15.91 11.672a.375.375 0 010 .656l-5.603 3.113a.375.375 0 01-.557-.328V8.887c0-.286.307-.466.557-.327l5.603 3.112z" />

</svg>

);

Services folder :

db.ts

const DB\_NAME = 'SkillAssessmentDB';

const DB\_VERSION = 1;

const STORE\_NAME = 'trainingMaterials';

let db: IDBDatabase;

export const initDB = (): Promise<boolean> => {

return new Promise((resolve, reject) => {

if (db) {

return resolve(true);

}

const request = indexedDB.open(DB\_NAME, DB\_VERSION);

request.onerror = (event) => {

console.error('IndexedDB error:', request.error);

reject(false);

};

request.onsuccess = (event) => {

db = request.result;

resolve(true);

};

request.onupgradeneeded = (event) => {

const db = (event.target as IDBOpenDBRequest).result;

if (!db.objectStoreNames.contains(STORE\_NAME)) {

db.createObjectStore(STORE\_NAME, { keyPath: 'id' });

}

};

});

};

export const addMaterial = (material: { id: string; data: string }): Promise<void> => {

return new Promise((resolve, reject) => {

if (!db) {

return reject('DB is not initialized.');

}

const transaction = db.transaction([STORE\_NAME], 'readwrite');

const store = transaction.objectStore(STORE\_NAME);

const request = store.put(material);

request.onsuccess = () => {

resolve();

};

request.onerror = () => {

console.error('Error adding material to DB:', request.error);

reject(request.error);

};

});

};

export const getMaterialData = (id: string): Promise<string | undefined> => {

return new Promise((resolve, reject) => {

if (!db) {

return reject('DB is not initialized.');

}

const transaction = db.transaction([STORE\_NAME], 'readonly');

const store = transaction.objectStore(STORE\_NAME);

const request = store.get(id);

request.onsuccess = () => {

resolve(request.result?.data);

};

request.onerror = () => {

console.error('Error getting material from DB:', request.error);

reject(request.error);

};

});

};

export const deleteMaterial = (id: string): Promise<void> => {

return new Promise((resolve, reject) => {

if (!db) {

return reject('DB is not initialized.');

}

const transaction = db.transaction([STORE\_NAME], 'readwrite');

const store = transaction.objectStore(STORE\_NAME);

const request = store.delete(id);

request.onsuccess = () => {

resolve();

};

request.onerror = () => {

console.error('Error deleting material from DB:', request.error);

reject(request.error);

};

});

};

export const getAllMaterials = (): Promise<{id: string, data: string}[]> => {

return new Promise((resolve, reject) => {

if (!db) {

return reject('DB is not initialized.');

}

const transaction = db.transaction([STORE\_NAME], 'readonly');

const store = transaction.objectStore(STORE\_NAME);

const request = store.getAll();

request.onsuccess = () => {

resolve(request.result);

};

request.onerror = () => {

console.error('Error getting all materials from DB:', request.error);

reject(request.error);

};

});

};

export const clearAllMaterials = (): Promise<void> => {

return new Promise((resolve, reject) => {

if (!db) {

return reject('DB is not initialized.');

}

const transaction = db.transaction([STORE\_NAME], 'readwrite');

const store = transaction.objectStore(STORE\_NAME);

const request = store.clear();

request.onsuccess = () => {

resolve();

};

request.onerror = () => {

console.error('Error clearing materials from DB:', request.error);

reject(request.error);

};

});

}

excelParser.ts

import { SkillCategory, SkillItem } from '../types';

declare const XLSX: any;

/\*\*

\* Parses a single worksheet that contains skill assessment data for multiple months.

\* It expects month names to be in row 4 (from column E to Y) and skill data

\* to start from row 5, with descriptions in column D and scores in the

\* corresponding month columns.

\* @param worksheet The XLSX worksheet object.

\* @returns A Map where keys are month names and values are the skill category data for that month.

\*/

const parseAssessmentsFromSheet = (worksheet: any): Map<string, SkillCategory[]> => {

const json: (string | number | null | undefined)[][] = XLSX.utils.sheet\_to\_json(worksheet, { header: 1 });

const assessments = new Map<string, SkillCategory[]>();

if (json.length < 5) return assessments; // Not enough data for headers and skills

const monthRow = json[3]; // Row 4 (0-indexed)

const monthColumns: { month: string, colIndex: number }[] = [];

// Columns E to Y correspond to indices 4 to 24

for (let i = 4; i <= 24; i++) {

const monthName = monthRow[i];

if (monthName && typeof monthName === 'string' && monthName.trim().length > 0) {

const trimmedMonth = monthName.trim();

monthColumns.push({ month: trimmedMonth, colIndex: i });

// Initialize categories for each found month

assessments.set(trimmedMonth, [

{ name: "مهارت های عمومی", items: [] },

{ name: "مهارت های تخصصی", items: [] },

{ name: "مهارت های ارتباطی", items: [] },

]);

}

}

if (monthColumns.length === 0) {

return assessments; // No valid months found in row 4

}

let currentCategoryIndex = 0;

const categoryNames = ["مهارت های عمومی", "مهارت های تخصصی", "مهارت های ارتباطی"];

// Start from row 5 (index 4)

for (let i = 4; i < json.length; i++) {

const row = json[i];

if (!row || row.length === 0) continue;

const description = row[3] as string; // Column D

const isSeparator = typeof description === 'string' && (description.trim().includes('مجموع') || description.trim().includes('درصد کل'));

if (isSeparator) {

if (currentCategoryIndex < categoryNames.length - 1) {

currentCategoryIndex++;

}

continue;

}

if (description && typeof description === 'string' && description.trim().length > 0) {

monthColumns.forEach(({ month, colIndex }) => {

const scoreValue = row[colIndex];

// Only process the cell if it contains a valid number greater than 0.

// This prevents empty cells and cells with 0 from being incorrectly treated as a registered score.

if (typeof scoreValue === 'number' && isFinite(scoreValue)) {

if (scoreValue > 0 && scoreValue <= 4) {

const skillItem: SkillItem = { description: description.trim(), score: scoreValue };

const monthCategories = assessments.get(month);

if (monthCategories && monthCategories[currentCategoryIndex]) {

monthCategories[currentCategoryIndex].items.push(skillItem);

}

}

}

});

}

}

// After parsing, filter out months that did not have any scores.

// An assessment is only considered "registered" if at least one skill has a score.

const validAssessments = new Map<string, SkillCategory[]>();

for (const [month, categories] of assessments.entries()) {

const hasAnyItems = categories.some(category => category.items.length > 0);

if (hasAnyItems) {

validAssessments.set(month, categories);

}

}

return validAssessments;

};

export const parseExcelData = (file: File): Promise<Map<string, SkillCategory[]>> => {

return new Promise((resolve, reject) => {

const reader = new FileReader();

reader.onload = (e) => {

try {

const data = new Uint8Array(e.target?.result as ArrayBuffer);

const workbook = XLSX.read(data, { type: 'array' });

const sheetName = workbook.SheetNames[0];

const worksheet = workbook.Sheets[sheetName];

const assessments = parseAssessmentsFromSheet(worksheet);

if (assessments.size === 0) {

throw new Error("داده ارزیابی معتبری یافت نشد. اطمینان حاصل کنید که نام ماه‌ها در ردیف 4 (از ستون E تا Y) و داده‌های مهارت از ردیف 5 به بعد قرار دارند.");

}

resolve(assessments);

} catch (error) {

console.error("Error parsing Excel file:", error);

reject(error instanceof Error ? error : new Error("Failed to parse Excel file."));

}

};

reader.onerror = (error) => reject(error);

reader.readAsArrayBuffer(file);

});

};

export const parseComprehensiveExcel = (file: File): Promise<{ [staffName: string]: Map<string, SkillCategory[]> }> => {

return new Promise((resolve, reject) => {

const reader = new FileReader();

reader.onload = (e) => {

try {

const data = new Uint8Array(e.target?.result as ArrayBuffer);

const workbook = XLSX.read(data, { type: 'array' });

const allStaffData: { [staffName: string]: Map<string, SkillCategory[]> } = {};

for (const sheetName of workbook.SheetNames) {

const worksheet = workbook.Sheets[sheetName];

const staffAssessments = parseAssessmentsFromSheet(worksheet);

if (staffAssessments.size > 0) {

allStaffData[sheetName.trim()] = staffAssessments;

}

}

if (Object.keys(allStaffData).length === 0) {

throw new Error("فایل اکسل جامع معتبر نیست. اطمینان حاصل کنید که نام هر شیت مطابق با نام پرسنل باشد و داده‌های ارزیابی در آن وجود داشته باشد.");

}

resolve(allStaffData);

} catch (error) {

console.error("Error parsing comprehensive Excel file:", error);

reject(error instanceof Error ? error : new Error("Failed to parse comprehensive Excel file."));

}

};

reader.onerror = (error) => reject(error);

reader.readAsArrayBuffer(file);

});

};

geminiService.ts

import { StaffMember, SkillItem } from '../types';

export const generateImprovementPlan = (

staff: StaffMember,

weakSkillsByCategory: { categoryName: string; skills: SkillItem[] }[],

supervisorMessage?: string,

managerMessage?: string

): string => {

const allWeakSkills = weakSkillsByCategory.flatMap(category =>

category.skills.map(skill => ({ description: skill.description, category: category.categoryName }))

);

if (allWeakSkills.length === 0) {

return "هیچ نقطه ضعفی برای ایجاد برنامه یافت نشد.";

}

// Section 1: List of skills to improve

const skillList = weakSkillsByCategory

.filter(cat => cat.skills.length > 0)

.map(category => {

const skills = category.skills.map(skill => `- ${skill.description}`).join('\n');

return `\*\*${category.categoryName}\*\*\n${skills}`;

})

.join('\n\n');

// Section 2: 30-day study plan

let studyPlan = '';

for (let i = 0; i < 30; i++) {

const skillToStudy = allWeakSkills[i % allWeakSkills.length];

studyPlan += `\*\*روز ${i + 1}:\*\* مطالعه و تمرین در مورد "${skillToStudy.description}" (از دسته ${skillToStudy.category})\n`;

}

const plan = `

\*\*${staff.name} عزیز\*\*، طبق نتایج استخراج شده از آخرین ارزیابی شما، برنامه آموزشی پیشنهادی به شرح زیر می‌باشد.

\n

\*\*موارد نیازمند بهبود\*\*

${skillList}

\n\n

\*\*برنامه مطالعه ماه پیش رو\*\*

در ماه پیش رو، پیشنهاد می‌شود طبق جدول زمان‌بندی زیر بر روی موارد مشخص شده تمرکز کنید:

${studyPlan}

`;

let finalPlan = plan.trim();

if (supervisorMessage && supervisorMessage.trim()) {

finalPlan += `

\n\n---

\n\n\*\*پیام سوپروایزر آموزشی:\*\*\n\n<QUOTE>${supervisorMessage}</QUOTE>

`;

}

if (managerMessage && managerMessage.trim()) {

finalPlan += `

\n\n---

\n\n\*\*پیام مسئول بخش:\*\*\n\n<QUOTE>${managerMessage}</QUOTE>

`;

}

return finalPlan;

};

Hooks folder :

useLocalStorage.ts

import { useState, useEffect } from 'react';

function useLocalStorage<T>(key: string, initialValue: T): [T, React.Dispatch<React.SetStateAction<T>>] {

const [storedValue, setStoredValue] = useState<T>(() => {

try {

const item = window.localStorage.getItem(key);

return item ? JSON.parse(item) : initialValue;

} catch (error) {

console.error(error);

return initialValue;

}

});

// Effect to update localStorage when the state changes

useEffect(() => {

try {

window.localStorage.setItem(key, JSON.stringify(storedValue));

} catch (error) {

console.error(`Error setting localStorage key “${key}”:`, error);

}

}, [key, storedValue]);

// Effect to listen for changes from other tabs

useEffect(() => {

const handleStorageChange = (e: StorageEvent) => {

if (e.key === key) {

try {

const newValue = e.newValue ? JSON.parse(e.newValue) : initialValue;

// Use updater function form to safely handle setting functions in state

setStoredValue(() => newValue);

} catch (error) {

console.error(error);

// If parsing fails, fall back to initial value

setStoredValue(() => initialValue);

}

}

};

window.addEventListener('storage', handleStorageChange);

return () => {

window.removeEventListener('storage', handleStorageChange);

};

}, [key, initialValue]);

return [storedValue, setStoredValue];

}

export default useLocalStorage;