Conform

```
pnpm install @conform-to/react @conform-to/zod
```

We use it to validate the input in the client side and the server side together, so we put it in the form and in server action.

In Server Action

```
'use server'

import { prisma } from '@/lib/prisma'
import ProjectSchema from '@/schema/ProjectSchema'
import { parseWithZod } from '@conform-to/zod'
import { redirect } from 'next/navigation'

export const AddProjectAction = async (
        prevState: unknown,
        formData: FormData
) => {
        const submission = parseWithZod(formData, {
                schema: ProjectSchema,
        })

        if (submission.status !== 'success') {
                return submission.reply()
        }

        try {
                await prisma.project.create({
                        data: {
                                title: submission.value.title,
                                city: submission.value.city,
                                country: submission.value.country,
                                description: submission.value.description,
                                image: submission.value.image,
                                style: submission.value.style,
                                client: {
                                        connect: {
                                                id: submission.value.clientId,
                                        },
                                },
                        },
                })
        } catch (error) {
                console.error(error)
        }
        redirect('/admin')
}
```

In Form Component

Before the return

```
const [lastResult, action] = useActionState(AddProjectAction, undefined)
const [form, fields] = useForm({
        lastResult,
        onValidate({ formData }) {
                return parseWithZod(formData, { schema: ProjectSchema })
        },
        shouldValidate: 'onBlur',
        shouldRevalidate: 'onInput',
})
```

In Form

```
<Form
        id={form.id}
        onSubmit={form.onSubmit}
        action={action}
        noValidate
        className='flex flex-col gap-4 max-w-4xl mx-auto py-12'
                >
```

In Input

There are many type for the Input fields, we will introduce the most important of them.

Input

```
<div className='flex flex-col gap-2'>
        <Label>Title</Label>
        <Input
                type='text'
                key={fields.title.key}
                name={fields.title.name}
                defaultValue={fields.title.initialValue}
        />
        <div className='text-destructive'>{fields.title.errors}</div>
</div>
```

Select with roll from prisma schema

```
// import RoleSchema from '@/prisma/generated/inputTypeSchemas/RoleSchema'

<Select
        key={fields.role.key}
        name={fields.role.name}
        defaultValue={fields.role.initialValue}
>
        <SelectTrigger className='w-full'>
                <SelectValue placeholder='Role' />
        </SelectTrigger>
        <SelectContent>
                <SelectItem value={RoleSchema.Enum.superAdmin}>
                        {RoleSchema.Enum.superAdmin}
                </SelectItem>
                <SelectItem value={RoleSchema.Enum.admin}>
                        {RoleSchema.Enum.admin}
                </SelectItem>
        </SelectContent>
</Select>
```

Select with roll from Database

- first we need to send the data base information via props from the form page to form component

```
const articleDb = await prisma.article.findMany({
  select: {
    id: true,
    title: true,
  },
  orderBy: { createdAt: 'desc' },
})
// ............................
// ............................
<AddSubtitleForm articleDb={articleDb} />
```

- Then we will receive the props in form component

```
type articleDbType = {
  articleDb: {
    id: string
    title: string
  }[]
}

export default function AddSubtitleForm(articleDb: articleDbType) {
  // .................................................
  // .................................................
}
```

- Then we use a Select component

```
<div className='flex flex-col gap-2'>
        <Label>Article</Label>
        <Select
                key={fields.articleId.key}
                name={fields.articleId.name}
                defaultValue={fields.articleId.initialValue}
        >
                <SelectTrigger className='w-full'>
                        <SelectValue placeholder='Article' />
                </SelectTrigger>
                <SelectContent>
                        {articleDb.articleDb.map(({ id, title }) => (
                                <SelectItem value={id} key={id}>
                                        {title}
                                </SelectItem>
                        ))}
                </SelectContent>
        </Select>
        <div className='text-destructive'>{fields.articleId.errors}</div>
</div>
```

Image Input with Uploadthing

```
<Label>Images</Label>
        <Card className=''>
                <CardContent className='flex flex-col gap-2 '>
                        <Input
                                type='hidden'
                                value={mainImage}
                                key={fields.image.key}
                                name={fields.image.name}
                                defaultValue={fields.image.initialValue as any}
                        />
                        {mainImage.length > 0 ? (
                                <div className='flex gap-5'>
                                        <div className='relative size-32'>
                                                <Image
                                                        src={mainImage}
                                                        alt={'product image'}
                                                        height={96}
                                                        width={96}
                                                        className='w-full h-full rounded-lg bor
                                                />
                                                <Button
                                                        type='button'
                                                        size={'icon'}
                                                        className='absolute -top-3 -right-3 siz
                                                        variant={'destructive'}
                                                        onClick={() => handleDelete()}
                                                >
                                                        <XIcon />
                                                </Button>
                                        </div>
                                </div>
                        ) : (
                                <UploadDropzone
                                        endpoint='imageUploader'
                                        onClientUploadComplete={(res) => {
                                                setMainImage((res[0] as { ufsUrl: string }).ufs
                                        }}
                                        onUploadError={(error: Error) => alert(`ERROR! ${error.
                                        className='ut-button:bg-primary ut-button:text-backgrou
                                />
                        )}
                        <div className='text-destructive'>{fields.image.errors}</div>
                </CardContent>
        </Card>
```

Submit Button

```
'use client'

import { Button } from '@/components/ui/button'
import { LoaderCircle } from 'lucide-react'
import { useFormStatus } from 'react-dom'

type SubmitButtonProps = {
        title: string,
}

export default function SubmitButton({ title }: SubmitButtonProps) {
        const status = useFormStatus()

        return (
                <Button type='submit' size={'wide'} className='mt-6'>
                        {status.pending ? <LoaderCircle className='animate-spin' /> : title}
                </Button>
        )
}
```