Review the process of how the default tenant admin user is created when the tenant first sig up.

Make sure the correct default roles and permission are set for the tenant admin user

Then after that, review in detail the DTOs, Schema, zod validation logic, and the server action for the process of creating a new user.

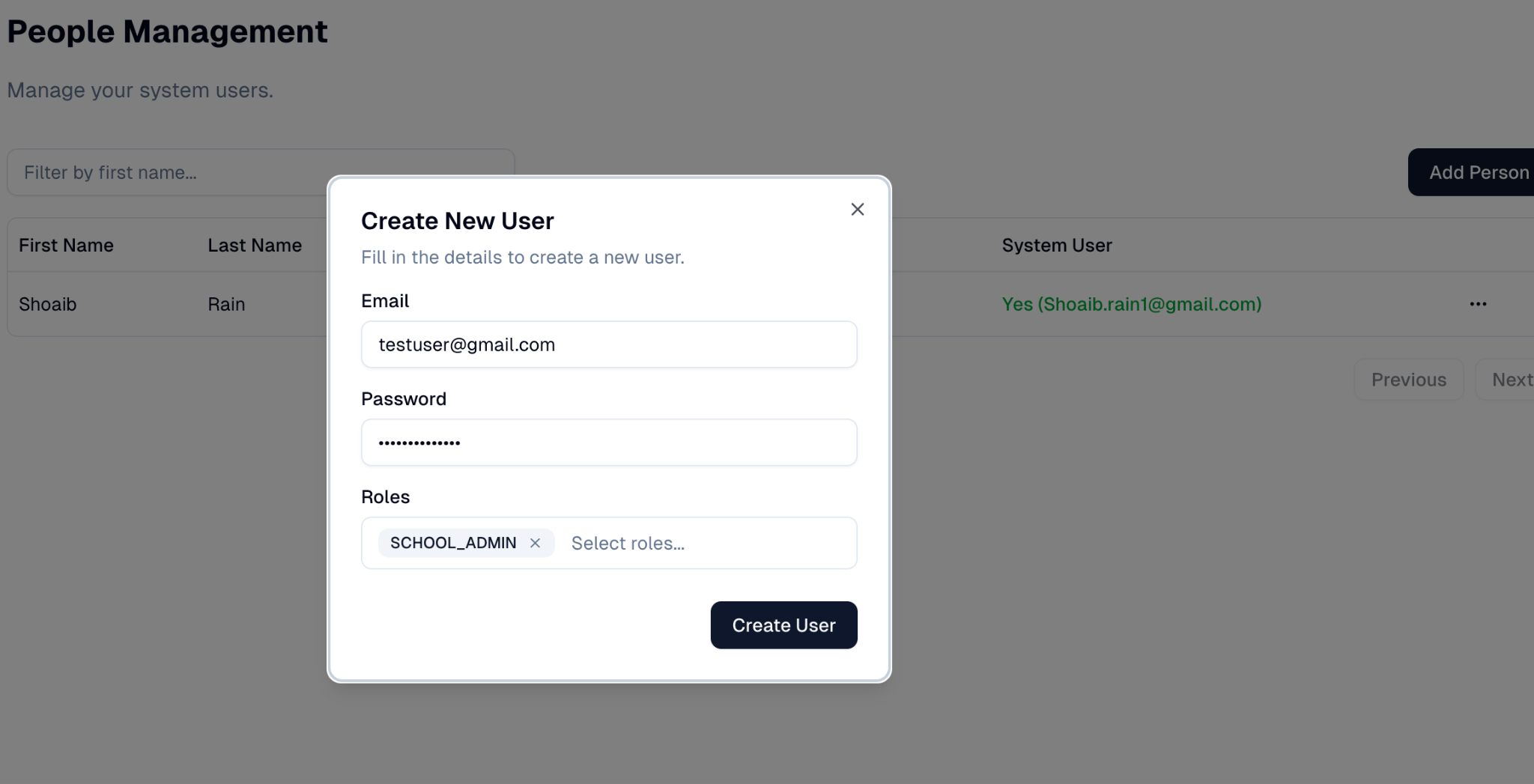
The idea is when the tenant sign up, they can invite/create a user to bring on their team, like school admin, teacher, etc…(Staff).

The database transaction layer code, server action layer code, schema definition and zod validation logic, form for user creation.

Currently, there are couple of issues with the current flow for user.   
When the tenant admin is created,

| First Name | Last Name | Type | School | Contact Email | System User |  |
| --- | --- | --- | --- | --- | --- | --- |
| Shoaib | Rain | staff | N/A | Shoaib.rain1@gmail.com | Yes (Shoaib.rain1@gmail.com) |  |

A Type of staff is given, which is incorrect. It should be more higher up level, like tenant\_admin.   
Currently,



In create new User form, after filling up the form, when “Create User” button is clicked, nothing happens. This form seems to be broken, and it does not have correct user feedback to notify if validation error. Nothing happens when I click the Button.

Analyze this form in great details without making any random assumption.  
"use client"

import React from 'react';

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

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

import {

Dialog,

DialogContent,

DialogHeader,

DialogTitle,

DialogFooter,

DialogDescription,

} from '@/components/ui/dialog';

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

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

import { toast } from 'sonner';

import { MultiSelect } from '@/components/ui/multi-select';

import { Form, FormControl, FormField, FormItem, FormLabel, FormMessage } from '@/components/ui/form';

import { UserFormSchema, UserFormInput } from '@/lib/dto/user';

import { upsertUserAction } from '@/lib/actions/user';

import type { roles, users } from '@/lib/db/schema/tenant';

import { useTenant } from '@/components/tenant-provider';

/\*\*

\* This form should be elegnat and enterprise looking Person Invite Form, where

\* users can be invited to join the platform by providing their email and selecting

\* their roles.

\*

\* This form should be able to create a user account just with email, like inviting new users.

\* The other relation with Person should be able to link the user to a person record later.

\*/

// Define the User type based on Drizzle's inference for the users table

export type User = (typeof users.$inferSelect) & {

roles: string; // This will be a comma-separated string of role names from getUsersWithRoles

};

// This should be PersonFormDialogProps

interface UserFormDialogProps {

isOpen: boolean;

setIsOpen: (open: boolean) => void;

user: User | null;

allRoles: (typeof roles.$inferSelect)[];

}

// Should be PersonFormDialog

export function UserFormDialog({ isOpen, setIsOpen, user, allRoles }: UserFormDialogProps) {

const isEditMode = !!user;

const { user: currentUserSession } = useTenant();

const canAssignRoles = currentUserSession.permissions.includes('user:assign\_roles');

const roleOptions = allRoles.map(role => ({

value: role.id.toString(),

label: role.name,

}));

const userRoleValues = user ? user.roles.split(', ').filter(Boolean).map((roleName: string) => { // Explicitly type roleName

const role = allRoles.find(r => r.name === roleName);

return role ? role.id.toString() : '';

}).filter(Boolean) : [];

// analyze and make sure we are using the correct default values and form logic for PersonFormDialog.

// based on the tenant schema

const form = useForm<UserFormInput>({

resolver: zodResolver(UserFormSchema),

defaultValues: {

id: user?.id,

email: user?.email || '',

password: '',

roles: userRoleValues,

},

});

const onSubmit = async (values: UserFormInput) => {

console.log("Submitting user form with values:", values);

const dataToSend = { ...values };

if (isEditMode && !dataToSend.password) {

delete dataToSend.password;

}

const result = await upsertUserAction(dataToSend);

if (result.success) {

toast.success(result.message);

setIsOpen(false);

} else {

toast.error(result.message);

if (result.errors?.email) {

form.setError('email', { type: 'server', message: result.errors.email[0] });

}

}

};

return (

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

<DialogContent className="sm:max-w-[425px]">

<DialogHeader>

<DialogTitle>{isEditMode ? 'Edit User' : 'Create New User'}</DialogTitle>

<DialogDescription>

{isEditMode ? "Update the user's details below." : "Fill in the details to create a new user."}

</DialogDescription>

</DialogHeader>

<Form {...form}>

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

<FormField

control={form.control}

name="email"

render={({ field }) => (

<FormItem>

<FormLabel>Email</FormLabel>

<FormControl><Input type="email" {...field} /></FormControl>

<FormMessage />

</FormItem>

)}

/>

<FormField

control={form.control}

name="password"

render={({ field }) => (

<FormItem>

<FormLabel>Password</FormLabel>

<FormControl><Input type="password" placeholder={isEditMode ? "Leave blank to keep current" : ""} {...field} /></FormControl>

<FormMessage />

</FormItem>

)}

/>

{canAssignRoles && (

<FormField

control={form.control}

name="roles"

render={({ field }) => (

<FormItem>

<FormLabel>Roles</FormLabel>

<FormControl>

<MultiSelect

options={roleOptions}

defaultValue={field.value}

onValueChange={field.onChange}

placeholder="Select roles..."

/>

</FormControl>

<FormMessage />

</FormItem>

)}

/>

)}

<DialogFooter>

<Button type="submit" disabled={form.formState.isSubmitting}>

{form.formState.isSubmitting ? 'Saving...' : isEditMode ? 'Save Changes' : 'Create User'}

</Button>

</DialogFooter>

</form>

</Form>

</DialogContent>

</Dialog>

);

}

I have provided you the all the codes for server actions, db schema, DTOs, components and all the necessary information.

This is modern, enterprise grade comprehensive school management application.

I want to implement a robust, secure and enterprise grade Person Onboarding process. This process involves creating a Person/User in the system eduX. The person could be Staffs (teacher,principle,counsler, etc…) and non staff (student, guardian).

The Form modal should be intuitive and easy to use with robust functionality, that provides drop down for staff and non-staff, and based on that it creates Person, creates User, etc.. and does the necessary validation and seeding the relationship. Etc…

For now lets only focus on the Staff person type, while building a solid scalable foundation that could allow to integrating student admission and enrollment process. The idea I have is that the process of creating A person should be easy, intuitive and then it redirects to that person settings page where further details can be added.

Analyze and read and understand all the source code attached, and then step by step guide me how to implement enterprise grade Person onboarding process.