"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfo.jsx"

import React, { useState, useEffect, useContext } from 'react';  
import { BabyContext } from '../../context/BabyContext';  
  
import TitleSub from '../TitleSub';  
import BabyInfoAddNewRecord from './BabyInfoAddNewRecord';  
import BabyInfoSearch from './BabyInfoSearch';  
import './BabyInfoSearch.css';  
import BabyInfoTranslate from './BabyInfoTranslate';  
import BabyInfoIdentify from './BabyInfoIdentify';  
import BabyInfoGA from './BabyInfoGA';  
import BabyInfoBio from './BabyInfoBio';  
import BabyInfoSave from './BabyInfoSave';  
  
const BabyInfo = () => {  
 const { updateBabyData } = useContext(BabyContext);  
  
 const savedData = JSON.parse(localStorage.getItem("selectedBaby"));  
 const defaultData = {  
 babyName: "",  
 babyNameArabic: "",  
 babyMRN: "",  
 visitNumber: "",  
 gender: "",  
 birthWeight: "",  
 dob: "",  
 gaWeeks: *0*,  
 gaDays: *0*,  
 passportId: "",  
 personalId: "",  
 birthCertificateId: "",  
 };  
  
 // Initialize state here, NOT inside useEffect  
 const [localData, setLocalData] = useState(savedData || defaultData);  
  
 // useEffect should be used to perform side effects like storing data into localStorage  
 useEffect(() => {  
 localStorage.setItem("activeComponent", "BabyInfo");  
 }, []);  
  
 return (  
 <div style={{ padding: '10px' }}>  
 <TitleSub sectionTitle="Baby Information" />  
  
 {/\* Add New Record + Search in the same line \*/}  
 <div style={{ display: 'flex', justifyContent: 'space-between', marginBottom: '15px' }}>  
 <BabyInfoAddNewRecord setLocalData={setLocalData} />  
 <BabyInfoSearch setLocalData={setLocalData} /> {/\* Search is inside BabyInfo \*/}  
 </div>  
  
 <BabyInfoTranslate localData={localData} setLocalData={setLocalData} />  
 <BabyInfoIdentify localData={localData} setLocalData={setLocalData} />  
 <BabyInfoGA localData={localData} setLocalData={setLocalData} />  
 <BabyInfoBio localData={localData} setLocalData={setLocalData} />  
 <BabyInfoSave localData={localData} updateBabyData={updateBabyData} />  
 </div>  
 );  
};  
  
export default BabyInfo;

"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoBio.jsx"

import React, { useContext } from 'react';  
import { BabyContext } from '../../context/BabyContext';  
  
const BabyInfoBio = () => {  
 const { babyData, updateBabyData } = useContext(BabyContext);  
  
 return (  
 <div>  
 <label>Gender:</label>  
 <select  
 value={babyData.gender || ''}  
 onChange={(e) => updateBabyData({ gender: e.target.value })}  
 >  
 <option value="">Select Gender</option>  
 <option value="Male">Male</option>  
 <option value="Female">Female</option>  
 <option value="Others">Others; unassigned (ambiguous)</option>  
 </select>  
  
 <label>Birth Weight:</label>  
 <select  
 value={babyData.birthWeight || ''}  
 onChange={(e) => updateBabyData({ birthWeight: e.target.value })}  
 >  
 <option value="">Select Weight</option>  
 {Array.from({ length: *5651* }, (\_, i) => *350* + i).map((weight) => (  
 <option key={weight} value={weight}>  
 {weight} grams  
 </option>  
 ))}  
 </select>  
 </div>  
 );  
};  
  
export default BabyInfoBio;

"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoIdentify.jsx"

import React, { useEffect, useContext } from "react";  
import { BabyContext } from "../../context/BabyContext";  
  
const BabyInfoIdentify = () => {  
 const { babyData, updateBabyData } = useContext(BabyContext);  
  
 // Auto-generate Visit Number based on MRN  
 useEffect(() => {  
 if (babyData.babyMRN && !babyData.visitNumber) {  
 const now = new Date();  
 const timestamp = now.toISOString().replace(/*[-:.TZ]*/g, "").slice(*0*, *12*);  
 const generatedVisit = `${babyData.babyMRN}-${timestamp}`;  
 updateBabyData({ visitNumber: generatedVisit });  
 }  
 }, [babyData.babyMRN, babyData.visitNumber, updateBabyData]);  
  
 return (  
 <div>  
 <label>MRN:</label>  
 <input  
 type="text"  
 value={babyData.babyMRN || ""}  
 onChange={(e) =>  
 updateBabyData({  
 babyMRN: e.target.value,  
 visitNumber: "", // Reset visit number when MRN changes  
 })  
 }  
 />  
  
 <label>Visit Number:</label>  
 <input type="text" value={babyData.visitNumber || ""} readOnly />  
  
 <label>Passport / Personal ID:</label>  
 <input  
 type="text"  
 value={babyData.passportId || ""}  
 onChange={(e) =>  
 updateBabyData({ passportId: e.target.value })  
 }  
 />  
  
 <label>Birth Certificate ID:</label>  
 <input  
 type="text"  
 value={babyData.birthCertificateId || ""}  
 onChange={(e) =>  
 updateBabyData({ birthCertificateId: e.target.value })  
 }  
 />  
 </div>  
 );  
};  
  
export default BabyInfoIdentify;

"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoGA.jsx"

import React, { useContext } from 'react';  
import { BabyContext } from '../../context/BabyContext';  
  
const BabyInfoGA = () => {  
 const { babyData, updateBabyData } = useContext(BabyContext);  
 const currentDate = new Date().toISOString().split('T')[*0*];  
  
 const calculateDOL = () => {  
 if (!babyData.dob) return '';  
 const dobDate = new Date(babyData.dob);  
 const currentDateObj = new Date();  
 const diffTime = currentDateObj - dobDate;  
 const diffDays = Math.floor(diffTime / (*1000* \* *3600* \* *24*));  
 return diffDays >= *0* ? `${diffDays} Days` : 'Invalid Date';  
 };  
  
 const calculateCGA = () => {  
 if (!babyData.dob || !babyData.gaWeeks || !babyData.gaDays) return '';  
 const dol = parseInt(calculateDOL().split(' ')[*0*], *10*);  
 if (isNaN(dol)) return '';  
 const totalWeeks = babyData.gaWeeks + Math.floor(dol / *7*);  
 const totalDays = (babyData.gaDays + (dol % *7*)) % *7*;  
 return `${totalWeeks} Weeks ${totalDays} Days`;  
 };  
  
 return (  
 <div>  
 <label>Date of Birth (DOB):</label>  
 <input  
 type="date"  
 value={babyData.dob || ''}  
 onChange={(e) => updateBabyData({ dob: e.target.value })}  
 />  
  
 <label>Current Date:</label>  
 <input  
 type="date"  
 value={currentDate}  
 readOnly  
 />  
  
 <label>Days of Life (DOL):</label>  
 <input  
 type="text"  
 value={calculateDOL()}  
 readOnly  
 />  
  
 <label>Gestational Age at Birth:</label>  
 <div style={{ display: 'flex', gap: '10px' }}>  
 <select  
 value={babyData.gaWeeks !== *22* ? babyData.gaWeeks : ""}  
 onChange={(e) => updateBabyData({ gaWeeks: Number(e.target.value) })}  
 >  
 <option value="">Select Week</option>  
 {Array.from({ length: *34* }, (\_, i) => *22* + i).map((week) => (  
 <option key={week} value={week}>{week} Weeks</option>  
 ))}  
 </select>  
 <select  
 value={babyData.gaDays || ""}  
 onChange={(e) => updateBabyData({ gaDays: Number(e.target.value) })}  
 >  
 <option value="">Select Day</option>  
 {Array.from({ length: *7* }, (\_, i) => i).map((day) => (  
 <option key={day} value={day}>{day} Days</option>  
 ))}  
 </select>  
 </div>  
  
 <label>Corrected Gestational Age (CGA):</label>  
 <input  
 type="text"  
 value={calculateCGA()}  
 readOnly  
 />  
 </div>  
 );  
};  
  
export default BabyInfoGA;

"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoSave.jsx"

import React, { useContext, useState } from 'react';  
import axios from 'axios';  
import { BabyContext } from '../../context/BabyContext';  
  
const BabyInfoSave = () => {  
 const { babyData, updateBabyData } = useContext(BabyContext);  
 const [saved, setSaved] = useState(*false*);  
  
 const calculateDOL = () => {  
 if (!babyData.dob) return '';  
 const dobDate = new Date(babyData.dob);  
 const currentDateObj = new Date();  
 const differenceInTime = currentDateObj - dobDate;  
 const differenceInDays = Math.floor(differenceInTime / (*1000* \* *3600* \* *24*));  
 return differenceInDays >= *0* ? `${differenceInDays} Days` : 'Invalid Date';  
 };  
  
 const calculateCGA = () => {  
 if (!babyData.dob || !babyData.gaWeeks || !babyData.gaDays) return '';  
 const dol = parseInt(calculateDOL().split(' ')[*0*], *10*);  
 if (isNaN(dol)) return '';  
 const totalWeeks = babyData.gaWeeks + Math.floor(dol / *7*);  
 const totalDays = (babyData.gaDays + (dol % *7*)) % *7*;  
 return `${totalWeeks} Weeks ${totalDays} Days`;  
 };  
  
 const handleSave = async () => {  
 const requiredFields = [  
 { field: "babyName", label: "Baby Name" },  
 { field: "babyMRN", label: "Baby MRN" },  
 { field: "visitNumber", label: "Visit Number" },  
 { field: "gender", label: "Gender" },  
 { field: "birthWeight", label: "Birth Weight" },  
 { field: "dob", label: "Date of Birth (DOB)" },  
 { field: "gaWeeks", label: "Gestational Age Weeks" },  
 { field: "gaDays", label: "Gestational Age Days" }  
 ];  
  
 const emptyField = requiredFields.find(({ field }) => !babyData[field]);  
  
 if (emptyField) {  
 alert(`Please complete the record: ${emptyField.label}`);  
 return;  
 }  
  
 const updatedData = {  
 ...babyData,  
 dol: calculateDOL(),  
 cga: calculateCGA()  
 };  
  
 try {  
 const response = await axios.post("http://localhost:5000/api/babies", updatedData);  
 updateBabyData(response.data);  
 setSaved(*true*);  
 alert("Record saved to backend successfully!");  
  
 // 👇 Clear the form after saving  
 updateBabyData({  
 babyName: "",  
 babyNameArabic: "",  
 babyMRN: "",  
 visitNumber: "",  
 gender: "",  
 birthWeight: "",  
 dob: "",  
 gaWeeks: *0*,  
 gaDays: *0*,  
 passportId: "",  
 personalId: "",  
 birthCertificateId: ""  
 });  
  
 } catch (error) {  
 console.error("Save failed:", error);  
 alert("Failed to save to backend.");  
 }  
 };  
  
 return (  
 <div>  
 <button onClick={handleSave}>Save Record</button>  
 {saved && <p style={{ color: 'green' }}>Successfully Saved</p>}  
 </div>  
 );  
};  
  
export default BabyInfoSave;

"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoAddNewRecord.jsx"

import React from "react";  
import axios from "axios";  
  
const BabyInfoAddNewRecord = ({ setLocalData }) => {  
 const handleAddNew = async () => {  
 const proceed = window.confirm(  
 "This will clear the current data. Do you want to proceed?"  
 );  
 if (!proceed) return;  
  
 // ✅ Corrected emptyData to match backend schema types  
 const emptyData = {  
 babyName: "",  
 babyNameArabic: "",  
 babyMRN: "",  
 visitNumber: "",  
 gender: "",  
 birthWeight: *0*, // 🟢 Changed from "" to 0 (number)  
 dob: "",  
 gaWeeks: *0*,  
 gaDays: *0*,  
 passportId: "",  
 personalId: "",  
 birthCertificateId: "",  
 };  
  
 try {  
 const response = await axios.post("http://localhost:5000/api/babies", emptyData);  
 console.log("New baby record added:", response.data);  
  
 // Clear local storage  
 localStorage.removeItem("selectedBaby");  
 localStorage.removeItem("babyName");  
 localStorage.removeItem("babyMRN");  
  
 setLocalData(emptyData);  
 window.alert("New record initiated successfully.");  
 } catch (error) {  
 console.error("Error adding new record:", error);  
 window.alert(`Failed: ${error.response?.data?.message || "Server error"}`);  
 }  
 };  
  
 return (  
 <button  
 onClick={handleAddNew}  
 style={{  
 backgroundColor: "green", // 🟢 Restored original color  
 color: "white",  
 border: "none",  
 padding: "8px 16px",  
 marginRight: "10px",  
 cursor: "pointer",  
 borderRadius: "5px",  
 }}  
 >  
 Add New Record  
 </button>  
 );  
};  
  
export default BabyInfoAddNewRecord;

"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoSearch.jsx"

import React, { useState } from "react";  
import axios from "axios";  
  
const BabyInfoSearch = ({ setLocalData }) => {  
 const [showSearchBox, setShowSearchBox] = useState(*false*);  
 const [query, setQuery] = useState("");  
  
 const handleSearchClick = () => {  
 const proceed = window.confirm(  
 "Searching a new record may override unsaved data. Proceed?"  
 );  
 setShowSearchBox(proceed); // ✅ Simplified logic  
 };  
  
 const handleSearch = async () => {  
 if (!query.trim()) {  
 alert("Please enter a search term.");  
 return;  
 }  
  
 try {  
 const response = await axios.get(`http://localhost:5000/api/babies/search`, {  
 params: { query },  
 });  
  
 const baby = response.data;  
  
 // 🟢 START OF CHANGED SECTION 🟢  
 if (baby) {  
 // Clear old data before setting new values  
 localStorage.removeItem("selectedBaby");  
 localStorage.removeItem("babyName");  
 localStorage.removeItem("babyMRN");  
  
 localStorage.setItem("selectedBaby", JSON.stringify(baby));  
 localStorage.setItem("babyName", baby.babyName || "");  
 localStorage.setItem("babyMRN", baby.babyMRN || "");  
  
 setLocalData(baby);  
 alert("Record found.");  
 } else {  
 // Clear data when no match found  
 localStorage.removeItem("selectedBaby");  
 setLocalData(*null*);  
 alert("No matching record found.");  
 }  
 // 🟢 END OF CHANGED SECTION 🟢  
  
 } catch (error) {  
 // 🟢 Improved error handling  
 alert(error.response?.data?.message || "Search failed. Check console.");  
 console.error("Search error:", error.response?.data || error);  
 }  
  
 setShowSearchBox(*false*);  
 };  
  
 // Rest of the file remains unchanged ▼▼▼  
 const handleCloseSearchBox = () => {  
 setShowSearchBox(*false*);  
 };  
  
 return (  
 <>  
 <button  
 onClick={handleSearchClick}  
 style={{  
 backgroundColor: "#007bff",  
 color: "white",  
 border: "none",  
 padding: "8px 16px",  
 cursor: "pointer",  
 borderRadius: "5px",  
 }}  
 >  
 Search Record  
 </button>  
  
 {showSearchBox && (  
 <div style={{ /\* existing popup styles \*/ }}>  
 {/\* existing popup content \*/}  
 </div>  
 )}  
 </>  
 );  
};  
  
export default BabyInfoSearch;

"C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoSearch.css"

/\* C:\Users\yahya\Halebi2\Frontend\src\components\BabyInfo\BabyInfoSearch.css \*/  
  
.baby-search-button {  
 padding: 6px 12px;  
 background-color: #0078D4;  
 color: white;  
 border: none;  
 border-radius: 4px;  
 margin-left: 10px;  
 cursor: pointer;  
 font-size: 12px;  
}  
  
.baby-search-popup {  
 position: fixed;  
 top: 20%;  
 left: 50%;  
 transform: translate(-50%, -20%);  
 background-color: #f9f9f9;  
 padding: 15px;  
 border: 2px solid #0078D4;  
 border-radius: 6px;  
 box-shadow: 0 0 12px rgba(0, 0, 0, 0.25);  
 z-index: 9999;  
 width: 300px;  
}  
  
.baby-search-input {  
 width: 100%;  
 padding: 8px;  
 margin-bottom: 10px;  
 font-size: 14px;  
 border: 1px solid #ccc;  
 border-radius: 4px;  
}  
  
.baby-search-find {  
 width: 100%;  
 padding: 6px;  
 background-color: #0078D4;  
 color: white;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
 font-size: 13px;  
}

"C:\Users\yahya\Halebi2\Backend\models\baby.js"

const mongoose = require('mongoose');

const babySchema = new mongoose.Schema({

babyName: { type: String, required: false },

babyNameArabic: { type: String, required: false },

babyMRN: { type: String, required: false },

visitNumber: { type: String, required: false },

gender: { type: String, required: false },

birthWeight: { type: Number, required: false },

dob: { type: Date, required: false },

gaWeeks: { type: Number, required: false },

gaDays: { type: Number, required: false },

passportId: { type: String, required: false },

personalId: { type: String, required: false },

birthCertificateId: { type: String, required: false },

});

module.exports = mongoose.model('Baby', babySchema);

"C:\Users\yahya\Halebi2\Backend\routes\babyRoutes.js"

const express = require('express');

const router = express.Router();

const Baby = require('../models/baby');

// ✅ Fixed: Remove "|| null" for numerical fields

router.post("/", async (req, res) => {

try {

const newBaby = new Baby({

babyName: req.body.babyName || "",

babyNameArabic: req.body.babyNameArabic || "",

babyMRN: req.body.babyMRN || "",

visitNumber: req.body.visitNumber || "",

gender: req.body.gender || "",

birthWeight: req.body.birthWeight, // 🟢 Direct assignment (no default)

dob: req.body.dob || "",

gaWeeks: req.body.gaWeeks,

gaDays: req.body.gaDays,

passportId: req.body.passportId || "",

personalId: req.body.personalId || "",

birthCertificateId: req.body.birthCertificateId || "",

});

await newBaby.save();

res.status(201).json(newBaby);

} catch (error) {

console.error("Save error:", error);

res.status(500).json({ message: "Failed to save baby record." });

}

});

module.exports = router;

"C:\Users\yahya\Halebi2\Backend\server.js"

const express = require("express");

const connectDB = require('./config/db');

const cors = require("cors");

const babyRoutes = require('./routes/babyRoutes');

const motherRoutes = require('./routes/motherRoutes');

const adminRoutes = require('./routes/adminRoutes');

const nurseRoutes = require('./routes/nurseRoutes');

const parentRoutes = require('./routes/parentRoutes');

const facePhotoRoutes = require('./routes/facePhotoRoutes');

const footPrintRoutes = require('./routes/footPrintRoutes');

const retinaPrintRoutes = require('./routes/retinaPrintRoutes');

const motherFingerPrintRoutes = require('./routes/motherFingerPrintRoutes');

const motherIDRoutes = require('./routes/motherIDRoutes');

const qrCodeRoutes = require('./routes/qrCodeRoutes');

connectDB();

const app = express();

const PORT = 5000;

app.use(cors());

app.use(express.json());

app.use("/api/babies", babyRoutes);

app.use("/api/mothers", motherRoutes);

app.use("/api/admins", adminRoutes);

app.use("/api/nurses", nurseRoutes);

app.use("/api/parents", parentRoutes);

app.use("/api/facephotos", facePhotoRoutes);

app.use("/api/footprints", footPrintRoutes);

app.use("/api/retinaprints", retinaPrintRoutes);

app.use("/api/motherfingerprints", motherFingerPrintRoutes);

app.use("/api/motherids", motherIDRoutes);

app.use("/api/qrcodes", qrCodeRoutes);

app.get("/", (req, res) => {

res.send("Backend is running successfully.");

});

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});