**Record management Module (RMM)**

**Create Classroom Card:**

"use client"

import { Button } from "@/components/ui/button"

import {

    Card,

    CardContent,

    CardDescription,

    CardFooter,

    CardHeader,

    CardTitle

} from "@/components/ui/card"

import { Input } from "@/components/ui/input"

import { Label } from "@/components/ui/label"

import {

    Select,

    SelectContent,

    SelectItem,

    SelectTrigger,

    SelectValue

} from "@/components/ui/select"

import { Textarea } from "@/components/ui/textarea"

import { trackOptions } from "@/lib/constants"

import { useConvexMutation } from "@convex-dev/react-query"

import { zodResolver } from "@hookform/resolvers/zod"

import { useMutation } from "@tanstack/react-query"

import { useQuery } from "convex/react"

import { useForm } from "react-hook-form"

import { toast } from "sonner"

import { z } from "zod"

import { api } from "../../../../../../convex/\_generated/api"

import { Id } from "../../../../../../convex/\_generated/dataModel"

import { useState } from "react"

export const roomTypes = ["REGULAR", "LABORATORY", "COMPUTER\_LABORATORY"] as const;

export const roomSchema = z.object({

    name: z.string().min(1, "Room name is required"),

    capacity: z.coerce.number().min(1, "Capacity must be at least 1"),

    type: z.enum(["REGULAR", "LABORATORY", "COMPUTER\_LABORATORY"]),

    teacherId: z.string().min(1, "Teacher is required"),

    description: z.string().optional(),

    features: z.array(z.string()).optional(),

    gradeLevel: z.string().min(1, "Grade level is required").optional(),

    track: z.enum(["core", "academic", "immersion", "tvl", "sports", "arts"]).optional(),

});

export type RoomFormData = z.infer<typeof roomSchema>;

export const AddClassRoomCard = () => {

    const [selectedGradeLevelId, setSelectedGradeLevelId] = useState<Id<"gradeLevels"> | null>(null);

    const teachers = useQuery(api.classroom.getTeachers);

    const gradeLevels = useQuery(api.gradeLevel.get);

    const { register, handleSubmit, watch, setValue, formState: { errors } } = useForm<RoomFormData>({

        resolver: zodResolver(roomSchema)

    });

    const { mutate: createRoom, isPending } = useMutation({

        mutationFn: useConvexMutation(api.classroom.create),

        onSuccess: () => {

            toast.success("Classroom created successfully");

        },

        onError: (error) => {

            toast.error(error.message);

        }

    });

    const handleTeacherSelect = (teacherId: string) => {

        const selectedTeacher = teachers?.find(t => t.\_id === teacherId);

        const gradeLevel = watch("gradeLevel")

        if (selectedTeacher) {

            setValue("teacherId", teacherId);

            setValue("name", `${gradeLevel}-${selectedTeacher.lastName}`);

        }

    };

    const handleGradeLevelSelect = (level: string) => {

        const teacher = teachers?.find(t => t.\_id === watch("teacherId"));

        const gradeLevel = gradeLevels?.find(g => g.level === level)

        if (teacher && gradeLevel) {

            setValue("gradeLevel", level);

            setValue("name", `${level}-${teacher.lastName}`);

            setSelectedGradeLevelId(gradeLevel.\_id as Id<"gradeLevels">)

        }

    };

    const handleTrackSelect = (track: "core" | "academic" | "immersion" | "tvl" | "sports" | "arts") => {

        const teacher = teachers?.find(t => t.\_id === watch("teacherId"));

        const gradeLevel = watch("gradeLevel");

        if (teacher && (gradeLevel?.startsWith("Grade 11") || gradeLevel?.startsWith("Grade 12"))) {

            setValue("name", `${gradeLevel}-${track}-${teacher.lastName}`);

        }

    };

    return (

        <Card className="flex flex-col h-fit">

            <CardHeader>

                <CardTitle>Add Class Room</CardTitle>

                <CardDescription>

                    Fill out the form to add a room

                </CardDescription>

            </CardHeader>

            <form onSubmit={

                handleSubmit(data => {

                    // Remove gradeLevel from data since it's just for display

                    /\* eslint-disable-next-line @typescript-eslint/no-unused-vars \*/

                    const { gradeLevel, ...submitData } = data;

                    createRoom({

                        ...submitData,

                        gradeLevelId: selectedGradeLevelId ?? undefined,

                        teacherId: data.teacherId as Id<"users">

                    });

                })

            }>

                <CardContent className="grid gap-8 mt-7">

                    <div className="grid gap-2">

                        <Label>Assign Teacher</Label>

                        <Select onValueChange={handleTeacherSelect}>

                            <SelectTrigger>

                                <SelectValue placeholder="Select teacher" />

                            </SelectTrigger>

                            <SelectContent>

                                {teachers?.map(teacher => (

                                    <SelectItem key={teacher.\_id} value={teacher.\_id}>

                                        {teacher.lastName}, {teacher.firstName}

                                    </SelectItem>

                                ))}

                            </SelectContent>

                        </Select>

                        {errors.teacherId && (

                            <p className="text-sm text-red-500">{errors.teacherId.message}</p>

                        )}

                    </div>

                    <div className="grid gap-2">

                        <Label>Grade Level</Label>

                        <Select onValueChange={handleGradeLevelSelect}>

                            <SelectTrigger>

                                <SelectValue placeholder="Select grade level" />

                            </SelectTrigger>

                            <SelectContent>

                                {gradeLevels?.map((grade) => (

                                    <SelectItem key={grade.\_id} value={grade.level}>

                                        {grade.level}

                                    </SelectItem>

                                ))}

                            </SelectContent>

                        </Select>

                        {errors.gradeLevel && (

                            <p className="text-sm text-red-500">{errors.gradeLevel.message}</p>

                        )}

                    </div>

                    {(watch("gradeLevel")?.startsWith("Grade 11") || watch("gradeLevel")?.startsWith("Grade 12")) && (

                        <div className="grid gap-2">

                            <Label>Track</Label>

                            <Select onValueChange={handleTrackSelect}>

                                <SelectTrigger>

                                    <SelectValue placeholder="Select track" />

                                </SelectTrigger>

                                <SelectContent>

                                    {trackOptions.map((track) => (

                                        <SelectItem key={track.value} value={track.value}>

                                            {track.label}

                                        </SelectItem>

                                    ))}

                                </SelectContent>

                            </Select>

                        </div>

                    )}

                    <div className="grid gap-2">

                        <Label>Room Name</Label>

                        <Input {...register("name")} disabled />

                        {errors.name && (

                            <p className="text-sm text-red-500">{errors.name.message}</p>

                        )}

                    </div>

                    <div className="grid gap-2">

                        <Label>Room Type</Label>

                        <Select onValueChange={(value) => setValue("type", value as typeof roomTypes[number])}>

                            <SelectTrigger>

                                <SelectValue placeholder="Select room type" />

                            </SelectTrigger>

                            <SelectContent>

                                {roomTypes.map((type) => (

                                    <SelectItem key={type} value={type}>

                                        {type.replace("\_", " ")}

                                    </SelectItem>

                                ))}

                            </SelectContent>

                        </Select>

                        {errors.type && (

                            <p className="text-sm text-red-500">{errors.type.message}</p>

                        )}

                    </div>

                    <div className="grid gap-2">

                        <Label>Capacity</Label>

                        <Input type="number" {...register("capacity")} />

                        {errors.capacity && (

                            <p className="text-sm text-red-500">{errors.capacity.message}</p>

                        )}

                    </div>

                    <div className="grid gap-2">

                        <Label>Description (Optional)</Label>

                        <Textarea {...register("description")} />

                    </div>

                </CardContent>

                <CardFooter>

                    <Button type="submit" disabled={isPending} className="text-white">

                        {isPending ? "Creating..." : "Create Classroom"}

                    </Button>

                </CardFooter>

            </form>

        </Card>

    )

}

**Create grade level card:**

"use client"

import { Button } from "@/components/ui/button"

import {

    Card,

    CardContent,

    CardDescription,

    CardFooter,

    CardHeader,

    CardTitle

} from "@/components/ui/card"

import { Input } from "@/components/ui/input"

import { Label } from "@/components/ui/label"

import { useConvexMutation } from "@convex-dev/react-query"

import { useMutation } from "@tanstack/react-query"

import { useState } from "react"

import { api } from "../../../../../../convex/\_generated/api"

import { toast } from "sonner"

export const AddGradeLevelCard = () => {

    const [gradeLevel, setGradeLevel] = useState("")

    const { mutate, isPending, isError } = useMutation({

        mutationFn: useConvexMutation(api.gradeLevel.create)

    })

    const onSubmit = async () => {

        if (gradeLevel.length <= 1) return toast.error("Invalid grade level")

        await mutate({

            gradeLevel

        })

        if (isError) return toast.error("Grade level already exists")

        setGradeLevel("")

        toast.success("Grade level created!")

    }

    return (

        <Card className="flex flex-col h-fit">

            <CardHeader>

                <CardTitle>Add Grade Level</CardTitle>

                <CardDescription>

                    Fill out the form to add a new grade level

                </CardDescription>

            </CardHeader>

            <CardContent className="grid gap-8 mt-7">

                <div className="grid gap-2">

                    <Label htmlFor="gradeLevel">Grade Level</Label>

                    <Input

                        id="gradeLevel"

                        type="text"

                        placeholder="Ex: Grade 11"

                        className="w-full"

                        disabled={isPending}

                        value={gradeLevel}

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

                    />

                </div>

            </CardContent>

            <CardFooter className="mt-auto">

                <Button

                    className="text-black"

                    onClick={onSubmit}

                    disabled={isPending}

                >

                    {isPending ? "Adding..." : "Add"}

                </Button>

            </CardFooter>

        </Card>

    )

}

**Intervention Settings page:**

"use client"

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Textarea } from "@/components/ui/textarea";

import { useConvexMutation } from "@convex-dev/react-query";

import { useMutation } from "@tanstack/react-query";

import { useQuery } from "convex/react";

import { useState } from "react";

import { api } from "../../../../../../convex/\_generated/api";

import { Id } from "../../../../../../convex/\_generated/dataModel";

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";

import { PlusCircle, AlertCircle, Pencil, Trash2 } from "lucide-react"

import { Alert, AlertDescription, AlertTitle } from "@/components/ui/alert";

import { ScrollArea } from "@/components/ui/scroll-area";

import { useConfirm } from "@/hooks/use-confirm";

import { toast } from "sonner";

type Intervention = {

    \_id: Id<"interventions">;

    name: string;

    description?: string;

};

type NewIntervention = {

    name: string;

    description: string;

};

export const InterventionSettings = () => {

    const [editMode, setEditMode] = useState<Id<"interventions"> | null>(null);

    const [newIntervention, setNewIntervention] = useState<NewIntervention>({

        name: '',

        description: ''

    });

    const [error, setError] = useState<string>('');

    const interventions = useQuery(api.interventions.get);

    const { mutate: create, isPending: isCreating } = useMutation({

        mutationFn: useConvexMutation(api.interventions.create),

    });

    const { mutate: update, isPending: isUpdating } = useMutation({

        mutationFn: useConvexMutation(api.interventions.update),

    })

    const { mutate: remove, isPending: isRemoving } = useMutation({

        mutationFn: useConvexMutation(api.interventions.remove),

    })

    const [DeleteConfirmDialog, confirmDelete] = useConfirm(

        "Delete Intervention",

        "Are you sure you want to delete this intervention?"

    );

    const handleInputChange = (

        e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement>

    ) => {

        const { name, value } = e.target;

        setNewIntervention(prev => ({ ...prev, [name]: value }));

        setError('');

    };

    const handleSubmit = async (e: React.FormEvent) => {

        e.preventDefault()

        try {

            if (editMode) {

                await update({ id: editMode, ...newIntervention })

                toast.success("Intervention updated successfully")

                setEditMode(null)

            } else {

                await create(newIntervention)

                toast.success("Intervention created successfully")

            }

            setNewIntervention({ name: "", description: "" })

        } catch (err) {

            setError(err instanceof Error ? err.message : "An error occurred")

            toast.error(err instanceof Error ? err.message : "An error occurred")

        }

    }

    const handleEdit = (intervention: Intervention) => {

        setEditMode(intervention.\_id)

        setNewIntervention({

            name: intervention.name,

            description: intervention.description || "",

        })

    }

    const handleDelete = async (id: Id<"interventions">) => {

        const confirmed = await confirmDelete()

        if (confirmed) {

            try {

                await remove({ id })

                toast.success("Intervention deleted successfully")

            } catch (err) {

                setError(err instanceof Error ? err.message : "An error occurred")

                toast.error(err instanceof Error ? err.message : "An error occurred")

            }

        }

    }

    if (!interventions) return <div className="flex justify-center items-center h-screen">Loading...</div>

    return (

        <div className="container mx-auto py-8">

            <DeleteConfirmDialog />

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

                <Card className="lg:col-span-1 h-fit">

                    <CardHeader>

                        <CardTitle>{editMode ? "Edit Intervention" : "Add New Intervention"}</CardTitle>

                        <CardDescription>

                            {editMode ? "Update the details of the intervention" : "Create a new intervention"}

                        </CardDescription>

                    </CardHeader>

                    <CardContent>

                        <form onSubmit={handleSubmit} className="space-y-4">

                            <Input

                                type="text"

                                name="name"

                                value={newIntervention.name}

                                onChange={handleInputChange}

                                placeholder="Intervention Name"

                                required

                            />

                            <Textarea

                                name="description"

                                value={newIntervention.description}

                                onChange={handleInputChange}

                                placeholder="Description"

                                className="h-24"

                            />

                            {error && (

                                <Alert variant="destructive">

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

                                    <AlertTitle>Error</AlertTitle>

                                    <AlertDescription>{error}</AlertDescription>

                                </Alert>

                            )}

                            <div className="flex gap-2">

                                <Button

                                    type="submit"

                                    disabled={isCreating || isUpdating}

                                    className="text-white"

                                >

                                    {editMode ? (

                                        <>

                                            <Pencil className="mr-2 h-4 w-4" />

                                            Update Intervention

                                        </>

                                    ) : (

                                        <>

                                            <PlusCircle className="mr-2 h-4 w-4" />

                                            Add Intervention

                                        </>

                                    )}

                                </Button>

                                {editMode && (

                                    <Button

                                        type="button"

                                        variant="outline"

                                        onClick={() => {

                                            setEditMode(null)

                                            setNewIntervention({ name: "", description: "" })

                                        }}

                                    >

                                        Cancel

                                    </Button>

                                )}

                            </div>

                        </form>

                    </CardContent>

                </Card>

                <Card className="lg:col-span-2">

                    <CardHeader>

                        <CardTitle>Current Interventions</CardTitle>

                        <CardDescription>List of all interventions in the system</CardDescription>

                    </CardHeader>

                    <CardContent>

                        <ScrollArea className="h-[calc(100vh-16rem)] pr-4">

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

                                {interventions.map((intervention: Intervention) => (

                                    <Card key={intervention.\_id}>

                                        <CardHeader className="flex flex-row items-start justify-between space-y-0">

                                            <div>

                                                <CardTitle>{intervention.name}</CardTitle>

                                                {intervention.description && <CardDescription>{intervention.description}</CardDescription>}

                                            </div>

                                            <div className="flex gap-2">

                                                <Button variant="outline" size="icon" onClick={() => handleEdit(intervention)}>

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

                                                </Button>

                                                <Button

                                                    variant="destructive"

                                                    size="icon"

                                                    onClick={() => handleDelete(intervention.\_id)}

                                                    disabled={isRemoving}

                                                >

                                                    <Trash2 className="h-4 w-4" />

                                                </Button>

                                            </div>

                                        </CardHeader>

                                    </Card>

                                ))}

                            </div>

                        </ScrollArea>

                    </CardContent>

                </Card>

            </div>

        </div>

    )

}

**Create School Period:**

"use client"

import { Button } from "@/components/ui/button"

import {

    Card,

    CardContent,

    CardDescription,

    CardFooter,

    CardHeader,

    CardTitle

} from "@/components/ui/card"

import { Input } from "@/components/ui/input"

import { Label } from "@/components/ui/label"

import { useConvexMutation } from "@convex-dev/react-query"

import { useMutation } from "@tanstack/react-query"

import { useState } from "react"

import { toast } from "sonner"

import { api } from "../../../../../../convex/\_generated/api"

export const AddSchoolPeriodCard = () => {

    const [period, setPeriod] = useState("")

    const [timeRange, setTimeRange] = useState("")

    const { mutate, isPending } = useMutation({

        mutationFn: useConvexMutation(api.schoolPeriod.create),

    })

    const onSubmit = async () => {

        if (period.length <= 1) return toast.error("Invalid period")

        if (timeRange.length <= 1) return toast.error("Invalid time range")

        await mutate({

            period,

            timeRange,

        })

        toast.success("Successfully created a school period.")

        setPeriod("")

        setTimeRange("")

    }

    return (

        <Card className="flex flex-col h-fit">

            <CardHeader>

                <CardTitle>Add School Period</CardTitle>

                <CardDescription>

                    Fill out the form to add a school period

                </CardDescription>

            </CardHeader>

            <CardContent className="grid gap-8 mt-7">

                <div className="grid gap-2">

                    <Label htmlFor="period">Period</Label>

                    <Input

                        id="period"

                        type="text"

                        placeholder="Ex: 1st Period"

                        className="w-full"

                        disabled={isPending}

                        value={period}

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

                    />

                </div>

                <div className="grid gap-2">

                    <Label htmlFor="tr" className="font-semibold">Time Range</Label>

                    <Input

                        id="tr"

                        type="text"

                        placeholder="Ex: 6:30AM - 7:30AM"

                        className="w-full"

                        disabled={isPending}

                        value={timeRange}

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

                    />

                </div>

            </CardContent>

            <CardFooter className="mt-auto">

                <Button

                    className="text-white"

                    onClick={onSubmit}

                    disabled={isPending}

                >

                    Add

                </Button>

            </CardFooter>

        </Card>

    )

}

**Schedule Card component:**

"use client"

import { DataTable } from "@/components/data-table";

import {

    Card,

    CardContent,

    CardDescription,

    CardHeader,

    CardTitle,

} from "@/components/ui/card";

import {

    Select,

    SelectContent,

    SelectItem,

    SelectTrigger,

    SelectValue,

} from "@/components/ui/select";

import { cn } from "@/lib/utils";

import { useQuery } from "convex/react";

import { useState } from "react";

import { api } from "../../../../../../convex/\_generated/api";

import { ScheduleColumns } from "./schedule-columns";

interface ScheduleCardTableProps {

    isSH?: boolean;

}

export const ScheduleCardTable = ({

    isSH

}: ScheduleCardTableProps) => {

    const [selectedDay, setSelectedDay] = useState<string>("all");

    const [filterType, setFilterType] = useState<"teacher" | "room">("teacher");

    const schedules = useQuery(api.schedules.get) as { day: string }[] | undefined;

    const days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];

    // Update the filtering logic

    const filteredSchedules = selectedDay !== "all"

        ? schedules?.filter((schedule) => schedule.day.includes(selectedDay))

        : schedules;

    return (

        <Card className={cn(!isSH ? "xl:col-span-2" : "xl:col-span-3")}>

            <CardHeader>

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

                    <div>

                        <CardTitle>Class Schedules</CardTitle>

                        <CardDescription>

                            Manage your class schedules here

                        </CardDescription>

                    </div>

                    <div className="flex gap-3">

                        <Select

                            value={selectedDay}

                            onValueChange={setSelectedDay}

                        >

                            <SelectTrigger className="w-[180px]">

                                <SelectValue placeholder="Filter by day" />

                            </SelectTrigger>

                            <SelectContent>

                                <SelectItem value="all">All Days</SelectItem>

                                {days.map((day) => (

                                    <SelectItem key={day} value={day}>

                                        {day}

                                    </SelectItem>

                                ))}

                            </SelectContent>

                        </Select>

                        <Select

                            value={filterType}

                            onValueChange={(value: "teacher" | "room") => setFilterType(value)}

                        >

                            <SelectTrigger className="w-[180px]">

                                <SelectValue placeholder="Filter by..." />

                            </SelectTrigger>

                            <SelectContent>

                                <SelectItem value="teacher">Teacher Name</SelectItem>

                                <SelectItem value="room">Room</SelectItem>

                            </SelectContent>

                        </Select>

                    </div>

                </div>

            </CardHeader>

            <CardContent>

                <DataTable

                    columns={ScheduleColumns}

                    // @ts-expect-error slight type issue

                    data={filteredSchedules ?? []}

                    filter={filterType}

                    placeholder={filterType === "teacher" ? "Filter by teacher..." : "Filter by room..."}

                />

            </CardContent>

        </Card>

    );

};

**Create school year card:**

"use client"

import { DatePickerWithRange } from "@/components/date-range-picker"

import { Button } from "@/components/ui/button"

import {

    Card,

    CardContent,

    CardDescription,

    CardFooter,

    CardHeader,

    CardTitle

} from "@/components/ui/card"

import { Input } from "@/components/ui/input"

import { Label } from "@/components/ui/label"

import { api } from "../../../../../../convex/\_generated/api"

import { useMutation } from "@tanstack/react-query"

import { useConvexMutation } from "@convex-dev/react-query"

import { useState } from "react"

import { DateRange } from "react-day-picker"

import { addDays } from "date-fns"

import { toast } from "sonner"

export const AddSchoolYearCard = () => {

    const [batchName, setBatchName] = useState("")

    const [dateRange, setDateRange] = useState<DateRange | undefined>({

        from: new Date(),

        to: addDays(new Date(), 365),

    })

    const { mutate, isPending } = useMutation({

        mutationFn: useConvexMutation(api.schoolYear.create),

        onSuccess: () => {

            toast.success("New school year added!")

        },

        onError: () => {

            toast.error("Error adding school year!")

        }

    })

    const onSubmit = async () => {

        if (batchName.length <= 3) {

            toast.error("Please add a valid batch name")

            return;

        }

        if (dateRange?.from && dateRange?.to) {

            await mutate({

                batchName,

                startDate: new Date(dateRange.from.getTime() - dateRange.from.getTimezoneOffset() \* 60000).toISOString().split('T')[0],

                endDate: new Date(dateRange.to.getTime() - dateRange.to.getTimezoneOffset() \* 60000).toISOString().split('T')[0],

                sy: `${new Date(dateRange.from).getFullYear()}-${new Date(dateRange.to).getFullYear()}`

            })

        }

    }

    return (

        <Card className="flex flex-col h-fit">

            <CardHeader>

                <CardTitle>Add School Year</CardTitle>

                <CardDescription>

                    Fill out the form to add a school year

                </CardDescription>

            </CardHeader>

            <CardContent className="grid gap-8 mt-7">

                <div className="grid gap-2">

                    <Label htmlFor="batchName" className="font-semibold">Batch Name</Label>

                    <Input

                        id="batchName"

                        type="text"

                        placeholder="Batch name"

                        className="w-full"

                        value={batchName}

                        required

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

                        disabled={isPending}

                    />

                </div>

                <div className="grid gap-2">

                    <Label htmlFor="SY" className="font-semibold">Start Date - End Date</Label>

                    <DatePickerWithRange

                        value={dateRange}

                        onChange={setDateRange}

                    />

                </div>

            </CardContent>

            <CardFooter className="mt-auto">

                <Button

                    className="text-white"

                    disabled={isPending}

                    onClick={onSubmit}

                >

                    {isPending ? "Adding..." : "Add School Year"}

                </Button>

            </CardFooter>

        </Card>

    )

}

**Create section form:**

"use client"

import { Button, buttonVariants } from "@/components/ui/button";

import {

    Card,

    CardContent,

    CardDescription,

    CardHeader,

    CardTitle

} from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

    Select,

    SelectContent,

    SelectItem,

    SelectTrigger,

    SelectValue

} from "@/components/ui/select";

import { SectionWithDetails } from "@/lib/types";

import { cn } from "@/lib/utils";

import { SectionFormData, sectionSchema } from "@/lib/validation/add-section-zod";

import { useConvexMutation } from "@convex-dev/react-query";

import { zodResolver } from "@hookform/resolvers/zod";

import { useMutation } from "@tanstack/react-query";

import { useQuery } from "convex/react";

import {

    ChevronLeft,

    MinusIcon,

    PlusIcon,

    Trash

} from "lucide-react";

import Link from "next/link";

import { useRouter } from "next/navigation";

import { useEffect, useState } from "react";

import { useFieldArray, useForm } from "react-hook-form";

import { toast } from "sonner";

import { api } from "../../../../../../convex/\_generated/api";

import { Id } from "../../../../../../convex/\_generated/dataModel";

import { AddScheduleModal } from "./add-schedule-modal";

import { EditScheduleModal } from "./edit-schedule-modal";

interface SectionFormProps {

    isEditing?: boolean;

    section?: SectionWithDetails;

}

export const SectionForm = ({ isEditing = false, section }: SectionFormProps) => {

    const [isSeniorHigh, setIsSeniorHigh] = useState<boolean | undefined>(false);

    const router = useRouter()

    const { register, control, handleSubmit, setValue, watch, reset, formState: { errors } } = useForm<SectionFormData>({

        resolver: zodResolver(sectionSchema),

        defaultValues: {

            name: "",

            roomId: "",

            advisorId: "",

            schoolYearId: "",

            gradeLevelId: "",

            classes: [{

                subjectId: "",

                teacherId: "",

                semester: "",

                track: ""

            }]

        }

    });

    const { fields, append, remove } = useFieldArray({

        control,

        name: "classes"

    });

    const { mutate: createSection, isPending } = useMutation({

        mutationFn: useConvexMutation(api.sections.create),

        onSuccess: () => {

            toast.success("Section created successfully");

            reset();

        },

        onError: (error) => {

            toast.error(error.message);

        }

    });

    const { mutate: updateSection, isPending: isUpdating } = useMutation({

        mutationFn: useConvexMutation(api.sections.update),

        onSuccess: () => {

            toast.success("Section updated successfully");

            router.push("/sysadmin-sections");

        },

        onError: (error) => {

            toast.error(error.message);

        }

    });

    const selectedGradeLevelId = watch("gradeLevelId");

    const teachers = useQuery(api.users.getTeachers);

    const subjects = useQuery(api.subjects.getSubjects);

    const schoolYears = useQuery(api.schoolYear.get);

    const gradeLevels = useQuery(api.gradeLevel.get);

    const schoolPeriods = useQuery(api.schoolPeriod.get);

    const rooms = useQuery(api.classroom.getAvailableRooms);

    const filteredSubjects = subjects?.filter(

        subject => subject.gradeLevelId === selectedGradeLevelId

    );

    const trackOptions = [

        { label: "Core Subject (All Track)", value: "core" },

        { label: "Academic Track", value: "academic" },

        { label: "Work Immersion/Culminating Activity", value: "immersion" },

        { label: "TVL Track", value: "tvl" },

        { label: "Sports Track", value: "sports" },

        { label: "Arts and Design Track", value: "arts" }

    ];

    useEffect(() => {

        if (isEditing && section) {

            reset({

                name: section.name,

                gradeLevelId: section.gradeLevelId,

                advisorId: section.advisorId,

                schoolYearId: section.schoolYearId,

                roomId: section.roomId,

                classes: section.classes.map(cls => ({

                    id: cls.\_id,

                    subjectId: cls.subjectId,

                    teacherId: cls.teacherId,

                    semester: cls.semester || "",

                    track: cls.track || "",

                    schedules: cls.schedules?.map(schedule => ({

                        days: schedule.day,

                        schoolPeriodId: schedule.schoolPeriodId,

                        roomId: schedule.roomId

                    })) || []

                }))

            });

            // Set senior high state

            const gradeLevel = gradeLevels?.find(g => g.\_id === section.gradeLevelId);

            const isSHS = gradeLevel?.level.includes("11") || gradeLevel?.level.includes("12");

            setIsSeniorHigh(isSHS);

        }

    }, [isEditing, section, reset, gradeLevels]);

    useEffect(() => {

        const advisorId = watch('advisorId');

        if (advisorId && fields.length > 0) {

            setValue(`classes.0.teacherId`, advisorId);

            const firstTeacherSelect = document.querySelector(`[name="classes.0.teacherId"]`);

            if (firstTeacherSelect) {

                firstTeacherSelect.setAttribute('disabled', 'true');

            }

        }

    }, [watch('advisorId'), fields.length, setValue]);

    const handleAddClass = () => {

        const formValues = watch();

        append({

            subjectId: "",

            teacherId: fields.length === 0 ? formValues.advisorId : "",

            semester: "",

            track: ""

        });

    };

    // const handleGradeLevelChange = (value: string) => {

    //     setValue("gradeLevelId", value)

    //     const selectedGradeLevel = gradeLevels?.find(g => g.\_id === value);

    //     const isSHS = selectedGradeLevel?.level.includes("11") || selectedGradeLevel?.level.includes("12");

    //     setIsSeniorHigh(isSHS);

    //     if (!isSHS) {

    //         // Clear semester and track for all classes

    //         fields.forEach((\_, index) => {

    //             setValue(`classes.${index}.semester`, "");

    //             setValue(`classes.${index}.track`, "");

    //         });

    //     }

    // };

    const handleRoomSelect = (roomId: string) => {

        const selectedRoom = rooms?.find(room => room.\_id === roomId);

        if (selectedRoom && selectedRoom.teacher && selectedRoom.gradeLevel) {

            setValue("roomId", roomId);

            setValue("advisorId", selectedRoom.teacher.\_id);

            setValue("gradeLevelId", selectedRoom.gradeLevel.\_id);

            if (fields.length > 0) {

                setValue("classes.0.teacherId", selectedRoom.teacher.\_id);

            }

            const isSHS = selectedRoom.gradeLevel.level.includes("11") ||

                selectedRoom.gradeLevel.level.includes("12");

            setIsSeniorHigh(isSHS);

        }

    };

    // const handleSemesterChange = (value: string, index: number) => {

    //     setValue(`classes.${index}.semester`, value);

    // };

    // const handleTrackChange = (value: string, index: number) => {

    //     setValue(`classes.${index}.track`, value);

    // };

    const onSubmit = (data: SectionFormData) => {

        if (isEditing && section) {

            updateSection({

                id: section.\_id,

                ...data,

                advisorId: data.advisorId as Id<"users">,

                gradeLevelId: data.gradeLevelId as Id<"gradeLevels">,

                schoolYearId: data.schoolYearId as Id<"schoolYears">,

                roomId: data.roomId as Id<"rooms">,

                classes: data.classes.map(cls => ({

                    ...(cls.id ? { id: cls.id as Id<"classes"> } : {}),

                    subjectId: cls.subjectId as Id<"subjects">,

                    teacherId: cls.teacherId as Id<"users">,

                    semester: cls.semester,

                    track: cls.track,

                    schedules: cls.schedules?.map(schedule => ({

                        days: schedule.days,

                        roomId: schedule.roomId as Id<"rooms">,

                        schoolPeriodId: schedule.schoolPeriodId as Id<"schoolPeriods">,

                    }))

                }))

            });

        } else {

            createSection({

                ...data,

                advisorId: data.advisorId as Id<"users">,

                gradeLevelId: data.gradeLevelId as Id<"gradeLevels">,

                roomId: data.roomId as Id<"rooms">,

                schoolYearId: data.schoolYearId as Id<"schoolYears">,

                classes: data.classes.map(cls => ({

                    ...cls,

                    subjectId: cls.subjectId as Id<"subjects">,

                    teacherId: cls.teacherId as Id<"users">,

                    schedules: cls.schedules?.map(schedule => ({

                        ...schedule,

                        roomId: schedule.roomId as Id<"rooms">,

                        schoolPeriodId: schedule.schoolPeriodId as Id<"schoolPeriods">,

                    }))

                }))

            });

        }

    };

    return (

        <div className="container mx-auto p-4">

            <main className="space-y-4 mt-8">

                <form onSubmit={handleSubmit(onSubmit)}>

                    <div className="mx-auto grid max-w-full flex-1 auto-rows-max gap-4">

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

                            <Link

                                href="/sysadmin-sections"

                                className={cn("h-7 w-7", buttonVariants({

                                    variant: "outline",

                                    size: "icon",

                                }))}>

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

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

                            </Link>

                            <h1 className="flex-1 shrink-0 whitespace-nowrap text-xl font-semibold tracking-tight sm:grow-0">

                                {isEditing ? "Edit Section" : "Add a Section"}

                            </h1>

                            {/\* Button for desktop/laptop users \*/}

                            <div className="hidden items-center gap-2 md:flex md:ml-auto">

                                <Button

                                    type="submit"

                                    size="sm"

                                    className="text-white"

                                    disabled={isEditing ? isUpdating : isPending}

                                >

                                    {isEditing

                                        ? (isUpdating ? "Updating..." : "Update Section")

                                        : (isPending ? "Creating..." : "Save Section")

                                    }

                                </Button>

                            </div>

                        </div>

                        <div className="grid gap-4 md:grid-cols-[1fr\_250px] lg:grid-cols-3 lg:gap-8">

                            <div className="grid auto-rows-max items-start gap-4 lg:col-span-3 lg:gap-8">

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Section Details</CardTitle>

                                        <CardDescription>

                                            Add section details

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-2">

                                                <Label htmlFor="name">Section Name</Label>

                                                <Input

                                                    {...register("name")}

                                                    placeholder="Section name"

                                                />

                                                {errors.name && (

                                                    <p className="text-sm text-red-500">{errors.name.message}</p>

                                                )}

                                            </div>

                                            <div className="grid gap-2">

                                                <Label>School Year</Label>

                                                <Select

                                                    onValueChange={(value) => setValue("schoolYearId", value)}

                                                    value={watch("schoolYearId")}

                                                >

                                                    <SelectTrigger>

                                                        <SelectValue placeholder="Select School Year" />

                                                    </SelectTrigger>

                                                    <SelectContent>

                                                        {schoolYears?.map((sy) => (

                                                            <SelectItem key={sy.\_id} value={sy.\_id}>

                                                                {sy.sy || `${sy.startDate.split('-')[0]}-${sy.endDate.split('-')[0]}`}

                                                            </SelectItem>

                                                        ))}

                                                    </SelectContent>

                                                </Select>

                                                {errors.schoolYearId && (

                                                    <p className="text-sm text-red-500">{errors.schoolYearId.message}</p>

                                                )}

                                            </div>

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

                                                <div className="grid gap-2">

                                                    <Label>Room</Label>

                                                    <Select

                                                        onValueChange={handleRoomSelect}

                                                        value={watch("roomId")}

                                                    >

                                                        <SelectTrigger>

                                                            <SelectValue placeholder="Select Room" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            {rooms?.map((room) => (

                                                                <SelectItem key={room.\_id} value={room.\_id}>

                                                                    {room.name} ({room.teacher?.lastName})

                                                                </SelectItem>

                                                            ))}

                                                        </SelectContent>

                                                    </Select>

                                                    {errors.roomId && (

                                                        <p className="text-sm text-red-500">{errors.roomId.message}</p>

                                                    )}

                                                </div>

                                                {/\* Make Adviser field read-only since it's auto-populated from room selection \*/}

                                                <div className="grid gap-2">

                                                    <Label>Adviser</Label>

                                                    <Select

                                                        disabled

                                                        value={watch("advisorId")}

                                                    >

                                                        <SelectTrigger>

                                                            <SelectValue>

                                                                {teachers?.find(t => t.id === watch("advisorId"))?.lastName},

                                                                {teachers?.find(t => t.id === watch("advisorId"))?.firstName}

                                                            </SelectValue>

                                                        </SelectTrigger>

                                                    </Select>

                                                    {errors.advisorId && (

                                                        <p className="text-sm text-red-500">{errors.advisorId.message}</p>

                                                    )}

                                                </div>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Classes Card \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Class Schedule</CardTitle>

                                        <CardDescription>Add subjects and their schedules</CardDescription>

                                    </CardHeader>

                                    <CardContent>

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

                                            {fields.map((field, index) => (

                                                <Card key={field.id} className="p-4 border border-gray-200">

                                                    <div className="grid gap-4 sm:grid-cols-2">

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

                                                            <Label htmlFor={`subject-${index}`}>Subject</Label>

                                                            <Select

                                                                onValueChange={(value) => setValue(`classes.${index}.subjectId`, value)}

                                                                disabled={!selectedGradeLevelId}

                                                                value={watch(`classes.${index}.subjectId`)}

                                                            >

                                                                <SelectTrigger id={`subject-${index}`}>

                                                                    <SelectValue

                                                                        placeholder={

                                                                            !selectedGradeLevelId

                                                                                ? "Select grade level first"

                                                                                : index === 0

                                                                                    ? "Adviser's subject"

                                                                                    : "Select subject"

                                                                        }

                                                                    />

                                                                </SelectTrigger>

                                                                <SelectContent>

                                                                    {filteredSubjects?.map((subject) => (

                                                                        <SelectItem key={subject.\_id} value={subject.\_id}>

                                                                            {subject.name}

                                                                        </SelectItem>

                                                                    ))}

                                                                </SelectContent>

                                                            </Select>

                                                            {errors.classes?.[index]?.subjectId && (

                                                                <p className="text-sm text-red-500">{errors.classes[index].subjectId.message}</p>

                                                            )}

                                                        </div>

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

                                                            <Label htmlFor={`teacher-${index}`}>{index === 0 ? "Adviser" : "Teacher"}</Label>

                                                            <Select

                                                                onValueChange={(value) => setValue(`classes.${index}.teacherId`, value)}

                                                                disabled={index === 0}

                                                                value={watch(`classes.${index}.teacherId`)}

                                                            >

                                                                <SelectTrigger id={`teacher-${index}`}>

                                                                    <SelectValue

                                                                        placeholder={index === 0 ? "Adviser will teach this subject" : "Select teacher"}

                                                                    />

                                                                </SelectTrigger>

                                                                <SelectContent>

                                                                    {teachers?.map((teacher) => (

                                                                        <SelectItem key={teacher.id} value={teacher.id}>

                                                                            {teacher.lastName}, {teacher.firstName}

                                                                        </SelectItem>

                                                                    ))}

                                                                </SelectContent>

                                                            </Select>

                                                            {errors.classes?.[index]?.teacherId && (

                                                                <p className="text-sm text-red-500">{errors.classes[index].teacherId.message}</p>

                                                            )}

                                                        </div>

                                                        {isSeniorHigh && (

                                                            <>

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

                                                                    <Label htmlFor={`semester-${index}`}>Semester</Label>

                                                                    <Select

                                                                        onValueChange={(value) => setValue(`classes.${index}.semester`, value)}

                                                                        value={watch(`classes.${index}.semester`)}

                                                                    >

                                                                        <SelectTrigger id={`semester-${index}`}>

                                                                            <SelectValue placeholder="Select Semester" />

                                                                        </SelectTrigger>

                                                                        <SelectContent>

                                                                            <SelectItem value="1st">1st Semester</SelectItem>

                                                                            <SelectItem value="2nd">2nd Semester</SelectItem>

                                                                        </SelectContent>

                                                                    </Select>

                                                                    {errors.classes?.[index]?.semester && (

                                                                        <p className="text-sm text-red-500">{errors.classes[index].semester.message}</p>

                                                                    )}

                                                                </div>

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

                                                                    <Label htmlFor={`track-${index}`}>Track</Label>

                                                                    <Select

                                                                        onValueChange={(value) => setValue(`classes.${index}.track`, value)}

                                                                        value={watch(`classes.${index}.track`)}

                                                                    >

                                                                        <SelectTrigger id={`track-${index}`}>

                                                                            <SelectValue placeholder="Select Track" />

                                                                        </SelectTrigger>

                                                                        <SelectContent>

                                                                            {trackOptions.map((track) => (

                                                                                <SelectItem key={track.value} value={track.value}>

                                                                                    {track.label}

                                                                                </SelectItem>

                                                                            ))}

                                                                        </SelectContent>

                                                                    </Select>

                                                                    {errors.classes?.[index]?.track && (

                                                                        <p className="text-sm text-red-500">{errors.classes[index].track.message}</p>

                                                                    )}

                                                                </div>

                                                            </>

                                                        )}

                                                    </div>

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

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

                                                            <Label>Schedules</Label>

                                                            {isEditing ? (

                                                                <EditScheduleModal

                                                                    initialSchedule={watch(`classes.${index}.schedules`)?.[0] || {

                                                                        days: [],

                                                                        schoolPeriodId: "",

                                                                        roomId: ""

                                                                    }}

                                                                    onScheduleEdit={(schedule) => {

                                                                        setValue(`classes.${index}.schedules`, [{

                                                                            days: Array.isArray(schedule.days) ? schedule.days : [schedule.days],

                                                                            schoolPeriodId: schedule.schoolPeriodId,

                                                                            roomId: schedule.roomId

                                                                        }]);

                                                                    }}

                                                                    schoolPeriods={schoolPeriods || []}

                                                                    rooms={rooms || []}

                                                                />

                                                            ) : (

                                                                <AddScheduleModal

                                                                    onScheduleAdd={(schedule) => {

                                                                        const schedules = watch(`classes.${index}.schedules`) || [];

                                                                        setValue(`classes.${index}.schedules`, [...schedules, schedule]);

                                                                    }}

                                                                    schoolPeriods={schoolPeriods || []}

                                                                    rooms={rooms || []}

                                                                />

                                                            )}

                                                        </div>

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

                                                            {watch(`classes.${index}.schedules`)?.map((schedule, scheduleIndex) => (

                                                                <div

                                                                    key={scheduleIndex}

                                                                    className="flex items-center justify-between p-2 bg-gray-50 rounded-md"

                                                                >

                                                                    <div className="space-y-1">

                                                                        <div className="text-sm font-medium">

                                                                            {Array.isArray(schedule.days)

                                                                                ? schedule.days.join(", ")

                                                                                : schedule.days}

                                                                        </div>

                                                                        <div className="text-sm text-muted-foreground">

                                                                            {schoolPeriods?.find(p => p.\_id === schedule.schoolPeriodId)?.timeRange} |

                                                                            {rooms?.find(r => r.\_id === schedule.roomId)?.name}

                                                                        </div>

                                                                    </div>

                                                                    <Button

                                                                        type="button"

                                                                        variant="ghost"

                                                                        size="sm"

                                                                        onClick={() => {

                                                                            const schedules = watch(`classes.${index}.schedules`);

                                                                            setValue(

                                                                                `classes.${index}.schedules`,

                                                                                schedules?.filter((\_, i) => i !== scheduleIndex)

                                                                            );

                                                                        }}

                                                                    >

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

                                                                    </Button>

                                                                </div>

                                                            ))}

                                                        </div>

                                                    </div>

                                                    {index > 0 && (

                                                        <Button type="button" variant="ghost" size="sm" className="mt-4" onClick={() => remove(index)}>

                                                            <MinusIcon className="h-4 w-4 mr-2" />

                                                            Remove Class

                                                        </Button>

                                                    )}

                                                </Card>

                                            ))}

                                            {!isEditing && (

                                                <Button type="button" variant="outline" onClick={handleAddClass} className="w-full">

                                                    <PlusIcon className="h-4 w-4 mr-2" />

                                                    Add Class

                                                </Button>

                                            )}

                                        </div>

                                    </CardContent>

                                </Card>

                            </div>

                        </div>

                        {/\* Button for mobile users \*/}

                        <div className="flex items-center justify-center gap-2 md:hidden">

                            <Button

                                size="sm"

                                className="text-white"

                            >

                                Save Product

                            </Button>

                        </div>

                    </div>

                </form>

            </main>

        </div >

    )

}

**Create subjects card:**

"use client"

import { Button } from "@/components/ui/button";

import {

    Card,

    CardContent,

    CardDescription,

    CardFooter,

    CardHeader,

    CardTitle

} from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { useConvexMutation } from "@convex-dev/react-query";

import { zodResolver } from "@hookform/resolvers/zod";

import { useMutation } from "@tanstack/react-query";

import { useForm } from "react-hook-form";

import { toast } from "sonner";

import \* as z from "zod";

import { api } from "../../../../../../convex/\_generated/api";

import { useQuery } from "convex/react";

import { Id } from "../../../../../../convex/\_generated/dataModel";

import { useState } from "react";

// const componentSchema = z.object({

//     written: z.number().min(0).max(100),

//     performance: z.number().min(0).max(100),

//     exam: z.number().min(0).max(100).optional()

// }).refine(data => {

//     const total = data.written + data.performance + (data.exam || 0);

//     return total === 100;

// }, "Component weights must total 100%");

const gradeWeightSchema = z.object({

    written: z.number().min(0).max(100),

    performance: z.number().min(0).max(100),

    exam: z.number().min(0).max(100).optional(),

}).refine(data => {

    const total = data.written + data.performance + (data.exam || 0);

    return total === 100;

}, "Component weights must total 100%");

export const subjectSchema = z.object({

    name: z.string().min(1, "Subject name is required"),

    gradeLevelId: z.string().min(1, "Grade level is required"),

    subjectCode: z.string().min(2).max(10),

    subjectCategory: z.enum(["core", "applied", "specialized"]),

    gradeWeights: gradeWeightSchema,

    // isMapeh: z.boolean().default(false),

    // mapehWeights: z.object({

    //     music: componentSchema,

    //     arts: componentSchema,

    //     pe: componentSchema,

    //     health: componentSchema

    // }).optional()

})

// .superRefine((data, ctx) => {

//     if (data.isMapeh && !data.mapehWeights) {

//         ctx.addIssue({

//             code: z.ZodIssueCode.custom,

//             message: "MAPEH weights are required for MAPEH subjects",

//             path: ["mapehWeights"]

//         });

//     }

//     if (!data.isMapeh && !data.gradeWeights) {

//         ctx.addIssue({

//             code: z.ZodIssueCode.custom,

//             message: "Grade weights are required for non-MAPEH subjects",

//             path: ["gradeWeights"]

//         });

//     }

// });

export type SubjectFormData = z.infer<typeof subjectSchema>;

export const AddSubjectsCard = () => {

    const [gradeLevel, setGradeLevel] = useState<string>("Grade 7")

    const { register, handleSubmit, setValue, getValues, formState: { errors } } = useForm<SubjectFormData>({

        resolver: zodResolver(subjectSchema),

        // defaultValues: {

        //     isMapeh: false

        // }

    });

    const gradeLevels = useQuery(api.gradeLevel.get);

    const subjects = [

        "English",

        "Mathematics",

        "Science",

        "Filipino",

        "Araling Panlipunan",

        "Edukasyon sa Pagpapakatao",

        "MAPEH",

        "Edukasyong Pantahanan at Pangkabuhayan",

        "Technology and Livelihood Education (TLE)",

        "Computer Programming"

    ];

    const { mutate: createSubject, isPending } = useMutation({

        mutationFn: useConvexMutation(api.subjects.create),

        onSuccess: () => {

            toast.success("Subject created successfully");

        },

        onError: (error) => {

            toast.error(error.message);

        }

    });

    const sortGradeLevels = gradeLevels?.sort((a, b) => Number(a.level) - Number(b.level))

    const findGradeLevel = () => {

        const gradeLevel = gradeLevels?.find(level => level.\_id === getValues('gradeLevelId'));

        return gradeLevel?.level || "Grade 7"

    };

    // const isMapeh = watch("isMapeh");

    // const handleMapehChange = (checked: boolean) => {

    //     setValue("isMapeh", checked);

    //     if (checked) {

    //         setValue("gradeWeights", undefined);

    //     } else {

    //         setValue("mapehWeights", undefined);

    //     }

    // };

    const onSubmit = async (data: SubjectFormData) => {

        // If it's a MAPEH subject, exclude gradeWeights from submission

        // const submitData = isMapeh

        //     ? { ...data, gradeWeights: undefined }

        //     : { ...data, mapehWeights: undefined };

        createSubject({

            ...data,

            gradeLevelId: data.gradeLevelId as Id<"gradeLevels">

        });

    };

    const isShs = gradeLevel === "Grade 11" || gradeLevel === "Grade 12";

    // const components = ["music", "arts", "pe", "health"] as const;

    return (

        <Card className="flex flex-col h-fit">

            <CardHeader>

                <CardTitle>Add New Subject</CardTitle>

                <CardDescription>Create a new subject with grade weights</CardDescription>

            </CardHeader>

            <form onSubmit={handleSubmit(onSubmit)}>

                <CardContent className="grid gap-8 mt-7">

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

                        <Label>Grade Level</Label>

                        <Select onValueChange={(value) => {

                            setValue("gradeLevelId", value)

                            setGradeLevel(findGradeLevel())

                        }}>

                            <SelectTrigger>

                                <SelectValue placeholder="Select Grade Level" />

                            </SelectTrigger>

                            <SelectContent>

                                {sortGradeLevels?.map((grade) => (

                                    <SelectItem

                                        key={grade.\_id}

                                        value={grade.\_id}

                                    >

                                        {grade.level}

                                    </SelectItem>

                                ))}

                            </SelectContent>

                        </Select>

                        {errors.gradeLevelId && (

                            <p className="text-sm text-red-500">{errors.gradeLevelId.message}</p>

                        )}

                    </div>

                    {isShs ? (

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

                            <Label>Subject Name</Label>

                            <Input {...register("name")} />

                            {errors.name && (

                                <p className="text-sm text-red-500">{errors.name.message}</p>

                            )}

                        </div>

                    ) : (

                        // to ensure the format of subject names

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

                            <Label>Subject Name</Label>

                            <Select onValueChange={(value) => {

                                setValue("name", value)

                            }}>

                                <SelectTrigger>

                                    <SelectValue placeholder="Select Subect" />

                                </SelectTrigger>

                                <SelectContent>

                                    {subjects?.map((subect) => (

                                        <SelectItem

                                            key={subect}

                                            value={subect}

                                        >

                                            {subect}

                                        </SelectItem>

                                    ))}

                                </SelectContent>

                            </Select>

                            {errors.name && (

                                <p className="text-sm text-red-500">{errors.name.message}</p>

                            )}

                        </div>

                    )}

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

                        <Label>Subject Code</Label>

                        <Input {...register("subjectCode")} />

                        {errors.subjectCode && (

                            <p className="text-sm text-red-500">{errors.subjectCode.message}</p>

                        )}

                    </div>

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

                        <Label>Category</Label>

                        <Select onValueChange={

                            (value) => setValue("subjectCategory", value as "core" | "applied" | "specialized")}>

                            <SelectTrigger>

                                <SelectValue placeholder="Select category" />

                            </SelectTrigger>

                            <SelectContent>

                                <SelectItem value="core">Core</SelectItem>

                                <SelectItem value="applied">Applied</SelectItem>

                                <SelectItem value="specialized">Specialized</SelectItem>

                            </SelectContent>

                        </Select>

                        {errors.subjectCategory && (

                            <p className="text-sm text-red-500">{errors.subjectCategory.message}</p>

                        )}

                    </div>

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

                        <Label>Grade Weights (Total must be 100%)</Label>

                        <div className="grid grid-cols-3 gap-4">

                            <div>

                                <Label>Written</Label>

                                <Input

                                    type="number"

                                    {...register("gradeWeights.written", { valueAsNumber: true })}

                                />

                                {errors?.gradeWeights?.written && (

                                    <p className="text-sm text-red-500">{errors.gradeWeights.written.message}</p>

                                )}

                            </div>

                            <div>

                                <Label>Performance</Label>

                                <Input

                                    type="number"

                                    {...register("gradeWeights.performance", { valueAsNumber: true })}

                                />

                                {errors?.gradeWeights?.performance && (

                                    <p className="text-sm text-red-500">{errors.gradeWeights.performance.message}</p>

                                )}

                            </div>

                            <div>

                                <Label>Exam</Label>

                                <Input

                                    type="number"

                                    {...register("gradeWeights.exam", { valueAsNumber: true })}

                                />

                                {errors?.gradeWeights?.exam && (

                                    <p className="text-sm text-red-500">{errors.gradeWeights.exam.message}</p>

                                )}

                            </div>

                        </div>

                    </div>

                </CardContent>

                <CardFooter>

                    <Button type="submit" disabled={isPending} className="text-white">

                        {isPending ? "Creating..." : "Create Subject"}

                    </Button>

                </CardFooter>

            </form>

        </Card>

    )

}

Enrollment Tab

'use client'

import React from 'react'

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from '@/components/ui/card'

import { DataTable } from '@/components/data-table'

import EnrollmentDialog from '../my-advisees/\_components/EnrollmentDialog'

import { enrollmentColumn } from './enrollment-data'

import { useQuery } from 'convex/react'

import { api } from '../../../../convex/\_generated/api'

import Loading from '@/app/components/Loading'

function StudentsPage() {

    const students = useQuery(api.students.getStudent)

    if(!students){

        return <Loading/>

    }

    return (

        <div className='container mx-auto p-4'>

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

                <div className="flex items-center">

                    <div className="hidden md:flex   ml-auto items-center gap-2">

                        <EnrollmentDialog />

                    </div>

                </div>

                <Card x-chunk="dashboard-06-chunk-0">

                    <CardHeader>

                        <CardTitle>List of Students</CardTitle>

                        <CardDescription className="flex justify-between">

                           Manage the list of students.

                        </CardDescription>

                    </CardHeader>

                    <CardContent className=''>

                        <div className="">

                            <DataTable

                                columns={enrollmentColumn}

                                // @ts-ignore

                                data={students}

                                filter="fullName"

                                placeholder="students by name"

                            />

                        </div>

                    </CardContent>

                </Card>

            </main>

        </div>

    )

}

export default StudentsPage

**Enrollment Form**

'use client'

import React, { useState } from 'react'

import {

    Dialog,

    DialogContent,

    DialogHeader,

    DialogTitle,

    DialogTrigger,

  } from "@/components/ui/dialog"

import { PlusCircleIcon } from 'lucide-react'

import EnrollmentForm from './Form'

function EnrollmentDialog() {

  const [isOpen, setIsOpen] = useState(false)

  return (

    <div>

        <Dialog open={isOpen} onOpenChange={setIsOpen}>

            <DialogTrigger onClick={()=>setIsOpen(true)} className='border shadow-md flex justify-center items-center gap-x-3 disabled:bg-blue-200 bg-blue-600 text-white border-gray-100 rounded-md px-2 py-1'>

                <PlusCircleIcon className="h-3.5 w-3.5" />

                <span className="sr-only sm:not-sr-only sm:whitespace-nowrap">

                    Enroll student

                </span>

            </DialogTrigger>

            <DialogContent className='max-w-6xl max-h-screen overflow-auto'>

              <DialogHeader>

                <DialogTitle>Enroll new student</DialogTitle>

              </DialogHeader>

              <EnrollmentForm onClose={()=>setIsOpen(false)}/>

            </DialogContent>

          </Dialog>

    </div>

  )

}

export default EnrollmentDialog

**Advisees Tab**

'use client'

import { StudentTable } from './\_components/StudentTable'

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from '@/components/ui/card'

function Studentpage() {

  return (

    <div className="container mx-auto p-4">

    <main className="">

        <Card x-chunk="dashboard-06-chunk-0">

            <CardHeader>

                <CardTitle>Advisory Class</CardTitle>

                <CardDescription className="flex justify-between">

                    List of students

                </CardDescription>

            </CardHeader>

            <CardContent className=''>

                <StudentTable/>

            </CardContent>

        </Card>

    </main>

    </div>

  )

}

export default Studentpage

**Student records (grades, values, attendance) tabs**

'use client'

import React, { useState } from 'react'

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"

import { ChevronLeft, File } from 'lucide-react'

import { Button, buttonVariants } from '@/components/ui/button'

import Link from 'next/link'

import { cn } from '@/lib/utils'

import {

    DropdownMenu,

    DropdownMenuContent,

    DropdownMenuItem,

    DropdownMenuLabel,

    DropdownMenuSeparator,

    DropdownMenuTrigger,

  } from "@/components/ui/dropdown-menu"

import JrGradesTemplate from '../\_components/JrGradesTemplate'

import SrGradesTemplate from '../\_components/SrGradesTemplate'

import { useQuery } from 'convex/react'

import { api } from '../../../../../convex/\_generated/api'

import { Doc, Id } from '../../../../../convex/\_generated/dataModel'

import Values from '../\_components/Values'

import Attendance from '../\_components/Attendance'

import SchoolForm9 from '../\_components/SchoolForm9'

import { useSearchParams } from 'next/navigation'

import SF10Template from '../\_components/SF10Template'

function Student({

    params

}:{

    params: {

        studentId: string

    }

}) {

    const [isOpen, setIsOpen] = useState(false)

    const [sf10Open, setSF10Open] = useState(false)

    const searchParams = useSearchParams();

    const isSHS = searchParams.get('isSHS')

    const sem = searchParams.get('sem')

    const student = useQuery(api.students.getStudentWithDetails,{

        id: params.studentId as Id<'students'>,

        isSHS: isSHS !== null ? isSHS  : undefined,

        sem: sem !== null ? sem : undefined

    })

    const attendance = useQuery(api.attendance.get, {

        studentId: student?.\_id,

        classId: student?.cLass?.\_id

    })

  return (

    <div className='p-3 md:p-10 text-primary'>

        <div className="bg-white w-full-full min-h-screen h-fit p-5 shadow-md">

            <Link

                href="/my-advisees"

                className={cn("h-7 w-7", buttonVariants({

                    variant: "outline",

                    size: "icon",

                }))}>

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

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

            </Link>

            <div className="w-full flex item-start justify-between mb-5 gap-x-2">

                <h1 className='text-lg md:text-xl text-primary font-semibold flex items-center'>Student Report Card</h1>

                <div className="space-y-3">

                    <DropdownMenu>

                        <DropdownMenuTrigger><Button variant={'link'}></Button>

                        <Button variant="outline" size="sm" className="h-7 gap-1">

                                    <File className="h-3.5 w-3.5" />

                                    <span className="sr-only sm:not-sr-only sm:whitespace-nowrap">

                                        School Reports

                                    </span>

                                </Button>

                        </DropdownMenuTrigger>

                        <DropdownMenuContent>

                            <DropdownMenuLabel>School Forms</DropdownMenuLabel>

                            <DropdownMenuSeparator />

                            <DropdownMenuItem onClick={()=>setIsOpen(true)}>

                                School Form 9 (SF9)

                            </DropdownMenuItem>

                            <DropdownMenuItem onClick={()=>setSF10Open(true)}>School Form 10 (SF10)</DropdownMenuItem>

                        </DropdownMenuContent>

                    </DropdownMenu>

                </div>

            </div>

            <div className="grid grid-cols-4 mb-5 text-sm md:text-sm font-semibold">

                {/\* <h1 className='col-span-4 md:col-span-2  text-text'>Student Number: <span className='text-sm md:text-lg font-semibold'> {student?} </span></h1> \*/}

                <h1 className='col-span-4 md:col-span-2  text-text'>LRN: <span className='text-sm  font-normal'> {student?.lrn} </span></h1>

                <h1 className='col-span-4 md:col-span-2  text-text'>Full Name: <span className='text-sm  font-normal'>{student?.lastName}, {student?.firstName} {student?.middleName} </span></h1>

                <h1 className='col-span-4 md:col-span-2  text-text'>Grade Level: <span className='text-sm  font-normal'>{student?.gradeLevel} </span></h1>

                <h1 className='col-span-4 md:col-span-2  text-text'>Adviser: <span className='text-sm font-normal'>{student?.advisor?.firstName} {student?.advisor?.middleName} {student?.advisor?.lastName} </span></h1>

                <h1 className='col-span-4 md:col-span-2  text-text'>Section: <span className='text-sm font-normal'>{student?.sectionDoc.name}</span></h1>

                {student?.gradeLevel === "11"  && (

                    <>

                        <h1 className='col-span-4 md:col-span-2  text-text'>Semister: <span className='text-sm md:text-lg font-normal'> 1st </span></h1>

                    </>

                )}

                {student?.gradeLevel === "12"  && (

                    <>

                        <h1 className='col-span-4 md:col-span-2  text-text'>Semister: <span className='text-sm md:text-lg font-normal'> 1st </span></h1>

                    </>

                )}

            </div>

            {student && (

                <>

                    <SchoolForm9 student={student} setIsOpen={setIsOpen} isOpen={isOpen}/>

                    <SF10Template student={student} setIsOpen={setSF10Open} isOpen={sf10Open}/>

                </>

            )}

            <Tabs defaultValue="grades" className="w-full">

                <TabsList>

                    <TabsTrigger value="grades">Grades</TabsTrigger>

                    <TabsTrigger value="values">Values</TabsTrigger>

                    <TabsTrigger value="attendance">Attendance</TabsTrigger>

                </TabsList>

                {/\* Grades Content \*/}

                {student?.sectionDoc.gradeLevel.level === "Grade 11" || student?.sectionDoc.gradeLevel.level === "Grade 12" ? (

                    <TabsContent value="grades" className='text-xs md:text-sm w-full gap-x-10'>

                        <Tabs defaultValue="1st">

                            <TabsList>

                                <TabsTrigger value="1st">1st Semester</TabsTrigger>

                                <TabsTrigger value="2nd">2nd Semester</TabsTrigger>

                            </TabsList>

                            <TabsContent value='1st'>

                                <SrGradesTemplate sem='1st' student={student} />

                            </TabsContent>

                            <TabsContent value='2nd'>

                                <SrGradesTemplate sem='2nd' student={student} />

                            </TabsContent>

                        </Tabs>

                    </TabsContent>

                ) :(

                    <TabsContent value="grades" className='text-xs md:text-sm w-full gap-x-10'>

                        {student && (

                            <JrGradesTemplate student={student} sf9={false}/>

                        )}

                    </TabsContent>

                )}

                {/\* {Values} \*/}

                <TabsContent value='values'>

                    <Values studentId={student?.\_id as Id<'students'>} classId={student?.cLass?.\_id as Id<'classes'>}/>

                </TabsContent>

                {/\* Attendance Content \*/}

                <TabsContent value="attendance">

                {student && (

                    <Attendance attendance={attendance as Doc<'attendance'>} studentId={student?.\_id as Id<'students'>} classId={student?.cLass?.\_id as Id<'classes'>}/>

                )}

                </TabsContent>

            </Tabs>

        </div>

    </div>

  )

}

export default Student

Promotion of student (Conditionally promoted/Retain/promoted)

import React, { useMemo, useState } from 'react'

import { Dialog, DialogContent, DialogFooter, DialogHeader, DialogTitle, DialogTrigger } from '@/components/ui/dialog'

import { Button } from '@/components/ui/button'

import { StudentWithDetails } from '@/lib/types'

import { FaLevelUpAlt } from 'react-icons/fa'

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip";

import { useMutation, useQuery } from 'convex/react'

import { api } from '../../../../../convex/\_generated/api'

import { toast } from 'sonner'

import { Id } from '../../../../../convex/\_generated/dataModel'

import { cn } from '@/lib/utils'

interface FinalizeGradesDialogProps {

    student:StudentWithDetails,

    averages: {

        classId: Id<'classes'> | undefined,

        subjectName: string,

        finalGrade: number | string

    }[] | null,

    generalAverage: number | string

}

function FinalizeGradesDialog({student, averages, generalAverage}:FinalizeGradesDialogProps ) {

    const [isOpen, setIsOpen ] = useState(false)

    const [isLoading, setIsLoading ] = useState(false)

    const [isDisabled, setIsDisabled ] = useState(false)

    const promoteStudent = useMutation(api.finalGrades.create)

    const isPromoted = useQuery(api.finalGrades.isStudentPromoted,{

        studentId: student.\_id as Id<'students'>,

        sectionId: student.sectionDoc?.\_id as Id<'sections'>,

        schoolYearId: student.sectionDoc?.schoolYearId as Id<'schoolYears'>,

        semester: student.cLass?.semester

    })

    const isShs = Number(student.sectionDoc?.gradeLevel?.level ?? 0) > 10

    const alreadyPromoted = useMemo(() => isPromoted === undefined ? true : isPromoted.hasPromoted ,[isPromoted])

    const studentGradeLevel = student.gradeLevelToEnroll

    const studentSem = student.semesterToEnroll

    const sectionGradeLevel = student.sectionDoc?.gradeLevel?.level

    const classSem = student.cLass?.semester

    const isRetained = useMemo(() => isPromoted?.hasPromoted ? isShs ? studentSem === classSem && true : studentGradeLevel === sectionGradeLevel ? true : false : false ,[studentSem, classSem, isShs, studentGradeLevel,isPromoted, sectionGradeLevel])

    const failedAverages = useMemo(() => {

        if (averages === null) {

            setIsDisabled(true);

            return null;

        }

        const notAllNumbers = averages.some((item) => typeof item.finalGrade === 'string')

        const noScore = averages.some((item)=> item.finalGrade === "")

        if(noScore) {

            setIsDisabled(true)

            return []

        }

        if(notAllNumbers){

            setIsDisabled(true)

            return []

        }

        const allNumberAverages = averages.filter((item) => (typeof item.finalGrade === "number"))

        // Filter averages less than or equal to 74

        const filteredAverages = allNumberAverages.filter(

            (item) => (item.finalGrade as number) <= 74

        );

        // If all averages are valid numbers, ensure `disabled` is false

        setIsDisabled(false);

        return filteredAverages;

    }, [averages, setIsDisabled]);

    const convertedTypesAve = averages?.map(a => {

        return {

            classId: a.classId as Id<'classes'>,

            subjectName: a.subjectName,

            forRemedial: typeof a.finalGrade === 'string' ? false : a.finalGrade <= 74 ? true : false,

            finalGrade: typeof a.finalGrade === 'string' ? 0 : a.finalGrade

        }

    })

    const removeClassIdUndefined = convertedTypesAve?.filter(a=> a.classId !== undefined) ?? []

    const studentName = `${student.firstName} ${student.middleName ? student.middleName: ""} ${student.lastName}`

    const handlePromote = () =>{

        setIsLoading(true)

        toast.promise(promoteStudent({

            sectionId: student.sectionDoc?.\_id as Id<'sections'>,

            advisorId: student.cLass?.teacherId as Id<'users'>,

            studentId: student.\_id,

            subjects: removeClassIdUndefined ?? [],

            schoolYearId: student.cLass?.schoolYearId,

            generalAverage: typeof generalAverage === 'string' ? 0 : generalAverage,

            semester: student.cLass?.semester,

        }),{

            loading: "Promoting student...",

            success: "Student promoted successfully.",

            error: "Failed to promote student."

        })

        setIsOpen(!isOpen)

        setIsLoading(false)

    }

  return (

    <Dialog open={isOpen} onOpenChange={setIsOpen}>

        <TooltipProvider>

            <Tooltip>

                <TooltipTrigger asChild>

                    <DialogTrigger disabled={ alreadyPromoted || isDisabled === true || isLoading} onClick={()=> setIsOpen(!isOpen)}>

                        <Button size="default" disabled={ alreadyPromoted || isDisabled === true || isLoading}  className={cn(isRetained && "bg-red-500 disabled:bg-red-500" ," self-end h-7 gap-1 bg-blue-600 text-white py-2 ml-auto")}>

                            <FaLevelUpAlt className="h-m3.5 w-3.5" />

                            <span className="sr-only sm:not-sr-only sm:whitespace-nowrap">

                                {isPromoted ? isPromoted.studentFinalFGrade?.promotionType : "Promotion"}

                            </span>

                        </Button>

                    </DialogTrigger>

                </TooltipTrigger>

                <TooltipContent className=' bg-white p-5 space-y-2 shadow-md'>

                    <p>To promote the student, you must complete and finalize the final rating for all subjects.</p>

                </TooltipContent>

            </Tooltip>

        </TooltipProvider>

        <DialogContent className=' max-h-screen overflow-auto text-primary'>

        <DialogHeader>

            <DialogTitle className='text-lg'>

                Promote to the next grade level?

            </DialogTitle>

        </DialogHeader>

        {failedAverages && failedAverages.length >= 1 ? (

            isShs ? (

                <>

                    <div className="space-y-3">

                        <p>{studentName} has <strong>failed ({failedAverages.length})</strong> of his subjects.</p>

                        <h1>Failed Subject(s):</h1>

                        {failedAverages.map((fs) => (

                            <div key={fs.subjectName} className="">

                                <h1 className='pl-5'>- <strong>{fs.subjectName} - ({fs.finalGrade})</strong></h1>

                            </div>

                        ))}

                        <p className='text-sm text-justify'>\* Must pass remedial classes for failed competencies in the subjects or learning areas to be allowed to enroll in the next semester. Otherwise, the learner must retake the subjects failed.</p>

                    </div>

                    <DialogFooter>

                        <Button variant={'default'} onClick={handlePromote} className=" text-white">Conditionally Promote</Button>

                    </DialogFooter>

                </>

            ) : (

                failedAverages.length >= 3 ? (

                    <>

                        <div className="space-y-3">

                            <p>{studentName} has <strong>failed ({failedAverages.length})</strong> of his subjects.</p>

                            <h1>Failed Subject(s):</h1>

                            {failedAverages.map((fs) => (

                                <div key={fs.subjectName} className="">

                                    <h1 className='pl-5'>- <strong>{fs.subjectName} - ({fs.finalGrade})</strong></h1>

                                </div>

                            ))}

                            <p className='text-sm text-justify'>\* Did not meet expectations in three or more learning areas. Retained in the same grade level.</p>

                        </div>

                        <DialogFooter>

                            <Button variant={'destructive'} onClick={handlePromote} className=" text-white">Retain</Button>

                        </DialogFooter>

                    </>

                ) : (

                    <>

                        <div className="space-y-3">

                            <p>{studentName} has <strong>failed ({failedAverages.length})</strong> of his subjects.</p>

                            <h1>Failed Subject(s):</h1>

                            {failedAverages.map((fs) => (

                                <div key={fs.subjectName} className="">

                                    <h1 className='pl-5'>- <strong>{fs.subjectName} - ({fs.finalGrade})</strong></h1>

                                </div>

                            ))}

                            <p className='text-sm text-justify'>\*Must enroll in remedial classes for that subject(s) to improve their grade</p>

                        </div>

                        <DialogFooter>

                            <Button variant={'default'} onClick={handlePromote} className=" text-white">Conditionally Promote</Button>

                        </DialogFooter>

                    </>

                )

            )

        ) : (

            <>

                <p className='capitalize'>{studentName} has passed all his subjects.</p>

                <DialogFooter>

                    <Button variant={'default'} onClick={handlePromote} className=" text-white">Promote</Button>

                </DialogFooter>

            </>

        )}

        </DialogContent>

    </Dialog>

  )

}

export default FinalizeGradesDialog

**SF9**

'use client'

import React, { useRef } from 'react'

import JrGradesTemplate from './JrGradesTemplate'

import { StudentWithDetails } from '@/lib/types'

import Values from './Values'

import { Id } from '../../../../../convex/\_generated/dataModel'

import {

    Dialog,

    DialogContent,

    DialogFooter,

    DialogHeader,

    DialogTitle,

  } from "@/components/ui/dialog"

import { Button } from '@/components/ui/button'

import { useReactToPrint } from 'react-to-print'

import { IoMdPrint } from 'react-icons/io'

import SrGradesTemplate from './SrGradesTemplate'

import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs'

import SF9FrontTemplate from './SF9FrontTemplate'

interface SchoolForm9Props {

    student: StudentWithDetails,

    isOpen: boolean,

    setIsOpen: (value: boolean) => void

}

function SchoolForm9({

    student,

    isOpen,

    setIsOpen

}: SchoolForm9Props) {

    const componentRef = useRef(null);

    const gradeLevel = student.sectionDoc?.gradeLevel?.level

    const isSHS = gradeLevel === "Grade 11" || gradeLevel === "Grade 12"

  const reactToPrintContent = () => {

    return componentRef.current;

  };

   const handlePrint = useReactToPrint({

      documentTitle: `School form 9`,

    });

  return (

    <Dialog open={isOpen} onOpenChange={setIsOpen}>

          <DialogContent className='max-w-screen-xl  max-h-screen overflow-auto text-black bg-white p-5'>

          <DialogHeader>

              <DialogTitle className='flex gap-x-3 items-center text-xs font-normal'>Schoo form 9</DialogTitle>

          </DialogHeader>

          <Tabs>

            <TabsList>

              <TabsList>

                <TabsTrigger value='front'>Front</TabsTrigger>

                <TabsTrigger value='back'>Back</TabsTrigger>

              </TabsList>

            </TabsList>

            <TabsContent value='front'>

              <div ref={componentRef} className=''>

                  <SF9FrontTemplate student={student}  />

              </div>

            </TabsContent>

            <TabsContent value='back'>

              <div ref={componentRef} className='grid grid-cols-2 gap-x-10 p-10  text-black bg-white w-fit'>

                  <div className="">

                      {isSHS ? (

                        <>

                        <div className="mb-3">

                          <h1 className='text-center text-xs'>REPORT ON LEARNING PROGRESS AND ACHIEVEMENT</h1>

                          <SrGradesTemplate student={student} sem='1st' sf9={true}/>

                        </div>

                        <div className="">

                          <h1 className='text-center text-xs'>REPORT ON LEARNING PROGRESS AND ACHIEVEMENT</h1>

                          <SrGradesTemplate student={student} sem='2nd' sf9={true}/>

                        </div>

                        </>

                      ): (

                        <JrGradesTemplate student={student} sf9={true}/>

                      )}

                  </div>

                  <div className='text-xs'>

                      <Values sf9={true} studentId={student?.\_id as Id<'students'>} classId={student?.cLass?.\_id as Id<'classes'>} isSHS={isSHS}/>

                  </div>

              </div>

            </TabsContent>

          </Tabs>

          <DialogFooter>

              <Button onClick={()=> setIsOpen(false)} variant={'ghost'}>Cancel</Button>

              <Button variant={'default'} onClick={()=> {handlePrint(reactToPrintContent)}} className='text-white flex gap-x-3'><IoMdPrint />Print</Button>

          </DialogFooter>

          </DialogContent>

    </Dialog>

  )

}

export default SchoolForm9

**SF10**

import { FinalGradesWithDetails, StudentWithDetails } from '@/lib/types'

import Image from 'next/image'

import React, { useRef } from 'react'

import KagawaranNgEdukasyon from '@/../public/images/kagawaran-ng-edukasyon-logo.png'

import DepedLogo from '@/../public/images/Logo-DepEd.png'

import { Dialog, DialogContent, DialogFooter, DialogHeader, DialogTitle } from '@/components/ui/dialog'

import { Button } from '@/components/ui/button'

import { useReactToPrint } from 'react-to-print'

import { IoMdPrint } from 'react-icons/io'

import { Label } from '@/components/ui/label'

import SHSSubjectsTemplate from './SHSSubjectsTemplate'

import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs'

import { Checkbox } from '@/components/ui/checkbox'

import { useQuery } from 'convex/react'

import { api } from '../../../../../convex/\_generated/api'

import { cn } from '@/lib/utils'

import JHSSubjectsTemplate from './JHSSubjectsTemplate'

interface SF10TemplateProps {

    student: StudentWithDetails,

    isOpen: boolean,

    setIsOpen: (value: boolean) => void,

}

function SF10Template({student,isOpen,setIsOpen}: SF10TemplateProps) {

    const componentRef = useRef(null);

    const studentFinalGrades = useQuery(api.finalGrades.getFinalGradesForSF10, {

        studentId: student.\_id

    })

    const shsEnrollments = useQuery(api.enrollments.shsAdmission,{

        studentId: student.\_id

    })

    const esCompleter = shsEnrollments?.elegibilityForEnrollment?.esCompleter

    const hsCompleter = shsEnrollments?.elegibilityForEnrollment?.hsCompleter

    const jhsCompleter = shsEnrollments?.elegibilityForEnrollment?.jhsCompleter

    const dateOfGraduation = shsEnrollments?.elegibilityForEnrollment?.date

    const schoolGraduated = shsEnrollments?.elegibilityForEnrollment?.school

    const schoolGraduatedAddress = shsEnrollments?.elegibilityForEnrollment?.schoolAddress

    const peptPasser = shsEnrollments?.elegibilityForEnrollment?.peptPasser

    const alsPasser = shsEnrollments?.elegibilityForEnrollment?.alsPasser

    const others = shsEnrollments?.elegibilityForEnrollment?.others

    const dateOfExam = shsEnrollments?.elegibilityForEnrollment?.dateOfExam

    const communityLearningCenter = shsEnrollments?.elegibilityForEnrollment?.communityLearningCenter

    const schoolId = shsEnrollments?.elegibilityForEnrollment?.schoolId

    const gradeLevel = student.sectionDoc?.gradeLevel?.level

    const isSHS = gradeLevel === "Grade 11" || gradeLevel === "Grade 12"

  const reactToPrintContent = () => {

    return componentRef.current;

  };

   const handlePrint = useReactToPrint({

      documentTitle: `School form 10`,

    });

  const formatDateString = (dateString: string) => {

    const date = new Date(dateString);

    const month = date.getMonth() + 1;

    const day = date.getDate();

    const year = date.getFullYear();

    return `${month < 10 ? '0' : ''}${month}/${day < 10 ? '0' : ''}${day}/${year}`;

  };

  return (

    <Dialog open={isOpen} onOpenChange={setIsOpen}>

    <DialogContent className='max-w-screen-xl  max-h-screen overflow-auto text-black bg-white p-5'>

    <DialogHeader>

        <DialogTitle className='flex gap-x-3 items-center text-xs font-normal'>School form 10</DialogTitle>

    </DialogHeader>

    <Tabs>

            <TabsList>

              <TabsTrigger value='front'>Front</TabsTrigger>

              <TabsTrigger value='back'>Back</TabsTrigger>

            </TabsList>

            <TabsContent value='front'>

            <div ref={componentRef} className='p-5 text-black text-sm'>

                <div className="grid grid-cols-12 h-fit text-sm ">

                    <div className="col-span-2 flex justify-center">

                        <Image src={KagawaranNgEdukasyon} alt='' className='object-contain size-14 '/>

                    </div>

                    <div className={cn(isSHS ? "text-sm": "text-xs","col-span-8 ")}>

                        <h1 className='uppercase text-center'>Republic of the Philippines</h1>

                        <h1 className='uppercase text-center'>Department of Education</h1>

                        {isSHS ? (

                            <h1 className='uppercase text-center font-semibold text-sm'>Senior High School Student Permanent Record</h1>

                        ):(

                            <div className="text-xs">

                                <h1 className='uppercase text-center font-semibold text-xs'>Learner&apos;s Permanent Academic Record for Junior High School(SF10)</h1>

                                <p className='italic text-[0.6rem] text-center'>(Formerly Form 137)</p>

                            </div>

                        )}

                    </div>

                    <div className="col-span-2 flex  justify-center">

                        <Image src={DepedLogo} alt=''  className='object-contain w-20 h-14 '/>

                    </div>

                </div>

                <div className="pb-1">

                    <h1 className='uppercase text-center font-semibold bg-gray-300 text-[0.7rem]'>Learner&apos;s Information</h1>

                    {isSHS ? (

                        <>

                        <div className="grid grid-cols-3 gap-x-2 uppercase text-[0.55rem] ">

                            <h1 className=' flex'>Last Name: <span className='border-b-black border-b flex-1 px-1'>{student.lastName} {student.extensionName}</span></h1>

                            <h1 className=' flex'>First Name: <span className='border-b-black border-b flex-1 px-1'>{student.firstName}</span></h1>

                            <h1 className=' flex'>MIddle Name: <span className='border-b-black border-b flex-1 px-1'>{student.middleName}</span></h1>

                        </div>

                        <div className="grid grid-cols-12 gap-x-2 text-[0.55rem] mt-[-2px]">

                          <h1 className='flex col-span-2'>LRN: <span className='border-b-black border-b flex-1 px-1'>{student.lrn}</span></h1>

                          <h1 className='flex col-span-4'>Date of Birth (MM/DD/YYYY): <span className='border-b-black border-b flex-1 px-1'>{formatDateString(student.birthDate)}</span></h1>

                          <h1 className='flex col-span-1'>Sex: <span className='border-b-black border-b flex-1 px-1'>{student.sex}</span></h1>

                          <h1 className='flex col-span-5'>Date of SHS Admission (MM/DD/YYYY): <span className='border-b-black border-b flex-1 px-1'>{student.birthDate}</span></h1>

                      </div>

                      </>

                    ):(

                    <>

                        <div className="grid grid-cols-4 gap-x-2 uppercase text-[0.6rem] my-1">

                            <h1 className=' flex'>Last Name: <span className='border-b-black border-b flex-1 px-1 text-center'>{student.lastName} {student.extensionName}</span></h1>

                            <h1 className=' flex'>First Name: <span className='border-b-black border-b flex-1 px-1'  text-center>{student.firstName}</span></h1>

                            <h1 className=' flex'>Name EXTN.(Jr,I,II): <span className='border-b-black border-b flex-1 px-1  text-center'>{student.extensionName}</span></h1>

                            <h1 className=' flex'>MIddle Name: <span className='border-b-black border-b flex-1 px-1  text-center'>{student.middleName}</span></h1>

                        </div>

                        <div className="grid grid-cols-12 gap-x-2  mt-[-2px] text-[0.6rem]">

                            <h1 className='flex col-span-5 capitalize'>Learner Reference Number (LRN): <span className='border-b-black border-b flex-1 px-1  text-center'>{student.lrn}</span></h1>

                            <h1 className='flex col-span-4'>Birthdate (MM/DD/YYYY): <span className='border-b-black border-b flex-1 px-1  text-center'>{formatDateString(student.birthDate)}</span></h1>

                            <h1 className='flex col-span-3'>Sex: <span className='border-b-black border-b flex-1 px-1  text-center'>{student.sex}</span></h1>

                        </div>

                     </>

                    )}

                </div>

                <div className="">

                    <h1 className='uppercase text-center bg-gray-300 text-[0.7rem] font-semibold'>Eligibility for SHS Enrollment</h1>

                    <div className={cn(isSHS ? "border-none p-0 " : "border border-black p-1 mt-1", "")}>

                        {isSHS ? (

                            <div className="grid grid-cols-12 mt-1">

                            <div className="col-span-4 flex gap-x-5 text-[0.55rem]">

                                <Checkbox

                                    id="highSchoolCompleter"

                                    className='rounded-none border-black border size-4 ml-0 '

                                                                />

                                <Label htmlFor="highSchoolCompleter" className='ml-[-15px] text-[0.6rem]'>High School Completer\*</Label>

                                <h1 className='flex w-1/3 items-baseline'>Gen. Ave: <input type="text" value={hsCompleter ? hsCompleter.genAve : ""} className='border-b border-b-black flex-1 w-1/3 px-3 h-3' /></h1>

                            </div>

                            <div className="col-span-5 flex gap-x-5 text-[0.55rem]">

                                <Checkbox

                                    id="juniorHighSchoolCompleter"

                                    className='rounded-none checked:bg-transparent bg-transparent'

                                    disabled={false}

                                    />

                                <Label htmlFor="juniorHighSchoolCompleter" className='ml-[-15px] text-[0.6rem]'>Junior High School Completer\*</Label>

                                <h1 className='flex w-1/3 items-baseline'>Gen. Ave: <input type="text" value={jhsCompleter ? jhsCompleter.genAve : ""} className='border-b border-b-black flex-1 w-1/3 px-3 h-3' /></h1>

                            </div>

                        </div>

                        ):(

                        <div className="grid grid-cols-12 mt-1 text-[0.6rem]">

                            <div className="col-span-4 flex gap-x-5 text-[0.6rem] mb-1">

                                <Checkbox

                                    id="elemSchoolCompleter"

                                    className='rounded-none border-black border size-4 ml-0 '

                                />

                                <Label htmlFor="elemSchoolCompleter" className='ml-[-15px] text-[0.6rem]'>Elementary School Completer\*</Label>

                            </div>

                            <h1 className='col-span-4 flex gap-x-3 w-1/2 items-baseline '>Gen. Ave: <input type="text" value={esCompleter ? esCompleter.genAve : ""} className='border-b border-b-black flex-1 text-center w-1/3 px-3 h-3' /></h1>

                            <h1 className='col-span-4 flex gap-x-3 items-baseline '>Citation(if Any): <input type="text" value={esCompleter ? esCompleter.citation : ""} className='border-b border-b-black text-center  flex-1 w-1/3 px-3 h-3' /></h1>

                        </div>

                        )}

                        {isSHS ? (

                        <div className="grid grid-cols-12 text-[0.55rem]  mt-[-4px]">

                            <h1 className='flex gap-x-1 items-baseline col-span-5'>Date of Gradeuation/Completion (MM/DD/YYYY): <input type="text" value={dateOfGraduation ? formatDateString(dateOfGraduation) : ""} className='border-b border-b-black flex-1  h-3 w-1/3 px-3' /></h1>

                            <h1 className='flex gap-x-1 items-baseline col-span-3'>Name of School: <input type="text" value={schoolGraduated ? schoolGraduated : ""} className='border-b border-b-black line-clamp-1 flex-1 w-1/3 px-3 h-3' /></h1>

                            <h1 className='flex gap-x-1 items-baseline col-span-4'>School Address: <input type="text" value={schoolGraduatedAddress ? schoolGraduatedAddress : ""} className='border-b line-clamp-1 border-b-black flex-1 w-1/3 px-3  h-3' /></h1>

                        </div>

                        ):(

                        <div className="grid grid-cols-12 gap-x-5 text-[0.6rem] ">

                            <h1 className='flex gap-x-1 items-baseline col-span-5 pl-5'>Name of Elementary School: <input type="text" value={schoolGraduatedAddress ? schoolGraduatedAddress : ""} className='border-b border-b-black flex-1  h-3 w-1/3 px-3' /></h1>

                            <h1 className='flex gap-x-1 items-baseline col-span-3'>School ID: <input type="text" value={schoolId ? schoolId : ""} className='border-b border-b-black line-clamp-1 flex-1 w-1/3 px-3 h-3' /></h1>

                            <h1 className='flex gap-x-1 items-baseline col-span-4'>Address of School: <input type="text" value={schoolGraduatedAddress ? schoolGraduatedAddress : ""} className='border-b line-clamp-1 border-b-black flex-1 w-1/3 px-3  h-3' /></h1>

                        </div>

                        )}

                    </div>

                    {!isSHS && (

                        <h1 className='text-[0.6rem] '>Other Credential Presented</h1>

                    )}

                    <div className={cn(!isSHS && "px-2" ,"grid grid-cols-12 pt-1  mt-[-2px]")}>

                        <div className="col-span-3 flex gap-x-5 text-[0.6rem]">

                            <Checkbox

                                id="peptPasser"

                                className='rounded-none ml-0 '

                                />

                            <Label htmlFor="peptPasser" className='ml-[-15px] text-[0.6rem]'>PEPT Passer\*</Label>

                            <h1 className='flex w-1/3 items-baseline'>Rating: <input type="text" className='border-b border-b-black flex-1 w-1/3 px-3 h-3' /></h1>

                        </div>

                        <div className="col-span-3 flex gap-x-5 text-[0.6rem]">

                            <Checkbox

                                id="alsAEPasser"

                                className='rounded-none ml-0 '

                                />

                            <Label htmlFor="alsAEPasser" className='ml-[-15px] text-[0.6rem]'>ALS A&E Passer\*</Label>

                            <h1 className='flex w-1/3 items-baseline'>Rating: <input type="text" className='border-b border-b-black flex-1 w-1/3 px-3 h-3' /></h1>

                        </div>

                        <div className="col-span-6 flex text-[0.6rem]">

                            <div className='flex gap-x-5 col-span-2'>

                            <Checkbox

                                    id="others"

                                    className='rounded-none ml-0'

                                />

                                <Label htmlFor="others" className='ml-[-15px] text-[0.6rem]'>Others\*</Label>

                            </div>

                            <h1 className='flex w-full items-baseline'>(Pls. specify): <input type="text" value={others ? others.description : ""} className='border-b border-b-black flex-1 w-1/3 px-3 h-3' /></h1>

                        </div>

                    </div>

                    <div className="grid grid-cols-12 gap-x-5 text-[0.6rem] mt-[-4px]">

                        <h1 className='flex col-span-6 items-baseline'>Date of Examination/Assessment (MM/DD/YYYY): <input type="text" value={dateOfExam ? formatDateString(dateOfExam) : ""} className='border-b border-b-black flex-1 px-3 h-3' /></h1>

                        <h1 className='flex col-span-6 items-baseline'>Name and Address of Community Learning Center: <input type="text" value={communityLearningCenter ? communityLearningCenter : ""} className='border-b border-b-black flex-1 px-3 h-3' /></h1>

                    </div>

                    {isSHS ? (

                        <>

                            <div className="grid grid-cols-12 italic text-[0.5rem] mt-[-1px]">

                                <h4 className='col-span-6 italic'>\*High School Completers are students who graduated from secondary school under the old curriculum</h4>

                                <h4 className='col-span-6 italic '>\*\*\*ALS A&E - Alternative Learning System Accreditation and equivalency Test for JHS</h4>

                            </div>

                            <h4 className=' italic text-[0.5rem] mt-[-6px]'>\*\*PEPT - Philippine educational Placement Test for JHS</h4>

                        </>

                    ) : (

                        <div className="mt-2"></div>

                    )}

                </div>

                {isSHS ? (

                    <div className="">

                        <div className="mb-5">

                            <SHSSubjectsTemplate student={student} finalGrades={studentFinalGrades as FinalGradesWithDetails[]} level={"11"} sem='1st'/>

                        </div>

                        <SHSSubjectsTemplate student={student} finalGrades={studentFinalGrades as FinalGradesWithDetails[]} level={"11"} sem={'2nd'}/>

                    </div>

                ): (

                    <div className="">

                        <h1 className='uppercase text-center bg-gray-300 text-[0.7rem] font-semibold'>Scholastic Record</h1>

                        <div className="mb-3 mt-3">

                            <JHSSubjectsTemplate student={student} finalGrades={studentFinalGrades as FinalGradesWithDetails[]} level='7' sf10/>

                        </div>

                        <JHSSubjectsTemplate student={student} finalGrades={studentFinalGrades as FinalGradesWithDetails[]} level='8' sf10/>

                        <div className="mt-2 border border-black p-1">

                            <h1 className='text-center font-semibold text-xs mb-3'>Certification</h1>

                            <div className="grid grid-cols-12 gap-x-2">

                                <h1 className='text-[0.6rem] col-span-5 flex items-baseline'>I CERTIFY that this is true record of <input type="text" className='border-b border-b-black flex-1 w-3 text-center' /></h1>

                                <h1 className='text-[0.6rem] col-span-2 flex items-baseline'>with LRN <input type="text" value={student.lrn} className='border-b border-b-black flex-1 w-3 text-center' /></h1>

                                <h1 className='text-[0.6rem] col-span-5 flex items-baseline'>and that he/she is eligible for admission to Grade <input type="number" className='border-b border-b-black flex-1 w-3 text-center' />.</h1>

                            </div>

                            <div className="grid grid-cols-12 gap-x-2">

                                <h1 className='text-[0.6rem] col-span-5 flex items-baseline'>Name of school: <input type="text" className='border-b border-b-black flex-1 w-3' /></h1>

                                <h1 className='text-[0.6rem] col-span-3 flex items-baseline'>School ID: <input type="number" className='border-b border-b-black flex-1 w-3' /></h1>

                                <h1 className='text-[0.6rem] col-span-4 flex items-baseline'>Last School Year Attended: <input type="text" className='border-b border-b-black flex-1 w-3' /></h1>

                            </div>

                            <div className="grid grid-cols-12 gap-x-5 text-[0.6rem]">

                                <input type="text" className='col-span-4 border-b border-b-black text-center mt-3' />

                                <input type="text" className='col-span-5 border-b border-b-black text-center mt-3' />

                                <h1 className='col-span-3 mt-3'></h1>

                                <h1 className='col-span-4 text-center '>Date</h1>

                                <h1 className='col-span-5 text-center'>Name of Principal/School Head over Printed Name</h1>

                                <h1 className='col-span-3 text-center'>(affix School seal heare)</h1>

                            </div>

                        </div>

                    </div>

                )}

            </div>

            </TabsContent>

            <TabsContent value='back'>

                {isSHS ? (

                    <div ref={componentRef} className='p-5 text-black text-sm'>

                      <div className="mb-5">

                          <SHSSubjectsTemplate student={student} level='12' sem='1st'/>

                      </div>

                      <SHSSubjectsTemplate student={student} level='12' sem='2nd'/>

                      <div className='my-2 w-full h-2 bg-gray-300'></div>

                      <div className="grid grid-cols-12 gap-x-5 text-[0.6rem] font-semibold">

                          <h1 className='col-span-10 flex items-baseline'>Track/Strand Accomplished: <input type="text" className='border-b-black border-b flex-1 px-3 uppercase h-5' /></h1>

                          <h1 className='col-span-2 flex items-baseline'>SHS Gen Average: <input type="number" className='border-b-black border-b flex-1 px-3 w-2 h-5 uppercase' /></h1>

                      </div>

                      <div className="grid grid-cols-12 gap-x-5 text-[0.6rem]  font-semibold">

                          <h1 className='col-span-8 flex items-baseline'>Awards/Honors Received: <input type="text" className='border-b-black border-b flex-1 px-3 uppercase h-5' /></h1>

                          <h1 className='col-span-4 flex items-baseline'>Date of SHS Graduation (MM/DD/YYYY): <input type="text" className='border-b-black border-b flex-1 w-1 h-5 uppercase' /></h1>

                      </div>

                      <div className="grid grid-cols-12 gap-x-5 text-[0.6rem]  font-semibold">

                          <h1 className='col-span-6 flex items-baseline'>Certified By:</h1>

                          <h1 className='col-span-6 flex items-baseline'>Place School Seal Here:</h1>

                      </div>

                      <div className="grid grid-cols-2">

                          <div className="pr-5">

                              <div className="grid grid-cols-12 text-[0.6rem] gap-x-10">

                                  <input type="text" className='border-b-black border-b col-span-6 text-center uppercase'/>

                                  <input type="text" className='border-b-black border-b col-span-4 text-center uppercase' />

                                  <h1 className='text-center col-span-6'>Signature of School Head Over Printed Name</h1>

                                  <h1 className='text-center col-span-4'>Date</h1>

                              </div>

                              <div className="p-2 border border-black text-[0.5rem]">

                                  <h1 className='text-xs font-semibold'>Note:</h1>

                                  <p className='leading-3'>This permanent record or a photocopy of this permanent record that bears the seal of the school and the original signature in ink of the school head shall be considered valid for all legal purposes. Any erasure or alteration made on this copy should be validated by the School head.</p>

                                  <p className='leading-3'>If the student transfers to another school, the originating school should produce one (1) certified true copy of this permanent record for safekeeping. The receiving school shall continue filling up the original form.</p>

                                  <p className='leading-3'>Upon graduation, the school from which the student graduated should keep the original form and produce one (1) certified true copy for the Division office.</p>

                              </div>

                          </div>

                          <div className="border-l-black border-l">

                          </div>

                      </div>

                  </div>

                ): (

                    <div ref={componentRef} className='p-5 text-black text-sm'>

                        <div className="text-[0.55rem] flex justify-between items-center">

                            <h1 className='uppercase'>SF10 JHS</h1>

                            <h1 className='uppercase'>Page 2 of <input type="number" className='border-b border-b-black text-center h-3' /></h1>

                        </div>

                        <div className="">

                            <JHSSubjectsTemplate student={student} finalGrades={studentFinalGrades as FinalGradesWithDetails[]} level='9' sf10/>

                            <JHSSubjectsTemplate student={student} finalGrades={studentFinalGrades as FinalGradesWithDetails[]} level='10' sf10/>

                            <JHSSubjectsTemplate student={student} finalGrades={studentFinalGrades as FinalGradesWithDetails[]} sf10/>

                        </div>

                        <h1 className="font-bold text-[0.6rem] leading-4">For Transfer Out / JHS Completer only</h1>

                        <div className=" border border-black p-1">

                            <h1 className='text-center font-semibold text-xs mb-1'>Certification</h1>

                            <div className="grid grid-cols-12 gap-x-2">

                                <h1 className='text-[0.6rem] col-span-6 flex items-baseline leading-3'>I CERTIFY that this is true record of <input type="text" className='border-b border-b-black flex-1 w-3 text-center h-3' /></h1>

                                <h1 className='text-[0.6rem] col-span-2 flex items-baseline leading-3'>with LRN <input type="text" value={student.lrn} className='border-b border-b-black flex-1 w-3 text-center h-3' /></h1>

                                <h1 className='text-[0.6rem] col-span-4 flex items-baseline leading-3'>and that he/she is eligible for admission to Grade <input type="number" className='border-b border-b-black flex-1 w-3 text-center h-3' />.</h1>

                            </div>

                            <div className="grid grid-cols-12 gap-x-2 mt-1">

                                <h1 className='text-[0.6rem] col-span-5 flex items-baseline leading-3'>Name of school: <input type="text" className='border-b border-b-black flex-1 w-3 h-3' /></h1>

                                <h1 className='text-[0.6rem] col-span-3 flex items-baseline leading-3'>School ID: <input type="number" className='border-b border-b-black flex-1 w-3 h-3' /></h1>

                                <h1 className='text-[0.6rem] col-span-4 flex items-baseline leading-3'>Last School Year Attended: <input type="text" className='border-b border-b-black flex-1 w-3 h-3' /></h1>

                            </div>

                            <div className="grid grid-cols-12 gap-x-5 text-[0.6rem]">

                                <input type="text" className='col-span-4 border-b border-b-black text-center mt-3' />

                                <input type="text" className='col-span-5 border-b border-b-black text-center mt-3' />

                                <h1 className='col-span-3 mt-3'></h1>

                                <h1 className='col-span-4 text-center '>Date</h1>

                                <h1 className='col-span-5 text-center'>Name of Principal/School Head over Printed Name</h1>

                                <h1 className='col-span-3 text-center'>(affix School seal heare)</h1>

                            </div>

                        </div>

                    </div>

                )}

            </TabsContent>

    </Tabs>

         <DialogFooter>

         <Button onClick={()=> setIsOpen(false)} variant={'ghost'}>Cancel</Button>

         <Button variant={'default'} onClick={()=> {handlePrint(reactToPrintContent)}} className='text-white flex gap-x-3'><IoMdPrint />Print</Button>

     </DialogFooter>

     </DialogContent>

</Dialog>

  )

}

export default SF10Template

Class Tab

'use client'

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

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from '@/components/ui/select';

import SectionsCard from './\_components/SectionsCard'

import Loading from '@/app/components/Loading'

import { ClassesWithDetails } from '@/lib/types'

import { useQuery } from 'convex/react';

import { api } from '../../../../convex/\_generated/api';

import { Id } from '../../../../convex/\_generated/dataModel';

import { Label } from '@/components/ui/label';

function ClassPage() {

  const sy = useQuery(api.schoolYear.get)

  const latestSY = sy ? sy[0].\_id : undefined

  const [selectedSY , setSelectedSY] = useState<Id<'schoolYears'> | undefined>(latestSY)

  useEffect(() => {

    if (latestSY) {

        setSelectedSY(latestSY)

    }

}, [latestSY])

  const classes = useQuery(api.classes.getTeacherClassesWithSchoolYear, {

    schoolYearId: selectedSY

  })

  return (

    <div className=' m-5 text-primary'>

        <h1 className="font-semibold font-serif text-lg text-center">Classes</h1>

        <div className='flex items-center justify-end gap-x-1'>

          <Label >School Year : </Label>

          <Select defaultValue={selectedSY} onValueChange={(value)=>{setSelectedSY(value as Id<'schoolYears'>)}}>

              <SelectTrigger className='w-40 bg-white'>

                  <SelectValue placeholder="Select school year" />

              </SelectTrigger>

              <SelectContent>

                {sy && sy.map((sy) =>(

                  <SelectItem key={sy.\_id} value={sy.\_id}>{sy.sy}</SelectItem>

                ))}

              </SelectContent>

          </Select>

        </div>

        {classes === undefined ? (<Loading/>) : classes && classes.length > 0 ? (

          <div className=' m-5 grid grid-cols-1 md:grid-cols-1 lg:grid-cols-2 xl:grid-cols-3 gap-5 md:gap-10'>

            {classes.map((cl) =>(

              <SectionsCard

                key={cl?.\_id}

                cls={cl as ClassesWithDetails}/>

            ))}

          </div>

        ) :(

          <div className="text-gray-500 text-sm text-center py-4">No assigned subject yet.</div>

        )}

    </div>

  )

}

export default SectionPage

**RECSYS (Recommender System)**

**Dialog when the subject submit the scores of student (Interventions)**  
'use client'

import { Button } from '@/components/ui/button'

import { Dialog, DialogContent, DialogDescription, DialogFooter, DialogHeader, DialogTitle } from '@/components/ui/dialog'

import React from 'react'

import { calculateInitialGrade, calculatePercentageScore, calculateTotalScore, calculateWeightedScore, convertToTransmutedGrade } from '@/lib/utils';

import { StudentsWithClassRecord } from '@/lib/types';

import { useMutation, useQuery } from 'convex/react';

import { api } from '../../../../../convex/\_generated/api';

import { toast } from 'sonner';

import { Id } from '../../../../../convex/\_generated/dataModel';

interface SubmitGradesDialogProps {

  isSGOpen: boolean,

  setIsSGOpen: (value:boolean) => void;

  studentsWithDetails: StudentsWithClassRecord

  subjectId: Id<"subjects"> | undefined

}

function SubmitGradesDialog({isSGOpen, setIsSGOpen ,studentsWithDetails, subjectId}: SubmitGradesDialogProps) {

  const submitQuarterlyGrade =  useMutation(api.quarterlyGrades.create)

  const appliedGW = useQuery(api.appliedGradeWeigths.get, {subjectId})

 if (studentsWithDetails.classRecords.length === 0) {

    return <></>;

  }

  const classRecord = studentsWithDetails.classRecords[0]

  const subject = studentsWithDetails.classRecords[0].cLass.subject

  const section = studentsWithDetails.sectionDoc

  const writtenWeight = appliedGW?.written ?? 0

  const performanceWeight = appliedGW?.performance ?? 0

  const examWeight = appliedGW?.exam ?? 0

  const subComponent = classRecord.subComponent

  const studentName = `${studentsWithDetails.lastName}, ${studentsWithDetails.firstName} ${studentsWithDetails.middleName}`

  const learningMode = appliedGW?.learningMode ?? "Face to face"

  const written = studentsWithDetails.classRecords[0].written

  const performance = studentsWithDetails.classRecords[0].performance

  const quarterlyExam = studentsWithDetails.classRecords[0].quarterlyExam

  const totalWritten = written.reduce((sum, assessment) => sum + assessment.highestScore, 0);

  const totalPerformance = performance.reduce((sum, assessment) => sum + assessment.highestScore, 0);

  const totalQE = quarterlyExam.reduce((sum, assessment) => sum + assessment.highestScore, 0);

  const transmutedGrade = convertToTransmutedGrade(

    calculateInitialGrade(

        calculateWeightedScore(calculatePercentageScore(calculateTotalScore(written), totalWritten ?? 0), writtenWeight?? 0),

        calculateWeightedScore(calculatePercentageScore(calculateTotalScore(performance), totalPerformance ?? 0), performanceWeight?? 0),

        calculateWeightedScore(calculatePercentageScore(calculateTotalScore(quarterlyExam), totalQE ?? 0), examWeight?? 0)

    ),

    Number(section?.gradeLevel?.level),

    learningMode,

    subject?.subjectCategory?.toLowerCase()

  )

  const needsIntervention = transmutedGrade <= 74

  function handleSubmit (){

    toast.promise(submitQuarterlyGrade({

      studentId: studentsWithDetails.\_id,

      gradeLevel: Number(section?.gradeLevel?.level.replace("Grade", "")),

      classId: classRecord.classId,

      quarter: classRecord.quarter,

      quarterlyGrade: transmutedGrade, // score

      needsIntervention: needsIntervention,

      classRecordId: classRecord.\_id,

      subComponent: subComponent

    }),{

    loading: 'Submitting grades...',

    success: 'Grades submitted successfully',

    error: 'Failed to submit grades :('

  })

  setIsSGOpen(false)

  }

  return (

    <Dialog open={isSGOpen}>

        <DialogContent>

        <DialogHeader>

            <DialogTitle className='text-primary'>Submit Grades?</DialogTitle>

            <DialogDescription className='space-y-2'>

              <div className="">

                <h1 className='text-sm font-semibold'>{studentName}</h1>

              </div>

              <div className="">

                Quarterly Grade: <span className={'text-primary'}>{transmutedGrade}</span>

                {transmutedGrade <= 74 ? (<span className='font-semibold uppercase ml-5 text-red-500'>Needs Intervention</span>) : (<span className='font-semibold uppercase ml-5 text-green-500'>Passed</span>)}

              </div>

            </DialogDescription>

        </DialogHeader>

        <DialogFooter>

            <Button onClick={()=> {

              setIsSGOpen(false)

            }} variant={'ghost'} className='text-primary'>No</Button>

            <Button  variant={'default'} onClick={handleSubmit} className='text-white'>Yes</Button>

        </DialogFooter>

        </DialogContent>

    </Dialog>

  )

}

export default SubmitGradesDialog

**List of students that need interventions per quarter**

'use client'

import React, { useState } from 'react'

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"

import { DataTable } from '@/components/data-table'

import { forImprovementsColumns } from './studentData'

import { useQuery } from 'convex/react'

import { api } from '../../../../../convex/\_generated/api'

import { ClassesWithDetails, SectionWithGradeLevel, StudentsWithQuarterlyGrades } from '@/lib/types'

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from '@/components/ui/select'

function NeedsImprovement({

    section,

    cls

}: {

    section: SectionWithGradeLevel

    cls: ClassesWithDetails

}) {

    const [selectedSubComponent ,setSelectedSubComponent] = useState<string>("Music")

    const isMapeh = cls.subject?.name.toLocaleUpperCase() === 'MAPEH'

    const studentNeedsIntervention = useQuery(api.quarterlyGrades.needIntervention, {

        gradeLevel: Number(section?.gradeLevel?.level.replace("Grade", "")),

        classId: cls.\_id,

        needsIntervention: true

    })

    const getUniqueQuarterlyGrades = (

        quarter: string,

        studentNeedsIntervention: StudentsWithQuarterlyGrades[]

      ) => {

        const filteredGrades = studentNeedsIntervention.map((student) => {

            if(isMapeh){

                const quarterlyGrade = student.quarterlyGrades.find(grade => grade.quarter === quarter && grade.subComponent === selectedSubComponent)

                if(quarterlyGrade === undefined) return null

                return {

                    ...student,

                    quarterlyGrade: quarterlyGrade

                }

            } else {

                const quarterlyGrade = student.quarterlyGrades.find(grade => grade.quarter === quarter)

                if(quarterlyGrade === undefined) return null

                return {

                    ...student,

                    quarterlyGrade: quarterlyGrade

                }

            }

        });

        const removeNull = filteredGrades.filter(g => g !== null)

        return removeNull;

      };

      const semester = cls.semester

      const firstQuarterIntervention = getUniqueQuarterlyGrades("1st", studentNeedsIntervention ?? []);

      const secondQuarterIntervention = getUniqueQuarterlyGrades("2nd", studentNeedsIntervention ?? []);

      const thirdQuarterIntervention = getUniqueQuarterlyGrades("3rd", studentNeedsIntervention ?? []);

      const fourthQuarterIntervention = getUniqueQuarterlyGrades("4th", studentNeedsIntervention ?? []);

  return (

    <div>

        <Tabs defaultValue={semester ? semester === "2nd" ? "3rd" : "1st" : "1st"} className='w-full'>

            <div className="flex justify-between ">

                <TabsList  className='space-x-3 bg-transparent '>

                {semester ? semester === "1st" ? (

                    <>

                    <TabsTrigger value='1st'  className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >1st</TabsTrigger>

                    <TabsTrigger value='2nd' className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >2nd</TabsTrigger>

                    </>

                ): (

                    <>

                    <TabsTrigger value='3rd' className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >3rd</TabsTrigger>

                    <TabsTrigger value='4th' className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >4th</TabsTrigger>

                    </>

                ): (

                    <>

                        <TabsTrigger value='1st'  className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >1st</TabsTrigger>

                        <TabsTrigger value='2nd' className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >2nd</TabsTrigger>

                        <TabsTrigger value='3rd' className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >3rd</TabsTrigger>

                        <TabsTrigger value='4th' className='font-medium text-xs md:text-md shadow-md border-b-2 data-[state=active]:border-b-primary data-[state=active]:text-primary' >4th</TabsTrigger>

                    </>

                )}

                </TabsList>

                {isMapeh && <SelectMapeh setSelectedSubComponent={setSelectedSubComponent} />}

            </div>

            <TabsContent value="1st">

                <DataTable

                    columns={forImprovementsColumns}

                    data={firstQuarterIntervention ?? []}

                    filter='fullName'

                    placeholder='by name'

                />

            </TabsContent>

            <TabsContent value="2nd">

                <DataTable

                    columns={forImprovementsColumns}

                    data={secondQuarterIntervention ?? []}

                    filter='fullName'

                    placeholder='by name'

                />

            </TabsContent>

            <TabsContent value="3rd">

                <DataTable

                    columns={forImprovementsColumns}

                    data={thirdQuarterIntervention ?? []}

                    filter='fullName'

                    placeholder='by name'

                />

            </TabsContent>

            <TabsContent value="4th">

                <DataTable

                    columns={forImprovementsColumns}

                    data={fourthQuarterIntervention ?? []}

                    filter='fullName'

                    placeholder='by name'

                />

            </TabsContent>

        </Tabs>

    </div>

  )

}

export default NeedsImprovement

const SelectMapeh = ({setSelectedSubComponent}:{

    setSelectedSubComponent: (value: string) => void;

}) =>{

    return(

        <div className="col-span-1">

            <Select defaultValue={'Music'} onValueChange={(value)=>{setSelectedSubComponent(value)}}>

                <SelectTrigger className='w-40'>

                    <SelectValue placeholder="Select component" />

                </SelectTrigger>

                <SelectContent>

                    <SelectItem value="Music">Music</SelectItem>

                    <SelectItem value="Arts">Arts</SelectItem>

                    <SelectItem value="Physical Education">Physical Education</SelectItem>

                    <SelectItem value="Health">Health</SelectItem>

                </SelectContent>

            </Select>

        </div>

    )

}

**Assessments tab**

'use client'

import WrittenWorks from './\_components/WrittenWorks'

import PerformanceTask from './\_components/PerformanceTask'

import QuarterExam from './\_components/QuarterExam'

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"

import {

    Card,

    CardContent,

  } from "@/components/ui/card"

import AssignSubjects from './\_components/AssignSubjects'

import { useQuery } from 'convex/react'

import { api } from '../../../../convex/\_generated/api'

import { AssessmentTypes } from '@/lib/types'

function AssessmentPage() {

    const schoolYears = useQuery(api.schoolYear.get)

    const latestSY = schoolYears && schoolYears?.length > 0 ? schoolYears[0].\_id : undefined

    const data = useQuery(api.assessments.getAssessments, {sy: latestSY})

    const getAssignSubjects = useQuery(api.subjects.getAssignSubjects, {sy: latestSY})

  return (

    <div className='m-5 '>

    <Tabs defaultValue="subjects" className="max-w-full">

        <TabsList className='text-primary text-xl tracking-wide mb-0 gap-0 rounded-b-none bg-transparent px-0'>

            <TabsTrigger value="subjects" className='font-medium rounded-b-none rounded-tr-none bg-white  text-lg data-[state=active]:border-b-4 border-b-primary data-[state=active]:text-primary data-[state=active]:bg-white'>Subjects</TabsTrigger>

            <TabsTrigger value="assessments" className='font-medium rounded-b-none rounded-tl-none bg-white  text-lg data-[state=active]:border-b-4 border-b-primary data-[state=active]:text-primary data-[state=active]:bg-white'>Assessments</TabsTrigger>

        </TabsList>

        <TabsContent value="subjects" className='w-full'>

            <Card className='rounded-t-none px-0'>

                <CardContent>

                    <AssignSubjects subjects={getAssignSubjects}/>

                </CardContent>

            </Card>

        </TabsContent>

        <TabsContent value="assessments" className='w-full'>

            <Card className='rounded-t-none px-0'>

                <CardContent className='px-0'>

                    <Tabs defaultValue="ww" className="max-w-full">

                        <TabsList>

                            <TabsTrigger value="ww">Written Works</TabsTrigger>

                            <TabsTrigger value="pt">Performance Task</TabsTrigger>

                            <TabsTrigger value="qe">Quarterly Exams</TabsTrigger>

                        </TabsList>

                            <TabsContent value="ww" className='w-full'>

                                <WrittenWorks assessments={data as AssessmentTypes[]}/>

                            </TabsContent>

                            <TabsContent value="pt" className='w-full'>

                                <PerformanceTask assessments={data as AssessmentTypes[]}/>

                            </TabsContent>

                            <TabsContent value="qe" className='w-full'>

                                <QuarterExam assessments={data as AssessmentTypes[]}/>

                            </TabsContent>

                    </Tabs>

                </CardContent>

            </Card>

        </TabsContent>

    </Tabs>

    </div>

  )

}

export default AssessmentPage

**Form to dynamically assign grade weights**

'use client'

import React, { useState } from 'react'

import { Dialog, DialogTitle, DialogTrigger, DialogContent, DialogDescription, DialogFooter } from '@/components/ui/dialog'

import { Settings } from 'lucide-react'

import { Form, FormControl, FormField, FormItem, FormLabel, FormMessage } from '@/components/ui/form'

import { Input } from '@/components/ui/input'

import { Button } from '@/components/ui/button'

import { useForm } from 'react-hook-form';

import { z } from 'zod';

import { zodResolver } from '@hookform/resolvers/zod';

import { GradeWeightsFormSchema } from '@/lib/validation/grade-weights-form';

import { toast } from 'sonner';

import { RadioGroup, RadioGroupItem } from '@/components/ui/radio-group'

import { Label } from '@/components/ui/label'

import { cn } from '@/lib/utils'

import { useMutation } from 'convex/react'

import { api } from '../../../../../convex/\_generated/api'

import { SubjectsWithAppliedGradeWeights } from '@/lib/types'

interface GradeWeightsFormProps{

    subject: SubjectsWithAppliedGradeWeights

}

function GradeWeightsForm({subject}: GradeWeightsFormProps) {

    const [isDialogOpen, setIsDialogOpen] = useState<boolean>(false)

    const [learningMode, setLearningMode] = useState<string>(subject.appliedGradeWeights?.learningMode ?? "Face to face")

    const addAppliedGW = useMutation(api.appliedGradeWeigths.create)

    const form = useForm<z.infer<typeof GradeWeightsFormSchema>>({

        resolver: zodResolver(GradeWeightsFormSchema),

        defaultValues: {

            learningMode: subject.appliedGradeWeights?.learningMode ?? "Face to face",

            written: subject.appliedGradeWeights?.written ?? subject.gradeWeights?.written,

            performance: subject.appliedGradeWeights?.performance ?? subject.gradeWeights?.performance,

            exam: subject.appliedGradeWeights?.exam ?? (learningMode === "Alternative Learning Module" ? undefined : subject.gradeWeights?.exam),

        }

    });

    const onSubmit = async (data: z.infer<typeof GradeWeightsFormSchema>) => {

        const isOneHundred = data.learningMode === "Face to face" ? (data.written + data.performance + (data.exam ?? 0)) === 100 : (data.written + data.performance) === 100

        if(!isOneHundred) {

            toast.error("The sum of all grade weights must equal 100%.");

            return

        }

        toast.promise(addAppliedGW({

            subjectId: subject.\_id,

            learningMode: data.learningMode,

            written: data.written,

            performance: data.performance,

            exam: data.exam

        }),{

            loading: `Saving ${subject.subjectCode} grade weights... Please wait.`,

            success:() => {

                return `${subject.subjectCode} grade weights were saved successfully.`

            },

            error: () =>{

                return `Failed to save ${subject.subjectCode}'s grade weights. Please try again.`

            }

        })

        setIsDialogOpen(false)

    };

  return (

    <Dialog open={isDialogOpen} onOpenChange={setIsDialogOpen}>

        <DialogTrigger onClick={()=> setIsDialogOpen(!isDialogOpen)}><Settings/></DialogTrigger>

        <DialogContent>

            <DialogTitle className='text-primary'>Set Grade Weights & Learning Mode for {subject.subjectCode}</DialogTitle>

            <DialogDescription className='text-xs'>We recommend this balanced weight distribution, but you can adjust it to fit your subject, teaching style, or learning mode.</DialogDescription>

            <div className="pt-2 border-t-2 border-t-gray-300">

                <Form {...form}>

                    <form onSubmit={form.handleSubmit(onSubmit)}>

                        <div className="text-primary space-y-2 mx-auto w-full">

                            <FormField

                                name="learningMode"

                                control={form.control}

                                render={({ field }) => (

                                    <FormItem className="space-y-3">

                                        <FormLabel className='mt-3 text-white px-2 py-1 font-semibold text-sm block w-full bg-primary'>Learning Mode</FormLabel>

                                        <FormControl>

                                            <RadioGroup

                                                defaultValue={subject.appliedGradeWeights?.learningMode ?? "Face to face"}

                                                className='pl-5 flex gap-x-5'

                                                onValueChange={(value) => {

                                                    field.onChange(value)

                                                    setLearningMode(value)

                                                    if (value === "Alternative Learning Module") {

                                                        form.setValue('exam', undefined)

                                                    }

                                                }}

                                            >

                                                <FormItem  className="space-x-2">

                                                    <FormControl>

                                                        <RadioGroupItem

                                                            value="Face to face"

                                                            className='rounded-none' />

                                                    </FormControl>

                                                    <Label htmlFor="Yes">Face to face</Label>

                                                </FormItem >

                                                <FormItem  className="space-x-2">

                                                    <FormControl>

                                                        <RadioGroupItem

                                                            value="Alternative Learning Module"

                                                            className='rounded-none'

                                                        />

                                                    </FormControl>

                                                    <Label htmlFor="No">Alternative Learning Module</Label>

                                                </FormItem >

                                            </RadioGroup>

                                        </FormControl>

                                    </FormItem>

                                )}

                            />

                            <div className="space-y-1 ">

                            <FormLabel className='text-white px-2 py-1 font-semibold text-sm block w-full bg-primary'>Set Grade Weights</FormLabel>

                            <div className="grid grid-cols-12 items-center w-full px-5">

                                <FormLabel className='col-span-4 font-semibold'>Written Works</FormLabel>

                                <FormField

                                    name={`written`}

                                    control={form.control}

                                    render={({ field }) => (

                                        <FormItem>

                                            <FormControl>

                                                <Input

                                                    type="number"

                                                    placeholder="Percent"

                                                    {...field}

                                                    className='w-20'

                                                />

                                            </FormControl>

                                            <FormMessage />

                                        </FormItem>

                                    )}

                                />

                            </div>

                            <div className="grid grid-cols-12 items-center w-full px-5">

                                <FormLabel className='col-span-4 font-semibold'>Performance Task</FormLabel>

                                <FormField

                                    name={`performance`}

                                    control={form.control}

                                    render={({ field }) => (

                                        <FormItem>

                                            <FormControl>

                                                <Input

                                                    type="number"

                                                    placeholder="Percent"

                                                    {...field}

                                                    className='w-20'

                                                />

                                            </FormControl>

                                            <FormMessage />

                                        </FormItem>

                                    )}

                                />

                            </div>

                            <div className={cn('grid grid-cols-12 items-center w-full px-5', learningMode === "Alternative Learning Module" && "hidden") }>

                                <FormLabel className='col-span-4 font-semibold'>Major Exam</FormLabel>

                                <FormField

                                    name={`exam`}

                                    control={form.control}

                                    render={({ field }) => (

                                        <FormItem  >

                                            <FormControl>

                                                <Input

                                                    type="number"

                                                    placeholder="Percent"

                                                    {...field}

                                                    className={cn('w-20')}/>

                                            </FormControl>

                                            <FormMessage />

                                        </FormItem>

                                    )}/>

                            </div>

                            </div>

                        </div>

                        <DialogFooter>

                            <Button type='button' variant="ghost" onClick={() =>{

                                 setIsDialogOpen(false)

                                 setLearningMode(subject.appliedGradeWeights?.learningMode ?? "Face to face")

                                 }} className="text-primary">

                                Close

                            </Button>

                            <Button type="submit" variant={'default'} className="text-white">Save</Button>

                        </DialogFooter>

                    </form>

                </Form>

            </div>

        </DialogContent>

    </Dialog>

  )

}

export default GradeWeightsForm

**User Access Rights Module (SUM)**

Authentication screen / Login Page:

"use client"

import { useState } from "react"

import { AuthFlow } from "../types"

import { useConvexAuth } from "convex/react"

import { RoleCheck } from "@/components/guards/logged-in"

import { SignInCardRegister } from "./sign-in-card-register"

import { SignUpCardRegister } from "./sign-up-card-register"

export const AuthScreenRegister = () => {

    const [state, setState] = useState<AuthFlow>("signIn")

    const { isAuthenticated } = useConvexAuth()

    if (isAuthenticated) {

        return <RoleCheck />

    }

    return (

        <div className="h-screen w-full lg:flex lg:flex-row">

            {/\* <div

                className="hidden lg:w-[50%] bg-primary/90 lg:flex lg:flex-col lg:justify-center lg:items-center text-white px-[76px]"

            >

                <Image

                    src="/images/black-logo.svg"

                    alt="Logo of Tanjay National High School"

                    width={500}

                    height={500}

                    className="invert"

                />

                <h1 className="text-[196px] font-bold">

                    ERMS.

                </h1>

                <h1 className="text-xl font-medium my-10">

                    Electronic Record Management System

                </h1>

            </div> \*/}

            <div className="hidden lg:w-[50%] bg-gradient-to-b from-primary to-primary/70 lg:flex lg:flex-col lg:justify-center lg:items-center text-white px-[76px]">

                <h1 className="text-3xl font-bold mb-4 text-center">Simplify your workflow with seamless electronic records management!</h1>

                <p className="text-lg font-medium">

                    Electronic Record Management System

                </p>

            </div>

            <div className="h-full w-full lg:w-[50%] flex flex-col flex-1 items-center justify-center bg-gradient-to-b from-primary to-primary/70">

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

                    <div className="md:h-auto md:w-[420px]">

                        {state === "signIn" ? <SignInCardRegister setState={setState} /> : <SignUpCardRegister setState={setState} />}

                    </div>

                </div>

            </div>

        </div>

    )

}

**Sign In Card:**

import { Button } from "@/components/ui/button"

import {

    Card,

    CardContent,

    CardDescription,

    CardHeader,

    CardTitle

} from "@/components/ui/card"

import { Input } from "@/components/ui/input"

import { Separator } from "@/components/ui/separator"

import { useAuthActions } from "@convex-dev/auth/react"

import { TriangleAlertIcon } from "lucide-react"

import { useState } from "react"

import { AuthFlow } from "../types"

export const SignInCardRegister = ({

    //eslint-disable-next-line @typescript-eslint/no-unused-vars

    setState

}: {

    setState: (state: AuthFlow) => void

}) => {

    const [email, setEmail] = useState("");

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

    const [pending, setPending] = useState<boolean>(false);

    //eslint-disable-next-line @typescript-eslint/no-unused-vars

    const [error, setError] = useState("");

    const { signIn } = useAuthActions()

    // const router = useRouter()

    const onSignIn = async (e: React.FormEvent<HTMLFormElement>) => {

        e.preventDefault()

        if (pending) return;

        setPending(true);

        setError("");

        try {

            await signIn("password", {

                email,

                password,

                flow: "signIn"

            });

            setError("");

        } catch (error) {

            console.error("Sign in error:", error);

            if (error instanceof Error) {

                if (error.message.includes("Failed to fetch")) {

                    setError("Connection error. Please check your internet connection and try again.");

                } else {

                    setError("Invalid email or password");

                }

            } else {

                setError("Invalid email or password");

            }

        } finally {

            setPending(false);

        }

    }

    return (

        <div className="relative h-full w-full flex items-center justify-center ">

            <Card className="w-full h-full p-8 z-50">

                <CardHeader className="px-0 pt-0">

                    <CardTitle className="text-primary">

                        Login to continue

                    </CardTitle>

                    <CardDescription>

                        Please fill in the form below to login to your account.

                    </CardDescription>

                </CardHeader>

                {!!error && (

                    <div className="bg-destructive/15 p-3 rounded-md flex items-center gap-x-2 text-sm text-destructive mb-6">

                        <TriangleAlertIcon className="size-4" />

                        {error}

                    </div>

                )}

                <CardContent className="space-y-5 px-0 pb-0">

                    <form onSubmit={onSignIn} className="space-y-2.5">

                        <Input

                            disabled={pending}

                            value={email}

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

                            placeholder="Email"

                            type="email"

                            required

                        />

                        <Input

                            disabled={pending}

                            value={password}

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

                            placeholder="Password"

                            type="password"

                            required

                        />

                        <Button

                            type="submit"

                            className="w-full text-white"

                            size={"lg"}

                            disabled={pending}

                        >

                            Continue

                        </Button>

                    </form>

                    <Separator />

                </CardContent>

            </Card>

        </div>

    )

}

**Role check component:**

"use client"

import { useRouter } from "next/navigation"

import { useEffect } from "react"

import { useConvexAuth } from "convex/react"

import { useCheckRole } from "@/features/current/api/use-check-role"

export function RoleCheck() {

    const router = useRouter()

    const { isAuthenticated, isLoading: isAuthLoading } = useConvexAuth()

    const { data: role, isLoading: isRoleLoading } = useCheckRole()

    useEffect(() => {

        if (!isAuthLoading && !isRoleLoading && isAuthenticated) {

            switch (role) {

                case "admin":

                    router.push("/sysadmin")

                    break

                case "teacher":

                    router.push("/dashboard")

                    break

                case "school-head":

                    router.push("/school-head")

                    break

                case "staff":

                    router.push("/sr-dashboard")

                    break

                default:

                    router.push("/")

            }

        }

    }, [isAuthenticated, isAuthLoading, isRoleLoading, role, router])

    return null

}

**Guard component:**

"use client"

import { useCheckRole } from "@/features/current/api/use-check-role"

import { useConvexAuth } from "convex/react"

import { useRouter } from "next/navigation"

import { useEffect } from "react"

export function SystemAdminGuard({ children }: { children: React.ReactNode }) {

    const router = useRouter()

    const { isAuthenticated, isLoading: isAuthLoading } = useConvexAuth()

    const { data: role, isLoading: isRoleLoading } = useCheckRole()

    useEffect(() => {

        if (!isAuthLoading && !isRoleLoading) {

            if (!isAuthenticated) {

                router.push("/")

                return

            }

            // Based on schema.ts roles: "admin", "teacher", "school-head", "staff"

            if (role !== "admin") {

                // Redirect non-admin users to appropriate routes

                switch (role) {

                    case "teacher":

                        router.push("/dashboard")

                        break

                    case "school-head":

                        router.push("/school-head")

                        break

                    case "staff":

                        router.push("/sr-dashboard")

                        break

                    default:

                        router.push("/auth")

                }

                return

            }

        }

    }, [isAuthenticated, isAuthLoading, isRoleLoading, role, router])

    // Show nothing while checking authentication and role

    if (isAuthLoading || isRoleLoading) {

        return null

    }

    // Only render children if authenticated and system admin

    if (isAuthenticated && role === "admin") {

        return <>{children}</>

    }

    return null

}

**Middleware configuration:**

import { convexAuthNextjsMiddleware } from "@convex-dev/auth/nextjs/server";

export default convexAuthNextjsMiddleware();

export const config = {

  // The following matcher runs middleware on all routes

  // except static assets.

  matcher: ["/((?!.\*\\..\*|\_next).\*)", "/", "/(api|trpc)(.\*)"],

};

**Authentication Configuration:**

import { Password } from "@convex-dev/auth/providers/Password";

import { convexAuth } from "@convex-dev/auth/server";

const CustomPassword = Password({

  profile(params) {

    return {

      email: params.email as string,

      firstName: params.firstName as string,

      lastName: params.lastName as string,

      role: params.role as "admin" | "teacher" | "school-head" | "staff",

      isActive: true,

      // department: (params.department as string) ?? null,

      // specialization: (params.specialization as string) ?? null,

      // yearsOfExperience: (params.yearsOfExperience as number) ?? null,

      // birthDate: params.birthDate as string,

    };

  },

});

export const { auth, signIn, signOut, store } = convexAuth({

  providers: [CustomPassword],

});

Create admin:

"use client";

import {

    Breadcrumb,

    BreadcrumbItem,

    BreadcrumbLink,

    BreadcrumbList,

    BreadcrumbPage,

    BreadcrumbSeparator,

} from "@/components/ui/breadcrumb";

import { Button, buttonVariants } from "@/components/ui/button";

import {

    Card,

    CardContent,

    CardDescription,

    CardHeader,

    CardTitle

} from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

    Select,

    SelectContent,

    SelectGroup,

    SelectItem,

    SelectLabel,

    SelectTrigger,

    SelectValue,

} from "@/components/ui/select";

import {

    ChevronLeft

} from "lucide-react";

import { UserAvatarUpload } from "@/components/shared/user-avatar-upload";

import { Barangay, City, fetchBarangays, fetchCities, fetchProvinces, fetchRegions, Province, Region } from "@/lib/address-api";

import { AdminFormData } from "@/lib/types";

import { cn } from "@/lib/utils";

import { useMutation } from "convex/react";

import Link from "next/link";

import { useRouter } from "next/navigation";

import { useEffect, useState } from "react";

import { useForm } from "react-hook-form";

import { toast } from "sonner";

import { api } from "../../../../../convex/\_generated/api";

const SystemAdminAddAdminPage = () => {

    const createAdmin = useMutation(api.users.createUser)

    const [isLoading, setIsLoading] = useState(false)

    const router = useRouter()

    const {

        register,

        handleSubmit,

        formState: { errors },

        watch,

        setValue

    } = useForm<AdminFormData>({

        defaultValues: {

            isActive: true

        }

    })

    const password = watch("password")

    const [regions, setRegions] = useState<Region[]>([]);

    const [provinces, setProvinces] = useState<Province[]>([]);

    const [cities, setCities] = useState<City[]>([]);

    const [barangays, setBarangays] = useState<Barangay[]>([]);

    const [isNCR, setIsNCR] = useState(false);

    const [imageStorageId, setImageStorageId] = useState<string | undefined>();

    useEffect(() => {

        fetchRegions().then(data => {

            setRegions(data);

        });

    }, []);

    const handleRegionChange = async (regionCode: string) => {

        setValue('province', '');

        setValue('city', '');

        setValue('barangay', '');

        setCities([]);

        setBarangays([]);

        const isNCRSelected = regionCode === '130000000';

        setIsNCR(isNCRSelected);

        const selectedRegion = regions.find(r => r.code === regionCode);

        if (selectedRegion) {

            setValue('region', selectedRegion.name);

        }

        if (isNCRSelected) {

            setValue('province', 'Metro Manila');

            const cityData = await fetchCities('130000000');

            setCities(cityData);

        } else {

            const provinceData = await fetchProvinces(regionCode);

            setProvinces(provinceData);

        }

    };

    const handleProvinceChange = async (provinceCode: string) => {

        setValue('city', '');

        setValue('barangay', '');

        const selectedProvince = provinces.find(p => p.code === provinceCode);

        if (selectedProvince) {

            setValue('province', selectedProvince.name);

        }

        const cityData = await fetchCities(watch('region'));

        setCities(cityData);

    };

    const handleCityChange = async (cityCode: string) => {

        setValue('barangay', '');

        const selectedCity = cities.find(c => c.code === cityCode);

        if (selectedCity) {

            setValue('city', selectedCity.name);

            if (selectedCity.postalCode) {

                setValue('postalCode', selectedCity.postalCode);

            }

        }

        try {

            const barangayData = await fetchBarangays(cityCode);

            setBarangays(barangayData);

        } catch (error) {

            toast.error('Failed to fetch barangays: ' + (error as Error).message);

        }

    };

    const handleBarangayChange = (barangayCode: string) => {

        const selectedBarangay = barangays.find(b => b.code === barangayCode);

        if (selectedBarangay) {

            setValue('barangay', selectedBarangay.name);

        }

    };

    const onSubmit = async (data: AdminFormData) => {

        if (data.password !== data.confirmPassword) {

            toast.error("Passwords do not match");

            return;

        }

        setIsLoading(true);

        try {

            await createAdmin({

                email: data.email,

                password: data.password,

                firstName: data.firstName,

                middleName: data.middleName,

                lastName: data.lastName,

                contactNumber: data.contactNumber,

                role: "admin",

                department: data.department,

                specialization: data.specialization,

                yearsOfExperience: data.yearsOfExperience,

                birthDate: data.birthDate as string,

                isActive: data.isActive,

                region: data.region,

                province: data.province,

                city: data.city,

                barangay: data.barangay,

                street: data.street,

                houseNumber: data.houseNumber,

                postalCode: data.postalCode,

                imageStorageId: imageStorageId,

            });

            toast.success("Admin created successfully");

            router.push("/sysadmin-admins");

        } catch (error) {

            toast.error("Failed to create admin: " + (error as Error).message);

        } finally {

            setIsLoading(false);

        }

    };

    return (

        <div className="container mx-auto p-4">

            <Breadcrumb className="hidden md:flex">

                <BreadcrumbList>

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin">Dashboard</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin-admins">System Administrators</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbPage>Add a System Administrator</BreadcrumbPage>

                    </BreadcrumbItem>

                </BreadcrumbList>

            </Breadcrumb>

            <main className="space-y-4 mt-8">

                <form onSubmit={handleSubmit(onSubmit)}>

                    <div className="mx-auto grid max-w-[59rem] flex-1 auto-rows-max gap-4">

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

                            <Link

                                href="/sysadmin-admins"

                                className={cn("h-7 w-7", buttonVariants({

                                    variant: "outline",

                                    size: "icon",

                                }))}>

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

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

                            </Link>

                            <h1 className="flex-1 shrink-0 whitespace-nowrap text-xl font-semibold tracking-tight sm:grow-0">

                                Add a System Administrator

                            </h1>

                            {/\* Button for desktop/laptop users \*/}

                            <div className="hidden items-center gap-2 md:ml-auto md:flex">

                                <Button

                                    type="submit"

                                    size="sm"

                                    className="text-white"

                                    disabled={isLoading}

                                >

                                    {isLoading ? "Creating..." : "Save"}

                                </Button>

                            </div>

                        </div>

                        <div className="grid gap-4 md:grid-cols-[1fr\_250px] lg:grid-cols-3 lg:gap-8">

                            <div className="grid auto-rows-max items-start gap-4 lg:col-span-2 lg:gap-8">

                                {/\* Fname, Lname, Mname, Desc \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Admin Details</CardTitle>

                                        <CardDescription>

                                            Add admin details

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="name">Email <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="email"

                                                    type="email"

                                                    className="w-full"

                                                    placeholder="zKj6w@example.com"

                                                    {...register("email", { required: "Email is required" })}

                                                />

                                                {errors.email && <p className="text-red-500">{errors.email.message}</p>}

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">First Name <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="name"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="John"

                                                        {...register("firstName", { required: "First name is required" })}

                                                    />

                                                    {errors.firstName && <p className="text-red-500">{errors.firstName.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Last Name <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="name"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Doe"

                                                        {...register("lastName", { required: "Last name is required" })}

                                                    />

                                                    {errors.lastName && <p className="text-red-500">{errors.lastName.message}</p>}

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="name">Middle Name (Optional)</Label>

                                                <Input

                                                    id="name"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Kennedy"

                                                    {...register("middleName")}

                                                />

                                            </div>

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

                                                {/\* password and current password \*/}

                                                <div className="grid gap-3">

                                                    <Label htmlFor="password">Password <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="password"

                                                        type="password"

                                                        {...register("password", {

                                                            required: "Password is required",

                                                            minLength: {

                                                                value: 8,

                                                                message: "Password must be at least 8 characters"

                                                            }

                                                        })}

                                                        className="w-full"

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                    />

                                                    {errors.password && <p className="text-red-500">{errors.password.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="confirmPassword">Confirm Password <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="confirmPassword"

                                                        type="password"

                                                        {...register("confirmPassword", {

                                                            validate: value =>

                                                                value === password || "Passwords do not match"

                                                        })}

                                                        className="w-full"

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                    />

                                                    {errors.confirmPassword && <p className="text-red-500">{errors.confirmPassword.message}</p>}

                                                </div>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Other details \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Other details</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="contact">Contact Number <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="contact"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Enter your contact number"

                                                    {...register("contactNumber", { required: "Contact number is required" })}

                                                    maxLength={11}

                                                />

                                                {errors.contactNumber && <p className="text-red-500">{errors.contactNumber.message}</p>}

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="birthDate">Birth Date <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="birthDate"

                                                    type="date"

                                                    className="w-full"

                                                    {...register("birthDate", {

                                                        required: "Birth date is required",

                                                        validate: (value) => {

                                                            if (!value) return "Birth date is required";

                                                            const birthDate = new Date(value);

                                                            const today = new Date();

                                                            let calculatedAge = today.getFullYear() - birthDate.getFullYear();

                                                            if (

                                                                today.getMonth() < birthDate.getMonth() ||

                                                                (today.getMonth() === birthDate.getMonth() && today.getDate() < birthDate.getDate())

                                                            ) {

                                                                calculatedAge--;

                                                            }

                                                            if (calculatedAge < 18) return "Must be at least 18 years old";

                                                            if (calculatedAge > 65) return "Must not exceed 65 years old";

                                                            return true;

                                                        }

                                                    })}

                                                />

                                                {errors.birthDate && <p className="text-red-500">{errors.birthDate.message}</p>}

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="department">Department (Optional)</Label>

                                                <Input

                                                    id="department"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Enter department"

                                                    {...register("department")}

                                                />

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="specialization">Specialization (Optional)</Label>

                                                    <Input

                                                        id="specialization"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Enter specialization"

                                                        {...register("specialization")}

                                                    />

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="experience">Years of Experience (Optional)</Label>

                                                    <Input

                                                        id="experience"

                                                        type="number"

                                                        className="w-full"

                                                        placeholder="Enter years of experience"

                                                        {...register("yearsOfExperience", {

                                                            valueAsNumber: true

                                                        })}

                                                    />

                                                </div>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Address details \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Address Details</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="region">Region (Optional)</Label>

                                                    <Select

                                                        onValueChange={(value) => {

                                                            setValue('region', value);

                                                            handleRegionChange(value);

                                                        }}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select region" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Region</SelectLabel>

                                                                {regions.map(region => (

                                                                    <SelectItem key={region.code} value={region.code}>

                                                                        {region.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="province">Province (Optional)</Label>

                                                    {isNCR ? (

                                                        <Input

                                                            value="Metro Manila"

                                                            disabled

                                                            className="bg-muted"

                                                            {...register("province")}

                                                        />

                                                    ) : (

                                                        <Select

                                                            onValueChange={(value) => {

                                                                setValue('province', value);

                                                                handleProvinceChange(value);

                                                            }}

                                                            disabled={!watch('region')}

                                                        >

                                                            <SelectTrigger className="w-full">

                                                                <SelectValue placeholder="Select province" />

                                                            </SelectTrigger>

                                                            <SelectContent>

                                                                <SelectGroup>

                                                                    <SelectLabel>Province</SelectLabel>

                                                                    {provinces.map(province => (

                                                                        <SelectItem key={province.code} value={province.code}>

                                                                            {province.name}

    </SelectItem>))}

  </SelectGroup>

  </SelectContent>

    </Select>

      )}

      </div>

    </div>

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

                <div className="grid gap-3">

                                 <Label htmlFor="city">City/Municipality (Optional)</Label>

                                                    <Select

                                                        onValueChange={(value) => {

                                                            setValue('city', value);

                                                            handleCityChange(value);

                                                        }}

                                                        disabled={!watch('region') || (!isNCR && !watch('province'))}

                                                    >

                                   <SelectTrigger className="w-full">

                           <SelectValue placeholder="Select city" />

                     </SelectTrigger>

                        <SelectContent>

                            <SelectGroup>

                    <SelectLabel>City</SelectLabel>

              {cities.map(city => (

             <SelectItem key={city.code} value={city.code}>

                                                                        {city.name}

                                                                    </SelectItem>

                                                                ))}

                                           </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="barangay">Barangay (Optional)</Label>

                                                    <Select

                                                        onValueChange={(value) => handleBarangayChange(value)}

                                                        disabled={!watch('city')}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select barangay" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Barangay</SelectLabel>

                                                                {barangays.map(barangay => (

                                                                    <SelectItem key={barangay.code} value={barangay.code}>

                                                                        {barangay.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="street">Street (Optional)</Label>

                                                    <Input

                                                        id="street"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Enter street"

                                                        {...register("street")}

                                                    />

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="houseNumber">House Number (Optional)</Label>

                                                    <Input

                                                        id="houseNumber"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Enter house number"

                                                        {...register("houseNumber")}

                                                    />

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="postalCode">Postal Code (Optional)</Label>

                                                <Input

                                                    id="postalCode"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Enter postal code"

                                                    {...register("postalCode")}

                                                />

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                            </div>

                            <div className="grid auto-rows-max items-start gap-4 lg:gap-8">

                                {/\* Admin status \*/}

                                <Card x-chunk="dashboard-07-chunk-3">

                                    <CardHeader>

                                        <CardTitle>Admin Status</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="status">Status <span className="text-red-500">\*</span></Label>

                                                <Select defaultValue="active" onValueChange={(value) => setValue("isActive", value === "active")}>

                                                    <SelectTrigger id="status" aria-label="Select status">

                                                        <SelectValue placeholder="Select status" />

                                                    </SelectTrigger>

                                                    <SelectContent>

                                                        <SelectItem value="active">Active</SelectItem>

                                                        <SelectItem value="inactive">Inactive</SelectItem>

                                                    </SelectContent>

                                                </Select>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Admin Image \*/}

                                <Card

                                    className="overflow-hidden" x-chunk="dashboard-07-chunk-4"

                                >

                                    <CardHeader>

                                        <CardTitle>Admin Image</CardTitle>

                                        <CardDescription>

                                            Upload an image of the admin (Optional)

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <UserAvatarUpload

                                            onImageUpload={(storageId) => setImageStorageId(storageId)}

                                            onImageRemove={() => setImageStorageId(undefined)}

                                        />

                                    </CardContent>

                                </Card>

                            </div>

                        </div>

                        {/\* Button for mobile users \*/}

                        <div className="flex items-center justify-center gap-2 md:hidden">

                            <Button

                                size="sm"

                                className="text-white"

                            >

                                Save Product

                            </Button>

                        </div>

                    </div>

                </form>

            </main>

        </div>

    )

}

export default SystemAdminAddAdminPage;

Create Principal:

"use client";

import { UserAvatarUpload } from "@/components/shared/user-avatar-upload";

import {

    Breadcrumb,

    BreadcrumbItem,

    BreadcrumbLink,

    BreadcrumbList,

    BreadcrumbPage,

    BreadcrumbSeparator,

} from "@/components/ui/breadcrumb";

import { Button, buttonVariants } from "@/components/ui/button";

import {

    Card,

    CardContent,

    CardDescription,

    CardHeader,

    CardTitle

} from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

    Select,

    SelectContent,

    SelectGroup,

    SelectItem,

    SelectLabel,

    SelectTrigger,

    SelectValue,

} from "@/components/ui/select";

import {

    Barangay,

    City,

    Province,

    Region,

    fetchBarangays,

    fetchCities,

    fetchProvinces,

    fetchRegions

} from "@/lib/address-api";

import { useMutation } from "convex/react";

import { useRouter } from "next/navigation";

import { useEffect, useState } from "react";

import { useForm } from "react-hook-form";

import { toast } from "sonner";

import { api } from "../../../../../convex/\_generated/api";

import {

    ChevronLeft

} from "lucide-react";

import { Textarea } from "@/components/ui/textarea";

import { PrincipalFormData, SchoolHeadType } from "@/lib/types";

import { cn } from "@/lib/utils";

import Link from "next/link";

const SystemAdminHandlePrincipalPage = () => {

    const createPrincipal = useMutation(api.users.createSchoolHead);

    const [isLoading, setIsLoading] = useState(false);

    const router = useRouter();

    const [imageStorageId, setImageStorageId] = useState<string | undefined>();

    const {

        register,

        handleSubmit,

        formState: { errors },

        watch,

        setValue

    } = useForm<PrincipalFormData>({

        defaultValues: {

            gender: '',

            schoolHeadType: 'junior-high' // default value

        }

    });

    // const password = watch("password");

    // Address state management

    const [regions, setRegions] = useState<Region[]>([]);

    const [provinces, setProvinces] = useState<Province[]>([]);

    const [cities, setCities] = useState<City[]>([]);

    const [barangays, setBarangays] = useState<Barangay[]>([]);

    const [isNCR, setIsNCR] = useState(false);

    // Fetch regions on component mount

    useEffect(() => {

        fetchRegions().then(data => {

            setRegions(data);

        });

    }, []);

    // Address handling functions (same as in admin page)

    const handleRegionChange = async (regionCode: string) => {

        setValue('province', '');

        setValue('city', '');

        setValue('barangay', '');

        setCities([]);

        setBarangays([]);

        const isNCRSelected = regionCode === '130000000';

        setIsNCR(isNCRSelected);

        const selectedRegion = regions.find(r => r.code === regionCode);

        if (selectedRegion) {

            setValue('region', selectedRegion.name);

        }

        if (isNCRSelected) {

            setValue('province', 'Metro Manila');

            const cityData = await fetchCities('130000000');

            setCities(cityData);

        } else {

            const provinceData = await fetchProvinces(regionCode);

            setProvinces(provinceData);

        }

    };

    const handleProvinceChange = async (provinceCode: string) => {

        setValue('city', '');

        setValue('barangay', '');

        setBarangays([]);

        const selectedProvince = provinces.find(p => p.code === provinceCode);

        if (selectedProvince) {

            setValue('province', selectedProvince.name);

        }

        const cityData = await fetchCities(provinceCode);

        setCities(cityData);

    };

    const handleCityChange = async (cityCode: string) => {

        setValue('barangay', '');

        const selectedCity = cities.find(c => c.code === cityCode);

        if (selectedCity) {

            setValue('city', selectedCity.name);

        }

        const barangayData = await fetchBarangays(cityCode);

        setBarangays(barangayData);

    };

    const handleBarangayChange = (barangayCode: string) => {

        const selectedBarangay = barangays.find(b => b.code === barangayCode);

        if (selectedBarangay) {

            setValue('barangay', selectedBarangay.name);

        }

    };

    const onSubmit = async (data: PrincipalFormData) => {

        if (data.password !== data.confirmPassword) {

            toast.error("Passwords do not match");

            return;

        }

        if (!data.gender) {

            toast.error("Please select a gender");

            return;

        }

        setIsLoading(true);

        try {

            await createPrincipal({

                email: data.email,

                password: data.password,

                firstName: data.firstName,

                middleName: data.middleName,

                lastName: data.lastName,

                contactNumber: data.contactNumber,

                birthDate: data.birthDate as string,

                gender: data.gender,

                description: data.description,

                region: data.region,

                province: data.province,

                city: data.city,

                barangay: data.barangay,

                street: data.street,

                houseNumber: data.houseNumber,

                postalCode: data.postalCode,

                imageStorageId: imageStorageId,

                schoolHeadType: data.schoolHeadType,

            });

            toast.success("Principal created successfully");

            router.push("/sysadmin-principal/list");

        } catch (error) {

            toast.error("Failed to create principal: " + (error as Error).message);

        } finally {

            setIsLoading(false);

        }

    };

    return (

        <div className="container mx-auto p-4">

            <Breadcrumb className="hidden md:flex">

                <BreadcrumbList>

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin">Dashboard</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin-principal/list">List of Principals</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbPage>Add Principal</BreadcrumbPage>

                    </BreadcrumbItem>

                </BreadcrumbList>

            </Breadcrumb>

            <main className="space-y-4 mt-8">

                <form onSubmit={handleSubmit(onSubmit)}>

                    <div className="mx-auto grid max-w-[59rem] flex-1 auto-rows-max gap-4">

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

                            <Link

                                href="/sysadmin-principal/list"

                                className={cn("h-7 w-7", buttonVariants({

                                    variant: "outline",

                                    size: "icon",

                                }))}>

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

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

                            </Link>

                            <h1 className="flex-1 shrink-0 whitespace-nowrap text-xl font-semibold tracking-tight sm:grow-0">

                                Add Principal

                            </h1>

                            {/\* Button for desktop/laptop users \*/}

                            <div className="hidden items-center gap-2 md:ml-auto md:flex">

                                <Button

                                    type="submit"

                                    size="sm"

                                    className="text-white"

                                    disabled={isLoading}

                                >

                                    {isLoading ? "Creating..." : "Save"}

                                </Button>

                            </div>

                        </div>

                        <div className="grid gap-4 md:grid-cols-[1fr\_250px] lg:grid-cols-3 lg:gap-8">

                            <div className="grid auto-rows-max items-start gap-4 lg:col-span-2 lg:gap-8">

                                {/\* Fname, Lname, Mname, Desc \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Principal Details</CardTitle>

                                        <CardDescription>

                                            Principal Details

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="name">Email <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="email"

                                                    type="email"

                                                    className="w-full"

                                                    placeholder="zKj6w@example.com"

                                                    {...register("email", { required: "Email is required" })}

                                                />

                                                {errors.email && <p className="text-red-500">{errors.email.message}</p>}

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">First Name <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="name"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="John"

                                                        {...register("firstName", { required: "First Name is required" })}

                                                    />

                                                    {errors.firstName && <p className="text-red-500">{errors.firstName.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Last Name <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="name"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Doe"

                                                        {...register("lastName", { required: "Last Name is required" })}

                                                    />

                                                    {errors.lastName && <p className="text-red-500">{errors.lastName.message}</p>}

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="name">Middle Name (Optional)</Label>

                                                <Input

                                                    id="name"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Kennedy"

                                                    {...register("middleName")}

                                                />

                                            </div>

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

                                                {/\* password and current password \*/}

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Password <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="password"

                                                        type="password"

                                                        className="w-full"

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                        {...register("password", { required: "Password is required" })}

                                                    />

                                                    {errors.password && <p className="text-red-500">{errors.password.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Confirm Password <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="cpassword"

                                                        type="password"

                                                        className="w-full"

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                        {...register("confirmPassword", { required: "Confirm Password is required" })}

                                                    />

                                                    {errors.confirmPassword && <p className="text-red-500">{errors.confirmPassword.message}</p>}

                                                </div>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Other details \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Other details</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="birthDate">Birth Date <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="birthDate"

                                                    type="date"

                                                    className="w-full"

                                                    {...register("birthDate", {

                                                        required: "Birth date is required",

                                                        validate: (value) => {

                                                            if (!value) return "Birth date is required";

                                                            const birthDate = new Date(value);

                                                            const today = new Date();

                                                            let age = today.getFullYear() - birthDate.getFullYear();

                                                            if (

                                                                today.getMonth() < birthDate.getMonth() ||

                                                                (today.getMonth() === birthDate.getMonth() && today.getDate() < birthDate.getDate())

                                                            ) {

                                                                age--;

                                                            }

                                                            if (age < 18) return "Must be at least 18 years old";

                                                            if (age > 65) return "Must not exceed 65 years old";

                                                            return true;

                                                        }

                                                    })}

                                                />

                                                {errors.birthDate && <p className="text-red-500">{errors.birthDate.message}</p>}

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="gender">Gender <span className="text-red-500">\*</span></Label>

                                                    <Select

                                                        onValueChange={(value) => setValue("gender", value)}

                                                        {...register("gender", { required: "Gender is required" })}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select gender" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Gender</SelectLabel>

                                                                <SelectItem value="male">Male</SelectItem>

                                                                <SelectItem value="female">Female</SelectItem>

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                    {errors.gender && <p className="text-red-500">{errors.gender.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    {/\* contact number \*/}

                                                    <Label htmlFor="contact">Contact Number <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="contact"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Enter your contact number"

                                                        {...register("contactNumber", { required: "Contact Number is required" })}

                                                        maxLength={11}

                                                    />

                                                    {errors.contactNumber && <p className="text-red-500">{errors.contactNumber.message}</p>}

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="description">Description (Optional)</Label>

                                                <Textarea

                                                    id="description"

                                                    placeholder="Enter something about the principal"

                                                    className="min-h-32"

                                                    {...register("description")}

                                                />

                                                <Label htmlFor="schoolHeadType">School Head Level <span className="text-red-500">\*</span></Label>

                                                <Select

                                                    onValueChange={(value: SchoolHeadType) => setValue("schoolHeadType", value)}

                                                    value={watch("schoolHeadType")}

                                                >

                                                    <SelectTrigger>

                                                        <SelectValue placeholder="Select school head type" />

                                                    </SelectTrigger>

                                                    <SelectContent>

                                                        <SelectGroup>

                                                            <SelectLabel>School Head Level</SelectLabel>

                                                            <SelectItem value="junior-high">Junior High School Head</SelectItem>

                                                            <SelectItem value="senior-high">Senior High School Head</SelectItem>

                                                        </SelectGroup>

                                                    </SelectContent>

                                                </Select>

                                                {errors.schoolHeadType && (

                                                    <p className="text-sm text-red-500">{errors.schoolHeadType.message}</p>

                                                )}

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Address details \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Address Details (Optional)</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="region">Region</Label>

                                                    <Select onValueChange={handleRegionChange}>

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select region" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Region</SelectLabel>

                                                                {regions.map(region => (

                                                                    <SelectItem key={region.code} value={region.code}>

                                                                        {region.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="province">Province</Label>

                                                    {isNCR ? (

                                                        <Input

                                                            value="Metro Manila"

                                                            disabled

                                                            className="bg-muted"

                                                        />

                                                    ) : (

                                                        <Select

                                                            onValueChange={handleProvinceChange}

                                                            disabled={!watch('region')}

                                                        >

                                                            <SelectTrigger className="w-full">

                                                                <SelectValue placeholder="Select province" />

                                                            </SelectTrigger>

                                                            <SelectContent>

                                                                <SelectGroup>

                                                                    <SelectLabel>Province</SelectLabel>

                                                                    {provinces.map(province => (

                                                                        <SelectItem key={province.code} value={province.code}>

                                                                            {province.name}

                                                                        </SelectItem>

                                                                    ))}

                                                                </SelectGroup>

                                                            </SelectContent>

                                                        </Select>

                                                    )}

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="city">City/Municipality</Label>

                                                    <Select

                                                        onValueChange={handleCityChange}

                                                        disabled={!watch('region') || (!isNCR && !watch('province'))}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select city" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>City</SelectLabel>

                                                                {cities.map(city => (

                                                                    <SelectItem key={city.code} value={city.code}>

                                                                        {city.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="barangay">Barangay</Label>

                                                    <Select

                                                        onValueChange={handleBarangayChange}

                                                        disabled={!watch('city')}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select barangay" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Barangay</SelectLabel>

                                                                {barangays.map(barangay => (

                                                                    <SelectItem key={barangay.code} value={barangay.code}>

                                                                        {barangay.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="strt">Street</Label>

                                                <Input

                                                    id="strt"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Enter street"

                                                    {...register("street")}

                                                />

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                            </div>

                            <div className="grid auto-rows-max items-start gap-4 lg:gap-8">

                                <Card className="overflow-hidden">

                                    <CardHeader>

                                        <CardTitle>Principal Image</CardTitle>

                                        <CardDescription>

                                            Upload principal image (Optional)

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <UserAvatarUpload

                                            onImageUpload={(storageId) => setImageStorageId(storageId)}

                                            onImageRemove={() => setImageStorageId(undefined)}

                                        />

                                    </CardContent>

                                </Card>

                            </div>

                        </div>

                        {/\* Button for mobile users \*/}

                        <div className="flex items-center justify-center gap-2 md:hidden">

                            <Button

                                type="submit"

                                size="sm"

                                className="text-white"

                                disabled={isLoading}

                            >

                                {isLoading ? "Creating..." : "Save"}

                            </Button>

                        </div>

                    </div>

                </form>

            </main>

        </div>

    )

}

export default SystemAdminHandlePrincipalPage;

Create School Registrar:

"use client"

import {

    Breadcrumb,

    BreadcrumbItem,

    BreadcrumbLink,

    BreadcrumbList,

    BreadcrumbPage,

    BreadcrumbSeparator,

} from "@/components/ui/breadcrumb";

import { Button, buttonVariants } from "@/components/ui/button";

import {

    Card,

    CardContent,

    CardDescription,

    CardHeader,

    CardTitle

} from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

    Select,

    SelectContent,

    SelectGroup,

    SelectItem,

    SelectLabel,

    SelectTrigger,

    SelectValue,

} from "@/components/ui/select";

import {

    ChevronLeft

} from "lucide-react";

import { UserAvatarUpload } from "@/components/shared/user-avatar-upload";

import {

    Barangay,

    City,

    Province,

    Region,

    fetchBarangays,

    fetchCities,

    fetchProvinces,

    fetchRegions

} from "@/lib/address-api";

import { cn } from "@/lib/utils";

import { useMutation } from "convex/react";

import Link from "next/link";

import { useRouter } from "next/navigation";

import { useEffect, useState } from "react";

import { useForm } from "react-hook-form";

import { toast } from "sonner";

import { api } from "../../../../../convex/\_generated/api";

export interface RegistrarFormData {

    email: string;

    password: string;

    confirmPassword: string;

    firstName: string;

    lastName: string;

    middleName?: string;

    employeeId?: string;

    contactNumber: string;

    birthDate: string;

    gender: string;

    yearsOfExperience?: number;

    description?: string; // missing

    region?: string;

    province?: string;

    city?: string;

    barangay?: string;

    street?: string;

}

const SystemAdminAddRegistrarPage = () => {

    const createRegistrar = useMutation(api.users.createStaff);

    const [isLoading, setIsLoading] = useState(false);

    const router = useRouter();

    const [imageStorageId, setImageStorageId] = useState<string | undefined>();

    const {

        register,

        handleSubmit,

        formState: { errors },

        watch,

        setValue

    } = useForm<RegistrarFormData>({

        defaultValues: {

            gender: '',

        }

    });

    console.log(watch())

    // Address state management

    const [regions, setRegions] = useState<Region[]>([]);

    const [provinces, setProvinces] = useState<Province[]>([]);

    const [cities, setCities] = useState<City[]>([]);

    const [barangays, setBarangays] = useState<Barangay[]>([]);

    const [isNCR, setIsNCR] = useState(false);

    // Fetch regions on component mount

    useEffect(() => {

        fetchRegions().then(data => {

            setRegions(data);

        });

    }, []);

    // Address handling functions (same as in admin page)

    const handleRegionChange = async (regionCode: string) => {

        setValue('province', '');

        setValue('city', '');

        setValue('barangay', '');

        setCities([]);

        setBarangays([]);

        const isNCRSelected = regionCode === '130000000';

        setIsNCR(isNCRSelected);

        const selectedRegion = regions.find(r => r.code === regionCode);

        if (selectedRegion) {

            setValue('region', selectedRegion.name);

        }

        if (isNCRSelected) {

            setValue('province', 'Metro Manila');

            const cityData = await fetchCities('130000000');

            setCities(cityData);

        } else {

            const provinceData = await fetchProvinces(regionCode);

            setProvinces(provinceData);

        }

    };

    const handleProvinceChange = async (provinceCode: string) => {

        setValue('city', '');

        setValue('barangay', '');

        setBarangays([]);

        const selectedProvince = provinces.find(p => p.code === provinceCode);

        if (selectedProvince) {

            setValue('province', selectedProvince.name);

        }

        const cityData = await fetchCities(provinceCode);

        setCities(cityData);

    };

    const handleCityChange = async (cityCode: string) => {

        setValue('barangay', '');

        const selectedCity = cities.find(c => c.code === cityCode);

        if (selectedCity) {

            setValue('city', selectedCity.name);

        }

        const barangayData = await fetchBarangays(cityCode);

        setBarangays(barangayData);

    };

    const handleBarangayChange = (barangayCode: string) => {

        const selectedBarangay = barangays.find(b => b.code === barangayCode);

        if (selectedBarangay) {

            setValue('barangay', selectedBarangay.name);

        }

    };

    const onSubmit = async (data: RegistrarFormData) => {

        if (data.password !== data.confirmPassword) {

            toast.error("Passwords do not match");

            return;

        }

        if (!data.gender) {

            toast.error("Please select a gender");

            return;

        }

        setIsLoading(true);

        try {

            await createRegistrar({

                email: data.email,

                password: data.password,

                firstName: data.firstName,

                middleName: data.middleName,

                lastName: data.lastName,

                contactNumber: data.contactNumber,

                birthDate: data.birthDate,

                gender: data.gender,

                description: data.description,

                region: data.region,

                province: data.province,

                city: data.city,

                barangay: data.barangay,

                street: data.street,

                imageStorageId: imageStorageId,

                department: "Registrar's Office",

                employeeId: data.employeeId,

                yearsOfExperience: data.yearsOfExperience ? Number(data.yearsOfExperience) : undefined,

            });

            toast.success("Registrar created successfully");

            router.push("/sysadmin-registrar");

        } catch (error) {

            toast.error("Failed to create registrar: " + (error as Error).message);

        } finally {

            setIsLoading(false);

        }

    };

    return (

        <div className="container mx-auto p-4">

            <Breadcrumb className="hidden md:flex">

                <BreadcrumbList>

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin">Dashboard</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin-registrar">Registrar</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbPage>Add a School Registrar</BreadcrumbPage>

                    </BreadcrumbItem>

                </BreadcrumbList>

            </Breadcrumb>

            <main className="space-y-4 mt-8">

                <form onSubmit={handleSubmit(onSubmit)}>

                    <div className="mx-auto grid max-w-[59rem] flex-1 auto-rows-max gap-4">

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

                            <Link

                                href="/sysadmin-registrar"

                                className={cn("h-7 w-7", buttonVariants({

                                    variant: "outline",

                                    size: "icon",

                                }))}>

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

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

                            </Link>

                            <h1 className="flex-1 shrink-0 whitespace-nowrap text-xl font-semibold tracking-tight sm:grow-0">

                                Add a School Registrar

                            </h1>

                            {/\* Button for desktop/laptop users \*/}

                            <div className="hidden items-center gap-2 md:ml-auto md:flex">

                                <Button

                                    type="submit"

                                    size="sm"

                                    className="text-white"

                                    disabled={isLoading}

                                >

                                    {isLoading ? "Creating..." : "Save"}

                                </Button>

                            </div>

                        </div>

                        <div className="grid gap-4 md:grid-cols-[1fr\_250px] lg:grid-cols-3 lg:gap-8">

                            <div className="grid auto-rows-max items-start gap-4 lg:col-span-2 lg:gap-8">

                                {/\* Fname, Lname, Mname, Desc \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Registrar Details</CardTitle>

                                        <CardDescription>

                                            Provide the details of the registrar.

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="name">Email</Label>

                                                <Input

                                                    {...register("email", {

                                                        required: "Email is required",

                                                        pattern: {

                                                            value: /^[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i,

                                                            message: "Invalid email address"

                                                        }

                                                    })}

                                                />

                                                {errors.email && <p className="text-red-500">{errors.email.message}</p>}

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">First Name</Label>

                                                    <Input

                                                        id="name"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="John"

                                                        {...register("firstName", { required: "First Name is required" })}

                                                    />

                                                    {errors.firstName && <p className="text-red-500">{errors.firstName.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Last Name</Label>

                                                    <Input

                                                        id="name"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Doe"

                                                        {...register("lastName", { required: "Last Name is required" })}

                                                    />

                                                    {errors.lastName && <p className="text-red-500">{errors.lastName.message}</p>}

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Middle Name (Optional)</Label>

                                                    <Input

                                                        id="name"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Kennedy"

                                                        {...register("middleName")}

                                                    />

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="empId">Employee ID</Label>

                                                    <Input

                                                        id="empId"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="2023-002"

                                                        {...register("employeeId")}

                                                    />

                                                    {errors.employeeId && <p className="text-red-500">{errors.employeeId.message}</p>}

                                                </div>

                                            </div>

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

                                                {/\* password and current password \*/}

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Password</Label>

                                                    <Input

                                                        id="password"

                                                        type="password"

                                                        className="w-full"

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                        {...register("password", { required: "Password is required" })}

                                                    />

                                                    {errors.password && <p className="text-red-500">{errors.password.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="name">Confirm Password</Label>

                                                    <Input

                                                        id="cpassword"

                                                        type="password"

                                                        className="w-full"

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                        {...register("confirmPassword", { required: "Confirm Password is required" })}

                                                    />

                                                    {errors.confirmPassword && <p className="text-red-500">{errors.confirmPassword.message}</p>}

                                                </div>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Other details \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Other details</CardTitle>

                                        <CardDescription>

                                            Provide the details of the registrar.

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="birthDate">Birth Date <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="birthDate"

                                                    type="date"

                                                    className="w-full"

                                                    {...register("birthDate", {

                                                        required: "Birth date is required",

                                                        validate: (value) => {

                                                            if (!value) return "Birth date is required";

                                                            const birthDate = new Date(value);

                                                            const today = new Date();

                                                            let age = today.getFullYear() - birthDate.getFullYear();

                                                            if (

                                                                today.getMonth() < birthDate.getMonth() ||

                                                                (today.getMonth() === birthDate.getMonth() && today.getDate() < birthDate.getDate())

                                                            ) {

                                                                age--;

                                                            }

                                                            if (age < 18) return "Must be at least 18 years old";

                                                            if (age > 65) return "Must not exceed 65 years old";

                                                            return true;

                                                        }

                                                    })}

                                                />

                                                {errors.birthDate && <p className="text-red-500">{errors.birthDate.message}</p>}

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="gender">Gender <span className="text-red-500">\*</span></Label>

                                                    <Select

                                                        onValueChange={(value) => setValue("gender", value)}

                                                        {...register("gender", { required: "Gender is required" })}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select gender" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Gender</SelectLabel>

                                                                <SelectItem value="male">Male</SelectItem>

                                                                <SelectItem value="female">Female</SelectItem>

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                    {errors.gender && <p className="text-red-500">{errors.gender.message}</p>}

                                                </div>

                                                <div className="grid gap-3">

                                                    {/\* contact number \*/}

                                                    <Label htmlFor="contact">Contact Number <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="contact"

                                                        type="text"

                                                        className="w-full"

                                                        placeholder="Enter your contact number"

                                                        {...register("contactNumber", { required: "Contact Number is required" })}

                                                        maxLength={11}

                                                    />

                                                    {errors.contactNumber && <p className="text-red-500">{errors.contactNumber.message}</p>}

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                {/\* yrs of exp \*/}

                                                <Label htmlFor="yrsOfExp">Years of Experience</Label>

                                                <Input

                                                    id="yrsOfExp"

                                                    type="number"

                                                    className="w-full"

                                                    placeholder="Enter years of experience"

                                                    {...register("yearsOfExperience")}

                                                />

                                                {errors.yearsOfExperience && <p className="text-red-500">{errors.yearsOfExperience.message}</p>}

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Address details \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Address Details (Optional)</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="region">Region</Label>

                                                    <Select onValueChange={handleRegionChange}>

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select region" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Region</SelectLabel>

                                                                {regions.map(region => (

                                                                    <SelectItem key={region.code} value={region.code}>

                                                                        {region.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="province">Province</Label>

                                                    {isNCR ? (

                                                        <Input

                                                            value="Metro Manila"

                                                            disabled

                                                            className="bg-muted"

                                                        />

                                                    ) : (

                                                        <Select

                                                            onValueChange={handleProvinceChange}

                                                            disabled={!watch('region')}

                                                        >

                                                            <SelectTrigger className="w-full">

                                                                <SelectValue placeholder="Select province" />

                                                            </SelectTrigger>

                                                            <SelectContent>

                                                                <SelectGroup>

                                                                    <SelectLabel>Province</SelectLabel>

                                                                    {provinces.map(province => (

                                                                        <SelectItem key={province.code} value={province.code}>

                                                                            {province.name}

                                                                        </SelectItem>

                                                                    ))}

                                                                </SelectGroup>

                                                            </SelectContent>

                                                        </Select>

                                                    )}

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="city">City/Municipality</Label>

                                                    <Select

                                                        onValueChange={handleCityChange}

                                                        disabled={!watch('region') || (!isNCR && !watch('province'))}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select city" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>City</SelectLabel>

                                                                {cities.map(city => (

                                                                    <SelectItem key={city.code} value={city.code}>

                                                                        {city.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="barangay">Barangay</Label>

                                                    <Select

                                                        onValueChange={handleBarangayChange}

                                                        disabled={!watch('city')}

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select barangay" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Barangay</SelectLabel>

                                                                {barangays.map(barangay => (

                                                                    <SelectItem key={barangay.code} value={barangay.code}>

                                                                        {barangay.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="strt">Street</Label>

                                                <Input

                                                    id="strt"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Enter street"

                                                    {...register("street")}

                                                />

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                            </div>

                            <div className="grid auto-rows-max items-start gap-4 lg:gap-8">

                                <Card

                                    className="overflow-hidden" x-chunk="dashboard-07-chunk-4"

                                >

                                    <CardHeader>

                                        <CardTitle>School Registrar Image</CardTitle>

                                        <CardDescription>

                                            Upload an image of the school registrar

                                        </CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <UserAvatarUpload

                                            onImageUpload={(storageId) => setImageStorageId(storageId)}

                                            onImageRemove={() => setImageStorageId(undefined)}

                                        />

                                    </CardContent>

                                </Card>

                            </div>

                        </div>

                        {/\* Button for mobile users \*/}

                        <div className="flex items-center justify-center gap-2 md:hidden">

                            <Button

                                type="submit"

                                size="sm"

                                className="text-white"

                                disabled={isLoading}

                            >

                                {isLoading ? "Creating..." : "Save"}

                            </Button>

                        </div>

                    </div>

                </form>

            </main>

        </div>

    )

}

export default SystemAdminAddRegistrarPage;

Create Teachers:

"use client";

import {

    Breadcrumb,

    BreadcrumbItem,

    BreadcrumbLink,

    BreadcrumbList,

    BreadcrumbPage,

    BreadcrumbSeparator,

} from "@/components/ui/breadcrumb";

import { Button, buttonVariants } from "@/components/ui/button";

import {

    Card,

    CardContent,

    CardDescription,

    CardHeader,

    CardTitle

} from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import {

    Select,

    SelectContent,

    SelectGroup,

    SelectItem,

    SelectLabel,

    SelectTrigger,

    SelectValue,

} from "@/components/ui/select";

import {

    ChevronLeft

} from "lucide-react";

import { UserAvatarUpload } from "@/components/shared/user-avatar-upload";

import { Barangay, City, Province, Region, fetchBarangays, fetchCities, fetchProvinces, fetchRegions } from "@/lib/address-api";

import { cn } from "@/lib/utils";

import { useMutation } from "convex/react";

import Link from "next/link";

import { useRouter } from "next/navigation";

import { useEffect, useState } from "react";

import { useForm } from "react-hook-form";

import { toast } from "sonner";

import { api } from "../../../../../convex/\_generated/api";

import { Id } from "../../../../../convex/\_generated/dataModel";

import { schoolSubjects } from "../../../../../data/teachers-data";

const SystemAdminAddTeacherPage = () => {

    const createTeacher = useMutation(api.users.createTeacher);

    const [isLoading, setIsLoading] = useState(false);

    const router = useRouter();

    const [imageStorageId, setImageStorageId] = useState<string | undefined>();

    const [regions, setRegions] = useState<Region[]>([]);

    const [provinces, setProvinces] = useState<Province[]>([]);

    const [cities, setCities] = useState<City[]>([]);

    const [barangays, setBarangays] = useState<Barangay[]>([]);

    const [isNCR, setIsNCR] = useState(false);

    const { setValue, watch } = useForm();

    // const [selectedSubjects, setSelectedSubjects] = useState<SchoolSubjects[]>([])

    useEffect(() => {

        fetchRegions().then(data => {

            setRegions(data);

        });

    }, []);

    const handleRegionChange = async (regionCode: string) => {

        setCities([]);

        setBarangays([]);

        const isNCRSelected = regionCode === '130000000';

        setIsNCR(isNCRSelected);

        const selectedRegion = regions.find(r => r.code === regionCode);

        if (selectedRegion) {

            setValue('region', selectedRegion.name);

        }

        if (isNCRSelected) {

            setValue('province', 'Metro Manila');

            const cityData = await fetchCities('130000000');

            setCities(cityData);

        } else {

            const provinceData = await fetchProvinces(regionCode);

            setProvinces(provinceData);

        }

    };

    const handleProvinceChange = async (provinceCode: string) => {

        setValue('city', '');

        setValue('barangay', '');

        setBarangays([]);

        const selectedProvince = provinces.find(p => p.code === provinceCode);

        if (selectedProvince) {

            setValue('province', selectedProvince.name);

        }

        const cityData = await fetchCities(provinceCode);

        setCities(cityData);

    };

    const handleCityChange = async (cityCode: string) => {

        setValue('barangay', '');

        const selectedCity = cities.find(c => c.code === cityCode);

        if (selectedCity) {

            setValue('city', selectedCity.name);

        }

        try {

            const barangayData = await fetchBarangays(cityCode);

            setBarangays(barangayData);

        } catch (error) {

            toast.error('Failed to fetch barangays: ' + (error as Error).message);

        }

    };

    const handleBarangayChange = (barangayCode: string) => {

        const selectedBarangay = barangays.find(b => b.code === barangayCode);

        if (selectedBarangay) {

            setValue('barangay', selectedBarangay.name);

        }

    };

    const handleSubmit = async (e: React.FormEvent<HTMLFormElement>) => {

        e.preventDefault();

        setIsLoading(true);

        try {

            const formData = new FormData(e.currentTarget);

            // Validate required fields

            const email = formData.get('email') as string;

            const password = formData.get('password') as string;

            const confirmPassword = formData.get('cpassword') as string;

            if (password !== confirmPassword) {

                toast.error("Passwords do not match");

                return;

            }

            await createTeacher({

                email,

                password,

                firstName: formData.get('firstName') as string,

                lastName: formData.get('lastName') as string,

                middleName: (formData.get('middleName') as string) || undefined,

                employeeId: formData.get('empId') as string,

                contactNumber: formData.get('contact') as string,

                birthDate: formData.get('bday') as string,

                gender: formData.get('gender') as string,

                specialization: formData.get('specialization') as string,

                yearsOfExperience: parseInt(formData.get('yrsOfExp') as string),

                // position: formData.get('position') as string,

                // advisoryClass: (formData.get('advisoryClass') as string) || undefined,

                subjectId: formData.getAll('subjects') as Id<'subjects'>[],

                // Use watch to get the address values

                region: watch('region'),

                province: watch('province'),

                city: watch('city'),

                barangay: watch('barangay'),

                street: formData.get('street') as string,

                ...(imageStorageId && { imageStorageId }),

            });

            toast.success("Teacher created successfully");

            router.push("/sysadmin-teachers");

        } catch (error) {

            toast.error("Failed to create teacher: " + (error as Error).message);

        } finally {

            setIsLoading(false);

        }

    };

    return (

        <div className="container mx-auto p-4">

            <Breadcrumb className="hidden md:flex">

                <BreadcrumbList>

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin">Dashboard</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbLink asChild>

                            <Link href="/sysadmin-teachers">Teachers</Link>

                        </BreadcrumbLink>

                    </BreadcrumbItem>

                    <BreadcrumbSeparator />

                    <BreadcrumbItem>

                        <BreadcrumbPage>Add a Teacher</BreadcrumbPage>

                    </BreadcrumbItem>

                </BreadcrumbList>

            </Breadcrumb>

            <main className="space-y-4 mt-8">

                <form id="teacher-form" onSubmit={handleSubmit}>

                    <div className="mx-auto grid max-w-[59rem] flex-1 auto-rows-max gap-4">

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

                            <Link

                                href="/sysadmin-teachers"

                                className={cn("h-7 w-7", buttonVariants({

                                    variant: "outline",

                                    size: "icon",

                                }))}>

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

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

                            </Link>

                            <h1 className="flex-1 shrink-0 whitespace-nowrap text-xl font-semibold tracking-tight sm:grow-0">

                                Add a Teacher

                            </h1>

                            {/\* Button for desktop/laptop users \*/}

                            <div className="hidden items-center gap-2 md:ml-auto md:flex">

                                <Button

                                    type="submit"

                                    form="teacher-form"

                                    size="sm"

                                    className="text-white"

                                    disabled={isLoading}

                                >

                                    {isLoading ? "Creating..." : "Save"}

                                </Button>

                            </div>

                        </div>

                        <div className="grid gap-4 md:grid-cols-[1fr\_250px] lg:grid-cols-3 lg:gap-8">

                            <div className="grid auto-rows-max items-start gap-4 lg:col-span-2 lg:gap-8">

                                {/\* Teacher Details Card \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Teacher Details</CardTitle>

                                        <CardDescription>Add teacher details</CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="email">Email <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="email"

                                                    name="email"

                                                    type="email"

                                                    required

                                                    placeholder="email@example.com"

                                                />

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="firstName">First Name <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="firstName"

                                                        name="firstName"

                                                        type="text"

                                                        required

                                                        placeholder="John"

                                                    />

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="lastName">Last Name <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="lastName"

                                                        name="lastName"

                                                        type="text"

                                                        required

                                                        placeholder="Doe"

                                                    />

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="middleName">Middle Name</Label>

                                                    <Input

                                                        id="middleName"

                                                        name="middleName"

                                                        type="text"

                                                        placeholder="Middle Name"

                                                    />

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="empId">Employee ID <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="empId"

                                                        name="empId"

                                                        type="text"

                                                        required

                                                        placeholder="2023-002"

                                                    />

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="password">Password <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="password"

                                                        name="password"

                                                        type="password"

                                                        required

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                    />

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="cpassword">Confirm Password <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="cpassword"

                                                        name="cpassword"

                                                        type="password"

                                                        required

                                                        placeholder="\*\*\*\*\*\*\*\*"

                                                    />

                                                </div>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                {/\* Other Details Card \*/}

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Other Details</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

                                            <div className="grid gap-3">

                                                <Label htmlFor="bday">Birth Date <span className="text-red-500">\*</span></Label>

                                                <Input

                                                    id="bday"

                                                    name="bday"

                                                    type="date"

                                                    required

                                                />

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="gender">Gender <span className="text-red-500">\*</span></Label>

                                                    <Select name="gender" required>

                                                        <SelectTrigger>

                                                            <SelectValue placeholder="Select gender" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectItem value="male">Male</SelectItem>

                                                            <SelectItem value="female">Female</SelectItem>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="contact">Contact Number <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="contact"

                                                        name="contact"

                                                        type="tel"

                                                        required

                                                        placeholder="09123456789"

                                                    />

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="specialization">Specialization <span className="text-red-500">\*</span></Label>

                                                    <Select name="specialization" required>

                                                        <SelectTrigger className="w-[276px]">

                                                            <SelectValue placeholder="Select specialization" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            {schoolSubjects.map((subject) => (

                                                                <SelectItem key={subject.value} value={subject.value}>

                                                                    {subject.label}

                                                                </SelectItem>

                                                            ))}

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="yrsOfExp">Years of Experience <span className="text-red-500">\*</span></Label>

                                                    <Input

                                                        id="yrsOfExp"

                                                        name="yrsOfExp"

                                                        type="number"

                                                        required

                                                        min="0"

                                                        placeholder="Years of experience"

                                                    />

                                                </div>

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                                <Card>

                                    <CardHeader>

                                        <CardTitle>Address Details</CardTitle>

                                    </CardHeader>

                                    <CardContent>

                                        <div className="grid gap-6">

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="region">Region (Optional)</Label>

                                                    <Select

                                                        onValueChange={handleRegionChange}

                                                        name="region"

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select region" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Region</SelectLabel>

                                                                {regions.map(region => (

                                                                    <SelectItem key={region.code} value={region.code}>

                                                                        {region.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="province">Province (Optional)</Label>

                                                    {isNCR ? (

                                                        <Input

                                                            value="Metro Manila"

                                                            disabled

                                                            className="bg-muted"

                                                            name="province"

                                                        />

                                                    ) : (

                                                        <Select

                                                            onValueChange={handleProvinceChange}

                                                            disabled={!regions.length}

                                                            name="province"

                                                        >

                                                            <SelectTrigger className="w-full">

                                                                <SelectValue placeholder="Select province" />

                                                            </SelectTrigger>

                                                            <SelectContent>

                                                                <SelectGroup>

                                                                    <SelectLabel>Province</SelectLabel>

                                                                    {provinces.map(province => (

                                                                        <SelectItem key={province.code} value={province.code}>

                                                                            {province.name}

                                                                        </SelectItem>

                                                                    ))}

                                                                </SelectGroup>

                                                            </SelectContent>

                                                        </Select>

                                                    )}

                                                </div>

                                            </div>

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

                                                <div className="grid gap-3">

                                                    <Label htmlFor="city">City/Municipality (Optional)</Label>

                                                    <Select

                                                        onValueChange={handleCityChange}

                                                        disabled={!provinces.length && !isNCR}

                                                        name="city"

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select city" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>City</SelectLabel>

                                                                {cities.map(city => (

                                                                    <SelectItem key={city.code} value={city.code}>

                                                                        {city.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                                <div className="grid gap-3">

                                                    <Label htmlFor="barangay">Barangay (Optional)</Label>

                                                    <Select

                                                        onValueChange={handleBarangayChange}

                                                        disabled={!cities.length}

                                                        name="barangay"

                                                    >

                                                        <SelectTrigger className="w-full">

                                                            <SelectValue placeholder="Select barangay" />

                                                        </SelectTrigger>

                                                        <SelectContent>

                                                            <SelectGroup>

                                                                <SelectLabel>Barangay</SelectLabel>

                                                                {barangays.map(barangay => (

                                                                    <SelectItem key={barangay.code} value={barangay.code}>

                                                                        {barangay.name}

                                                                    </SelectItem>

                                                                ))}

                                                            </SelectGroup>

                                                        </SelectContent>

                                                    </Select>

                                                </div>

                                            </div>

                                            <div className="grid gap-3">

                                                <Label htmlFor="street">Street (Optional)</Label>

                                                <Input

                                                    id="street"

                                                    name="street"

                                                    type="text"

                                                    className="w-full"

                                                    placeholder="Enter street"

                                                />

                                            </div>

                                        </div>

                                    </CardContent>

                                </Card>

                            </div>

                            {/\* Right Column \*/}

                            <div className="grid auto-rows-max items-start gap-4 lg:gap-8">

                                {/\* Teacher Image Card \*/}

                                <Card className="overflow-hidden">

                                    <CardHeader>

                                        <CardTitle>Teacher Image</CardTitle>

                                        <CardDescription>Upload teacher image (Optional)</CardDescription>

                                    </CardHeader>

                                    <CardContent>

                                        <UserAvatarUpload

                                            onImageUpload={(storageId) => setImageStorageId(storageId)}

                                            onImageRemove={() => setImageStorageId(undefined)}

                                        />

                                    </CardContent>

                                </Card>

                            </div>

                        </div>

                        {/\* Mobile Save Button \*/}

                        <div className="flex items-center justify-center gap-2 md:hidden">

                            <Button

                                type="submit"

                                size="sm"

                                className="text-white"

                                disabled={isLoading}

                            >

                                {isLoading ? "Creating..." : "Save"}

                            </Button>

                        </div>

                    </div>

                </form>

            </main>

        </div>

    );

};

export default SystemAdminAddTeacherPage;

**Enterprise Reports and Analytics (ERA)**

Dashboard Page:

"use client"

import {

  Breadcrumb,

  BreadcrumbItem,

  BreadcrumbList,

  BreadcrumbPage,

  BreadcrumbSeparator,

} from "@/components/ui/breadcrumb"

import {

  Card,

  CardContent,

  CardHeader,

  CardTitle

} from "@/components/ui/card"

import { useQuery } from "convex/react"

import { GraduationCapIcon, UserCheckIcon, Users2Icon } from "lucide-react"

import { api } from "../../../../convex/\_generated/api"

import EraInterventionCard from "./\_components/era-intervention-card"

import { EraSparCard } from "./\_components/era-spar-card"

function DashboardPage() {

  const counts = useQuery(api.students.getTeacherStudentsCount);

  return (

    <div className='containter mx-auto p-4 space-y-3'>

      <Breadcrumb className="hidden md:flex mb-6">

        <BreadcrumbList>

          <BreadcrumbItem>

            <BreadcrumbPage className="text-muted-foreground">Dashboard</BreadcrumbPage>

          </BreadcrumbItem>

          <BreadcrumbSeparator />

          <BreadcrumbItem>

            <BreadcrumbPage>Overview</BreadcrumbPage>

          </BreadcrumbItem>

        </BreadcrumbList>

      </Breadcrumb>

      <div className="grid gap-4 md:grid-cols-3 mb-8">

        <Card className="bg-foreground text-white">

          <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

            <CardTitle className="text-sm font-medium">Total Students</CardTitle>

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

          </CardHeader>

          <CardContent>

            <div className="text-2xl font-bold">{counts?.totalStudents || 0}</div>

            <p className="text-xs ">Total Students</p>

          </CardContent>

        </Card>

        <Card className="bg-primary-foreground text-white">

          <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

            <CardTitle className="text-sm font-medium">At-Risk Students</CardTitle>

            <Users2Icon className="h-4 w-4 " />

          </CardHeader>

          <CardContent>

            <div className="text-2xl font-bold">{counts?.atRiskStudents || 0}</div>

            <p className="text-xs ">At-Risk Students</p>

          </CardContent>

        </Card>

        <Card className="bg-primary text-white">

          <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

            <CardTitle className="text-sm font-medium">Pending Interventions</CardTitle>

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

          </CardHeader>

          <CardContent>

            <div className="text-2xl font-bold">{counts?.pendingInterventions || 0}</div>

            <p className="text-xs ">Pending Interventions</p>

          </CardContent>

        </Card>

      </div>

      <EraSparCard />

      <EraInterventionCard />

      {/\* <EraPostInterventionCard /> \*/}

    </div>

  )

}

export default DashboardPage

ERA – Student Performance Analytics and Reports:

"use client"

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select"

import { useQuery } from "convex/react"

import { useState } from "react"

import { StudentPerformanceChart } from "./student-performance-chart"

import { Id } from "../../../../../convex/\_generated/dataModel"

import { api } from "../../../../../convex/\_generated/api"

export const EraSparCard = () => {

    const [selectedClass, setSelectedClass] = useState<Id<"classes"> | null>(null);

    const [selectedStudent, setSelectedStudent] = useState<Id<"students"> | null>(null);

    const [selectedSubject, setSelectedSubject] = useState<Id<"subjects"> | null>(null);

    // Get teacher's classes with section details

    const classes = useQuery(api.classes.getTeacherClasses);

    // Get students for selected class

    const studentsWithGrades = useQuery(api.students.getStudentsWithGrades, {

        classId: selectedClass ?? undefined,

        subjectId: selectedSubject ?? undefined

    });

    // Get subjects for selected class

    // const subjects = useQuery(api.subjects.getTeacherSubjects);

    if (!classes) {

        return (

            <Card>

                <CardHeader>

                    <CardTitle>Student Performance Analytics and Reports</CardTitle>

                </CardHeader>

                <CardContent>Loading...</CardContent>

            </Card>

        );

    }

    // console.log(`Classes: ${JSON.stringify(classes)}`);

    console.log(`Students: ${JSON.stringify(studentsWithGrades)}`);

    // console.log(`Subjects: ${JSON.stringify(subjects)}`);

    return (

        <Card>

            <CardHeader>

                <CardTitle>Student Performance Analytics and Reports</CardTitle>

            </CardHeader>

            <CardContent>

                <div className="flex flex-col space-y-4 md:flex-row md:space-x-4 md:space-y-0 mb-6">

                    <Select

                        value={selectedClass?.toString() ?? ""}

                        onValueChange={(value) => {

                            setSelectedClass(value as Id<"classes">);

                            setSelectedStudent(null); // Reset student when class changes

                            setSelectedSubject(null); // Reset subject when class changes

                        }}

                    >

                        <SelectTrigger className="w-[200px]">

                            <SelectValue placeholder="Select Section" />

                        </SelectTrigger>

                        <SelectContent>

                            {classes.map((cls) => (

                                <SelectItem

                                    key={cls?.\_id}

                                    value={cls?.\_id as string}

                                >

                                    {cls?.section?.name} - {cls?.subject?.name}

                                </SelectItem>

                            ))}

                        </SelectContent>

                    </Select>

                    {selectedClass && (

                        <Select

                            value={selectedStudent?.toString() ?? ""}

                            onValueChange={(value) => {

                                setSelectedStudent(value as Id<"students">);

                                // Automatically set subject to the class subject

                                const selectedClassData = classes.find(c => c?.\_id === selectedClass);

                                if (selectedClassData) {

                                    setSelectedSubject(selectedClassData.subjectId);

                                }

                            }}

                        >

                            <SelectTrigger className="w-[200px]">

                                <SelectValue placeholder="Select Student" />

                            </SelectTrigger>

                            <SelectContent>

                                {studentsWithGrades?.map((student) => (

                                    <SelectItem

                                        key={student.id}

                                        value={student.id}

                                    >

                                        {student.name}

                                    </SelectItem>

                                ))}

                            </SelectContent>

                        </Select>

                    )}

                </div>

                {selectedStudent && selectedSubject && selectedClass && (

                    <StudentPerformanceChart

                        studentId={selectedStudent}

                        subjectId={selectedSubject}

                        classId={selectedClass}

                        key={`${selectedStudent}-${selectedSubject}-${selectedClass}`}

                    />

                )}

            </CardContent>

        </Card>

    );

};

ERA – Student Performance Chart:

"use client"

import { Bar, BarChart, XAxis, YAxis } from "recharts"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { ChartContainer, ChartTooltip, ChartTooltipContent } from "@/components/ui/chart"

import { useQuery } from "convex/react"

import { api } from "../../../../../convex/\_generated/api"

import { Id } from "../../../../../convex/\_generated/dataModel"

interface StudentPerformanceChartProps {

    studentId: Id<"students">

    subjectId: Id<"subjects">

    classId: Id<"classes">

}

export function StudentPerformanceChart({

    studentId,

    subjectId,

    classId

}: StudentPerformanceChartProps) {

    const students = useQuery(api.students.getStudentsWithGrades, {

        classId,

        subjectId

    });

    const subject = useQuery(api.subjects.getById, { id: subjectId });

    console.log('Students Data:', students);

    console.log('Selected Student:', studentId);

    console.log('Selected Class:', classId);

    console.log('Selected Subject:', subjectId);

    if (!students || !subject) {

        return (

            <Card className="lg:w-fit">

                <CardHeader>

                    <CardTitle className="text-text">Student Performance</CardTitle>

                    <CardDescription className="text-muted-foreground">

                        Loading data...

                    </CardDescription>

                </CardHeader>

                <CardContent>

                    <div className="flex items-center justify-center h-[200px]">

                        Loading...

                    </div>

                </CardContent>

            </Card>

        );

    }

    const studentData = students.find(s => s.id === studentId);

    if (!studentData || !studentData.grades || studentData.grades.length === 0) {

        return (

            <Card className="lg:w-fit">

                <CardHeader>

                    <CardTitle className="text-text">Student Performance</CardTitle>

                    <CardDescription className="text-muted-foreground">

                        {subject.name}

                    </CardDescription>

                </CardHeader>

                <CardContent>

                    <div className="flex items-center justify-center h-[200px]">

                        No grades recorded for this student yet

                    </div>

                </CardContent>

            </Card>

        );

    }

    const chartData = studentData.grades

        .map(grade => ({

            quarter: `Q${grade.quarter}`,

            score: grade.quarterlyGrade

        }))

        .sort((a, b) => a.quarter.localeCompare(b.quarter));

    const chartConfig = {

        score: {

            label: "Score",

            color: "hsl(var(--pieChart-1))",

        },

    }

    return (

        <Card className="lg:w-fit">

            <CardHeader>

                <CardTitle className="text-text">Student Performance</CardTitle>

                <CardDescription className="text-muted-foreground">

                    Quarterly scores for {subject.name}

                </CardDescription>

            </CardHeader>

            <CardContent>

                <div className="w-full overflow-x-auto">

                    <div className="min-w-[300px]">

                        <ChartContainer config={chartConfig} className="h-[200px] sm:h-[300px]">

                            <BarChart accessibilityLayer data={chartData}>

                                <XAxis

                                    dataKey="quarter"

                                    tickLine={false}

                                    axisLine={false}

                                    tickMargin={10}

                                    stroke="hsl(142, 76%, 36%)"

                                />

                                <YAxis

                                    tickLine={false}

                                    axisLine={false}

                                    stroke="hsl(142, 76%, 36%)"

                                    domain={[0, 100]}

                                    ticks={[0, 20, 40, 60, 80, 100]}

                                />

                                <ChartTooltip

                                    cursor={false}

                                    content={<ChartTooltipContent />}

                                />

                                <Bar

                                    dataKey="score"

                                    fill="var(--color-score)"

                                    radius={[4, 4, 0, 0]}

                                />

                            </BarChart>

                        </ChartContainer>

                    </div>

                </div>

            </CardContent>

        </Card>

    )

}

ERA – Intervention Card:

"use client"

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select"

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"

import { useQuery } from "convex/react"

import { useState } from "react"

import { ActualPerformanceChart } from "./actual-performance-chart"

import { ActualInterventionPerformanceChart } from "./actual-intervention-performance-chart"

import { CombinedInterventionChart } from "./combined-intervention-chart"

import { Id } from "../../../../../convex/\_generated/dataModel"

import { api } from "../../../../../convex/\_generated/api"

export default function EraInterventionCard() {

    const [selectedClass, setSelectedClass] = useState<Id<"classes"> | null>(null);

    const [selectedStudent, setSelectedStudent] = useState<Id<"students"> | null>(null);

    const [selectedSubject, setSelectedSubject] = useState<Id<"subjects"> | null>(null);

    // Get teacher's classes with section details

    const classes = useQuery(api.classes.getTeacherClasses);

    // Get students for selected class

    const studentsWithGrades = useQuery(api.students.getStudentsWithGrades, {

        classId: selectedClass ?? undefined,

        subjectId: selectedSubject ?? undefined

    });

    if (!classes) {

        return (

            <Card>

                <CardHeader>

                    <CardTitle>Intervention Analytics and Reports</CardTitle>

                </CardHeader>

                <CardContent>Loading...</CardContent>

            </Card>

        );

    }

    return (

        <Card>

            <CardHeader>

                <CardTitle>Intervention Analytics and Reports</CardTitle>

            </CardHeader>

            <CardContent>

                <div className="flex flex-col space-y-4 md:flex-row md:space-x-4 md:space-y-0 mb-6">

                    <Select

                        value={selectedClass?.toString() ?? ""}

                        onValueChange={(value) => {

                            setSelectedClass(value as Id<"classes">);

                            setSelectedStudent(null); // Reset student when class changes

                            setSelectedSubject(null); // Reset subject when class changes

                        }}

                    >

                        <SelectTrigger className="w-[200px]">

                            <SelectValue placeholder="Select Section" />

                        </SelectTrigger>

                        <SelectContent>

                            {classes.map((cls) => (

                                <SelectItem

                                    key={cls?.\_id}

                                    value={cls?.\_id as string}

                                >

                                    {cls?.section?.name} - {cls?.subject?.name}

                                </SelectItem>

                            ))}

                        </SelectContent>

                    </Select>

                    {selectedClass && (

                        <Select

                            value={selectedStudent?.toString() ?? ""}

                            onValueChange={(value) => {

                                setSelectedStudent(value as Id<"students">);

                                // Automatically set subject to the class subject

                                const selectedClassData = classes.find(c => c?.\_id === selectedClass);

                                if (selectedClassData) {

                                    setSelectedSubject(selectedClassData.subjectId);

                                }

                            }}

                        >

                            <SelectTrigger className="w-[200px]">

                                <SelectValue placeholder="Select Student" />

                            </SelectTrigger>

                            <SelectContent>

                                {studentsWithGrades?.map((student) => (

                                    <SelectItem

                                        key={student.id}

                                        value={student.id}

                                    >

                                        {student.name}

                                    </SelectItem>

                                ))}

                            </SelectContent>

                        </Select>

                    )}

                </div>

                {selectedStudent && selectedSubject && selectedClass && (

                    <Tabs defaultValue="actual" className="space-y-4">

                        <TabsList className="flex flex-wrap justify-start gap-2 mb-[96px] md:mb-3">

                            <TabsTrigger value="actual" className="flex-grow sm:flex-grow-0">

                                Actual

                            </TabsTrigger>

                            <TabsTrigger value="actual-intervention" className="flex-grow sm:flex-grow-0">

                                Actual Intervention

                            </TabsTrigger>

                            <TabsTrigger value="combined" className="flex-grow sm:flex-grow-0">

                                Combined

                            </TabsTrigger>

                        </TabsList>

                        <TabsContent value="actual">

                            <ActualPerformanceChart

                                studentId={selectedStudent}

                                subjectId={selectedSubject}

                                classId={selectedClass}

                            />

                        </TabsContent>

                        <TabsContent value="actual-intervention">

                            <ActualInterventionPerformanceChart

                                studentId={selectedStudent}

                                subjectId={selectedSubject}

                                classId={selectedClass}

                            />

                        </TabsContent>

                        <TabsContent value="combined">

                            <CombinedInterventionChart

                                studentId={selectedStudent}

                                subjectId={selectedSubject}

                                classId={selectedClass}

                            />

                        </TabsContent>

                    </Tabs>

                )}

            </CardContent>

        </Card>

    );

}

ERA – Actual Performance Chart (Intervention) :

"use client"

import { Bar, BarChart, XAxis, YAxis } from "recharts"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { ChartContainer, ChartTooltip, ChartTooltipContent } from "@/components/ui/chart"

import { useQuery } from "convex/react"

import { api } from "../../../../../convex/\_generated/api"

import { Id } from "../../../../../convex/\_generated/dataModel"

interface ActualPerformanceChartProps {

  studentId: Id<"students">

  subjectId: Id<"subjects">

  classId: Id<"classes">

}

export function ActualPerformanceChart({

  studentId,

  subjectId,

  classId

}: ActualPerformanceChartProps) {

  const students = useQuery(api.students.getStudentsWithGrades, {

    classId,

    subjectId

  });

  const subject = useQuery(api.subjects.getById, { id: subjectId });

  if (!students || !subject) {

    return (

      <Card className="lg:w-fit">

        <CardHeader>

          <CardTitle className="text-text">Actual Performance</CardTitle>

          <CardDescription className="text-muted-foreground">

            Loading data...

          </CardDescription>

        </CardHeader>

        <CardContent>

          <div className="flex items-center justify-center h-[200px]">

            Loading...

          </div>

        </CardContent>

      </Card>

    );

  }

  const studentData = students.find(s => s.id === studentId);

  if (!studentData || !studentData.grades || studentData.grades.length === 0) {

    return (

      <Card className="lg:w-fit">

        <CardHeader>

          <CardTitle className="text-text">Actual Performance</CardTitle>

          <CardDescription className="text-muted-foreground">

            {subject.name}

          </CardDescription>

        </CardHeader>

        <CardContent>

          <div className="flex items-center justify-center h-[200px]">

            No grades recorded for this student yet

          </div>

        </CardContent>

      </Card>

    );

  }

  const chartData = studentData.grades

    .map(grade => ({

      quarter: `Q${grade.quarter}`,

      score: grade.quarterlyGrade

    }))

    .sort((a, b) => a.quarter.localeCompare(b.quarter));

  const chartConfig = {

    score: {

      label: "Score",

      color: "hsl(var(--pieChart-1))",

    },

  }

  return (

    <Card className="lg:w-fit">

      <CardHeader>

        <CardTitle className="text-text">Actual Performance</CardTitle>

        <CardDescription className="text-muted-foreground">

          Quarterly scores for {subject.name}

        </CardDescription>

      </CardHeader>

      <CardContent>

        <div className="w-full overflow-x-auto">

          <div className="min-w-[300px]">

            <ChartContainer config={chartConfig} className="h-[200px] sm:h-[300px]">

              <BarChart accessibilityLayer data={chartData}>

                <XAxis

                  dataKey="quarter"

                  tickLine={false}

                  axisLine={false}

                  tickMargin={10}

                  stroke="hsl(142, 76%, 36%)"

                />

                <YAxis

                  tickLine={false}

                  axisLine={false}

                  stroke="hsl(142, 76%, 36%)"

                  domain={[0, 100]}

                  ticks={[0, 20, 40, 60, 80, 100]}

                />

                <ChartTooltip cursor={false} content={<ChartTooltipContent />} />

                <Bar

                  dataKey="score"

                  fill="var(--color-score)"

                  radius={[4, 4, 0, 0]}

                />

              </BarChart>

            </ChartContainer>

          </div>

        </div>

      </CardContent>

    </Card>

  );

}

ERA – Actual Intervention Performance Chart:

"use client"

import { Bar, BarChart, XAxis, YAxis } from "recharts"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { ChartContainer, ChartTooltip, ChartTooltipContent } from "@/components/ui/chart"

import { useQuery } from "convex/react"

import { api } from "../../../../../convex/\_generated/api"

import { Id } from "../../../../../convex/\_generated/dataModel"

interface ActualPerformanceChartProps {

  studentId: Id<"students">

  subjectId: Id<"subjects">

  classId: Id<"classes">

}

export function ActualPerformanceChart({

  studentId,

  subjectId,

  classId

}: ActualPerformanceChartProps) {

  const students = useQuery(api.students.getStudentsWithGrades, {

    classId,

    subjectId

  });

  const subject = useQuery(api.subjects.getById, { id: subjectId });

  if (!students || !subject) {

    return (

      <Card className="lg:w-fit">

        <CardHeader>

          <CardTitle className="text-text">Actual Performance</CardTitle>

          <CardDescription className="text-muted-foreground">

            Loading data...

          </CardDescription>

        </CardHeader>

        <CardContent>

          <div className="flex items-center justify-center h-[200px]">

            Loading...

          </div>

        </CardContent>

      </Card>

    );

  }

  const studentData = students.find(s => s.id === studentId);

  if (!studentData || !studentData.grades || studentData.grades.length === 0) {

    return (

      <Card className="lg:w-fit">

        <CardHeader>

          <CardTitle className="text-text">Actual Performance</CardTitle>

          <CardDescription className="text-muted-foreground">

            {subject.name}

          </CardDescription>

        </CardHeader>

        <CardContent>

          <div className="flex items-center justify-center h-[200px]">

            No grades recorded for this student yet

          </div>

        </CardContent>

      </Card>

    );

  }

  const chartData = studentData.grades

    .map(grade => ({

      quarter: `Q${grade.quarter}`,

      score: grade.quarterlyGrade

    }))

    .sort((a, b) => a.quarter.localeCompare(b.quarter));

  const chartConfig = {

    score: {

      label: "Score",

      color: "hsl(var(--pieChart-1))",

    },

  }

  return (

    <Card className="lg:w-fit">

      <CardHeader>

        <CardTitle className="text-text">Actual Performance</CardTitle>

        <CardDescription className="text-muted-foreground">

          Quarterly scores for {subject.name}

        </CardDescription>

      </CardHeader>

      <CardContent>

        <div className="w-full overflow-x-auto">

          <div className="min-w-[300px]">

            <ChartContainer config={chartConfig} className="h-[200px] sm:h-[300px]">

              <BarChart accessibilityLayer data={chartData}>

                <XAxis

                  dataKey="quarter"

                  tickLine={false}

                  axisLine={false}

                  tickMargin={10}

                  stroke="hsl(142, 76%, 36%)"

                />

                <YAxis

                  tickLine={false}

                  axisLine={false}

                  stroke="hsl(142, 76%, 36%)"

                  domain={[0, 100]}

                  ticks={[0, 20, 40, 60, 80, 100]}

                />

                <ChartTooltip cursor={false} content={<ChartTooltipContent />} />

                <Bar

                  dataKey="score"

                  fill="var(--color-score)"

                  radius={[4, 4, 0, 0]}

                />

              </BarChart>

            </ChartContainer>

          </div>

        </div>

      </CardContent>

    </Card>

  );

}

ERA – Combined Intervention Chart:

"use client"

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card"

import { ChartContainer, ChartTooltip } from "@/components/ui/chart"

import { useQuery } from "convex/react"

import { Bar, BarChart, Legend, XAxis, YAxis } from "recharts"

import { api } from "../../../../../convex/\_generated/api"

import { Id } from "../../../../../convex/\_generated/dataModel"

import { InterventionStatsPanel } from "./intervention-stats-panel"

interface CombinedInterventionChartProps {

    studentId: Id<"students">

    subjectId: Id<"subjects">

    classId: Id<"classes">

}

export function CombinedInterventionChart({

    studentId,

    subjectId,

    classId

}: CombinedInterventionChartProps) {

    // Get both regular and intervention grades

    const students = useQuery(api.students.getStudentsWithGrades, {

        classId,

        subjectId

    });

    const subject = useQuery(api.subjects.getById, { id: subjectId });

    if (!students || !subject) {

        return (

            <Card className="lg:w-fit">

                <CardHeader>

                    <CardTitle className="text-text">Combined Performance</CardTitle>

                    <CardDescription className="text-muted-foreground">

                        Loading data...

                    </CardDescription>

                </CardHeader>

                <CardContent>

                    <div className="flex items-center justify-center h-[200px]">

                        Loading...

                    </div>

                </CardContent>

            </Card>

        );

    }

    const studentData = students.find(s => s.id === studentId);

    if (!studentData || !studentData.grades || studentData.grades.length === 0) {

        return (

            <Card className="lg:w-fit">

                <CardHeader>

                    <CardTitle className="text-text">Combined Performance</CardTitle>

                    <CardDescription className="text-muted-foreground">

                        {subject.name}

                    </CardDescription>

                </CardHeader>

                <CardContent>

                    <div className="flex items-center justify-center h-[200px]">

                        No grades recorded for this student yet

                    </div>

                </CardContent>

            </Card>

        );

    }

    // eslint-disable-next-line @typescript-eslint/no-explicit-any

    const calculateStats = (grades: any[]) => {

        const interventionGrades = grades.filter(g => g.needsIntervention);

        const totalInterventions = interventionGrades.length;

        let averageImprovement = 0;

        let successRate = 0;

        const interventionMethods = new Map<string, number>();

        if (totalInterventions > 0) {

            // Calculate improvements

            const improvements = interventionGrades.map(g =>

                g.interventionGrade - g.quarterlyGrade

            );

            averageImprovement =

                improvements.reduce((a, b) => a + b, 0) / improvements.length;

            // Calculate success rate

            const successfulInterventions = interventionGrades.filter(

                g => g.interventionGrade > g.quarterlyGrade

            ).length;

            successRate = (successfulInterventions / totalInterventions) \* 100;

            // Count intervention methods

            interventionGrades.forEach(g => {

                if (g.interventionUsed && Array.isArray(g.interventionUsed)) {

                    g.interventionUsed.forEach((method: string) => {

                        interventionMethods.set(

                            method,

                            (interventionMethods.get(method) || 0) + 1

                        );

                    });

                }

            });

        }

        // Get top 3 most used interventions

        const mostUsedInterventions = Array.from(interventionMethods.entries())

            .sort((a, b) => b[1] - a[1])

            .slice(0, 3)

            .map(([method]) => method);

        return {

            totalInterventions,

            averageImprovement,

            successRate,

            mostUsedInterventions

        };

    };

    // Combine and format data for comparison

    const chartData = studentData.grades.map(grade => ({

        quarter: `Q${grade.quarter}`,

        actual: grade.quarterlyGrade,

        intervention: grade.needsIntervention ? grade.interventionGrade : null,

    })).sort((a, b) => a.quarter.localeCompare(b.quarter));

    const chartConfig = {

        actual: {

            label: "Actual Grade",

            color: "hsl(var(--pieChart-1))",

        },

        intervention: {

            label: "After Intervention",

            color: "hsl(var(--chart-2))",

        }

    }

    return (

        <Card className="lg:w-fit">

            <CardHeader>

                <CardTitle className="text-text">Combined Performance</CardTitle>

                <CardDescription className="text-muted-foreground">

                    Comparison of actual vs intervention grades for {subject.name}

                </CardDescription>

            </CardHeader>

            <CardContent>

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

                    <InterventionStatsPanel stats={calculateStats(studentData.grades)} />

                    <div className="w-full overflow-x-auto">

                        <div className="min-w-[300px]">

                            <ChartContainer config={chartConfig} className="h-[300px]">

                                <BarChart

                                    accessibilityLayer

                                    data={chartData}

                                    margin={{ top: 20, right: 30, left: 20, bottom: 5 }}

                                >

                                    <XAxis

                                        dataKey="quarter"

                                        tickLine={false}

                                        axisLine={false}

                                        tickMargin={10}

                                        stroke="hsl(142, 76%, 36%)"

                                    />

                                    <YAxis

                                        tickLine={false}

                                        axisLine={false}

                                        stroke="hsl(142, 76%, 36%)"

                                        domain={[0, 100]}

                                        ticks={[0, 20, 40, 60, 80, 100]}

                                    />

                                    <ChartTooltip

                                        cursor={false}

                                        content={({ active, payload }) => {

                                            if (active && payload && payload.length) {

                                                return (

                                                    <div className="rounded-lg border bg-background p-2 shadow-sm">

                                                        <div className="grid grid-cols-1 gap-2">

                                                            <div className="flex flex-col">

                                                                <span className="text-[0.70rem] uppercase text-muted-foreground">

                                                                    {payload[0].payload.quarter}

                                                                </span>

                                                                <span className="font-bold text-muted-foreground">

                                                                    Actual: {payload[0].payload.actual}

                                                                </span>

                                                                {payload[0].payload.intervention && (

                                                                    <span className="font-bold text-muted-foreground">

                                                                        After Intervention: {payload[0].payload.intervention}

                                                                    </span>

                                                                )}

                                                                {payload[0].payload.intervention && (

                                                                    <span className="text-[0.70rem] text-muted-foreground">

                                                                        Improvement: {(payload[0].payload.intervention - payload[0].payload.actual).toFixed(1)}%

                                                                    </span>

                                                                )}

                                                            </div>

                                                        </div>

                                                    </div>

                                                );

                                            }

                                            return null;

                                        }}

                                    />

                                    <Legend

                                        verticalAlign="top"

                                        height={36}

                                    />

                                    <Bar

                                        dataKey="actual"

                                        fill="var(--color-actual)"

                                        radius={[4, 4, 0, 0]}

                                    />

                                    <Bar

                                        dataKey="intervention"

                                        fill="var(--color-intervention)"

                                        radius={[4, 4, 0, 0]}

                                    />

                                </BarChart>

                            </ChartContainer>

                        </div>

                    </div>

                </div>

            </CardContent>

        </Card>

    );

}