NextGen Tech

* Muhammad Abdullah 22-NTU-CS-1358
* Maryam Sameen 22-NTU-CS-1354
* Muhammad Kaif 22-NTU-CS-1364



Department of Computer Science, Faisalabad

BS-AI 6th

Submitted To: Mr. Zahid Javed

# 1. Problem Statement

NextGen Tech is a lightweight admin dashboard designed for a tech startup. It enables administrators to efficiently manage content in two core areas: blog posts and contact entries. Built using Next.js, Tailwind CSS, and TypeScript, the application provides a responsive and modern interface for performing CRUD (Create, Read, Update, Delete) operations on both data types.

# 2. Screenshots

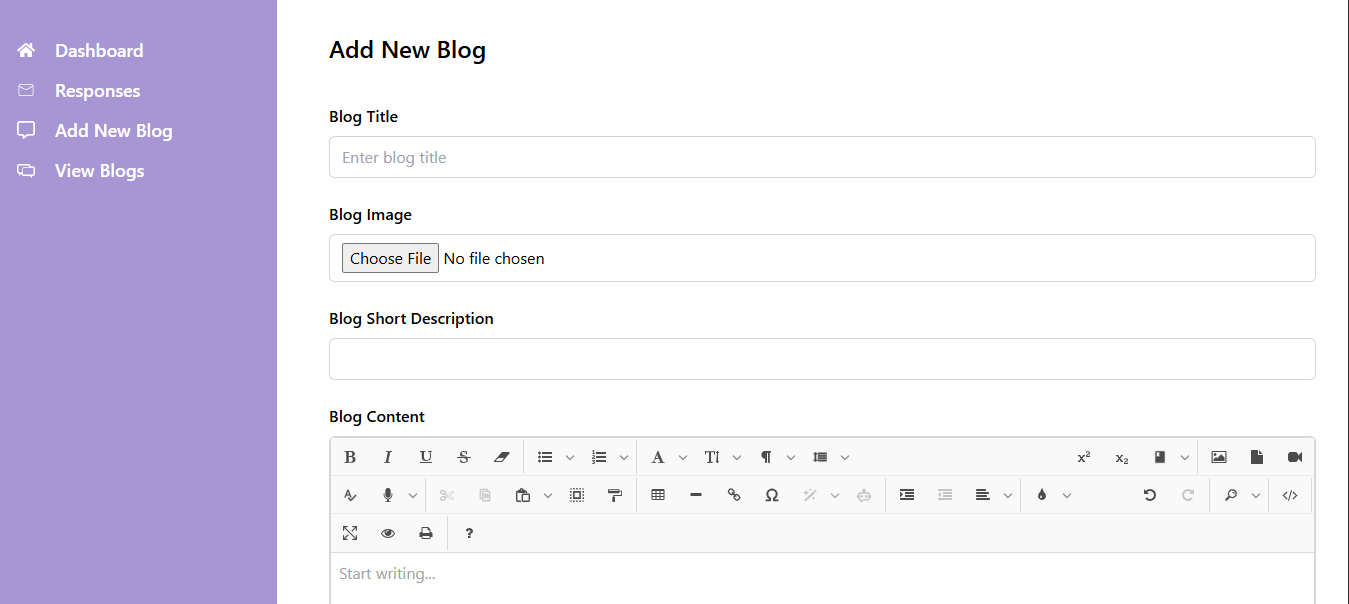


*Figure 1:Home Page*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 2:Admin Panel*



*Figure 3:Add New Blog*

A screenshot of a cartoon

AI-generated content may be incorrect.

*Figure 4:Contact Form*

# 3. Code Sections

## app/page.tsx

// import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';  
"use client"  
import HeroCarousel from "./components/HeroCarousel";  
import StatusCards from "./components/StatusCards";  
import LoginComp from "./components/LoginComp";  
import Footer from "./components/Footer";  
import Header from "./components/Header";  
import Services from "./components/Services";  
import Blogs from "./components/Blogs";  
import AboutUs from "./components/AboutUs";  
import ContactUs from "./components/ContactUs"  
import { useUser } from "@clerk/nextjs";  
  
export default function App() {  
 const {user} = useUser();  
 console.log("user", user);  
   
 return (  
 <div className="home-page-wrapper">  
 <Header user={user}/>  
 <HeroCarousel />  
 <StatusCards />  
 <LoginComp />  
 <Blogs />  
 <Services />  
 <AboutUs />  
 <ContactUs />  
 <Footer />  
 </div>  
 );  
}

## app/layout.tsx

import type { Metadata } from "next";  
import localFont from "next/font/local";  
import "./globals.css";  
import { ClerkProvider } from "@clerk/nextjs";  
import { ToastContainer } from "react-toastify";  
  
  
const geistSans = localFont({  
 src: "./fonts/GeistVF.woff",  
 variable: "--font-geist-sans",  
 weight: "100 900",  
});  
const geistMono = localFont({  
 src: "./fonts/GeistMonoVF.woff",  
 variable: "--font-geist-mono",  
 weight: "100 900",  
});  
  
export const metadata: Metadata = {  
 title: "Tech Startup",  
 description: " Tech Startup",  
};  
  
export default function RootLayout({  
 children,  
}: Readonly<{  
 children: React.ReactNode;  
}>) {  
 return (  
 <ClerkProvider>  
   
 <html lang="en">  
   
 <body className={`${geistSans.variable} ${geistMono.variable}`}>  
 <ToastContainer />  
 {children}  
 </body>  
 </html>  
 </ClerkProvider>  
 );  
}

## app/admin-panel/page.tsx

"use client";  
import { useRouter } from "next/navigation";  
import { useEffect, useState } from "react";  
  
export default function Page() {  
 const router = useRouter();  
 const [data, setData] = useState<any>(false);  
 useEffect(() => {  
 const fetchData = async () => {  
 const data = await fetch("/api/dashboard-data", {  
 cache: "no-store",  
 });  
 const res = await data.json();  
 setData(res);  
 };  
 fetchData();  
 }, []);  
 const cards = [  
 {  
 title: "Responses",  
 count: data?.messagesCount || 0, // Replace with dynamic value later  
 route: "/admin-panel/messages",  
 },  
 {  
 title: "Blogs",  
 count: data?.blogsCount || 0, // Replace with dynamic value later  
 route: "/admin-panel/blogs",  
 },  
 ];  
 return (  
 <div className="p-8 w-full max-w-full">  
 <h1 className="text-[28px] mb-6 text-[#333]">Dashboard</h1>  
 <div className="flex flex-wrap gap-6">  
 {cards.map((card, index) => (  
 <div

## app/admin-panel/add-blog/page.tsx

import Spinner from "@/app/components/Spinner";  
import dynamic from "next/dynamic";  
import { Suspense } from "react";  
  
const AddBlogForm = dynamic(  
 () => import("../../components/Adminpanel/AddBlogForm"),  
 {  
 ssr: false,  
 }  
);  
  
export default function Page() {  
 return (  
 <Suspense fallback={<Spinner />}>  
 <AddBlogForm />  
 </Suspense>  
 );  
}

## database.ts

 import mongoose from "mongoose";

const connectDatabase = async () => {

    const uri = process.env.MONGODB\_URI

    try {

        if (!uri) {

            throw new Error('MONGODB\_URI environment variable is not defined');

        }

        await mongoose.connect(uri);

        console.log("Databse Connected Successfully.");

    } catch (error) {

        console.log(error);

    }

};

export default connectDatabase;

### App/Components/Adminpanel/Addblogform.tsx

"use client";

import { useFormik } from "formik";

import { useRouter, useSearchParams } from "next/navigation";

import { Suspense, useEffect, useRef, useState } from "react";

import { toast } from "react-toastify";

import \* as Yup from "yup";

import dynamic from "next/dynamic";

import Spinner from "../Spinner";

const JoditEditor = dynamic(() => import("jodit-react"), {

ssr: false,

});

export default function Page() {

const editor = useRef(null);

const id = useSearchParams().get("id");

const [blog, setBlog] = useState<any>(null);

const router = useRouter();

const [loading, setLoading] = useState(false);

const validationSchema = Yup.object({

title: Yup.string().required("Title is required"),

shortDescription: Yup.string()

.required("Short Description is required")

.max(60, "Description can be max 60 characters"),

content: Yup.string().required("Content is required"),

});

useEffect(() => {

if (id) {

const fetchBlog = async () => {

const res = await fetch(`/api/blogs/${id}`);

const data = await res.json();

setBlog(data);

};

fetchBlog();

}

}, [id]);

const handleSubmit = async () => {

const { title, image, content, shortDescription } = formik.values;

const formData = new FormData();

formData.append("title", title);

if (image instanceof File) {

formData.append("image", image);

} else if (id && blog?.image) {

formData.append("existingImage", blog.image);

}

formData.append("description", content);

formData.append("shortDescription", shortDescription);

try {

setLoading(true);

const endpoint = id ? `/api/blogs/${id}` : "/api/blogs";

const response = await fetch(endpoint, {

method: id ? "PATCH" : "POST",

body: formData,

});

if (response?.ok) {

toast.success(`Blog ${id ? "updated" : "added"} successfully`);

formik.resetForm();

setLoading(false);

router.push("/admin-panel/blogs");

}

} catch (error) {

toast.error("Error uploading blog");

console.log("error", error);

}

};

const formik:any = useFormik({

initialValues: {

title: blog?.title || "",

image: null,

content: blog?.description || "",

shortDescription: blog?.shortDescription || "",

},

validationSchema,

onSubmit: handleSubmit,

enableReinitialize: true,

});

return (

<div className="p-8 w-full">

<div className="w-full">

<Suspense fallback={<Spinner />}>

<form onSubmit={formik.handleSubmit} className="flex flex-col gap-6">

<h2 className="text-2xl font-semibold mb-4">Add New Blog</h2>

<div className="flex flex-col">

<label htmlFor="title" className="font-medium mb-2">

Blog Title

</label>

<input

type="text"

id="title"

value={formik.values.title}

onChange={formik.handleChange}

name="title"

onBlur={formik.handleBlur}

placeholder="Enter blog title"

className="px-3 py-2 border border-gray-300 rounded-md text-black bg-white"

/>

{formik.touched.title && formik.errors.title && (

<p className="text-red-500 text-sm mt-1">

{formik.errors.title}

</p>

)}

</div>

<div className="flex flex-col">

<label htmlFor="image" className="font-medium mb-2">

Blog Image

</label>

<input

type="file"

id="image"

accept="image/\*"

name="image"

onChange={(e: any) =>

formik.setFieldValue("image", e.target.files[0])

}

className="px-3 py-2 border border-gray-300 rounded-md bg-white"

/>

{formik.touched.image && formik.errors.image && (

<p className="text-red-500 text-sm mt-1">

{formik.errors.image}

</p>

)}

</div>

<div className="flex flex-col">

<label htmlFor="shortDescription" className="font-medium mb-2">

Blog Short Description

</label>

<input

type="text"

id="shortDescription"

name="shortDescription"

onChange={formik.handleChange}

onBlur={formik.handleBlur}

value={formik.values.shortDescription}

className="px-3 py-2 border border-gray-300 rounded-md text-black bg-white"

/>

{formik.touched.shortDescription &&

formik.errors.shortDescription && (

<p className="text-red-500 text-sm mt-1">

{formik.errors.shortDescription}

</p>

)}

</div>

<div className="flex flex-col">

<label className="font-medium mb-2">Blog Content</label>

<div className="border border-gray-300 rounded-md overflow-hidden">

<JoditEditor

ref={editor}

value={formik.values.content}

onChange={(newContent) =>

formik.setFieldValue("content", newContent)

}

/>

</div>

{formik.touched.content && formik.errors.content && (

<p className="text-red-500 text-sm mt-1">

{formik.errors.content}

</p>

)}

</div>

<button

type="submit"

className="px-6 py-3 bg-[#4e2fa9] text-white rounded-lg font-medium max-w-xs hover:bg-[#3c2390] transition duration-300"

>

{loading ? <Spinner /> : "Save Blog"}

</button>

</form>

</Suspense>

</div>

</div>

);

}