app\layout.tsx

```
import type { Metadata } from 'next';

import { Inter, Poppins, Noto_Sans_KR } from 'next/font/google';

import './globals.css';

import Status from './_components/Status';

import Header from './_components/Header';

import Footer from './_components/Footer';



const inter = Inter({ subsets: ['latin'] });

const poppins = Poppins({

  subsets: ['latin'],

  weight: ['400', '600', '800'],

});

const notoSansKR = Noto_Sans_KR({

  subsets: ['latin'],

  weight: ['400', '600', '800'],

});



export const cls = (...classnames: string[]) => {

  return classnames.join(' ');

};



export const metadata: Metadata = {

  title: 'Create Next App',

  description: 'Generated by create next app',

};
```

```
export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      <body className={notoSansKR.className}>
        <div className="">
          <Header />
          <main className="pt-16 pb-16 min-h-screen tracking-tight text-gray-800 leading-tight">
            {children}
          </main>
          {/* footer */}
          <Footer />
        </div>
      </body>
    </html>
  );
}
```

app\loading.tsx

```
import './loading.css';



export default function Loading() {

  return (

    <div className="flex flex-col items-center justify-center h-height-minus-146 gap-6">

      <h2 className="text-2xl text-gray-100 animate-pulse">loading...</h2>

      {/* <div className="loadingContainer flex">

        <div className="loadingBar"></div>

        <div className="loadingBar"></div>

        <div className="loadingBar"></div>

        <div className="loadingBar"></div>

        <div className="loadingBar"></div>

      </div> */}

    </div>

  );

}
```

app\page.tsx

```
// app/page.tsx

import { getcurrentUserFromCookies } from '@/lib/cookies';

import createSupabaseServerClient from '@/lib/supabse/server';



import dayjs from 'dayjs';

import timezone from 'dayjs/plugin/timezone';

import utc from 'dayjs/plugin/utc';

import { DolphinBasic } from './_items/dolphin_item';

import { findCategoryNameById } from './post/_action/category';

import TopPosts from './_components/Topposts';

import AdFixedPage from './_components/Ad_Bottom';

import Link from 'next/link';



dayjs.extend(utc);

dayjs.extend(timezone);



export default async function Home() {

  const currentUser = getcurrentUserFromCookies();



  const supabase = await createSupabaseServerClient();



  // 7? ? ?? ??

  const now = dayjs().tz('Asia/Seoul');

  const sevenDayAgo = now.subtract(7, 'day').tz('Asia/Seoul').format(); // ??? ??

  console.log('oneDat', sevenDayAgo);
```

```
const result = await supabase
  .from('post')
  .select(
    `
    id, title, created_at, views, comments, author_id, author_name, author_email, author_avatar_url,
    parent_category_id, child_category_id
  `
  )
  .gt('created_at', sevenDayAgo) // ?? 1? ??? ???
  .order('views', { ascending: false }) // ??? ???? ??
  .limit(5); // ?? 5? ???? ???

const postsData = result.data || [];

const postsWithCategoryNames = await Promise.all(
  postsData.map(async (post) => {
    const parentCategoryName = await findCategoryNameById(post.parent_category_id);
    const childCategoryName = await findCategoryNameById(post.child_category_id);

    return {
      ...post,
      parent_category_name: parentCategoryName,
      child_category_name: childCategoryName,
    };
  })
);
```

```
  return (
    <div className="flex flex-col items-center ">
      <h2 className="text-xl font-bold mt-4">??? ????</h2>
      <Link href="/goodluck">
        <div className="mx-auto w-80 h-24 flex flex-col relative border-[1px] border-red-200 p-1 mt-4  rounded-xl">
          <img src="/lotteryAd.png" alt="" />
        </div>
      </Link>
      <h2 className="text-xl font-bold mt-4">??? ?? ???</h2>
      <TopPosts posts={postsWithCategoryNames} userId={currentUser?.id ?? null} />
      <div className="blank_gap w-full h-8" />
      <AdFixedPage />
      <h2 className="text-xl font-bold mt-8 ">All Posts</h2>
      <DolphinBasic width={100} color="text-red-300" />
      <DolphinBasic width={100} color="text-blue-300" />
      <DolphinBasic width={100} color="text-violet-300" />
    </div>
  );
}
```

app\ad\ad-content.tsx

```
'use client';


import { useEffect } from 'react';

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


export default function AdContnetPage() {

  const router = useRouter();

  const searchParams = useSearchParams();

  const target = searchParams.get('target');


  useEffect(() => {

    if (target) {

      console.log('Navigating to target in 1.5 seconds:', target);


      const timer = setTimeout(() => {

        console.log('Navigating now:', new Date().toISOString());

        const startTime = performance.now();


        router.replace(target);


        const endTime = performance.now();

        console.log(`Navigation to ${target} completed in ${endTime - startTime} ms`);

      }, 1500);


      return () => {

        console.log('Clearing timeout');
        clearTimeout(timer);

      };

    }

  }, [target, router]);



  return null;

}
```

app\ad\page.tsx

```
import { Suspense } from 'react';

import AdContnetPage from './ad-content';



export default function AdPage() {

  return (

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

      <div className="animate-slide-in-up">

        <p className="py-4 text-gray-800 font-semibold">?? ??? New</p>

        <img src="mainad-1.webp" alt="" className="w-screen-minus-32 aspect-square object-cover " />

        <Suspense fallback={<div>Loading...</div>}>

          <AdContnetPage />

        </Suspense>

      </div>

    </div>

  );

}



/* 'use client';



import { useEffect } from 'react';

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



export default function AdPage() {

  const router = useRouter();

  const searchParams = useSearchParams();

  const target = searchParams.get('target');
```

```
useEffect(() => {

  if (target) {

    console.log('Navigating to target in 1.5 seconds:', target);



    const timer = setTimeout(() => {

      console.log('Navigating now:', new Date().toISOString());

      const startTime = performance.now();



      router.replace(target);



      const endTime = performance.now();

      console.log(`Navigation to ${target} completed in ${endTime - startTime} ms`);

    }, 1500);



    return () => {

      console.log('Clearing timeout');

      clearTimeout(timer);

    };

  }

}, [target, router]);



return (

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

    <p className="py-4 text-gray-800 font-semibold">?? ???</p>

    <img src="mainad-1.png" alt="" className="w-screen-minus-32 aspect-square object-cover" />

  </div>
  );

}

 */
```

app\api\get-ip\route.ts

```
import { NextRequest, NextResponse } from 'next/server';



export async function GET(request: NextRequest) {

  const forwarded = request.headers.get('x-forwarded-for');

  let ip = forwarded ? forwarded.split(/, /)[0] : request.ip || 'Unknown IP';



  // IPv4 ??? IPv6 ???? ??? ?? ??

  if (ip.startsWith('::ffff:')) {

    ip = ip.substring(7);

  }



  console.log('ip', ip);

  return NextResponse.json({ ip });

}
```

app\api\user\route.ts

```
import { NextRequest, NextResponse } from 'next/server';

import { cookies } from 'next/headers';



export async function GET() {

  const cookieStore = cookies();

  const currentUserCookie = cookieStore.get('currentUser');



  if (currentUserCookie) {

    const currentUser = JSON.parse(currentUserCookie.value);

    return NextResponse.json(currentUser);

  }



  return NextResponse.json(null);

}
```

app\auth\page.tsx

```
import React from 'react';

import { AuthForm } from './components/AuthForm';

import { redirect } from 'next/navigation';

import SignOut from './components/SignOut';

import { CurrentUser, getCurrentUserInfo } from '@/lib/cookies';

import { revalidatePath } from 'next/cache';



export default async function page() {

  const currentUser: CurrentUser | null = getCurrentUserInfo();

  /* if (currentUser) {

    revalidatePath('/post');

    redirect('/post');

  } */

  return (

    <div className="flex justify-center mt-16 ">

      <div className="w-full mx-6 flex flex-col items-center gap-16">

        <img src="/logo-benefiboard.svg" alt="" />

        <SignOut />

        <AuthForm />

      </div>

    </div>

  );

}
```

app\auth\actions\index.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';

import { cookies } from 'next/headers';


export async function signUpWithEmailAndPassword(data: {

  email: string;

  password: string;

  confirm: string;

}) {

  const supabase = await createSupabaseServerClient();


  const { data: result, error } = await supabase.auth.signUp({

    email: data.email,

    password: data.password,

  });


  if (error) {

    return { success: false, message: error.message };

  }


  return { success: true, message: '????? ?????? ???? ??????' };

}


export async function signInWithEmailAndPassword(data: { email: string; password: string }) {

  const supabase = await createSupabaseServerClient();
  const { data: result, error } = await supabase.auth.signInWithPassword({
    email: data.email,
    password: data.password,
  });

  if (error) {
    return { success: false, message: error.message };
  }

  if (result.session) {
    cookies().set('access_token', result.session.access_token, {
      path: '/',
      httpOnly: true,
      secure: process.env.NODE_ENV === 'production',
      maxAge: result.session.expires_in,
    });

    const userResponse = await supabase
      .from('userdata')
      .select('id, username, avatar_url, email')
      .eq('id', result.user?.id)
      .single();

    if (userResponse.error) {
      return { success: false, message: userResponse.error.message };
    }
```

```
    cookies().set('currentUser', JSON.stringify(userResponse.data), {
      httpOnly: true,
      secure: process.env.NODE_ENV === 'production',
      maxAge: 60 * 60 * 24 * 7,
      path: '/',
      sameSite: 'lax',
    });

    return { success: true, message: '??? ??' };
  }

  return { success: false, message: '?? ??? ??????.' };
}
```

app\auth\auth-code-error\page.tsx

```
const AuthCodeErrorPage = () => {

  return (

    <div>

      <h1>Authentication Error</h1>

      <p>There was an error during the authentication process. Please try again.</p>

    </div>

  );

};


export default AuthCodeErrorPage;
```

app\auth\callback\route.ts

```
import { cookies } from 'next/headers';

import { NextResponse } from 'next/server';

import { type CookieOptions, createServerClient } from '@supabase/ssr';



export async function GET(request: Request) {

  const { searchParams, origin } = new URL(request.url);

  const code = searchParams.get('code');

  // if "next" is in param, use it as the redirect URL

  const next = searchParams.get('next') ?? '/';



  if (code) {

    const cookieStore = cookies();

    const supabase = createServerClient(

      process.env.NEXT_PUBLIC_SUPABASE_URL!,

      process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,

      {

        cookies: {

          get(name: string) {

            return cookieStore.get(name)?.value;

          },

          set(name: string, value: string, options: CookieOptions) {

            cookieStore.set({ name, value, ...options });

          },

          remove(name: string, options: CookieOptions) {

            cookieStore.delete({ name, ...options });

          },
        },
      }
    );
    const { error } = await supabase.auth.exchangeCodeForSession(code);
    if (!error) {
      return NextResponse.redirect(`${origin}${next}`);
    }
  }

  // return the user to an error page with instructions
  return NextResponse.redirect(`${origin}/auth/auth-code-error`);
}
```

app\auth\components\AuthForm.tsx

```
'use client';



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

import SignInForm from './SignInForm';

import RegisterForm from './RegisterForm';

import GithubButton from './OAuthForm_Github';

import GoogleButton from './OAuthForm_Google';



export function AuthForm() {

  return (

    <div className="w-full space-y-5">

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

        <TabsList className="grid w-full grid-cols-2">

          <TabsTrigger value="signin">SignIn</TabsTrigger>

          <TabsTrigger value="register">Register</TabsTrigger>

        </TabsList>

        <TabsContent value="signin">

          <SignInForm />

        </TabsContent>

        <TabsContent value="register">

          <RegisterForm />

        </TabsContent>

      </Tabs>

      <GithubButton />

      <GoogleButton />

    </div>
  );

}
```

app\auth\components\OAuthForm_Github.tsx

```
'use client';


import { createBrowserClient } from '@supabase/ssr';

import React from 'react';

import { redirect } from 'next/navigation';

import { FaGithub } from 'react-icons/fa';

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


export default function GithubButton() {

  const supabase = createBrowserClient(

    process.env.NEXT_PUBLIC_SUPABASE_URL!,

    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!

  );


  const loginWithGithub = async () => {

    await supabase.auth.signInWithOAuth({

      provider: 'github',

      options: {

        redirectTo: `${location.origin}/auth/callback`,

      },

    });

  };


  return (

    <Button onClick={loginWithGithub} className="w-full flex gap-2">

      <FaGithub size="1.6rem" fill="#eee" />
      GitHub with SignIn

    </Button>

  );

}
```

app\auth\components\OAuthForm_Google.tsx

```
'use client';


import { createBrowserClient } from '@supabase/ssr';

import React from 'react';

import { redirect } from 'next/navigation';

import { FaGithub, FaGoogle } from 'react-icons/fa';

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


export default function GoogleButton() {

  const supabase = createBrowserClient(

    process.env.NEXT_PUBLIC_SUPABASE_URL!,

    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!

  );


  const loginWithGoogle = async () => {

    await supabase.auth.signInWithOAuth({

      provider: 'google',

      options: {

        redirectTo: `${location.origin}/auth/callback`,

      },

    });

  };


  return (

    <Button onClick={loginWithGoogle} className="w-full flex gap-2">

      <FaGoogle size="1.6rem" fill="#eee" />
      Google with SignIn

    </Button>

  );

}
```

app\auth\components\RegisterForm.tsx

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

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

import * as z from 'zod';

import { AiOutlineLoading3Quarters } from 'react-icons/ai';



import {

  Form,

  FormControl,

  FormField,

  FormItem,

  FormLabel,

  FormMessage,

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

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



import { toast } from '@/components/ui/use-toast';

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

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

import { signUpWithEmailAndPassword } from '../actions';



const FormSchema = z

  .object({

    email: z.string().email({

      message: '??? ??? ???? ????.',

    }),

    password: z.string().min(6, {
      message: '????? 6?? ????? ???.',
    }),
    confirm: z.string().min(6, {
      message: '????? 6?? ????? ???.',
    }),
  })
  .refine((data) => data.confirm === data.password, {
    message: '????? ???? ????.',
    path: ['confirm'],
  });

export default function RegisterForm() {
  const form = useForm<z.infer<typeof FormSchema>>({
    resolver: zodResolver(FormSchema),
    defaultValues: {
      email: '',
      password: '',
      confirm: '',
    },
  });

  async function onSubmit(data: z.infer<typeof FormSchema>) {
    const { result, error } = await signUpWithEmailAndPassword(data);
    console.log('signUp', result, error);

    if (error) {
      toast({
```

```
      variant: 'destructive',
      title: '???? ??',
      description: (
        <pre className="mt-2 w-[340px] rounded-md bg-slate-950 p-4">
          <code className="text-white">{error.message}</code>
        </pre>
      ),
    });
    alert(error.message);
  } else {
    toast({
      title: '???? ??',
      description: (
        <pre className="mt-2 w-[340px] rounded-md bg-slate-950 p-4">
          <code className="text-white">????? ?????? ???? ??????</code>
        </pre>
      ),
    });
  }
}

return (
  <Form {...form}>
    <form onSubmit={form.handleSubmit(onSubmit)} className="w-full space-y-6">
      <FormField
        control={form.control}
        name="email"
  render={({ field }) => (
    <FormItem>
      <FormLabel>Email</FormLabel>
      <FormControl>
        <Input
          placeholder="example@gmail.com"
          {...field}
          type="email"
          onChange={field.onChange}
        />
      </FormControl>
      <FormMessage />
    </FormItem>
  )}
/>
<FormField
  control={form.control}
  name="password"
  render={({ field }) => (
    <FormItem>
      <FormLabel>Password</FormLabel>
      <FormControl>
        <Input
          placeholder="password"
          {...field}
          type="password"
          onChange={field.onChange}
        />
      </FormControl>
      <FormMessage />
    </FormItem>
  )}
/>
<FormField
  control={form.control}
  name="confirm"
  render={({ field }) => (
    <FormItem>
      <FormLabel>Confirm Password</FormLabel>
      <FormControl>
        <Input
          placeholder="Confirm Password"
          {...field}
          type="password"
          onChange={field.onChange}
        />
      </FormControl>
      <FormMessage />
    </FormItem>
  )}
/>
<Button type="submit" className="w-full flex gap-2">
  Register
  <AiOutlineLoading3Quarters className={cn('animate-spin')} />
```

```
        </Button>
      </form>
    </Form>
  );
}
 */

'use client';

import { zodResolver } from '@hookform/resolvers/zod';
import { useForm } from 'react-hook-form';
import * as z from 'zod';
import { AiOutlineLoading3Quarters } from 'react-icons/ai';
import { useState } from '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 { cn } from '@/lib/utils';
import { signUpWithEmailAndPassword } from '../actions';
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogFooter,
  DialogTitle,
  DialogDescription,
} from '@/components/ui/dialog';


const FormSchema = z
  .object({
    email: z.string().email({
      message: '??? ??? ???? ????.',
    }),
    password: z.string().min(6, {
      message: '????? 6?? ????? ???.',
    }),
    confirm: z.string().min(6, {
      message: '????? 6?? ????? ???.',
    }),
  })
  .refine((data) => data.confirm === data.password, {
    message: '????? ???? ????.',
    path: ['confirm'],
  });
```

```
export default function RegisterForm() {

  const form = useForm<z.infer<typeof FormSchema>>({

    resolver: zodResolver(FormSchema),

    defaultValues: {

      email: '',

      password: '',

      confirm: '',

    },

  });



  const [dialogOpen, setDialogOpen] = useState(false);

  const [dialogMessage, setDialogMessage] = useState('');

  const [dialogTitle, setDialogTitle] = useState('');



  async function onSubmit(data: z.infer<typeof FormSchema>) {

    const { success, message } = await signUpWithEmailAndPassword(data);

    console.log('signUp', success, message);



    setDialogTitle(success ? '???? ??' : '???? ??');

    setDialogMessage(message);

    setDialogOpen(true);

  }



  return (

    <>

      <Form {...form}>

        <form onSubmit={form.handleSubmit(onSubmit)} className="w-full space-y-6">
```

```
<FormField
  control={form.control}
  name="email"
  render={({ field }) => (
    <FormItem>
      <FormLabel>Email</FormLabel>
      <FormControl>
        <Input
          placeholder="example@gmail.com"
          {...field}
          type="email"
          onChange={field.onChange}
        />
      </FormControl>
      <FormMessage />
    </FormItem>
  )}
/>
<FormField
  control={form.control}
  name="password"
  render={({ field }) => (
    <FormItem>
      <FormLabel>Password</FormLabel>
      <FormControl>
        <Input
          placeholder="password"
          {...field}
          type="password"
          onChange={field.onChange}
        />
      </FormControl>
      <FormMessage />
    </FormItem>
  )}
/>
<FormField
  control={form.control}
  name="confirm"
  render={({ field }) => (
    <FormItem>
      <FormLabel>Confirm Password</FormLabel>
      <FormControl>
        <Input
          placeholder="Confirm Password"
          {...field}
          type="password"
          onChange={field.onChange}
        />
      </FormControl>
      <FormMessage />
    </FormItem>
  )}
/>
```

```
          <Button type="submit" className="w-full flex gap-2">
            Register
            <AiOutlineLoading3Quarters className={cn('animate-spin')} />
          </Button>
        </form>
      </Form>

      <Dialog open={dialogOpen} onOpenChange={setDialogOpen}>
        <DialogContent>
          <DialogHeader>
            <DialogTitle>{dialogTitle}</DialogTitle>
            <DialogDescription>
              <pre className="mt-2 w-[340px] rounded-md bg-slate-950 p-4">
                <code className="text-white">{dialogMessage}</code>
              </pre>
            </DialogDescription>
          </DialogHeader>
          <DialogFooter>
            <Button onClick={() => setDialogOpen(false)}>Close</Button>
          </DialogFooter>
        </DialogContent>
      </Dialog>
    </>
  );
}
```

app\auth\components\SignInForm.tsx

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

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

import * as z from 'zod';

import { AiOutlineLoading3Quarters } from 'react-icons/ai';



import {

  Form,

  FormControl,

  FormField,

  FormItem,

  FormLabel,

  FormMessage,

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

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

import { toast } from '@/components/ui/use-toast';

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

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

import { signInWithEmailAndPassword } from '../actions';

import { useTransition } from 'react';

import { useRouter } from 'next/navigation';



const FormSchema = z.object({

  email: z.string().email(),

  password: z.string().min(1, {

    message: 'Password is required.',

  }),
});


export default function SignInForm() {
  const [isPending, startTransition] = useTransition();
  const router = useRouter();


  const form = useForm<z.infer<typeof FormSchema>>({
    resolver: zodResolver(FormSchema),
    defaultValues: {
      email: '',
      password: '',
    },
  });


  async function onSubmit(data: z.infer<typeof FormSchema>) {
    startTransition(async () => {
      const result = await signInWithEmailAndPassword(data);


      if (!result.success) {
        toast({
          variant: 'destructive',
          title: 'Error',
          description: (
            <pre className="mt-2 w-[340px] rounded-md bg-slate-950 p-4">
              <code className="text-white">{result.message}</code>
            </pre>
          ),
        });
      } else {
        toast({
          title: 'Success',
          description: (
            <pre className="mt-2 w-[340px] rounded-md bg-slate-950 p-4">
              <code className="text-white">??? ??</code>
            </pre>
          ),
        });
        router.refresh(); // ???? ?????? ?? ??? ??
      }
    });
  }

  return (
    <Form {...form}>
      <form onSubmit={form.handleSubmit(onSubmit)} className="w-full space-y-6">
        <FormField
          control={form.control}
          name="email"
          render={({ field }) => (
            <FormItem>
              <FormLabel>Email</FormLabel>
              <FormControl>
                <Input
                  placeholder="example@gmail.com"
            {...field}
            type="email"
            onChange={field.onChange}
          />
        </FormControl>
        <FormMessage />
      </FormItem>
    )}
  />
  <FormField
    control={form.control}
    name="password"
    render={({ field }) => (
      <FormItem>
        <FormLabel>Password</FormLabel>
        <FormControl>
          <Input
            placeholder="password"
            {...field}
            type="password"
            onChange={field.onChange}
          />
        </FormControl>
        <FormMessage />
      </FormItem>
    )}
  />
```

```
        <Button type="submit" className="w-full flex gap-2">

          SignIn

          <AiOutlineLoading3Quarters className={cn('animate-spin', { hidden: !isPending })} />

        </Button>

      </form>

    </Form>

  );

}
```

app\auth\components\SignOut.tsx

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

import createSupabaseServerClient from '@/lib/supabse/server';

import { redirect } from 'next/navigation';



export default async function SignOut() {

  const logout = async () => {

    'use server';



    const supabase = await createSupabaseServerClient();

    await supabase.auth.signOut();

    redirect('/goodbye');

  };



  return (

    <form action={logout}>

      <Button>SignOut</Button>

    </form>

  );

}
```

app\goodbye\page.tsx

```
import React from 'react';



export default function GoodbyePage() {

  return (

    <div className="flex flex-col gap-8 items-center  h-screen w-screen">

      <h2 className="mt-32   text-2xl font-semibold  animate-bounce">?? ?? ??? ????</h2>

    </div>

  );

}
```

app\goodluck\page.tsx

```
import { getCurrentUserInfo } from '@/lib/cookies';

import GoodluckTabs from './_components/GoodluckTabs';

import createSupabaseServerClient from '@/lib/supabse/server';

import AdAlert from '../post/_component/AdAlert';



export default async function GoodluckPage() {

  const currentUser = getCurrentUserInfo();

  const supabase = await createSupabaseServerClient();

  // Fetching the round data

  const { data: roundData, error: roundError } = await supabase

    .from('user_rounds')

    .select('*')

    .eq('user_id', currentUser?.id)

    .single();



  if (roundError) {

    console.log('Error fetching user round data:', roundError.message);

  }



  const postId = null;



  return (

    <div className="flex flex-col items-center pb-16">

      <GoodluckTabs />

      {/* AdAlert ????? ?? */}

      {currentUser && roundData && (
        <div className="fixed top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2 z-50">
          <AdAlert userId={currentUser.id} postId={postId} initialRoundData={roundData} />
        </div>
      )}
    </div>
  );
}
```

app\goodluck_components\GoodluckTabs.tsx

```
'use client';


import { useState } from 'react';

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

import TitleComponent from './TitleComponent';

import LotteryComponent from './Lottery';

import PensionComponent from './Pension';


const GoodluckTabs = () => {

  const [activeTab, setActiveTab] = useState<string>('lottery');


  const handleTabChange = (value: string) => {

    setActiveTab(value);

  };


  const getTitle = (tab: string) => {

    switch (tab) {

      case 'lottery':

        return '??';

      case 'pension':

        return '????';

      default:

        return '';

    }

  };
  return (
    <div className="flex flex-col items-center w-96 px-6">
      <div className="self-start w-full">
        <TitleComponent title={getTitle(activeTab)} />
      </div>
      <Tabs defaultValue="lottery" className="w-full" onValueChange={handleTabChange}>
        <TabsList className="grid w-full grid-cols-2">
          <TabsTrigger value="lottery">??</TabsTrigger>
          <TabsTrigger value="pension">????</TabsTrigger>
        </TabsList>
        <TabsContent value="lottery">
          <div className="flex items-center justify-center">
            <LotteryComponent />
          </div>
        </TabsContent>
        <TabsContent value="pension">
          <div className="flex items-center justify-center">
            <PensionComponent />
          </div>
        </TabsContent>
      </Tabs>
    </div>
  );
};

export default GoodluckTabs;
```

app\goodluck_components\Lottery.tsx

```
'use client';



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

import './lottery.css';

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

import { BiRevision, BiSave } from 'react-icons/bi';



export default function LotteryComponent() {

  const [isAnimating, setIsAnimating] = useState(false);

  const [stoppedBalls, setStoppedBalls] = useState<(number | null)[]>(Array(6).fill(null));

  const [resultsHistory, setResultsHistory] = useState<number[][]>([]);



  const handleClick = () => {

    setIsAnimating(true);

    setStoppedBalls(Array(6).fill(null)); // ??? ????? ?? ? ? ???

    setTimeout(() => {

      stopBalls();

    }, 3000);

  };



  const getRandomNumber = (existingNumbers: number[]) => {

    let number;

    do {

      number = Math.floor(Math.random() * 45) + 1;

    } while (existingNumbers.includes(number));

    return number;
  };


  const stopBalls = () => {
    let newNumbers: number[] = [];
    stoppedBalls.forEach((_, index) => {
      setTimeout(() => {
        setStoppedBalls((prev) => {
          const newState = [...prev];
          const newNumber = getRandomNumber(newNumbers);
          newState[index] = newNumber; // ?? ?? ??
          newNumbers.push(newNumber);

          if (index === stoppedBalls.length - 1) {
            setResultsHistory((prevResults) => [...prevResults, newNumbers]); // ??? ?? ????? ??
            console.log('Results:', newNumbers); // ??? ??? ??
          }
          return newState;
        });
      }, index * 300);
    });
  };


  const handleReset = () => {
    setIsAnimating(false);
    setStoppedBalls(Array(6).fill(null));
    setResultsHistory([]);
  };
```

```
const handleCopy = () => {
  const textToCopy = resultsHistory
    .map((resultSet, idx) => `${idx + 1}??: ${resultSet.join(', ')}`)
    .join('\n');

  if (navigator.clipboard && navigator.clipboard.writeText) {
    navigator.clipboard
      .writeText(textToCopy)
      .then(() => {
        alert('?? ??? ????? ???????.');
      })
      .catch((err) => {
        console.error('???? ??? ???????:', err);
        alert('???? ??? ???? ????. ???? ??? ???.');
      });
  } else {
    // Fallback method using execCommand for mobile compatibility
    const textArea = document.createElement('textarea');
    textArea.value = textToCopy;
    textArea.style.position = 'fixed'; // Avoid scrolling to bottom
    textArea.style.opacity = '0'; // Hide the element
    document.body.appendChild(textArea);
    textArea.focus();
    textArea.select();
    try {
      const successful = document.execCommand('copy');
```

```
        if (successful) {
          alert('?? ??? ????? ???????.');
        } else {
          alert('???? ??? ???? ????. ???? ??? ???.');
        }
      } catch (err) {
        console.error('???? ?? ??:', err);
        alert('???? ??? ???? ????. ???? ??? ???.');
      }
      document.body.removeChild(textArea);
    }
  };

  useEffect(() => {
    console.log('Updated Results History:', resultsHistory); // resultsHistory? ????? ??? ??? ??
  }, [resultsHistory]);

  return (
    <div className="pt-12 flex flex-col items-center justify-center w-screen">
      {/* <h2>LotteryPage</h2> */}


      {/* ?? ????? */}
      <div className="flex gap-2 border-b-[1px] border-gray-400 pb-10">
        {[...Array(6)].map((_, index) => (
          <div key={index} className="wrapper">
            <div
                        className={`ball bg-gray-50 ${isAnimating && stoppedBalls[index] === null ? 'animate ' : ''}
```

```
${stoppedBalls[index] !== null ? 'stopped ' : ''}`}
              style={{ animationDelay: `${index * 0.3}s` }}
            >
              {stoppedBalls[index] !== null && (
                <span className="number text-lg font-semibold">{stoppedBalls[index]}</span>
              )}
            </div>
          </div>
        ))}
      </div>

      {/* ?? */}
      <Button onClick={handleClick} className="mt-12 ">
        ?? ?? ????
      </Button>

      {/* ?? */}
      <div className="mt-12 ease-in-out duration-300 ">
        {/* <h3>?? ?? ??</h3> */}
        {resultsHistory.length > 0 ? (
          <div className="flex flex-col gap-2 px-6 ">
            {resultsHistory.map((resultSet, resultSetIdx) => (
              <div className="grid grid-cols-10 items-center gap-4 " key={resultSetIdx}>
                <p className="col-span-2 text-start ">{resultSetIdx + 1}??</p>
                <div className="col-span-8 grid grid-cols-6 gap-2">
                  {resultSet
                    .sort((a, b) => a - b)
              .map((result, idx) => (
                <div
                  key={idx}
                  className="flex items-center justify-center w-8 h-8 rounded-full bg-blue-100 font-semibold text-center"
                >
                  <span>{result}</span>
                </div>
              ))}
          </div>
          {/* <Button variant="outline" className="col-span-2">
            ??
          </Button> */}
        </div>
      ))}
    </div>
  ) : (
    <></>
  )}
</div>


{/* ??? */}
{resultsHistory.length > 0 && (
  <div className="w-full mt-12 px-12 grid grid-cols-2 gap-4 ">
    <Button variant="outline" className="col-span-1 gap-2" onClick={handleReset}>
      <BiRevision /> ????
    </Button>
    <Button variant="outline" className="col-span-1 gap-2" onClick={handleCopy}>
```

```
            <BiSave /> ????

          </Button>

        </div>

      )}

    </div>

  );

}
```

app\goodluck_components\Pension.tsx

```
'use client';


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

import './lottery.css';

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

import { BiRevision, BiSave } from 'react-icons/bi';


export default function PensionComponent() {

  const [isAnimating, setIsAnimating] = useState(false);

  const [stoppedBalls, setStoppedBalls] = useState<(number | null)[]>(Array(6).fill(null));

  const [resultsHistory, setResultsHistory] = useState<number[][]>([]);


  const handleClick = () => {

    setIsAnimating(true);

    setStoppedBalls(Array(6).fill(null)); // ??? ????? ?? ? ? ???

    setTimeout(() => {

      stopBalls();

    }, 3000);

  };


  const getRandomDigit = () => {

    return Math.floor(Math.random() * 10); // 0?? 9??? ?? ?? ??

  };


  const stopBalls = () => {

    let newNumbers: number[] = [];
  stoppedBalls.forEach((_, index) => {
    setTimeout(() => {
      setStoppedBalls((prev) => {
        const newState = [...prev];
        const newNumber = getRandomDigit();
        newState[index] = newNumber; // ?? ?? ??
        newNumbers.push(newNumber);

        if (index === stoppedBalls.length - 1) {
          setResultsHistory((prevResults) => [...prevResults, newNumbers]); // ??? ?? ????? ??
          console.log('Results:', newNumbers); // ??? ??? ??
        }
        return newState;
      });
    }, index * 300);
  });
};

const handleReset = () => {
  setIsAnimating(false);
  setStoppedBalls(Array(6).fill(null));
  setResultsHistory([]);
};

const handleCopy = () => {
  const textToCopy = resultsHistory
    .map((resultSet, idx) => `${idx + 1}??: ${resultSet.join(', ')}`)
    .join('\n');


  if (navigator.clipboard && navigator.clipboard.writeText) {
    navigator.clipboard
      .writeText(textToCopy)
      .then(() => {
        alert('???? ??? ????? ???????.');
      })
      .catch((err) => {
        console.error('???? ??? ???????:', err);
        alert('???? ??? ???? ????. ???? ??? ???.');
      });
  } else {
    // Fallback method using execCommand for mobile compatibility
    const textArea = document.createElement('textarea');
    textArea.value = textToCopy;
    textArea.style.position = 'fixed'; // Avoid scrolling to bottom
    textArea.style.opacity = '0'; // Hide the element
    document.body.appendChild(textArea);
    textArea.focus();
    textArea.select();
    try {
      const successful = document.execCommand('copy');
      if (successful) {
        alert('???? ??? ????? ???????.');
      } else {
        alert('???? ??? ???? ????. ???? ??? ???.');
```

```
        }
      } catch (err) {
        console.error('???? ?? ??:', err);
        alert('???? ??? ???? ????. ???? ??? ???.');
      }
      document.body.removeChild(textArea);
    }
  };

  useEffect(() => {
    console.log('Updated Results History:', resultsHistory); // resultsHistory? ????? ??? ??? ??
  }, [resultsHistory]);

  return (
    <div className="pt-12 flex flex-col items-center justify-center w-screen">
      {/* <h2>LotteryPage</h2> */}

      {/* ?? ????? */}
      <div className="flex gap-2 border-b-[1px] border-gray-400 pb-10">
        {[...Array(6)].map((_, index) => (
          <div key={index} className="wrapper">
            <div
                    className={`ball  bg-gray-50  ${isAnimating  &&  stoppedBalls[index]  ===  null  ?  'animate  '  :  ''}
${stoppedBalls[index] !== null ? 'stopped ' : ''}`}
              style={{ animationDelay: `${index * 0.3}s` }}
            >
              {stoppedBalls[index] !== null && (
          <span className="number text-lg font-semibold">{stoppedBalls[index]}</span>

        )}

      </div>

    </div>

  ))}

</div>



{/* ?? */}

<Button onClick={handleClick} className="mt-12">

  ???? ?? ????

</Button>



{/* ?? */}

<div className="mt-12 ease-in-out duration-300 ">

  {/* <h3>???? ?? ??</h3> */}

  {resultsHistory.length > 0 ? (

    <div className="flex flex-col gap-2 px-6">

      {resultsHistory.map((resultSet, resultSetIdx) => (

        <div className="grid grid-cols-10 items-center gap-1 " key={resultSetIdx}>

          <p className="col-span-3 text-start text-sm tracking-tighter">

            {resultSetIdx + 1}?? ???

          </p>

          {/* <div className="col-span-2 flex flex-col items-center">

            <p className="text-sm">1??</p>

            <p className="text-sm">???</p>

          </div> */}

          <div className="col-span-7 grid grid-cols-6 gap-2">
```

```
            {resultSet.map((result, idx) => (
              <div
                key={idx}
                className="flex items-center justify-center w-8 h-8 rounded-full bg-blue-100 font-semibold text-center"
              >
                <span>{result}</span>
              </div>
            ))}
          </div>
          {/* <Button variant="outline" className="col-span-2">
            ??
          </Button> */}
        </div>
      ))}
    </div>
  ) : (
    <></>
  )}
</div>


{/* ??? */}
{resultsHistory.length > 0 && (
  <div className="w-full mt-12 px-12 grid grid-cols-2 gap-4 ">
    <Button variant="outline" className="col-span-1 gap-2" onClick={handleReset}>
      <BiRevision /> ????
    </Button>
    <Button variant="outline" className="col-span-1 gap-2" onClick={handleCopy}>
```

```
            <BiSave /> ????

          </Button>

        </div>

      )}

    </div>

  );

}
```

app\goodluck_components\TitleComponent.tsx

```
const TitleComponent = ({ title }: { title: string }) => (

  <div className="flex gap-1 items-center h-10 border-b-[1px] border-gray-200 py-2">

    <p className="leading-tight text-sm text-gray-600">??? ?? {'>'}</p>

    <p className="leading-tight text-sm font-bold text-gray-600">{title}</p>

  </div>

);

export default TitleComponent;
```

app\message\page.tsx

```
import { CurrentUser, getCurrentUserInfo } from '@/lib/cookies';

import MessageList from './_component/MessageList';



export default function MessagePage() {

  const currentUser: CurrentUser | null = getCurrentUserInfo();



  if (!currentUser) {

    return <div>???? ?????.</div>;

  }



  return <MessageList currentUserId={currentUser.id} />;

}
```

app\message\create\page.tsx

```
// pages/message/create.tsx

import { redirect } from 'next/navigation';

import MessageForm from '../_component/MessageForm';

import { CurrentUser, getCurrentUserInfo } from '@/lib/cookies';

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

import { Terminal } from 'lucide-react';

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

import Link from 'next/link';



export default async function MessageCreatePage() {

  const currentUser: CurrentUser | null = getCurrentUserInfo();



  if (!currentUser) {

    console.log('nonono');

    return (

      <div className="w-screen h-height-minus-146 flex flex-col justify-center items-center">

        <div className="flex flex-col items-center justify-center gap-6 mx-6  shadow-lg">

          <Alert>

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

            <AlertTitle>??? ? ??? ?????? ?</AlertTitle>

            <AlertDescription>?? ???? ?? ????. ??? ???.</AlertDescription>

            <AlertDescription>

              <div className="mt-4 pt-4 border-t-[1px] border-gray-200 ">

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

                  <Button variant="outline" asChild>

                    <Link href="/message">?????</Link>
                  </Button>
                  <Button variant="secondary" asChild>
                    <Link href="/auth">??? ????</Link>
                  </Button>
                </div>
              </div>
            </AlertDescription>
          </Alert>
        </div>
      </div>
    );
  }

  return (
    <div>
      <MessageForm
        sender_id={currentUser.id}
        sender_name={currentUser.username || ''}
        sender_avatar_url={currentUser.avatar_url || ''}
        sender_email={currentUser.email || ''}
      />
    </div>
  );
}
```

app\message_action\message.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';

import { MessageType } from '../_component/InfiniteScrollMessages';


export async function fetchMessages(

  page: number,

  filter: 'received' | 'sent' | 'starred',

  userId: string

): Promise<MessageType[]> {

  const supabase = await createSupabaseServerClient();

  const pageSize = 10; // ???? ??? ?


  let query = supabase

    .from('message')

    .select(

      `

      id,

      sender_id,

      receiver_id,

      title,

      content,

      created_at,

      is_starred_by_sender,

      is_starred_by_receiver,

      read,
      deleted_by_sender,
      deleted_by_receiver,
      userdata:userdata!message_sender_id_fkey (
        username,
        avatar_url
      )
    `
    )
    .order('created_at', { ascending: false })
    .range((page - 1) * pageSize, page * pageSize - 1);

  if (filter === 'received') {
    query = query.eq('receiver_id', userId).eq('deleted_by_receiver', false);
  } else if (filter === 'sent') {
    query = query.eq('sender_id', userId).eq('deleted_by_sender', false);
  } else if (filter === 'starred') {
    query = query.or(

`and(sender_id.eq.${userId},is_starred_by_sender.eq.true,deleted_by_sender.eq.false),and(receiver_id.eq.${userId},is_
starred_by_receiver.eq.true,deleted_by_receiver.eq.false)`
    );
  }

  const { data, error } = await query;

  if (error) {
    console.error('Error fetching messages:', error);
    return [];
  }

  console.log('Fetched messages:', data);

  // Map userdata to correct type
  const formattedData = data.map((message: any) => ({
    ...message,
    userdata: {
      username: message.userdata.username,
      avatar_url: message.userdata.avatar_url,
    },
  }));

  return formattedData as MessageType[];
}

export async function toggleStarMessage(messageId: string, isStarred: boolean, userId: string) {
  const supabase = await createSupabaseServerClient();

  // ??? ?? ????
  const { data: message, error: fetchError } = await supabase
    .from('message')
    .select('sender_id, receiver_id')
    .eq('id', messageId)
    .single();
  if (fetchError) {

    console.error('Error fetching message:', fetchError);

    throw fetchError;

  }



  // ??? ?? ????

  const updateFields =

    message.sender_id === userId

      ? { is_starred_by_sender: !isStarred }

      : { is_starred_by_receiver: !isStarred };



  const { data, error } = await supabase.from('message').update(updateFields).eq('id', messageId);



  if (error) {

    console.error('Error toggling star on message:', error);

    throw error;

  }



  return data;

}



/* ??? ?? */

export async function createMessage(formData: FormData): Promise<void> {

  const supabase = await createSupabaseServerClient();



  const sender_id = formData.get('sender_id');

  const receiver_id = formData.get('receiver_id');
  if (!sender_id || !receiver_id) {
    throw new Error('Invalid sender_id or receiver_id');
  }

  const { data, error } = await supabase.from('message').insert({
    sender_id: sender_id,
    receiver_id: receiver_id,
    title: formData.get('title'),
    content: formData.get('content'),
    sender_name: formData.get('sender_name'),
    sender_avatar_url: formData.get('sender_avatar_url'),
    sender_email: formData.get('sender_email'),
  });

  if (error) {
    console.error('Error creating message:', error);
    throw error;
  }
}

/*  */
export async function deleteMessage(messageId: string, userId: string) {
  const supabase = await createSupabaseServerClient();

  // ??? ?? ????
  const { data: message, error: fetchError } = await supabase
```

```
    .from('message')

    .select('sender_id, receiver_id')

    .eq('id', messageId)

    .single();



  if (fetchError) {

    console.error('Error fetching message:', fetchError);

    throw fetchError;

  }



  // ??? ?? ????

  const updateFields =

    message.sender_id === userId ? { deleted_by_sender: true } : { deleted_by_receiver: true };



  const { data, error } = await supabase.from('message').update(updateFields).eq('id', messageId);



  if (error) {

    console.error('Error updating delete status on message:', error);

    throw error;

  }



  return data;

}



export async function markMessageAsRead(messageId: string, userId: string) {

  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase
```

```
    .from('message')

    .update({ read: true })

    .eq('id', messageId)

    .eq('receiver_id', userId);


  if (error) {

    console.error('Error marking message as read:', error);

    throw error;

  }


  return data;

}


/* ??? ?? ?? ???? ?? */

export async function searchUsers(query: string) {

  const supabase = await createSupabaseServerClient();


  const { data, error } = await supabase

    .from('userdata')

    .select('id, username')

    .ilike('username', `%${query}%`)

    .limit(10);


  if (error) {

    console.error('Error searching users:', error);

    return [];

  }
  return data;

}



/* ??? ?? ?? ID? ?? */

export async function fetchUserById(userId: string) {

  const supabase = await createSupabaseServerClient();



  const { data, error } = await supabase

    .from('userdata')

    .select('id, username, avatar_url')

    .eq('id', userId)

    .single();



  if (error) {

    console.error('Error fetching user by id:', error);

    return null;

  }



  return data;

}
```

app\message_component\FixedIconMessage.tsx

```
// components/FixedIconGroup.tsx

'use client';



import { useState } from 'react';

import { BiPencil, BiUpArrowAlt, BiDownArrowAlt, BiDotsVerticalRounded } from 'react-icons/bi';

import Link from 'next/link';



export default function FixedIconMessage() {

  const [isExpanded, setIsExpanded] = useState(false);



  const toggleIcons = () => {

    setIsExpanded(!isExpanded);

  };



  return (

    <div className="flex flex-col gap-4 fixed bottom-20 right-4 items-center">

      {isExpanded && (

        <>

          <div className="w-11 h-11 bg-emerald-200 rounded-full flex items-center justify-center shadow-lg">

            <Link href="/message/create">

              <BiPencil size={32} color="white" />

            </Link>

          </div>

        </>

      )}

      <div
```

```
        className="w-12 h-12 bg-emerald-200 rounded-full flex items-center justify-center shadow-lg cursor-pointer"

        onClick={toggleIcons}

      >

        <BiDotsVerticalRounded size={36} color="white" />

      </div>

    </div>

  );

}
```

app\message_component\InfiniteScrollMessages.tsx

```
'use client';


import { useEffect, useRef, useState } from 'react';

import { useRouter } from 'next/navigation';

import {

  fetchMessages,

  deleteMessage,

  markMessageAsRead,

  toggleStarMessage,

} from '../_action/message';

import { BiReply, BiStar, BiTrash } from 'react-icons/bi';

import {

  Dialog,

  DialogContent,

  DialogHeader,

  DialogTitle,

  DialogDescription,

  DialogTrigger,

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

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


export type MessageType = {

  id: string;

  sender_id: string;

  receiver_id: string;

  title: string;
  content: string;

  created_at: string;

  read: boolean;

  is_starred_by_sender: boolean;

  is_starred_by_receiver: boolean;

  deleted_by_sender: boolean;

  deleted_by_receiver: boolean;

  userdata: {

    username: string;

    avatar_url: string;

  };

};


type InfiniteScrollMessagesProps = {

  initialMessages: MessageType[];

  filter: 'received' | 'sent' | 'starred';

  currentUserId: string;

};


function formatKoreanDate(dateString: string): string {

  return dateString.replace('T', ' ').substring(0, 19);

}


export default function InfiniteScrollMessages({

  initialMessages,

  filter,

  currentUserId,
}: InfiniteScrollMessagesProps) {

  const [messages, setMessages] = useState<MessageType[]>(initialMessages);

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

  const [page, setPage] = useState(2); // Page 1 is already loaded in initialMessages

  const [hasMore, setHasMore] = useState(true); // ???? ? ??? ??

  const ref = useRef<HTMLDivElement>(null);

  const router = useRouter();



  useEffect(() => {

    setMessages(initialMessages);

  }, [initialMessages]);



  useEffect(() => {

    console.log('messages:', messages); // ??? ?? ?? ?? ??

  }, [messages]);



  const handleObserver = async (entries: IntersectionObserverEntry[]) => {

    const target = entries[0];

    if (target.isIntersecting && !loading && hasMore) {

      setLoading(true);

      const newMessages = await fetchMessages(page, filter, currentUserId);

      if (newMessages.length > 0) {

        setMessages((prev) => [...prev, ...newMessages]);

        setPage((prev) => prev + 1); // Increment page number

      } else {

        setHasMore(false); // ? ?? ??? ???? ??? false? ??

      }
      setLoading(false);
    }
  };

  useEffect(() => {
    const observer = new IntersectionObserver(handleObserver, {
      root: null,
      rootMargin: '20px',
      threshold: 0,
    });

    if (ref.current && hasMore) observer.observe(ref.current);
    return () => {
      if (ref.current) observer.unobserve(ref.current);
    };
  }, [loading, hasMore]);

  const handleDeleteMessage = async (id: string) => {
    await deleteMessage(id, currentUserId);
    setMessages((prevMessages) => prevMessages.filter((message) => message.id !== id));
  };

  const handleReadMessage = async (id: string) => {
    await markMessageAsRead(id, currentUserId);
    setMessages((prevMessages) =>
      prevMessages.map((message) => (message.id === id ? { ...message, read: true } : message))
    );
```

```
};


const handleToggleStarMessage = async (e: React.MouseEvent, id: string, isStarred: boolean) => {
  e.stopPropagation(); // ??? ?? ??
  await toggleStarMessage(id, isStarred, currentUserId);
  setMessages((prevMessages) =>
    prevMessages.map((message) =>
      message.id === id
        ? {
            ...message,
            is_starred_by_sender:
              message.sender_id === currentUserId ? !isStarred : message.is_starred_by_sender,
            is_starred_by_receiver:
              message.receiver_id === currentUserId ? !isStarred : message.is_starred_by_receiver,
          }
        : message
    )
  );
};


return (
  <div>
    {messages.length > 0 ? (
      messages.map((message) => (
        <Dialog key={message.id} onOpenChange={() => handleReadMessage(message.id)}>
          <DialogTrigger asChild>
            <div className="py-4 flex items-center gap-4 bg-white border-b-[1px] border-gray-200 cursor-pointer">
```

```
{/* ??? ???*/}
<BiStar
  size={24}
  className={
    message.sender_id === currentUserId
      ? message.is_starred_by_sender
        ? 'text-yellow-500'
        : 'text-gray-400'
      : message.is_starred_by_receiver
        ? 'text-yellow-500'
        : 'text-gray-400'
  }
  onClick={(e) =>
    handleToggleStarMessage(
      e,
      message.id,
      message.sender_id === currentUserId
        ? message.is_starred_by_sender
        : message.is_starred_by_receiver
    )
  }
/>
{/* ?? ??? ?? */}
<div className="flex-1 flex flex-col gap-1">
  <p className="font-semibold line-clamp-1">{message.title}</p>
  <div className="flex gap-2">
    <p className="text-xs text-gray-600">{message.userdata.username}</p>
    <p className="text-xs text-gray-600">{formatKoreanDate(message.created_at)}</p>
  </div>
</div>
{/* ?? ? ?? ??? */}
<div className="flex gap-1">
  <BiReply size={24} />
  <Dialog>
    <DialogTrigger asChild>
      <button onClick={(e) => e.stopPropagation()}>
        <BiTrash size={24} />
      </button>
    </DialogTrigger>
    <DialogContent>
      <DialogHeader>
        <DialogTitle>?? ??</DialogTitle>
      </DialogHeader>
      <DialogDescription>
        ??? ? ???? ????????? ? ??? ??? ? ????.
      </DialogDescription>
      <div className="mt-4 flex justify-end space-x-2">
        <DialogTrigger asChild>
          <Button variant="outline">??</Button>
        </DialogTrigger>
        <Button
          variant="destructive"
          onClick={() => handleDeleteMessage(message.id)}
        >
```

```
              ??
            </Button>
          </div>
        </DialogContent>
      </Dialog>
    </div>
  </div>
</DialogTrigger>
<DialogContent>
  <div className="flex flex-col gap-4">
    <div className="flex items-center gap-2">
      <img
        src={message.userdata.avatar_url || '/default-avatar.png'}
        alt="avatar"
        className="w-10 h-10 rounded-full"
      />
      <div>
        <p className="font-bold">{message.userdata.username}</p>
        <p className="text-sm text-gray-600">{formatKoreanDate(message.created_at)}</p>
      </div>
    </div>
    <DialogHeader>
      <DialogTitle>{message.title}</DialogTitle>
    </DialogHeader>
    <DialogDescription>
      {message.content} {message.userdata.username} ?? ??? ??????.
    </DialogDescription>
```

```
</div>
<div className="mt-4 flex justify-end space-x-2">
  <Button
    variant="outline"
    onClick={() => router.push(`/message/create?receiver_id=${message.sender_id}`)}
  >
    <BiReply className="mr-1" /> ??
  </Button>
  <Dialog>
    <DialogTrigger asChild>
      <Button variant="destructive">
        <BiTrash className="mr-1" /> ??
      </Button>
    </DialogTrigger>
    <DialogContent>
      <DialogHeader>
        <DialogTitle>?? ??</DialogTitle>
      </DialogHeader>
      <DialogDescription>
        ??? ? ???? ????????? ? ??? ??? ? ????.
      </DialogDescription>
      <div className="mt-4 flex justify-end space-x-2">
        <DialogTrigger asChild>
          <Button variant="outline">??</Button>
        </DialogTrigger>
        <Button variant="destructive" onClick={() => handleDeleteMessage(message.id)}>
          ??
```

```
                  </Button>
                 </div>
                </DialogContent>
               </Dialog>
              </div>
             </DialogContent>
            </Dialog>
           ))
          ) : (
           <p className="hover:text-red-200 text-blue-400">No messages</p>
          )}
          {loading && <p>Loading...</p>}
          <div ref={ref} />
         </div>
        );
       }
```

app\message_component\MessageDetailDialog.tsx

```
'use client';


import {

  Dialog,

  DialogContent,

  DialogHeader,

  DialogTitle,

  DialogDescription,

  DialogTrigger,

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

import { useEffect, useState } from 'react';

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

import { BiReply, BiTrash } from 'react-icons/bi';

import { MessageType } from './InfiniteScrollMessages';

import { deleteMessage, markMessageAsRead } from '../_action/message';

import { useRouter } from 'next/navigation';


interface MessageDetailDialogProps {

  message: MessageType;

  onClose: () => void;

  onDelete: (id: string) => void;

}


const formatKoreanDate = (dateString: string): string => {

  return dateString.replace('T', ' ').substring(0, 19);

};
```

```
const MessageDetailDialog: React.FC<MessageDetailDialogProps> = ({
  message,
  onClose,
  onDelete,
}) => {
  const [dialogOpen, setDialogOpen] = useState(false);
  const router = useRouter();

  useEffect(() => {
    if (!message.read) {
      markMessageAsRead(message.id);
    }
  }, [message.id, message.read]);

  const handleDelete = async () => {
    await deleteMessage(message.id);
    onDelete(message.id);
    onClose();
  };

  const handleReply = () => {
    router.push(`/message/create?receiver_id=${message.sender_id}`);
  };

  return (
    <Dialog open={dialogOpen} onOpenChange={setDialogOpen}>
```

```
<DialogTrigger asChild>
  <div onClick={() => setDialogOpen(true)} className="cursor-pointer flex-1 py-4">
    <p className="font-semibold line-clamp-1">{message.title}</p>
  </div>
</DialogTrigger>
<DialogContent>
  <DialogHeader>
    <DialogTitle>{message.title}</DialogTitle>
  </DialogHeader>
  <div className="flex flex-col gap-4">
    <div className="flex items-center gap-2">
      <img
        src={message.userdata.avatar_url || '/default-avatar.png'}
        alt="avatar"
        className="w-10 h-10 rounded-full"
      />
      <div>
        <p className="font-bold">{message.userdata.username}</p>
        <p className="text-sm text-gray-600">{formatKoreanDate(message.created_at)}</p>
      </div>
    </div>
    <DialogDescription>{message.content}</DialogDescription>
  </div>
  <div className="mt-4 flex justify-end space-x-2">
    <Button variant="outline" onClick={handleReply}>
      <BiReply className="mr-1" /> ??
    </Button>
```

```
          <Button variant="destructive" onClick={handleDelete}>
            <BiTrash className="mr-1" /> ??
          </Button>
        </div>
      </DialogContent>
    </Dialog>
  );
};

export default MessageDetailDialog;
```

app\message_component\MessageFilter.tsx

```
// MessageFilter.tsx



'use client';



interface MessageFilterProps {

  filter: 'received' | 'sent' | 'starred';

  setFilter: (filter: 'received' | 'sent' | 'starred') => void;

}



const MessageFilter: React.FC<MessageFilterProps> = ({ filter, setFilter }) => {

  return (

    <div className="grid grid-cols-3 h-12 ">

      <div

        className={`bg-white border-b-[1px] flex justify-center items-center cursor-pointer ${

          filter === 'received' ? 'border-gray-400 font-bold' : 'border-gray-200'

        }`}

        onClick={() => setFilter('received')}

      >

        <p className="text-center">?? ??</p>

      </div>

      <div

        className={`bg-white border-b-[1px] flex justify-center items-center cursor-pointer ${

          filter === 'sent' ? 'border-gray-400 font-bold' : 'border-gray-200'

        }`}

        onClick={() => setFilter('sent')}

      >
    <p className="text-center">?? ??</p>
   </div>
   <div
    className={`bg-white border-b-[1px] flex justify-center items-center cursor-pointer ${
     filter === 'starred' ? 'border-gray-400 font-bold' : 'border-gray-200'
    }`}
    onClick={() => setFilter('starred')}
   >
    <p className="text-center">???</p>
   </div>
  </div>
 );
};

export default MessageFilter;
```

app\message_component\MessageForm.tsx

```
'use client';


import { useState, FormEvent, ChangeEvent, useEffect } from 'react';

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

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

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

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

import { createMessage, searchUsers, fetchUserById } from '../_action/message';

// fetchUserById ?? ???

import {

  Dialog,

  DialogContent,

  DialogDescription,

  DialogHeader,

  DialogTitle,

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

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


export type UserPropsType = {

  sender_id: string;

  sender_name?: string;

  sender_email?: string;

  sender_avatar_url?: string;

};


export default function MessageForm({
  sender_id,
  sender_name,
  sender_avatar_url,
  sender_email,
}: UserPropsType) {
  const [receiverId, setReceiverId] = useState('');
  const [receiverUsername, setReceiverUsername] = useState('');
  const [usernameSuggestions, setUsernameSuggestions] = useState<
    { id: string; username: string }[]
  >([]);
  const [title, setTitle] = useState('');
  const [content, setContent] = useState('');
  const [dialogOpen, setDialogOpen] = useState(false);
  const [dialogMessage, setDialogMessage] = useState('');
  const router = useRouter();
  const searchParams = useSearchParams();
  const receiverIdFromParams = searchParams.get('receiver_id');

  useEffect(() => {
    if (receiverIdFromParams) {
      (async () => {
        const user = await fetchUserById(receiverIdFromParams);
        if (user) {
          setReceiverId(user.id);
          setReceiverUsername(user.username);
        }
      })();
    }
  }, [receiverIdFromParams]);


  const handleUsernameChange = async (e: ChangeEvent<HTMLInputElement>) => {
    const query = e.target.value;
    setReceiverUsername(query);


    if (query.length > 1) {
      const users = await searchUsers(query);
      setUsernameSuggestions(users);
    } else {
      setUsernameSuggestions([]);
    }
  };


  const handleUserSelect = (userId: string, username: string) => {
    setReceiverId(userId);
    setReceiverUsername(username);
    setUsernameSuggestions([]);
  };


  const totalFormData = async (e: FormEvent) => {
    e.preventDefault();
    const form = e.currentTarget as HTMLFormElement;
    const formData = new FormData(form);


    formData.append('sender_id', sender_id);
    formData.append('receiver_id', receiverId);

    formData.append('title', title);

    formData.append('content', content);

    if (sender_name) {

      formData.append('sender_name', sender_name);

    }

    if (sender_avatar_url) {

      formData.append('sender_avatar_url', sender_avatar_url);

    }

    if (sender_email) {

      formData.append('sender_email', sender_email);

    }



    try {

      await createMessage(formData);

      setDialogMessage('???? ????? ???????.');

    } catch (error) {

      setDialogMessage('??? ?? ? ??? ??????.');

    } finally {

      setDialogOpen(true);

    }

  };



  const handleDialogClose = () => {

    setDialogOpen(false);

    router.push('/message');

  };
```

```
return (
  <div className="mt-4 mx-auto px-6">
    <h1 className="text-2xl font-bold mb-4 text-gray-800">?? ?? ???</h1>
    <form onSubmit={totalFormData} className="space-y-4">
      <div className="space-y-2">
        <Label htmlFor="receiver_username">?? ??</Label>
        <Input
          type="text"
          name="receiver_username"
          id="receiver_username"
          required
          value={receiverUsername}
          onChange={handleUsernameChange}
          disabled={!!receiverIdFromParams} // receiver_id? ?? ? ????
        />
        {usernameSuggestions.length > 0 && (
          <ul className="bg-white border border-gray-200 mt-2">
            {usernameSuggestions.map((user) => (
              <li
                key={user.id}
                className="p-2 cursor-pointer hover:bg-gray-200"
                onClick={() => handleUserSelect(user.id, user.username)}
              >
                {user.username}
              </li>
            ))}
      </ul>
    )}
  </div>
  <div className="space-y-2">
    <Label htmlFor="title">??</Label>
    <Input
      type="text"
      name="title"
      id="title"
      required
      value={title}
      onChange={(e) => setTitle(e.target.value)}
    />
  </div>
  <div className="space-y-2">
    <Label htmlFor="content">??</Label>
    <Textarea
      name="content"
      id="content"
      required
      value={content}
      onChange={(e) => setContent(e.target.value)}
    />
  </div>
  <Button type="submit">?? ???</Button>
</form>
```

```
      {/* Dialog */}

      <Dialog open={dialogOpen} onOpenChange={setDialogOpen}>

        <DialogContent>

          <DialogHeader>

            <DialogTitle>?? ?? ??</DialogTitle>

            <DialogDescription>{dialogMessage}</DialogDescription>

          </DialogHeader>

          <div className="mt-4 flex justify-end space-x-2">

            <Button variant="outline" onClick={handleDialogClose}>

              ??

            </Button>

          </div>

        </DialogContent>

      </Dialog>

    </div>

  );

}
```

app\message_component\MessageList.tsx

```
// MessageList.tsx


'use client';


import { useState, useEffect } from 'react';

import { fetchMessages } from '../_action/message';

import InfiniteScrollMessages from './InfiniteScrollMessages';

import FixedIconMessage from './FixedIconMessage';

import MessageFilter from './MessageFilter';

import { MessageType } from './InfiniteScrollMessages';


interface MessageListProps {

  currentUserId: string;

}


const MessageList: React.FC<MessageListProps> = ({ currentUserId }) => {

  const [initialMessages, setInitialMessages] = useState<MessageType[]>([]);

  const [filter, setFilter] = useState<'received' | 'sent' | 'starred'>('received');


  useEffect(() => {

    const fetchInitialMessages = async () => {

      const messages = await fetchMessages(1, filter, currentUserId);

      setInitialMessages(messages);

    };


    fetchInitialMessages();
  }, [filter, currentUserId]);


  useEffect(() => {
    console.log('initialMessages:', initialMessages); // ??? ?? ?? ?? ??
  }, [initialMessages]);


  return (
    <div>
      <MessageFilter filter={filter} setFilter={setFilter} />
      <div className="flex flex-col px-4 pt-4 ">
        <InfiniteScrollMessages
          initialMessages={initialMessages}
          filter={filter}
          currentUserId={currentUserId}
        />
      </div>
      <FixedIconMessage />
    </div>
  );
};


export default MessageList;
```

app\post\page.tsx

```
// app/post/page.tsx


import createSupabaseServerClient from '@/lib/supabse/server';

import SearchBar from './_component/SearchBar';

import { CurrentUser, getCurrentUserInfo } from '@/lib/cookies';

import PopularitySwitchClient from './_component/PopularitySwitchClient';

import FixedIconGroup from './_component/FixedIconGroup';

import { findCategoryNameById } from './_action/category';

import InfiniteScrollPosts from './_component/InfiniteScrollPosts';


export default async function PostPage() {

  const supabase = await createSupabaseServerClient();

  const result = await supabase

    .from('post')

    .select(

      `

    id, title, created_at, views, comments, author_id, author_name, author_email, author_avatar_url,

    parent_category_id, child_category_id

  `

    )

    .order('created_at', { ascending: false })

    .limit(10);


  const postsData = result.data || [];


  const initialPosts = postsData.map((post) => {
    return {
      ...post,
      parent_category_name: findCategoryNameById(post.parent_category_id),
      child_category_name: findCategoryNameById(post.child_category_id),
    };
  });


  const currentUser: CurrentUser | null = getCurrentUserInfo();


  return (
    <div className="pt-4">
      <SearchBar />
      {/* <PopularitySwitchClient initialPosts={initialPosts} userId={currentUser?.id ?? null} /> */}
      <div className="flex flex-col px-4 pt-4 ">
        <InfiniteScrollPosts initialPosts={initialPosts} userId={currentUser?.id ?? null} />
      </div>
      <FixedIconGroup />
    </div>
  );
}
```

app\post\types.ts

```
export type PostType = {

  id: string;

  author_id: string;

  created_at: string;

  parent_category_name?: string; // ?? ???? ??

  child_category_name?: string; // ?? ???? ??

  author_name?: string;

  author_email?: string;

  author_avatar_url?: string;

  views?: number;

  comments?: number;

  title?: string;

};
```

app\post\create\page.tsx

```
import PostForm from '../_component/PostForm';

import { CurrentUser, getCurrentUserInfo } from '@/lib/cookies';

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

import { Terminal } from 'lucide-react';

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

import Link from 'next/link';



export default async function PostCreatePage() {

  const currentUser: CurrentUser | null = getCurrentUserInfo();



  if (!currentUser) {

    console.log('nonono');

    return (

      <div className="w-screen h-height-minus-146 flex flex-col justify-center items-center">

        <div className="flex flex-col items-center justify-center gap-6 mx-6  shadow-lg">

          <Alert>

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

            <AlertTitle>??? ? ??? ?????? ?</AlertTitle>

            <AlertDescription>?? ???? ?? ????. ??? ???.</AlertDescription>

            <AlertDescription>

              <div className="mt-4 pt-4 border-t-[1px] border-gray-200 ">

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

                  <Button variant="outline" asChild>

                    <Link href="/post">?????</Link>

                  </Button>

                  <Button variant="secondary" asChild>
                    <Link href="/auth">??? ????</Link>

                  </Button>

                </div>

              </div>

            </AlertDescription>

          </Alert>

        </div>

      </div>

    );

  }


  return (

    <div>

      <PostForm

        user_id={currentUser.id}

        user_name={currentUser.username || ''}

        user_avatar_url={currentUser.avatar_url || ''}

        user_email={currentUser.email || ''}

      />

    </div>

  );

}
```

app\post\detail\[id]\page.tsx

```
import { getCurrentUserInfo } from '@/lib/cookies';

import { Suspense, lazy } from 'react';

import Breadcrumbs from '../../_component/Breadcrumbs';

import createSupabaseServerClient from '@/lib/supabse/server';

import AdAlert from '../../_component/AdAlert';

import { calculatePoints } from '../../_action/adPoint';

import AdFixedPage from '@/app/_components/Ad_Bottom';



// ?? ??? ????

const PostHeader = lazy(() => import('../../_component/PostHeader'));

const PostContent = lazy(() => import('../../_component/PostContent'));

const ExternalLinks = lazy(() => import('../../_component/ExternalLinksComponent'));

const ClientPostDetail = lazy(() => import('../../_component/ClientPostDetail'));



export default async function PostDetailPage({ params }: { params: { id: string } }) {

  const { id } = params;



  const supabase = await createSupabaseServerClient();

  const { data: post, error } = await supabase.from('post').select('*').eq('id', id).single();

  const currentUser = getCurrentUserInfo();

  console.log('currentUser[id]', currentUser);



  if (error) {

    return <div>Error loading post: {error.message}</div>;

  }
```

```
// Fetching the round data

let roundData;

if (currentUser) {

  const { data, error: roundError } = await supabase

    .from('user_rounds')

    .select('*')

    .eq('user_id', currentUser.id)

    .single();



  if (roundError) {

    console.log('Error fetching user round data:', roundError.message);

  } else {

    roundData = data;

  }

} else {

  // ????? ?? ???? ?? ?? ??? ???? ??

  roundData = {

    round_points: calculatePoints(),

    current_round_index: 0,

  };

}



const animationExecuted =

  typeof window !== 'undefined' && localStorage.getItem(`post_${id}_animation_executed`);



return (

  <div className="relative mx-4">
```

```
<Breadcrumbs category={post?.category} />

<Suspense fallback={<div>Loading Post Header...</div>}>

  <PostHeader

    title={post?.title}

    author_name={post?.author_name}

    author_email={post?.author_email}

    author_avatar_url={post?.author_avatar_url}

    author_id={post?.author_id}

    author={post?.author}

    views={post?.views}

    comments={post?.comments}

    created_at={post?.created_at}

    point={currentUser?.point ?? 0}

  />

</Suspense>

<AdFixedPage />

{post?.images ? (

  <div className="flex flex-col gap-4">

    {post.images.map((image: string, index: number) => (

      <img key={index} src={image} alt={`Post Image ${index + 1}`} />

    ))}

  </div>

) : null}

<Suspense fallback={<div>Loading Post Content...</div>}>

  <PostContent content={post?.content} />

</Suspense>

<Suspense fallback={<div>Loading External Links...</div>}>
```

```
        <ExternalLinks linkUrl1={post?.linkUrl1} linkUrl2={post?.linkUrl2} />

      </Suspense>

      <Suspense fallback={<div>Loading Post Details...</div>}>

        <ClientPostDetail

          postId={id}

          initialUser={

            currentUser ? { ...currentUser, current_points: currentUser.point ?? 0 } : null

          }

          initialPost={post}

        />

      </Suspense>

      {/* AdAlert ????? ?? */}

      {roundData && !animationExecuted && (

        <div className="fixed top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2 z-50">

          <AdAlert userId={currentUser?.id ?? null} postId={id} initialRoundData={roundData} />

        </div>

      )}

    </div>

  );

}
```

app\post\edit\[id]\page.tsx

```
import { redirect } from 'next/navigation';

import { CurrentUser, getCurrentUserInfo } from '@/lib/cookies';

import EditForm from '../../_component/EditForm';



export default async function PostEditPage() {

  const currentUser: CurrentUser | null = getCurrentUserInfo();



  if (!currentUser) {

    console.log('nonono');

    return redirect('post');

  }



  return (

    <div>

      <h2>PostEditPage</h2>

      {currentUser.id ? <p>Hello {currentUser.id}</p> : null}

      {currentUser.username ? <p>Hello {currentUser.username}</p> : null}

      {currentUser.email ? <p>Hello {currentUser.email}</p> : null}

      {currentUser.avatar_url ? <p>Hello {currentUser.avatar_url}</p> : null}

      <EditForm

        user_id={currentUser.id}

        user_name={currentUser.username || ''}

        user_avatar_url={currentUser.avatar_url || ''}

      />

    </div>

  );
}
```

app\post\search\page.tsx

```
// app/post/search/page.tsx

import createSupabaseServerClient from '@/lib/supabse/server';

import SearchBar from '../_component/SearchBar';

import { CurrentUser, getCurrentUserInfo } from '@/lib/cookies';

import InfiniteScrollPosts from '../_component/InfiniteScrollPosts';

import FixedIconGroup from '../_component/FixedIconGroup';

import { PostType } from '../types';



export async function searchPosts(query: string): Promise<PostType[]> {

  const supabase = await createSupabaseServerClient();

  const result = await supabase

    .from('post')

    .select('*')

    .ilike('title', `%${query}%`)

    .order('created_at', { ascending: false })

    .limit(10);



  return result.data || [];

}



export default async function PostSearchPage({

  searchParams,

}: {

  searchParams: { query: string };

}) {

  const query = searchParams.query || '';
  const initialPosts = await searchPosts(query);



  const currentUser: CurrentUser | null = getCurrentUserInfo();



  return (

    <div>

      <SearchBar initialQuery={query} />

      <div className="grid grid-cols-2 h-12 ">

        <div className="bg-white border-b-[1px] border-gray-400 flex justify-center items-center">

          <p className="font-bold text-center">Search Results</p>

        </div>

      </div>

      <div className="flex flex-col px-4 pt-4 ">

        <InfiniteScrollPosts

          initialPosts={initialPosts}

          userId={currentUser?.id ?? null}

          searchTerm={query}

        />

      </div>

      <FixedIconGroup />

    </div>

  );

}
```

app\post\[categoryId]\page.tsx

```
//app>post>[catergoryId]\page.tsx

import { getCurrentUserInfo } from '@/lib/cookies';

import createSupabaseServerClient from '@/lib/supabse/server';

import { calculatePoints } from '../_action/adPoint';

import { findCategoryNameById } from '../_action/category';

import SearchBar from '../_component/SearchBar';

import InfiniteScrollPosts from '../_component/InfiniteScrollPosts';

import FixedIconGroup from '../_component/FixedIconGroup';



export default async function PostListPageByCategory({

  params,

}: {

  params: { categoryId: string };

}) {

  const { categoryId } = params;



  const supabase = await createSupabaseServerClient();

  const { data: posts, error } = await supabase

    .from('post')

    .select('*')

    .eq('parent_category_id', categoryId) // ???? ID? ???

    .order('created_at', { ascending: false })

    .limit(10);



  const currentUser = getCurrentUserInfo();
const postsData = posts || [];


const initialPosts = await Promise.all(

  postsData.map(async (post) => {

    const parentCategoryName = await findCategoryNameById(post.parent_category_id);

    const childCategoryName = await findCategoryNameById(post.child_category_id);

    return {

      ...post,

      parent_category_name: parentCategoryName,

      child_category_name: childCategoryName,

    };

  })

);


if (error) {

  return <div>Error loading post: {error.message}</div>;

}


// Fetching the round data

let roundData;

if (currentUser) {

  const { data, error: roundError } = await supabase

    .from('user_rounds')

    .select('*')

    .eq('user_id', currentUser.id)

    .single();
  if (roundError) {

    console.log('Error fetching user round data:', roundError.message);

  } else {

    roundData = data;

  }

} else {

  // ????? ?? ???? ?? ?? ??? ???? ??

  roundData = {

    round_points: calculatePoints(),

    current_round_index: 0,

  };

}



// console.log('categoryInitial', initialPosts);

// console.log('Category ID:', categoryId); // ???? ID ?? ??



return (

  <div className="pt-4">

    <p>???? ???</p>

    <SearchBar />

    <div className="grid grid-cols-2 h-12">

      <div className="bg-white border-b-[1px] border-gray-400 flex justify-center items-center">

        <p className="font-bold text-center">??</p>

      </div>

      <div className="bg-white border-b-[1px] border-gray-200 flex justify-center items-center">

        <p className="font-light text-center">??</p>

      </div>
      </div>

      <div className="flex flex-col px-4 pt-4 ">

        <InfiniteScrollPosts

          initialPosts={initialPosts}

          userId={currentUser?.id ?? null}

          categoryId={categoryId} // ???? ID ??

        />

      </div>

      <FixedIconGroup />

    </div>

  );

}
```

app\post_action\adPoint.ts

```
/* 1? = 20point */


export const calculatePoints = (): number[] => {
  const rounds = 7 + Math.floor(Math.random() * 3); // 7 ~ 9 ??? ??
  const points: number[] = new Array(rounds).fill(0);
  let sum = 0;
  let bigPointIndex = Math.floor(Math.random() * rounds);


  for (let i = 0; i < rounds; i++) {
    if (i === bigPointIndex) {
      points[i] = 10 + Math.floor(Math.random() * 6); // 10 ~ 15 ??? ??
    } else {
      points[i] = Math.floor(Math.random() * 4); // 0 ~ 3 ??? ??
    }
    sum += points[i];
  }


  // ??? 20? ??? ?? ??
  while (sum !== 20) {
    sum = 0;
    bigPointIndex = Math.floor(Math.random() * rounds);
    for (let i = 0; i < rounds; i++) {
      if (i === bigPointIndex) {
        points[i] = 10 + Math.floor(Math.random() * 6); // 10 ~ 15 ??? ??
      } else {
        points[i] = Math.floor(Math.random() * 4); // 0 ~ 3 ??? ??
      }
      sum += points[i];
    }
  }

  return points;
};

/* export async function fetchIpAddress(): Promise<string> {
  try {
  const response = await fetch('/api/get-ip');
  const data = await response.json();
  return data.ip;
  } catch (error) {
  console.error('Failed to fetch IP address:', error);
  return 'Unknown IP';
  }
  } */

export async function fetchIpAddress(): Promise<string> {
  try {
    const response = await fetch('/api/get-ip');
    if (!response.ok) {
      throw new Error('Failed to fetch IP address');
    }
    const data = await response.json();
    return data.ip;
  } catch (error) {

    console.error('Failed to fetch IP address:', error);

    return 'Unknown IP';

  }

}
```

app\post_action\adPointSupabase.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';

import { calculatePoints, fetchIpAddress } from './adPoint';


export const saveRoundData = async (userId: string | null, roundPoints: number[]) => {

  const totalPoints = roundPoints.reduce((acc, val) => acc + val, 0);

  const rounds = roundPoints.length;

  const ipAddress = await fetchIpAddress();


  const newRoundData = {

    userId: userId ?? 'anonymous',

    ipAddress,

    timestamp: new Date().toISOString(),

    round: rounds,

    pointsArray: roundPoints,

    totalPoints,

  };


  // console.log('New round data:', newRoundData);


  // ???? ?? ??? ??? ??? ??

  await batchLogRounds([newRoundData]);

};


export async function batchLogRounds(roundsData: any[]) {
```

```
try {

  console.log('Starting to log rounds to Supabase');

  console.log('Rounds data:', roundsData);



  const saveRoundsData = roundsData[0];

  console.log('saveRoundsData:', saveRoundsData);



  // Ensure points_array is in a format Supabase can handle

  const pointsArray = `{${saveRoundsData.pointsArray.join(',')}}`;



  const formRoundsData = {

    user_id: saveRoundsData.userId,

    ip_address: saveRoundsData.ipAddress,

    timestamp: saveRoundsData.timestamp,

    round: saveRoundsData.round,

    points_array: pointsArray,

    total_points: saveRoundsData.totalPoints,

  };



  const supabase = await createSupabaseServerClient();



  console.log('formRoundsData:', formRoundsData);



  const { data, error } = await supabase.from('round_points').insert([formRoundsData]);



  if (data) {

    console.log('Round points logged successfully', data);
    }


    if (error) {
      console.error('Error logging round points:', error);
    }
  } catch (err) {
    console.error('Unexpected error during logging rounds:', err);
  }
}


export async function getUserRoundData(userId: string) {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase
    .from('user_rounds')
    .select('*')
    .eq('user_id', userId)
    .single();


  if (error) {
    if (error.code === 'PGRST116') {
      console.log('No existing round data found, initializing new round.');
      return {
        user_id: userId,
        current_round_index: 0,
        round_points: calculatePoints(),
        last_updated: new Date().toISOString(),
      };
    }
    console.error('Error fetching user round data:', error);
    return null;
  }

  return data;
}

export async function saveUserRoundData(
  userId: string,
  currentRoundIndex: number,
  roundPoints: number[]
) {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.from('user_rounds').upsert({
    user_id: userId,
    current_round_index: currentRoundIndex,
    round_points: roundPoints,
  });

  if (error) {
    console.error('Error saving user round data:', error);
  }
  return data;
}

// ?? ?? ??? ??
```

```
export async function addUserPoints(userId: string, points: number) {
 const supabase = await createSupabaseServerClient();
 const { data, error } = await supabase.from('user_points').select('*').eq('user_id', userId);

 if (error) {
  console.error('Error fetching user points:', error);
  return null;
 }

 if (data && data.length > 0) {
  // Update existing record
  const { data: updateData, error: updateError } = await supabase
   .from('user_points')
   .update({
    total_points: data[0].total_points + points,
    current_points: data[0].current_points + points,
    updated_at: new Date().toISOString(),
   })
   .eq('user_id', userId);

  if (updateError) {
   console.error('Error updating user points:', updateError);
  } else {
   console.log('User points updated successfully Server', updateData);
   console.log('User points Server', points);
  }
```

```
    return updateData;
  } else {
    // Insert new record
    const { data: insertData, error: insertError } = await supabase.from('user_points').insert({
      user_id: userId,
      total_points: points,
      current_points: points,
      created_at: new Date().toISOString(),
      updated_at: new Date().toISOString(),
    });

    if (insertError) {
      console.error('Error adding user points:', insertError);
    }

    return insertData;
  }
}

// ??? ?? ??? ??
export async function addWritingPoints(userId: string, points: number) {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.from('user_points').select('*').eq('user_id', userId);

  if (error) {
    console.error('Error fetching user points:', error);
    return null;
```

```
}


if (data && data.length > 0) {

  // Update existing record

  const { data: updateData, error: updateError } = await supabase

    .from('user_points')

    .update({

      total_points: data[0].total_points + points,

      current_points: data[0].current_points + points,

      writing_points: (data[0].writing_points || 0) + points, // writing_points ?? ????

      updated_at: new Date().toISOString(),

    })

    .eq('user_id', userId);


  if (updateError) {

    console.error('Error updating user points:', updateError);

  } else {

    console.log('User writing points updated successfully', updateData);

  }


  return updateData;

} else {

  // Insert new record

  const { data: insertData, error: insertError } = await supabase.from('user_points').insert({

    user_id: userId,

    total_points: points,

    current_points: points,
      writing_points: points, // writing_points ?? ??
      created_at: new Date().toISOString(),
      updated_at: new Date().toISOString(),
    });

    if (insertError) {
      console.error('Error adding user points:', insertError);
    }

    return insertData;
  }
}

export async function useUserPoints(userId: string, points: number, type: 'use' | 'donate') {
  const column = type === 'use' ? 'total_used_points' : 'total_donated_points';

  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.from('user_points').select('*').eq('user_id', userId);

  if (error) {
    console.error('Error fetching user points:', error);
    return null;
  }

  if (data && data.length > 0) {
    // Define the type for updateValues with an index signature
    const updateValues: {
      current_points: number;

      updated_at: string;

      [key: string]: any;

    } = {

      current_points: data[0].current_points - points,

      updated_at: new Date().toISOString(),

    };



    updateValues[column] = data[0][column] + points;



    const { data: updateData, error: updateError } = await supabase

      .from('user_points')

      .update(updateValues)

      .eq('user_id', userId);



    if (updateError) {

      console.error(`Error ${type === 'use' ? 'using' : 'donating'} user points:`, updateError);

    }



    return updateData;

  } else {

    console.error(`No user points record found for user_id: ${userId}`);

    return null;

  }

}
```

app\post_action\category.ts

```
export const categories = [
  { id: '37f26765-90c4-49e1-88ba-b2486b3632f7', name: '?????' },
  { id: 'ce7a39ac-f403-4750-b640-890bbcdf0dd3', name: '?????' },
  { id: '6a7414fc-369a-49b4-a2ab-b14c95efaf97', name: '????' },
  { id: 'a9649df0-2838-469c-8b40-48e14c55559a', name: '????' },
  { id: 'f189e104-6614-49d7-b3fd-e68d5988844b', name: '?????' },
  { id: '379ca741-04ad-447b-8c64-f848f3185e42', name: '??????' },
  { id: 'aab129c4-09d8-4b7c-ac78-5479d84acd3c', name: '?????' },
  { id: 'e6bea5c9-ec9b-41dd-9e42-29cd2784c6c0', name: '????' },
  { id: 'e2e75c17-c035-4881-9bef-25ce290d1438', name: '????' },
  { id: '28b8269c-5c34-4855-9a30-65df174ca05f', name: '?????' },
  { id: '62fbcfa4-4be6-4d81-ae5a-d12fae9f60fc', name: '??????' },
  { id: 'b883bfae-7d09-4a82-bd12-83d6ffd1644e', name: '??????' },
  { id: 'f8f66553-ef1d-4ef1-bbab-17d4ebddb601', name: '?????' },
  {
    id: 'ea948e43-096a-4fc3-aced-0db2d9665d39',
    name: '????',
    parent_id: '37f26765-90c4-49e1-88ba-b2486b3632f7',
  },
  {
    id: '1b5123e2-0450-4809-b0d1-95f44d931394',
    name: '??????',
    parent_id: '37f26765-90c4-49e1-88ba-b2486b3632f7',
  },
  {
    id: '3c310eb4-afa9-4d9d-979b-f6d99a54a141',
    name: '????',
    parent_id: '37f26765-90c4-49e1-88ba-b2486b3632f7',
  },
  {
    id: '35f0f3b7-e306-44f4-b69c-bde6f0c6ad84',
    name: '?????',
    parent_id: '37f26765-90c4-49e1-88ba-b2486b3632f7',
  },
  {
    id: 'e03ce7f8-4da5-4dbb-b0de-ad717f69758c',
    name: '?????',
    parent_id: 'ce7a39ac-f403-4750-b640-890bbcdf0dd3',
  },
  {
    id: '332d70a8-8c92-4160-9913-937fec239f98',
    name: '????',
    parent_id: 'ce7a39ac-f403-4750-b640-890bbcdf0dd3',
  },
  {
    id: 'e334cbfc-1418-4548-8560-1aed5df9f71a',
    name: '?????',
    parent_id: 'ce7a39ac-f403-4750-b640-890bbcdf0dd3',
  },
  {
    id: '8c22c56b-6f81-4450-8d5e-3ee1b2fe1a88',
    name: '????',
    parent_id: '6a7414fc-369a-49b4-a2ab-b14c95efaf97',
},

{

  id: '9b1f3382-7b35-4a56-b96b-e2232549d19e',

  name: '????',

  parent_id: '6a7414fc-369a-49b4-a2ab-b14c95efaf97',

},

{

  id: 'df4a7b6a-a56d-4a11-a075-525c885f75e1',

  name: '?????',

  parent_id: '6a7414fc-369a-49b4-a2ab-b14c95efaf97',

},

{

  id: '8a8a7f0f-0f04-459b-8083-f5d6ea836d7a',

  name: '?????',

  parent_id: '6a7414fc-369a-49b4-a2ab-b14c95efaf97',

},

{

  id: '04eb091b-f930-4386-af01-956d6fe13ade',

  name: '?????',

  parent_id: '6a7414fc-369a-49b4-a2ab-b14c95efaf97',

},

{

  id: '95a9c758-7e48-4072-bd5f-5983b8f0db58',

  name: '????',

  parent_id: 'a9649df0-2838-469c-8b40-48e14c55559a',

},

{
  id: '44adb9d9-7898-4879-bcb1-b58da01733ac',

  name: '????',

  parent_id: 'a9649df0-2838-469c-8b40-48e14c55559a',

},

{

  id: 'c59dc6bd-e13e-48c1-9739-ab3ed98aa373',

  name: '????',

  parent_id: 'a9649df0-2838-469c-8b40-48e14c55559a',

},

{

  id: '75b9162d-09fc-446d-825e-6d3c186b1c05',

  name: '?????',

  parent_id: 'a9649df0-2838-469c-8b40-48e14c55559a',

},

{

  id: '99621938-93bf-4ec1-88fa-982358c1630e',

  name: '?????',

  parent_id: 'f189e104-6614-49d7-b3fd-e68d5988844b',

},

{

  id: '520b4baa-d9b9-4c30-87b7-461109b49931',

  name: '????',

  parent_id: 'f189e104-6614-49d7-b3fd-e68d5988844b',

},

{

  id: '2b8b537e-99b4-4fd5-9fde-ce55d19e3316',

  name: '????',
  parent_id: 'f189e104-6614-49d7-b3fd-e68d5988844b',

},

{

  id: '6968c34c-3a2f-4e3e-b895-98c2e8f1626c',

  name: '????',

  parent_id: 'f189e104-6614-49d7-b3fd-e68d5988844b',

},

{

  id: '65c5a4aa-f33e-4172-81f2-ef1a0e27d0d4',

  name: '??????',

  parent_id: '379ca741-04ad-447b-8c64-f848f3185e42',

},

{

  id: '6827821e-f135-47eb-ba3b-154b093aca62',

  name: '????',

  parent_id: '379ca741-04ad-447b-8c64-f848f3185e42',

},

{

  id: 'b8272f3c-298a-47b7-a8a1-4b99f7a4ca7f',

  name: '?????',

  parent_id: '379ca741-04ad-447b-8c64-f848f3185e42',

},

{

  id: 'b2874adf-a016-4125-97e1-c0cdec20dc4c',

  name: '????',

  parent_id: '379ca741-04ad-447b-8c64-f848f3185e42',

},
```

{

  id: 'a915f2d2-0e82-4916-af2d-9bd8a79d4465',

  name: '????',

  parent_id: 'aab129c4-09d8-4b7c-ac78-5479d84acd3c',

},

{

  id: '1445c550-02b3-444c-9d19-4a29df1889de',

  name: '????',

  parent_id: 'aab129c4-09d8-4b7c-ac78-5479d84acd3c',

},

{

  id: 'f2cbc83e-ce67-4162-937a-e4e35afec218',

  name: '????',

  parent_id: 'aab129c4-09d8-4b7c-ac78-5479d84acd3c',

},

{

  id: '90dc9fa4-43bf-4b4e-81bd-6ae308a07747',

  name: '????',

  parent_id: 'aab129c4-09d8-4b7c-ac78-5479d84acd3c',

},

{

  id: 'bd0f17c2-a076-4a31-bf6f-1661e4a763b5',

  name: '???',

  parent_id: 'e6bea5c9-ec9b-41dd-9e42-29cd2784c6c0',

},

{

  id: 'b131ed29-3ca9-465a-b479-f83b87ef3218',

```
  name: '?????',
  parent_id: 'e6bea5c9-ec9b-41dd-9e42-29cd2784c6c0',
},
{
  id: '7f3b5dc1-3757-40ee-a287-2a1cf80870c8',
  name: '????',
  parent_id: 'e6bea5c9-ec9b-41dd-9e42-29cd2784c6c0',
},
{
  id: 'b2964f8f-d3e0-4838-abc5-607b294979ac',
  name: '????',
  parent_id: 'e6bea5c9-ec9b-41dd-9e42-29cd2784c6c0',
},
{
  id: '57bf4363-76b3-4fd7-95b5-2f6ba024ed3e',
  name: '???',
  parent_id: 'e2e75c17-c035-4881-9bef-25ce290d1438',
},
{
  id: '3c8d3c25-b459-41f7-ba07-669a33978e3e',
  name: '????',
  parent_id: 'e2e75c17-c035-4881-9bef-25ce290d1438',
},
{
  id: '245c0183-9a8c-47f3-af01-dccb5de27ba9',
  name: '???',
  parent_id: 'e2e75c17-c035-4881-9bef-25ce290d1438',
},

{

  id: 'a67f5d36-70eb-4b6e-a062-a98a4084b16f',

  name: '?????',

  parent_id: 'e2e75c17-c035-4881-9bef-25ce290d1438',

},

{

  id: '8186c226-6a91-4786-947a-d58da8602b60',

  name: '?????',

  parent_id: '28b8269c-5c34-4855-9a30-65df174ca05f',

},

{

  id: 'b43d06b3-683c-438e-a917-d71c914628b2',

  name: '?????',

  parent_id: '28b8269c-5c34-4855-9a30-65df174ca05f',

},

{

  id: 'b4e935db-0688-43df-b00e-4df1fa175861',

  name: '???',

  parent_id: '28b8269c-5c34-4855-9a30-65df174ca05f',

},

{

  id: '44fa4a2d-6f5c-4209-8d1d-081e019a51e6',

  name: '?????',

  parent_id: '28b8269c-5c34-4855-9a30-65df174ca05f',

},

{
  id: '52f7157a-4d6f-4ea1-a558-6f7d8096734d',

  name: '????',

  parent_id: '62fbcfa4-4be6-4d81-ae5a-d12fae9f60fc',

},

{

  id: '86664a55-83ce-486c-b8c6-19ecb1b324ad',

  name: '????',

  parent_id: '62fbcfa4-4be6-4d81-ae5a-d12fae9f60fc',

},

{

  id: '2ce2ef7c-046a-410b-b3b8-4aa2c90276f0',

  name: '????',

  parent_id: '62fbcfa4-4be6-4d81-ae5a-d12fae9f60fc',

},

{

  id: '12730e33-d413-4c1b-9fdf-0f29957d7147',

  name: '????',

  parent_id: '62fbcfa4-4be6-4d81-ae5a-d12fae9f60fc',

},

{

  id: 'e7adb8ea-7239-4560-89fc-cc727fec2899',

  name: '???',

  parent_id: '62fbcfa4-4be6-4d81-ae5a-d12fae9f60fc',

},

{

  id: '24fed77a-ca24-4093-ac33-a2bb88cfaeef',

  name: '????',
    parent_id: 'b883bfae-7d09-4a82-bd12-83d6ffd1644e',
  },
  {
    id: 'f7f4ea03-aff0-4e1a-9466-485b855fd6be',
    name: '?????',
    parent_id: 'b883bfae-7d09-4a82-bd12-83d6ffd1644e',
  },
  {
    id: 'a1e85b80-badf-4b32-9ba9-c96d534b18d6',
    name: '??',
    parent_id: 'b883bfae-7d09-4a82-bd12-83d6ffd1644e',
  },
  {
    id: '17b93710-e019-4ea1-8ef1-898a2614da4b',
    name: '??',
    parent_id: 'f8f66553-ef1d-4ef1-bbab-17d4ebddb601',
  },
  {
    id: '7d110e42-50f9-4744-bdf1-0fcea03e63cd',
    name: '????',
    parent_id: 'f8f66553-ef1d-4ef1-bbab-17d4ebddb601',
  },
];


export const parentCategoriesArray = categories.filter((category) => !category.parent_id);


export const childCategoriesArray = categories.filter((category) => category.parent_id);
export const findCategoryNameById = (id: string) => {

  const category = categories.find((category) => category.id === id);

  return category ? category.name : '';

};
```

app\post_action\comments.ts

```
// _action/comments.ts
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';


// ?? ??
export async function addComment(
  postId: string,
  userId: string,
  author: string,
  content: string,
  parentId: number | null = null
) {
  const supabase = await createSupabaseServerClient();


  // ?? ??
  const { data: commentData, error: commentError } = await supabase
    .from('comments')
    .insert([{ post_id: postId, user_id: userId, author, content, parent_id: parentId }]);


  if (commentError) {
    throw new Error(commentError.message);
  }


  // ?? ?? ? ????
  const { data: postData, error: fetchError } = await supabase
```

```
    .from('post')
    .select('comments')
    .eq('id', postId)
    .single();

  if (fetchError) {
    throw new Error('Failed to fetch post comments count');
  }

  // ?? ? ??
  const newCommentsCount = (postData.comments || 0) + 1;

  const { error: updateError } = await supabase
    .from('post')
    .update({ comments: newCommentsCount })
    .eq('id', postId);

  if (updateError) {
    throw new Error('Failed to update post comments count');
  }

  return commentData;
}

// ?? ??
export const updateComment = async (commentId: number, content: string) => {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.from('comments').update({ content }).eq('id', commentId);


  if (error) {

    throw new Error(error.message);

  }


  return data;

};


// ?? ??

export async function deleteComment(commentId: number) {

  const supabase = await createSupabaseServerClient();


  // ??? ?? ???? post_id? ??

  const { data: comment, error: fetchError } = await supabase

    .from('comments')

    .select('post_id')

    .eq('id', commentId)

    .single();


  if (fetchError) {

    throw new Error(fetchError.message);

  }


  // ?? ??

  const { data, error } = await supabase.from('comments').delete().eq('id', commentId);
```

```
if (error) {
  throw new Error(error.message);
}

// ?? ?? ? ????
const { data: postData, error: fetchPostError } = await supabase
  .from('post')
  .select('comments')
  .eq('id', comment.post_id)
  .single();

if (fetchPostError) {
  throw new Error('Failed to fetch post comments count');
}

// ?? ? ??
const newCommentsCount = (postData.comments || 0) - 1;

const { error: updateError } = await supabase
  .from('post')
  .update({ comments: newCommentsCount })
  .eq('id', comment.post_id);

if (updateError) {
  throw new Error('Failed to update post comments count');
}
```

```
  return data;
}


// ?? ????
export const fetchCommentsByPostId = async (postId: string) => {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase
    .from('comments')
    .select('*')
    .eq('post_id', postId)
    .order('created_at', { ascending: true });


  if (error) {
    throw new Error(error.message);
  }


  return data;
};
```

app\post_action\detailpost.ts

```
/* 'use server';



import createSupabaseServerClient from '@/lib/supabse/server';



export async function fetchPostById(id: string) {

  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase.from('post').select('*').eq('id', id).single();



  if (error) {

    throw new Error(error.message);

  }



  return data;

}

 */
```

app\post_action\image.ts

```
'use server';

import createSupabaseServerClient from '@/lib/supabse/server';


// ?? ???? ???? ??

function generateUniqueId() {

  return `${Date.now()}-${Math.floor(Math.random() * 100000)}`;

}


// ?? ??? ???? ???? ??

function sanitizeFileName(fileName: string): string {

  return fileName.replace(/[^a-zA-Z0-9._-]/g, '_');

}


// ??? ???

export async function uploadImageToSupabase(file: File, user_id: string) {

  const supabase = await createSupabaseServerClient();

  const uniqueSuffix = `${Date.now()}-${Math.floor(Math.random() * 100000)}`;

  const sanitizedFileName = sanitizeFileName(file.name);

  const fileName = `${user_id}/${uniqueSuffix}-${sanitizedFileName}`;

  console.log('Uploading file:', fileName); // ?? ??

  const { data, error } = await supabase.storage.from('images').upload(fileName, file);


  if (error) {

    console.error('Error uploading image:', error); // ?? ??

    throw new Error(error.message);

  }
  const url = supabase.storage.from('images').getPublicUrl(fileName).data.publicUrl;

  console.log('Uploaded file URL:', url); // ?? ??



  return { url };

}



// ??? ??

export async function deleteImageFromSupabase(url: string) {

  const supabase = await createSupabaseServerClient();

  // URL?? ??? ?? ??? ??

  const fileName = url.split('/').slice(-2).join('/'); // ??? ? ?? ?? ??? ???? ?? ?? ??

  console.log('Deleting file:', fileName); // ?? ??

  const { error } = await supabase.storage.from('images').remove([fileName]);



  if (error) {

    console.error('Error deleting image:', error); // ?? ??

    throw new Error(error.message);

  }



  console.log('Image deleted:', fileName); // ?? ??

}



// ??? ???

export async function uploadAvatarToSupabase(file: File, path: string) {

  const supabase = await createSupabaseServerClient();

  const sanitizedFileName = sanitizeFileName(file.name);
  const fileName = `${path}/${sanitizedFileName}`;
  console.log('Uploading file:', fileName); // ?? ??
  const { data, error } = await supabase.storage.from('avatars').upload(fileName, file);

  if (error) {
    console.error('Error uploading avatars:', error); // ?? ??
    throw new Error(error.message);
  }

  const url = supabase.storage.from('avatars').getPublicUrl(fileName).data.publicUrl;
  console.log('avatars Uploaded file URL:', url); // ?? ??

  return { url };
}

// ??? ??
export async function deleteAvatarFromSupabase(url: string) {
  const supabase = await createSupabaseServerClient();
  // URL?? ??? ?? ??? ??
  const fileName = url.split('/').slice(-3).join('/'); // ??? ? ?? ?? ??? ???? ?? ?? ??
  console.log('Deleting file:', fileName); // ?? ??
  const { error } = await supabase.storage.from('avatars').remove([fileName]);

  if (error) {
    console.error('Error deleting avatars:', error); // ?? ??
    throw new Error(error.message);
  }
  console.log('avatars deleted:', fileName); // ?? ??

}
```

app\post_action\incrementViews.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';


export async function incrementViews(postId: string) {
  const supabase = await createSupabaseServerClient();


  const { data, error } = await supabase.from('post').select('views').eq('id', postId).single();


  if (error) {
    throw new Error('Failed to fetch post views');
  }


  const newViews = (data.views || 0) + 1;


  const { error: updateError } = await supabase
    .from('post')
    .update({ views: newViews })
    .eq('id', postId);


  if (updateError) {
    throw new Error('Failed to update post views');
  }


  return newViews;
}
```

app\post_action\infinityScrollPost.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';

import { PostType } from '../types';


export const fetchMorePosts = async (page: number, categoryId?: string) => {

  const supabase = await createSupabaseServerClient();


  let query = supabase

    .from('post')

    .select('*')

    .order('created_at', { ascending: false })

    .range((page - 1) * 10, page * 10 - 1);


  if (categoryId) {

    query = query.eq('parent_category_id', categoryId);

  }


  const { data, error } = await query;


  if (error) {

    console.error('Error fetching posts:', error);

    return [];

  }


  return data;
};


export async function fetchSearchPosts(searchTerm: string, page: number): Promise<PostType[]> {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase
    .from('post')
    .select('*')
    .ilike('title', `%${searchTerm}%`)
    .order('created_at', { ascending: false })
    .range((page - 1) * 10, page * 10 - 1);

  if (error) {
    console.error('Error fetching search posts:', error);
    return [];
  }

  return data || [];
}
```

app\post_action\like.ts

```
// app/post/_action/likes.ts

'use server';



import createSupabaseServerClient from '@/lib/supabse/server';



// ??? ??? ????

export const fetchLikesDislikes = async (post_id: string) => {

  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase

    .from('post_likes')

    .select('liked, user_id')

    .eq('post_id', post_id);



  if (error) {

    console.error('Error fetching likes/dislikes:', error);

    throw error;

  }



  const likes = data.filter((item) => item.liked).length;

  const dislikes = data.filter((item) => !item.liked).length;

  const likeUsers = data; // ???? ?? ??? ??? ??



  //console.log('likesUsers:', likeUsers); // ?? ??



  return { likes, dislikes, likeUsers };

};
```

```
// ??? ??? ????

export const toggleLike = async (post_id: string, user_id: string, liked: boolean) => {

  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase

    .from('post_likes')

    .select('*')

    .eq('post_id', post_id)

    .eq('user_id', user_id)

    .single();



  if (error && error.code !== 'PGRST116') {

    console.error('Error fetching like/dislike:', error);

    throw error; // Ignore "Row not found" error

  }



  if (data) {

    // Toggle existing like/dislike

    const { error: updateError } = await supabase

      .from('post_likes')

      .update({ liked })

      .eq('id', data.id);



    if (updateError) {

      console.error('Error updating like/dislike:', updateError);

      throw updateError;

    }
  } else {
    // Insert new like/dislike
    const { error: insertError } = await supabase
      .from('post_likes')
      .insert({ post_id, user_id, liked });

    if (insertError) {
      console.error('Error inserting like/dislike:', insertError);
      throw insertError;
    }
  }
};
```

app\post_action\logReadPost.ts

```
'use server';



import createSupabaseServerClient from '@/lib/supabse/server';



export async function batchLogReadPosts(readPosts: any[]) {

  console.log('batchLogReadPosts', readPosts);

  const supabase = await createSupabaseServerClient();



  const formattedReadPosts = readPosts.map((post) => ({

    user_id: post.userId,

    post_id: post.postId,

    ip_address: post.ipAddress,

    timestamp: post.timestamp, // assuming this field exists in your table schema

  }));



  const { data, error } = await supabase.from('read_posts').insert(formattedReadPosts);



  if (error) {

    console.error('Error logging read posts:', error);

  }

}
```

app\post_action\post.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';

import { revalidatePath } from 'next/cache';

import { redirect } from 'next/navigation';

import { uploadImageToSupabase, deleteImageFromSupabase } from './image';


export async function createPost(formData: FormData) {

  const supabase = await createSupabaseServerClient();


  // FormData?? ?? ? ??

  const title = formData.get('title') as string;

  const content = formData.get('content') as string;

  const user_id = formData.get('user_id') as string;

  const user_name = formData.get('user_name') as string;

  const user_avatar_url = formData.get('user_avatar_url') as string;

  const user_email = formData.get('user_email') as string;

  const linkUrl1 = formData.get('linkUrl1') as string;

  const linkUrl2 = formData.get('linkUrl2') as string;

  const parent_category_id = formData.get('parent_category_id') as string;

  const child_category_id = formData.get('child_category_id') as string;


  // FormData?? ??? ?? ????

  const imageFiles = formData.getAll('images') as File[];


  // ??? ???
```

```
let imageUrls: string[] = [];

if (imageFiles.length > 0) {

  const validImageFiles = imageFiles.filter((image) => image.size > 0);



  if (validImageFiles.length > 0) {

    const uploadPromises = validImageFiles.map((image) => uploadImageToSupabase(image, user_id));

    const uploadResults = await Promise.all(uploadPromises);

    imageUrls = uploadResults.map((result) => result.url);

  }

}



// ??????? ??? ?? ??

const post = {

  title: title,

  content: content,

  author_id: user_id,

  author_name: user_name,

  author_avatar_url: user_avatar_url,

  author_email: user_email,

  linkUrl1: linkUrl1,

  linkUrl2: linkUrl2,

  images: imageUrls.length > 0 ? imageUrls : null,

  parent_category_id: parent_category_id || null,

  child_category_id: child_category_id || null,

};



// Supabase? ??? ??
```

```
  const { data, error } = await supabase.from('post').insert([post]);

  if (error) {
    throw new Error(error.message);
  } else {
    revalidatePath('/post');
    redirect('/post');
  }
}

export async function fetchPostById(post_id: string) {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.from('post').select('*').eq('id', post_id).single();

  if (error) {
    throw new Error(error.message);
  }

  return data;
}

export async function updatePostById(formData: FormData, id: string, user_id: string) {
  console.log('formData:', formData); // ?? ??
  const supabase = await createSupabaseServerClient();

  // FormData?? ?? ? ??
  const title = formData.get('title') as string;
const content = formData.get('content') as string;

const linkUrl1 = formData.get('linkUrl1') as string;

const linkUrl2 = formData.get('linkUrl2') as string;

const deletedImages = JSON.parse(formData.get('deletedImages') as string) as string[];

console.log('Updating post for id:', id); // ?? ??


// ?? ???? ???? ????

const { data: existingPost, error: fetchError } = await supabase

  .from('post')

  .select('images')

  .eq('id', id)

  .single();


if (fetchError) {

  console.error('Error fetching existing post:', fetchError); // ?? ??

  throw new Error(fetchError.message);

}


// ?? ????? ??? ???? ???? ?????? ??

const remainingImages = (existingPost.images || []).filter((url: string) => {

  const shouldDelete = deletedImages.includes(url);

  if (shouldDelete) {

    deleteImageFromSupabase(url); // ?????? ?? ??

  }

  return !shouldDelete;

});
```

```
// FormData?? ??? ?? ????
const imageFiles = formData.getAll('images') as File[];
console.log('Updating imageFiles:', imageFiles); // ?? ??

// ??? ???
let newImageUrls: string[] = [];
if (imageFiles.length > 0) {
  // ?? ??? ??? 0? ?? ??? ???? ???
  const validImageFiles = imageFiles.filter((image) => image.size > 0);

  if (validImageFiles.length > 0) {
    const uploadPromises = validImageFiles.map((image) => uploadImageToSupabase(image, user_id));
    const uploadResults = await Promise.all(uploadPromises);
    newImageUrls = uploadResults.map((result) => result.url);
    newImageUrls.forEach((url) => console.log('Image URL:', url)); // ?? ??
  } else {
    console.log('There are no valid image files to upload.');
  }
} else {
  console.log('No image files provided.');
}

// ????? ?? ??
const allImageUrls = [...remainingImages, ...newImageUrls];
const updates = {
  title: title,
  content: content,
    linkUrl1: linkUrl1,
    linkUrl2: linkUrl2,
    images: allImageUrls.length > 0 ? allImageUrls : null, // ??? URL ??
  };

  // Supabase? ??? ????
  const { data, error } = await supabase.from('post').update(updates).eq('id', id);

  if (error) {
    console.error('Error updating post:', error); // ?? ??
    throw new Error(error.message);
  } else {
    console.log('Post updated successfully:', data); // ?? ??
    revalidatePath('/post');
    redirect('/post');
  }

  return data;
}

export async function deletePostById(id: string) {
  const supabase = await createSupabaseServerClient();

  // ???? ??? ???? ?? ?????
  const { data: post, error: fetchError } = await supabase
    .from('post')
    .select('images')
    .eq('id', id)
    .single();

  if (fetchError) {
    console.error('Error fetching post:', fetchError); // ?? ??
    throw new Error(fetchError.message);
  }

  // ???? ?? ?? ?????? ?????
  if (post?.images) {
    const deletePromises = post.images.map((url: string) => deleteImageFromSupabase(url));
    await Promise.all(deletePromises);
  }

  // ???? ?????
  const { data, error } = await supabase.from('post').delete().eq('id', id);

  if (error) {
    throw new Error(error.message);
  } else {
    console.log('Post deleted successfully:', data);
    revalidatePath('/post');
    redirect('/post');
  }

  return data;
}
```

app\post_action\postData.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';


export const getUserViewedPosts = async (userId: string) => {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.from('views').select('post_id').eq('user_id', userId);


  if (error) {
    console.error('Error fetching viewed posts:', error);
    return [];
  }


  return data.map((view: { post_id: string }) => view.post_id);
};


export const getPostsData = async (limit: number = 10) => {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase.from('post').select('*').limit(limit);


  if (error) {
    console.error('Error fetching posts:', error);
    return [];
  }


  return data.map((post: any) => ({
    ...post,

    created_at: post.created_at ? post.created_at.toISOString() : null,

  }));

};
```

app\post_action\readPostAction.ts

```
'use server';



import { getCurrentUserInfo } from '@/lib/cookies';

import createSupabaseServerClient from '@/lib/supabse/server';



export async function recordPostRead(postId: string, ipAddress: string) {

  const supabase = await createSupabaseServerClient();

  const currentUser = await getCurrentUserInfo();



  const userId = currentUser ? currentUser.id : null;



  const { error } = await supabase

    .from('read_posts')

    .insert([{ user_id: userId, post_id: postId, ip_address: ipAddress }]);



  if (error) {

    console.error('Error recording post read:', error);

  }

}
```

app\post_action\rewardUtils.ts

```
export function generateRewards(cycleLength, totalPoints, bigRewardThreshold) {
  let rewards = Array(cycleLength).fill(0);
  let bigRewardIndex = Math.floor(Math.random() * cycleLength);
  let bigReward =
    Math.floor(Math.random() * (totalPoints - (cycleLength - 1) - bigRewardThreshold + 1)) +
    bigRewardThreshold;

  rewards[bigRewardIndex] = bigReward;
  let remainingPoints = totalPoints - bigReward;

  for (let i = 0; i < cycleLength; i++) {
    if (i !== bigRewardIndex) {
      let maxPossible = Math.min(remainingPoints, Math.floor(totalPoints / cycleLength));
      rewards[i] = Math.floor(Math.random() * (maxPossible + 1));
      remainingPoints -= rewards[i];
    }
  }

  let remainingIndices = rewards.reduce((indices, reward, index) => {
    if (index !== bigRewardIndex) indices.push(index);
    return indices;
  }, []);

  if (remainingPoints > 0) {
    let randomIndex = remainingIndices[Math.floor(Math.random() * remainingIndices.length)];
    rewards[randomIndex] += remainingPoints;
  }


  return rewards;

}
```

app\post_action\search.ts

```
'use server';


import createSupabaseServerClient from '@/lib/supabse/server';


export async function searchPosts(searchTerm: string) {
  const supabase = await createSupabaseServerClient();
  const { data, error } = await supabase
    .from('post')
    .select('*')
    .ilike('title', `%${searchTerm}%`)
    .limit(10);


  if (error) {
    throw new Error(error.message);
  }


  return data;
}
```

app\post_action\view.ts

```
// lib/actions/view.ts

import createSupabaseServerClient from '@/lib/supabse/server';

export const incrementViewCount = async (postId: string, userId: string | null) => {
  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase.from('views').insert({ post_id: postId, user_id: userId });

  if (error) {
    console.error('Error incrementing view count:', error);
  }

  return data;
};

export const getViewCount = async (postId: string) => {
  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase
    .from('views')
    .select('id', { count: 'exact' })
    .eq('post_id', postId);

  if (error) {
    console.error('Error fetching view count:', error);
  }


  return data ? data.length : 0;

};
```

app\post_component\AdAlert.tsx

```
'use client';


import { useEffect, useState, useCallback } from 'react';

import { AlertDialog } from '@/components/ui/alert-dialog'; // shadcn alert ????

import { addUserPoints } from '../_action/adPointSupabase';

import { PointAnimation } from './PointAnimation';

import { DolphinBasic } from '@/app/_items/dolphin_item';

import { BiMove } from 'react-icons/bi'; // ??? ??? ??


const SHOW_AD_FOR_READ_POSTS = true; // ?? ??? true?? ??? ???? ?? ???? ??? ???


export default function AdAlert({

  userId,

  postId,

  initialRoundData,

}: {

  userId: string | null;

  postId: string | null;

  initialRoundData: any;

}) {

  const [showAd, setShowAd] = useState(false);

  const [showAnimation, setShowAnimation] = useState(false);

  const [animationExecuted, setAnimationExecuted] = useState(false);

  const [touchStartX, setTouchStartX] = useState<number | null>(null);

  const [touchCurrentX, setTouchCurrentX] = useState<number | null>(null);

  const [mouseStartX, setMouseStartX] = useState<number | null>(null);
const [mouseCurrentX, setMouseCurrentX] = useState<number | null>(null);

const [isDragging, setIsDragging] = useState(false);


const handleAdClose = () => setShowAd(false);


const handleAnimationEnd = useCallback(

  (points: number) => {

    console.log('handleAnimationEnd called with points:', points);

    if (points > 0 && !animationExecuted) {

      setShowAd(true);

      setAnimationExecuted(true);

      if (userId) {

        addUserPoints(userId, points);

        console.log('Adding points client:', points);

      }

    }

  },

  [animationExecuted, userId]

);


useEffect(() => {

  console.log('useEffect called for postId:', postId);

  const animationExecuted = localStorage.getItem(`post_${postId}_animation_executed`);

  if (SHOW_AD_FOR_READ_POSTS || !animationExecuted) {

    setShowAnimation(true);

    setTimeout(() => {

      console.log('Setting animation executed in localStorage');
      localStorage.setItem(`post_${postId}_animation_executed`, 'true');
      setAnimationExecuted(true);
    }, 1500);
  }
}, [postId]);

const handleTouchStart = (e: React.TouchEvent) => {
  setTouchStartX(e.touches[0].clientX);
};

const handleTouchMove = (e: React.TouchEvent) => {
  if (touchStartX !== null) {
    const touchEndX = e.touches[0].clientX;
    setTouchCurrentX(touchEndX);

    if (Math.abs(touchStartX - touchEndX) > 100) {
      handleAdClose();
    }
  }
};

const handleTouchEnd = () => {
  setTouchStartX(null);
  setTouchCurrentX(null);
};

const handleMouseDown = (e: React.MouseEvent) => {
  setMouseStartX(e.clientX);
  setIsDragging(true);
};


const handleMouseMove = (e: React.MouseEvent) => {
  if (isDragging && mouseStartX !== null) {
    const mouseEndX = e.clientX;
    setMouseCurrentX(mouseEndX);


    if (Math.abs(mouseStartX - mouseEndX) > 100) {
      handleAdClose();
    }
  }
};


const handleMouseUp = () => {
  setMouseStartX(null);
  setMouseCurrentX(null);
  setIsDragging(false);
};


const handleMouseLeave = () => {
  setMouseStartX(null);
  setMouseCurrentX(null);
  setIsDragging(false);
};
```

```
const getTransformStyle = () => {
  let transformX = 0;


  if (touchStartX !== null && touchCurrentX !== null) {
    transformX = touchCurrentX - touchStartX;
  } else if (mouseStartX !== null && mouseCurrentX !== null) {
    transformX = mouseCurrentX - mouseStartX;
  }


  return `translate(-50%, -50%) translateX(${transformX}px)`;
};


return (
  <>
    {showAnimation && (
      <PointAnimation
        userId={userId}
        initialRoundData={initialRoundData}
        onAnimationEnd={handleAnimationEnd}
      />
    )}
    {showAd && (
      <>
        <div className="w-screen h-screen bg-black bg-opacity-50 z-40 relative" />
        <AlertDialog open={showAd} onOpenChange={handleAdClose}>
          <div
                className={`fixed top-1/2 left-1/2 p-4 w-80 h-80 bg-white z-50 ${isDragging ? 'cursor-grabbing' :
```

```
'cursor-grab'}`}
              style={{ transform: getTransformStyle() }}
              onTouchStart={handleTouchStart}
              onTouchMove={handleTouchMove}
              onTouchEnd={handleTouchEnd}
              onMouseDown={handleMouseDown}
              onMouseMove={handleMouseMove}
              onMouseUp={handleMouseUp}
              onMouseLeave={handleMouseLeave}
            >
              <div className="flex justify-between items-center">
                <h2 className="text-lg font-semibold">??</h2>
                <BiMove size={24} className="text-gray-500" />
              </div>
              <p>??? ?? ??? ????.</p>
              <DolphinBasic width={100} color="text-red-300" />
              <button onClick={() => (window.location.href = 'https://www.google.com')}>
                ?? ??
              </button>
              <button onClick={handleAdClose}>??</button>
            </div>
          </AlertDialog>
        </>
      )}
    </>
  );
}
```

app\post_component\Advertisement.tsx

```
import { EmojiYay } from '@/app/_emoji-gather/emoji-gather';

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



const Advertisement = ({ onButtonClick }: { onButtonClick: () => void }) => (

  <div className="my-4 flex flex-col gap-2 border-y-[1px] border-gray-200 py-4">

    <Button

      className="flex p-4 gap-4 items-center  mx-auto cursor-pointer  rounded-lg"

      onClick={onButtonClick}

    >

      {/* <EmojiYay size="36px" /> */}

      <p className=" font-semibold ">

        ??? ????

        <br />

        ??? ?? ???!

      </p>

      {/* <EmojiYay size="36px" /> */}

    </Button>

    <p className="text-gray-600 text-right mr-4">ad</p>

    <img src="/ad-nike1.jpg" alt="" className="w-full aspect-video object-cover " />

    <Button className="mx-8" onClick={onButtonClick}>

      ???? ??? ??

    </Button>

  </div>

);



export default Advertisement;
```

app\post_component\AuthorHoverCard.tsx

```
'use client';


import { useState } from 'react';

import { HoverCard, HoverCardContent, HoverCardTrigger } from '@/components/ui/hover-card';

import Link from 'next/link';

import { AiOutlineClose } from 'react-icons/ai';


interface AuthorHoverCardProps {

  author_avatar_url: string;

  author_name: string;

  author_id: string;

  triggerElement: React.ReactNode;

  point: number;

}


export default function AuthorHoverCardElement({

  author_avatar_url,

  author_name,

  author_id,

  triggerElement,

  point,

}: AuthorHoverCardProps) {

  const [isOpen, setIsOpen] = useState(false);


  const handleTouch = (e: React.TouchEvent) => {

    e.preventDefault(); // Prevent the default behavior
    setIsOpen(!isOpen);
  };


  const handleClose = () => {
    setIsOpen(false);
  };


  return (
    <HoverCard open={isOpen} onOpenChange={setIsOpen}>
      <HoverCardTrigger asChild>
        <div onTouchStart={handleTouch}>{triggerElement}</div>
      </HoverCardTrigger>
      {isOpen && (
        <div className="fixed inset-0 flex items-center justify-center z-50">
          <div className="relative w-80 bg-white shadow-lg p-4 rounded-lg">
            <div className="absolute top-2 right-2">
              <button onClick={handleClose} className="text-gray-500 hover:text-gray-700">
                <AiOutlineClose size={20} />
              </button>
            </div>
            <div className="flex flex-col justify-between space-y-4">
              <img
                src={author_avatar_url || '/money-3d-main.png'}
                alt="avatar_image"
                className="w-60 h-60 ring-1 rounded-full object-cover mx-auto mt-4"
              />
              <div className="flex flex-col justify-center space-y-1 py-4 border-y-[1px] border-gray-200">
```

```
                <h4 className="text font-semibold pb-2 ">{author_name}</h4>

                <p className="text-sm">Point : {point}</p>

                <p className="text-sm">Trophy : ??? ?? ? 999,999,999</p>

                <p className="text-sm">Support : ??? ? 999</p>

              </div>

              <Link href={`/profile/${author_id}`}>

                <p className="text-sm font-semibold">???...</p>

              </Link>

            </div>

          </div>

        </div>

      )}

    </HoverCard>

  );

}
```

app\post_component\BackButtonHandler.tsx

```
'use client';


import { useRouter } from 'next/navigation';

import { useEffect } from 'react';


export default function BackButtonHandler() {

  const router = useRouter();


  useEffect(() => {

    const handlePopState = () => {

      router.replace('/post');

    };


    window.addEventListener('popstate', handlePopState);


    return () => {

      window.removeEventListener('popstate', handlePopState);

    };

  }, [router]);


  return null;

}
```

app\post_component\Breadcrumbs.tsx

```
const Breadcrumbs = ({ category }: { category: string }) => (

  <div className="flex gap-1 items-center h-10 border-b-[1px] border-gray-200 ">

    <p className="leading-tight text-sm text-gray-600">?? {'>'}</p>

    <p className="leading-tight text-sm font-bold text-gray-600">{category || '?????'}</p>

  </div>

);



export default Breadcrumbs;
```

app\post_component\ClientPostDetail.tsx

```
// _component/ClientPostDetail.tsx

'use client';



import { useEffect, useState } from 'react';

import { useRouter } from 'next/navigation';

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

import PostLikeDislike from './PostLikeDislike';

import PostActions from './PostActions';

import Comments from './Comments';

import CommentDialog from './CommentDialog';

import Advertisement from './Advertisement';

//import { PointAnimation } from './PointAnimation';

import { incrementViews } from '../_action/incrementViews';

import { deletePostById } from '../_action/post';

import {

  fetchCommentsByPostId,

  addComment,

  updateComment,

  deleteComment,

} from '../_action/comments';

import { CurrentUser } from '@/lib/cookies';

import { EmojiYay } from '@/app/_emoji-gather/emoji-gather';



interface Comment {

  id: number;

  author: string;
  content: string;

  user_id: string;

  parent_id: number | null; // Add parent_id here

  children?: Comment[]; // Optional, to handle nested comments

}


interface PostDetailPageProps {

  postId: string;

  initialUser: CurrentUser | null;

  initialPost: any;

}


export default function ClientPostDetail({

  postId,

  initialUser,

  initialPost,

}: PostDetailPageProps) {

  const router = useRouter();

  const [showNumber, setShowNumber] = useState(false);

  const [comments, setComments] = useState<Comment[]>([]);

  const [isDialogOpen, setIsDialogOpen] = useState(false);

  const [editingComment, setEditingComment] = useState<{ id: number; content: string } | null>(

    null

  );

  const [views, setViews] = useState<number>(initialPost.views || 0);

  const [replyTo, setReplyTo] = useState<number | undefined>(undefined);

  const [user_id, setUser_id] = useState<string | null>(initialUser?.id || null);
  const [currentUser, setCurrentUser] = useState<CurrentUser | null>(initialUser);


  useEffect(() => {
    const updateViews = async () => {
      try {
        const newViews = await incrementViews(postId);
        setViews(newViews);
      } catch (error) {
        console.error('Failed to update views:', error);
      }
    };


    updateViews();
  }, [postId]);


  useEffect(() => {
    const getComments = async () => {
      try {
        const data = await fetchCommentsByPostId(postId);
        setComments(data);
      } catch (error) {
        console.error('Error fetching comments:', error);
      }
    };


    getComments();
  }, [postId]);
```

```
const handleDeleteConfirm = async () => {
  if (user_id === initialPost?.author_id) {
    try {
      await deletePostById(postId);
      alert('???? ????? ???????.');
      router.push('/post');
    } catch (error) {
      console.error('??? ?? ? ?? ??:', error);
      alert('??? ?? ? ??? ??????.');
    }
  } else {
    alert('? ???? ??? ??? ????.');
  }
};


const handleDialogSubmit = async (content: string, parentId: number | null = null) => {
  if (!user_id) {
    console.error('User ID is null');
    return;
  }


  const authorName = currentUser?.username || currentUser?.email || 'Unknown';


  try {
    if (editingComment) {
      await updateComment(editingComment.id, content);
    } else {
      await addComment(postId, user_id, authorName, content, parentId);
    }
    const data = await fetchCommentsByPostId(postId);
    setComments(data);
    setEditingComment(null);
    setReplyTo(undefined);
  } catch (error) {
    console.error('Error submitting comment:', error);
  }
};

const handleDeleteComment = async (commentId: number) => {
  try {
    await deleteComment(commentId);
    const data = await fetchCommentsByPostId(postId);
    setComments(data);
  } catch (error) {
    console.error('Error deleting comment:', error);
  }
};

const openEditDialog = (comment: { id: number; content: string }) => {
  setEditingComment(comment);
  setIsDialogOpen(true);
};
```

```
const openReplyDialog = (commentId: number) => {
  setReplyTo(commentId);
  setIsDialogOpen(true);
};


return (
  <div className="flex flex-col gap-4 ">
    {/* ????&???? */}
    {user_id === initialPost?.author_id && (
      <PostActions id={postId} onConfirmDelete={handleDeleteConfirm} />
    )}
    <hr />
    {/* ??? ??? */}
    <PostLikeDislike post_id={postId} user_id={user_id ?? ''} />
    {/* ??? ?? */}
    <hr className="mb-8" />
    {/* ?? */}
    <div className="flex flex-col pt-4 border-t-[1px] border-gray-200">
      <Comments
        comments={comments}
        onDelete={handleDeleteComment}
        onEdit={openEditDialog}
        onReply={openReplyDialog}
        currentUser={
          currentUser || {
            id: '',
            username: null,
            email: '',
            avatar_url: null,
            current_points: 0,
          }
        }
      />
      <Button variant="outline" className="mt-4 shadow-md" onClick={() => setIsDialogOpen(true)}>
        ????
      </Button>
    </div>

    <Advertisement onButtonClick={() => setShowNumber(true)} />
    {/* <PointAnimation showNumber={showNumber} setShowNumber={setShowNumber} userId={user_id} /> */}
    <CommentDialog
      isOpen={isDialogOpen}
      onClose={() => {
        setIsDialogOpen(false);
        setReplyTo(undefined);
      }}
      onSubmit={handleDialogSubmit}
      initialContent={editingComment ? editingComment.content : ''}
      isEdit={!!editingComment}
      parentId={replyTo}
    />
  </div>
 );
}
```

app\post_component\ClientPostDetailPage.tsx

```
// app/_component/ClientPostDetailPage.tsx


'use client';


import { useEffect, useState } from 'react';

import { PointAnimation } from './PointAnimation';


export default function ClientPostDetailPage({ postId }: { postId: string }) {

  const [showAnimation, setShowAnimation] = useState(false);

  const [initialRoundData, setInitialRoundData] = useState<any>(null);

  const userId = 'user_id_placeholder'; // ?? ??? ID? ??


  useEffect(() => {

    const animationExecuted = !!localStorage.getItem(`post_${postId}_animation_executed`);

    if (!animationExecuted) {

      setShowAnimation(true);

    }

  }, [postId]);


  useEffect(() => {

    async function fetchRoundData() {

      const response = await fetch(`/api/get-round-data?userId=${userId}`);

      const data = await response.json();

      setInitialRoundData(data);

      console.log('initial round data', initialRoundData);

    }
    fetchRoundData();
  }, [userId]);

  if (!showAnimation || !initialRoundData) return null;

  return <PointAnimation userId={userId} initialRoundData={initialRoundData} />;
}
```

app\post_component\CommentDialog.tsx

```
'use client';


// components/CommentDialog.tsx

import { useState, useEffect } from 'react';

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

import {

  Dialog,

  DialogContent,

  DialogDescription,

  DialogFooter,

  DialogHeader,

  DialogTitle,

  DialogTrigger,

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

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

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


interface CommentDialogProps {

  isOpen: boolean;

  onClose: () => void;

  onSubmit: (content: string, parentId?: number) => void;

  initialContent?: string;

  isEdit?: boolean;

  parentId?: number;

}
```

```
const CommentDialog: React.FC<CommentDialogProps> = ({
  isOpen,
  onClose,
  onSubmit,
  initialContent = '',
  isEdit = false,
  parentId,
}) => {
  const [content, setContent] = useState(initialContent);

  useEffect(() => {
    setContent(initialContent);
  }, [initialContent]);

  const handleSubmit = () => {
    onSubmit(content, parentId);
    setContent('');
    onClose();
  };

  return (
    <Dialog open={isOpen} onOpenChange={onClose}>
      <DialogContent className="sm:max-w-[425px]">
        <DialogHeader>
          <DialogTitle>{isEdit ? 'Edit Comment' : 'Add Comment'}</DialogTitle>
          <DialogDescription>
            {isEdit ? 'Edit your comment below.' : 'Add your comment below.'}
          </DialogDescription>
        </DialogHeader>
        <div className="grid gap-4 py-4">
          <div className="grid grid-cols-4 items-center gap-4">
            <Label htmlFor="comment" className="text-right">
              Comment
            </Label>
            <Input
              id="comment"
              value={content}
              onChange={(e) => setContent(e.target.value)}
              className="col-span-3"
            />
          </div>
        </div>
        <DialogFooter>
          <Button type="submit" onClick={handleSubmit}>
            {isEdit ? 'Save changes' : 'Add Comment'}
          </Button>
        </DialogFooter>
      </DialogContent>
    </Dialog>
  );
};

export default CommentDialog;
```

app\post_component\Comments.tsx

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

import { CurrentUser } from '@/lib/cookies';

import IconToggle from './IconToggleComments';

import React from 'react';



interface Comment {

  id: number;

  author: string;

  content: string;

  user_id: string;

  parent_id: number | null;

  children?: Comment[];

}



interface CommentsProps {

  comments: Comment[];

  onDelete: (commentId: number) => void;

  onEdit: (comment: { id: number; content: string }) => void;

  onReply: (commentId: number) => void;

  currentUser: CurrentUser;

}



function buildCommentTree(comments: Comment[]): Comment[] {

  const commentMap: { [key: number]: Comment } = {};

  const roots: Comment[] = [];
  comments.forEach((comment) => {
    comment.children = [];
    commentMap[comment.id] = comment;

    if (comment.parent_id === null) {
      roots.push(comment);
    } else {
      const parentComment = commentMap[comment.parent_id];
      if (parentComment) {
        parentComment.children!.push(comment);
      }
    }
  });

  return roots;
}

const Comments: React.FC<CommentsProps> = ({
  comments,
  onDelete,
  onEdit,
  onReply,
  currentUser,
}) => {
  const renderComments = (comments: Comment[]) => {
    return comments.map((comment, index) => (
      <React.Fragment key={comment.id}>
      <div className="my-4 flex flex-col">
        <div className="flex">
          <div className="flex-1 flex flex-col items-start">
            <div className="font-semibold">{comment.author}</div>
            <div>{comment.content}</div>
          </div>
          <IconToggle
            onEdit={() => onEdit({ id: comment.id, content: comment.content })}
            onDelete={() => onDelete(comment.id)}
            onReply={() => onReply(comment.id)}
            showEditDelete={comment.user_id === currentUser.id}
          />
        </div>
        {comment.children && comment.children.length > 0 && (
          <div className="ml-4 mt-2">{renderComments(comment.children)}</div>
        )}
      </div>
      {index < comments.length - 1 && <hr className="border-t border-gray-200 my-2" />}
    </React.Fragment>
  ));
};

const commentTree = buildCommentTree(comments);

return (
  <div>
    {commentTree.length === 0 ? (
        <p className="text-gray-600 mb-4">?? {comments.length}</p>
      ) : (
        <>
          <p className="text-gray-600 mb-4">?? {comments.length}</p>
          <hr className="border-t border-gray-200 mb-4" />
          {renderComments(commentTree)}
        </>
      )}
    </div>
  );
};

export default Comments;
```

app\post_component\EditForm.tsx

```
'use client';



import { useEffect, useState, FormEvent, ChangeEvent, useRef } from 'react';

import { useParams, useRouter } from 'next/navigation';

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

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

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

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

import { fetchPostById, updatePostById } from '../_action/post';

import { UserPropsType } from './PostForm';



export default function EditForm({ user_id, user_name, user_avatar_url }: UserPropsType) {

  const { id } = useParams() as { id: string };

  const router = useRouter();



  const [post, setPost] = useState<any>(null);

  const [title, setTitle] = useState('');

  const [content, setContent] = useState('');

  const [linkUrl1, setLinkUrl1] = useState('');

  const [linkUrl2, setLinkUrl2] = useState('');

  const [imagePreviews, setImagePreviews] = useState<string[]>([]);

  const [images, setImages] = useState<File[]>([]);

  const [deletedImages, setDeletedImages] = useState<string[]>([]);

  const fileInputRef = useRef<HTMLInputElement>(null);



  useEffect(() => {
    const fetchPost = async () => {

      const data = await fetchPostById(id);

      console.log('editData', data);

      setPost(data);

      setTitle(data.title);

      setContent(data.content);

      setLinkUrl1(data.linkUrl1 || '');

      setLinkUrl2(data.linkUrl2 || '');

      setImagePreviews(data.images || []);

    };


    fetchPost();

  }, [id]);


  const handleImageChange = (e: ChangeEvent<HTMLInputElement>) => {

    const files = e.target.files;

    if (files && files.length > 0) {

      const fileArray = Array.from(files).slice(0, 5); // ?? 5? ???

      const validImages = fileArray.filter(

        (file) => file.size > 0 && file.type.startsWith('image/')

      );

      setImages((prevImages) => [...prevImages, ...validImages]);

      setImagePreviews((prevPreviews) => [

        ...prevPreviews,

        ...validImages.map((file) => URL.createObjectURL(file)),

      ]);
    // ?? ??? ???? ?? ??? ?????.
    if (fileInputRef.current) {
      fileInputRef.current.value = '';
    }
  }
};

const handleImageRemove = (index: number, isExisting: boolean) => {
  if (isExisting) {
    setDeletedImages((prev) => [...prev, imagePreviews[index]]);
    setImagePreviews((prevPreviews) => prevPreviews.filter((_, i) => i !== index));
  } else {
    setImages((prevImages) => prevImages.filter((_, i) => i !== index));
    setImagePreviews((prevPreviews) => prevPreviews.filter((_, i) => i !== index));
  }
};

const editPost = async (event: FormEvent) => {
  event.preventDefault();
  if (user_id !== post.author_id) {
    console.log('You are not authorized to edit this post.');
    return;
  }

  const formData = new FormData();
  formData.append('title', title);
  formData.append('content', content);
  formData.append('linkUrl1', linkUrl1);

  formData.append('linkUrl2', linkUrl2);



  images.forEach((image) => {

    if (image.size > 0) {

      formData.append('images', image);

    }

  });



  formData.append('deletedImages', JSON.stringify(deletedImages));



  await updatePostById(formData, id, user_id);



  console.log('Post updated');

  router.push('/post');

};



if (!post) {

  return <p>Loading...</p>;

}



return (

  <div className="mt-4 mx-auto px-6">

    <h1 className="text-2xl font-bold mb-4 text-gray-800">??? ?? ???</h1>

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

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

        <Label htmlFor="title">Title</Label>
        <Input
          type="text"
          name="title"
          id="title"
          value={title}
          onChange={(e) => setTitle(e.target.value)}
          required
        />
      </div>
      <div className="space-y-2">
        <Label htmlFor="content">Content</Label>
        <Textarea
          name="content"
          id="content"
          value={content}
          onChange={(e) => setContent(e.target.value)}
          required
        />
      </div>
      <div className="space-y-2">
        <Label htmlFor="images">Images (?? 5?)</Label>
        <Input
          type="file"
          name="images"
          id="images"
          multiple
          accept="image/*"
```

```
      onChange={handleImageChange}
      ref={fileInputRef}
    />
  </div>
  <div className="flex space-x-2">
    {imagePreviews.map((src, index) => (
      <div key={index} className="relative w-24 h-24">
        <img src={src} alt={`Preview ${index + 1}`} className="w-full h-full object-cover" />
        <button
          type="button"
          onClick={() => handleImageRemove(index, index < post.images.length)}
          className="absolute top-0 right-0 w-6 h-6 bg-red-500 text-white rounded-full flex items-center justify-center"
        >
          ×
        </button>
      </div>
    ))}
  </div>
  <div className="space-y-2">
    <Label htmlFor="linkUrl1">?? Url #1</Label>
    <Input
      type="text"
      name="linkUrl1"
      id="linkUrl1"
      value={linkUrl1}
      onChange={(e) => setLinkUrl1(e.target.value)}
    />
```

```
          <Label htmlFor="linkUrl2">?? Url #2</Label>

          <Input

            type="text"

            name="linkUrl2"

            id="linkUrl2"

            value={linkUrl2}

            onChange={(e) => setLinkUrl2(e.target.value)}

          />

        </div>

        <Button type="submit">Edit Post</Button>

      </form>

    </div>

  );

}
```

app\post_component\ExternalLinksComponent.tsx

```
// components/ExternalLinks.tsx

'use client';



import Link from 'next/link';

import { useState, useEffect } from 'react';



interface ExternalLinksProps {

  linkUrl1?: string;

  linkUrl2?: string;

}



const isYouTubeUrl = (url: string) => {

  const youtubeRegex = /^https?:\/\/(www\.)?(youtube\.com|youtu\.be)\/.+$/;

  return youtubeRegex.test(url);

};



const getYouTubeEmbedUrl = (url: string) => {

  const videoIdMatch = url.match(/[?&]v=([^&]+)/) || url.match(/youtu\.be\/([^?]+)/);

  const videoId = videoIdMatch ? videoIdMatch[1] : null;

  return videoId ? `https://www.youtube-nocookie.com/embed/${videoId}` : null;

};



export default function ExternalLinks({ linkUrl1, linkUrl2 }: ExternalLinksProps) {

  const [iframeSrc1, setIframeSrc1] = useState<string | null>(null);

  const [iframeSrc2, setIframeSrc2] = useState<string | null>(null);

  const [isYouTube1, setIsYouTube1] = useState<boolean>(false);
const [isYouTube2, setIsYouTube2] = useState<boolean>(false);


useEffect(() => {
  if (linkUrl1) {
    const embedUrl1 = isYouTubeUrl(linkUrl1) ? getYouTubeEmbedUrl(linkUrl1) : linkUrl1;
    setIframeSrc1(embedUrl1);
    setIsYouTube1(isYouTubeUrl(linkUrl1));
  }


  if (linkUrl2) {
    const embedUrl2 = isYouTubeUrl(linkUrl2) ? getYouTubeEmbedUrl(linkUrl2) : linkUrl2;
    setIframeSrc2(embedUrl2);
    setIsYouTube2(isYouTubeUrl(linkUrl2));
  }
}, [linkUrl1, linkUrl2]);


return (
  <div className="flex flex-col gap-6">
    {iframeSrc1 && (
      <div className="flex flex-col mb-4">
        <div
          className=""
          style={{
            width: '100vw',
            paddingTop: isYouTube1 ? '56.25%' : '0',
            height: isYouTube1 ? '0' : '640px',
            marginLeft: '-50vw',
    left: '50%',
    position: 'relative',
  }}
>
  <iframe
    src={iframeSrc1}
    style={{
      width: '100%',
      height: isYouTube1 ? '100%' : '640px',
      position: isYouTube1 ? 'absolute' : 'static',
      top: '0',
      left: isYouTube1 ? '50%' : '0',
      transform: isYouTube1 ? 'translateX(-50%)' : 'none',
    }}
    title="?? ?? 1"
    allowFullScreen
    sandbox="allow-scripts allow-same-origin allow-presentation"
  ></iframe>
</div>
<div
  className="flex flex-col gap-1 justify-center py-4 pl-2  w-screen border-y-[1px] border-gray-200"
  style={{ marginLeft: '-50vw', left: '50%', position: 'relative' }}
>
  <h2 className="font-semibold">
    ???? 1.{' '}
    <span className="text-xs font-light tracking-tighter pl-1">
      ??? ?? ??? ?? ??? ??
```

```
          </span>
        </h2>
        {linkUrl1 && (
          <Link href={linkUrl1}>
            <p className="text-sm text-gray-600 underline line-clamp-1">{linkUrl1}</p>
          </Link>
        )}
      </div>
    </div>
  )}

  {iframeSrc2 && (
    <div className="flex flex-col mb-4">
      <div
        className=""
        style={{
          width: '100vw',
          paddingTop: isYouTube2 ? '56.25%' : '0',
          height: isYouTube2 ? '0' : '640px',
          marginLeft: '-50vw',
          left: '50%',
          position: 'relative',
        }}
      >
        <iframe
          src={iframeSrc2}
          style={{
```

```
      width: '100%',
      height: isYouTube2 ? '100%' : '640px',
      position: isYouTube2 ? 'absolute' : 'static',
      top: '0',
      left: isYouTube2 ? '50%' : '0',
      transform: isYouTube2 ? 'translateX(-50%)' : 'none',
    }}
    title="?? ?? 2"
    allowFullScreen
    sandbox="allow-scripts allow-same-origin allow-presentation"
  ></iframe>
</div>
<div
  className="flex flex-col gap-1 justify-center py-4 pl-2 w-screen  border-y-[1px]  border-gray-200"
  style={{ marginLeft: '-50vw', left: '50%', position: 'relative' }}
>
  <h2 className="font-semibold">
    ???? 2.{' '}
    <span className="text-xs font-light tracking-tighter pl-1">
      ??? ?? ??? ?? ??? ??
    </span>
  </h2>
  {linkUrl2 && (
    <Link href={linkUrl2}>
      <p className="text-sm text-gray-600 underline line-clamp-1">{linkUrl2}</p>
    </Link>
  )}
```

```
          </div>

        </div>

      )}

    </div>

  );

}
```

app\post_component\FixedIconGroup.tsx

```
// components/FixedIconGroup.tsx
'use client';

import { useState } from 'react';
import { BiPencil, BiUpArrowAlt, BiDownArrowAlt, BiDotsVerticalRounded } from 'react-icons/bi';
import Link from 'next/link';

export default function FixedIconGroup() {
  const [isExpanded, setIsExpanded] = useState(false);

  const toggleIcons = () => {
    setIsExpanded(!isExpanded);
  };

  const scrollToTop = () => {
    window.scrollTo({ top: 0, behavior: 'smooth' });
  };

  const scrollToBottom = () => {
    window.scrollTo({ top: document.body.scrollHeight, behavior: 'smooth' });
  };

  return (
    <div className="flex flex-col gap-4 fixed bottom-20 right-4 items-center">
      {isExpanded && (
        <>
```

```
          <div className="w-11 h-11 bg-emerald-200 rounded-full flex items-center justify-center shadow-lg">
            <Link href="/post/create">
              <BiPencil size={32} color="white" />
            </Link>
          </div>
          <div
            className="w-11 h-11 bg-emerald-200 rounded-full flex items-center justify-center shadow-lg cursor-pointer"
            onClick={scrollToTop}
          >
            <BiUpArrowAlt size={32} color="white" />
          </div>
          <div
            className="w-11 h-11 bg-emerald-200 rounded-full flex items-center justify-center shadow-lg cursor-pointer"
            onClick={scrollToBottom}
          >
            <BiDownArrowAlt size={32} color="white" />
          </div>
        </>
      )}
      <div
        className="w-12 h-12 bg-emerald-200 rounded-full flex items-center justify-center shadow-lg cursor-pointer"
        onClick={toggleIcons}
      >
        <BiDotsVerticalRounded size={36} color="white" />
      </div>
    </div>
  );
```

```
}
```

app\post_component\IconToggleComments.tsx

```
'use client';


import React, { useState } from 'react';

import { BiDotsVerticalRounded } from 'react-icons/bi';


interface IconToggleProps {

  onEdit: () => void;

  onDelete: () => void;

  onReply: () => void;

  showEditDelete: boolean;

}


const IconToggle: React.FC<IconToggleProps> = ({ onEdit, onDelete, onReply, showEditDelete }) => {

  const [isExpanded, setIsExpanded] = useState(false);


  const toggleIcons = () => {

    setIsExpanded(!isExpanded);

  };


  return (

    <div className="relative flex flex-col items-end">

      <div className="cursor-pointer" onClick={toggleIcons}>

        <BiDotsVerticalRounded

          size={36}

          className={isExpanded ? 'text-emerald-200' : 'text-gray-400'}

        />
      </div>
      {isExpanded && (
        <div className=" absolute right-0 top-full  flex flex-col items-end gap-3 p-2 z-10 bg-white">
          {showEditDelete && (
            <>
              <p
                onClick={onEdit}
                  className="text-sm tracking-tighter bg-emerald-200  border-gray-200 cursor-pointer min-w-16 min-h-8  flex
items-center justify-center shadow-md rounded-md"
              >
                ??
              </p>
              <p
                onClick={onDelete}
                  className="text-sm tracking-tighter bg-emerald-200  border-gray-200 cursor-pointer min-w-16 min-h-8  flex
items-center justify-center shadow-md rounded-md"
              >
                ??
              </p>
            </>
          )}
          <p
            onClick={onReply}
                className="text-sm tracking-tighter bg-emerald-200  border-gray-200 cursor-pointer min-w-16 min-h-8  flex
items-center justify-center shadow-md rounded-md"
          >
            ??
          </p>

        </div>

      )}

    </div>

  );

};


export default IconToggle;
```

app\post_component\InfiniteScrollPosts.tsx

```
'use client';



import { useEffect, useRef, useState } from 'react';

import Link from 'next/link';

import { useRouter } from 'next/navigation';

import { SlBubble, SlEye, SlHeart } from 'react-icons/sl';

import { listformatDate } from '@/lib/utils/formDate';

import { fetchMorePosts, fetchSearchPosts } from '../_action/infinityScrollPost';

import { PostType } from '../types';

import { addWritingPoints } from '../_action/adPointSupabase';



type InfiniteScrollPostsProps = {

  initialPosts: PostType[];

  userId?: string | null;

  searchTerm?: string;

  categoryId?: string; // ???? ID? ????? ??

};



export default function InfiniteScrollPosts({

  initialPosts,

  userId,

  searchTerm,

  categoryId, // ???? ID? ????? ??

}: InfiniteScrollPostsProps) {

  const [posts, setPosts] = useState<PostType[]>(initialPosts);

  const [loading, setLoading] = useState(false);
  const [page, setPage] = useState(2);

  const [hasMore, setHasMore] = useState(true);

  const [readPosts, setReadPosts] = useState<string[]>([]);

  const ref = useRef<HTMLDivElement>(null);

  const router = useRouter();


  //console.log('post writerId', posts[0].author_id);


  useEffect(() => {

    const clearLocalStorageDaily = () => {

      const lastClear = localStorage.getItem('lastClear');

      const now = new Date().getTime();


      if (!lastClear || now - parseInt(lastClear) > 24 * 60 * 60 * 1000) {

        localStorage.setItem('lastClear', now.toString());

        localStorage.removeItem('roundsData');

        if (userId) {

          localStorage.removeItem(`readPosts_${userId}`);

        }

      }

    };


    clearLocalStorageDaily();

  }, [userId]);


  useEffect(() => {

    if (userId) {
      const readPostsKey = `readPosts_${userId}`;

      const storedReadPosts = JSON.parse(localStorage.getItem(readPostsKey) || '[]');

      setReadPosts(storedReadPosts);

    }

  }, [userId]);



  useEffect(() => {

    setPosts(initialPosts);

  }, [initialPosts]);



  const handleObserver = async (entries: IntersectionObserverEntry[]) => {

    const target = entries[0];

    if (target.isIntersecting && !loading && hasMore) {

      setLoading(true);

      const newPosts = searchTerm

        ? await fetchSearchPosts(searchTerm, page)

        : await fetchMorePosts(page, categoryId); // categoryId? ???? ???

      if (newPosts.length > 0) {

        setPosts((prev) => [...prev, ...newPosts]);

        setPage((prev) => prev + 1);

      } else {

        setHasMore(false);

      }

      setLoading(false);

    }

  };
```

```
useEffect(() => {
  const observer = new IntersectionObserver(handleObserver, {
    root: null,
    rootMargin: '20px',
    threshold: 0,
  });

  if (ref.current && hasMore) observer.observe(ref.current);
  return () => {
    if (ref.current) observer.unobserve(ref.current);
  };
}, [loading, hasMore]);

const prefetchPostDetail = async (postId: string) => {
  const targetUrl = `/post/detail/${postId}`;
  router.prefetch(targetUrl);
};

const handlePostClick = async (postId: string) => {
  const post = posts.find((p) => p.id === postId); // ??? ??? ??
  if (post) {
    await addWritingPoints(post.author_id, 3); // ??? ????? 3??? ??
  }

  const readPostsKey = `readPosts_${userId}`;
  const storedReadPosts = JSON.parse(localStorage.getItem(readPostsKey) || '[]');
  // ?? ?? ?? ? ??? ??? ID ??
  const updatedReadPosts = Array.from(new Set([...storedReadPosts, postId]));

  setReadPosts(updatedReadPosts);
  localStorage.setItem(readPostsKey, JSON.stringify(updatedReadPosts));

  // ??? ?? ??
  const detailUrl = `/post/detail/${postId}`;
  router.push(detailUrl);
};

const isPostRead = (postId: string) => {
  return readPosts.includes(postId);
};

return (
  <div>
    {posts.length ? (
      posts.map((post) => {
        const profileUrl = {
          pathname: `/profile/${post.author_id}`,
        };

        return (
          <div
            key={post.id}
            className="flex flex-col py-2 bg-white border-b-[1px] border-gray-200"
  onClick={() => prefetchPostDetail(post.id)}
>
  <div className="flex justify-between items-center">
    <div className="categoryCreatorComments flex gap-2 flex-1 overflow-hidden">
      <div className="flex">
        <p className="text-xs leading-tight tracking-tight text-gray-600">
          {post.parent_category_name || '????'} &gt;
        </p>
        <p className="text-xs leading-tight tracking-tight text-gray-600 ml-1">
          {post.child_category_name || '????'}
        </p>
      </div>
    </div>
    <p className="text-xs text-gray-600">
      {listformatDate(post.created_at) || 'No time'}
    </p>
  </div>
  <div
    className="flex-1 pt-2 pb-1 cursor-pointer"
    onClick={() => handlePostClick(post.id)}
  >
    <p
      className={`font-semibold line-clamp-1 leading-tight tracking-tighter ${
        isPostRead(post.id) ? 'text-gray-400' : ''
      }`}
    >
      {post.title}
  </p>
</div>
<div className="flex gap-4 items-center overflow-hidden">
  <Link href={profileUrl} className="overflow-hidden flex-1">
    <p className="text-xs font-semibold leading-tight tracking-tight text-gray-600 truncate">
      {post.author_name || post.author_email || 'unknown'}
    </p>
  </Link>
  <div className="flex gap-1">
    <div className="flex items-center gap-[2px]">
      <SlHeart size={12} color="gray" />
      <p className="text-xs leading-tight tracking-tight text-gray-600">
        {post.views || '0'}
      </p>
    </div>
    <div className="flex items-center gap-[2px]">
      <SlEye size={14} color="gray" />
      <p className="text-xs leading-tight tracking-tight text-gray-600">
        {post.views || '0'}
      </p>
    </div>
    <div className="flex items-center gap-[2px]">
      <SlBubble size={12} color="gray" />
      <p className="text-xs leading-tight tracking-tight text-gray-600">
        {post.comments || '0'}
      </p>
    </div>
```

```
                </div>

              </div>

            </div>

          );

        })

      ) : (

        <p className="hover:text-red-200 text-blue-400">No posts</p>

      )}

      {loading && <p>Loading...</p>}

      <div ref={ref} />

    </div>

  );

}
```

app\post_component\PeriodSelector.tsx

```
// app/post/_component/PeriodSelector.tsx

'use client';



import { useState } from 'react';



interface PeriodSelectorProps {

  onSelectPeriod: (period: number) => void;

}



export default function PeriodSelector({ onSelectPeriod }: PeriodSelectorProps) {

  const [selectedPeriod, setSelectedPeriod] = useState(7);



  const handleSelect = (event: React.ChangeEvent<HTMLSelectElement>) => {

    const period = parseInt(event.target.value, 10);

    setSelectedPeriod(period);

    onSelectPeriod(period);

  };



  return (

    <div>

      <label htmlFor="period">Select period: </label>

      <select id="period" value={selectedPeriod} onChange={handleSelect}>

        <option value={1}>1 Day</option>

        <option value={7}>7 Days</option>

        <option value={30}>30 Days</option>

      </select>
    </div>
  );
}
```

app\post_component\PointAnimation.tsx

```
'use client';


import { useEffect, useState } from 'react';

import { calculatePoints } from '../_action/adPoint';

import { saveUserRoundData } from '../_action/adPointSupabase';


interface PointAnimationProps {

  userId: string | null;

  initialRoundData: any;

  onAnimationEnd?: (points: number) => void; // ??? ????? ??

}


export function PointAnimation({ userId, initialRoundData, onAnimationEnd }: PointAnimationProps) {

  const [randomNumber, setRandomNumber] = useState<number | null>(null);

  const [bonusNumber, setBonusNumber] = useState<number | null>(null);

  const [roundData, setRoundData] = useState(initialRoundData);

  const [animationExecuted, setAnimationExecuted] = useState(false); // ??? ??


  useEffect(() => {

    async function initializeRound() {

      let roundPoints: number[] = roundData.round_points;

      let currentRoundIndex = roundData.current_round_index;


      if (!roundPoints.length) {

        roundPoints = calculatePoints();

        if (userId) {
```

```
    await saveUserRoundData(userId, currentRoundIndex, roundPoints);
  }
  setRoundData({ round_points: roundPoints, current_round_index: currentRoundIndex });
}

const number = roundPoints[currentRoundIndex];
setRandomNumber(number);

currentRoundIndex += 1;
if (userId) {
  await saveUserRoundData(userId, currentRoundIndex, roundPoints);
}

if (currentRoundIndex >= roundPoints.length) {
  if (userId) {
    await saveUserRoundData(userId, 0, calculatePoints());
  }
}

const timer = setTimeout(() => {
  setRandomNumber(null);
  if (!animationExecuted) {
    setAnimationExecuted(true);
    if (onAnimationEnd) {
      onAnimationEnd(number);
    }
  }
```

```
      }, 2000);


      return () => clearTimeout(timer);
    }


    if (!animationExecuted) {
      initializeRound();
    }
  }, [userId, roundData, animationExecuted]);


  return (
    <>
      {randomNumber !== null && (
        <div
              className={`fixed flex flex-col justify-center items-center gap-4 top-1/2 left-1/2 transform -translate-x-1/2
-translate-y-1/2 z-50 animate-move-up ${
            randomNumber === 0 ? 'bg-gray-400' : randomNumber >= 10 ? 'bg-blue-600' : 'bg-red-600'
          } rounded-full p-4 w-52 h-52 aspect-square`}
        >
          <p className="text-6xl font-bold text-white">{randomNumber}</p>
          <p className="text-2xl text-white text-center">
            {randomNumber === 0 ? (
              <>
                ??? ?
                <br />
                ?????
              </>
            ) : randomNumber >= 10 ? (
              '????!!'
            ) : (
              '??? ??'
            )}
          </p>
          {userId ? (
            <p className="text-xl text-white">???? ???????!</p>
          ) : (
            <p className="text-xl text-white">????? ???? ?????...</p>
          )}
        </div>
      )}
      {bonusNumber !== null && (
            <div className="fixed flex flex-col justify-center items-center gap-4 top-1/2 left-1/2 transform -translate-x-1/2
-translate-y-1/2 z-50 animate-move-up bg-blue-600 rounded-full p-4 w-52 h-52 aspect-square">
          <p className="text-6xl font-bold text-white">{bonusNumber}</p>
          <p className="text-2xl text-white">??? ???</p>
        </div>
      )}
    </>
  );
}
```

app\post_component\PopularitySwitch.tsx

```
// app/post/_component/PopularitySwitch.tsx


'use client';


import { useState } from 'react';


interface PopularitySwitchProps {
  onSwitch: (orderBy: 'views' | 'created_at') => void;
}


export default function PopularitySwitch({ onSwitch }: PopularitySwitchProps) {
  const [active, setActive] = useState<'views' | 'created_at'>('created_at');


  const handleSwitch = (orderBy: 'views' | 'created_at') => {
    console.log('Switch clicked:', orderBy);
    setActive(orderBy);
    onSwitch(orderBy);
  };


  return (
    <div className="grid grid-cols-2 h-12">
      <div
        className={`flex justify-center items-center cursor-pointer ${
          active === 'views'
            ? 'bg-white border-b-[1px] border-gray-400'
            : 'bg-white border-b-[1px] border-gray-200'
        }`}
        onClick={() => handleSwitch('views')}
      >
        <p className={`text-center ${active === 'views' ? 'font-bold' : 'font-light'}`}>??</p>
      </div>
      <div
        className={`flex justify-center items-center cursor-pointer ${
          active === 'created_at'
            ? 'bg-white border-b-[1px] border-gray-400'
            : 'bg-white border-b-[1px] border-gray-200'
        }`}
        onClick={() => handleSwitch('created_at')}
      >
        <p className={`text-center ${active === 'created_at' ? 'font-bold' : 'font-light'}`}>
          ??
        </p>
      </div>
    </div>
  );
}
```

app\post_component\PopularitySwitchClient.tsx

```
'use client';


import { useState, useEffect } from 'react';

import InfiniteScrollPosts from './InfiniteScrollPosts';

import PopularitySwitch from './PopularitySwitch';

import { PostType } from '../types';


interface PopularitySwitchClientProps {

  initialPosts: PostType[];

  userId: string | null;

}


export default function PopularitySwitchClient({

  initialPosts,

  userId,

}: PopularitySwitchClientProps) {

  const [posts, setPosts] = useState<PostType[]>(initialPosts);

  const [orderBy, setOrderBy] = useState<'views' | 'created_at'>('created_at');


  useEffect(() => {

    const sortedPosts = [...initialPosts].sort((a, b) => {

      if (orderBy === 'views') {

        return (b.views || 0) - (a.views || 0);

      } else {

        return new Date(b.created_at).getTime() - new Date(a.created_at).getTime();

      }
    });

    setPosts(sortedPosts);

  }, [orderBy, initialPosts]);



  const handleSwitch = (orderBy: 'views' | 'created_at') => {

    setOrderBy(orderBy);

  };



  return (

    <>

      <PopularitySwitch onSwitch={handleSwitch} />

      <div className="flex flex-col px-4 pt-4 ">

        <InfiniteScrollPosts key={orderBy} initialPosts={posts} userId={userId} />

      </div>

    </>

  );

}
```

app\post_component\PostActions.tsx

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

import Link from 'next/link';

import {

  Dialog,

  DialogContent,

  DialogDescription,

  DialogHeader,

  DialogTitle,

  DialogTrigger,

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



const PostActions = ({ id, onConfirmDelete }: { id: string; onConfirmDelete: () => void }) => (

  <div className="grid grid-cols-2 items-center gap-2 w-full h-12">

    <Button variant="outline" asChild>

      <Link href={`/post/edit/${id}`}>????</Link>

    </Button>

    <Dialog>

      <DialogTrigger asChild>

        <Button variant="outline">????</Button>

      </DialogTrigger>

      <DialogContent>

        <DialogHeader>

          <DialogTitle>??? ??</DialogTitle>

          <DialogDescription>

            ??? ? ???? ????????? ? ??? ??? ? ????.

          </DialogDescription>
        </DialogHeader>

        <div className="mt-4 flex justify-end space-x-2">

          <DialogTrigger asChild>

            <Button variant="outline">??</Button>

          </DialogTrigger>

          <Button variant="destructive" onClick={onConfirmDelete}>

            ??

          </Button>

        </div>

      </DialogContent>

    </Dialog>

  </div>

);


export default PostActions;
```

app\post_component\PostContent.tsx

```
const PostContent = ({ content }: { content: string }) => (

  <section className="flex flex-col my-6 gap-8">

    {/* <img src="/mainad-1.png" alt="" className="w-full aspect-square object-cover" /> */}

    <p className="leading-normal ">{content}</p>

  </section>

);



export default PostContent;
```

app\post_component\PostForm.tsx

```
'use client';


import { useState, FormEvent, ChangeEvent, useRef } from 'react';

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

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

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

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

import { createPost } from '../_action/post';

import { categories, childCategoriesArray, parentCategoriesArray } from '../_action/category';

import {

  Select,

  SelectTrigger,

  SelectValue,

  SelectContent,

  SelectItem,

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


export type UserPropsType = {

  user_id: string;

  user_name?: string;

  user_email?: string;

  user_avatar_url?: string;

};


export default function PostForm({

  user_id,
  user_name,

  user_avatar_url,

  user_email,

}: UserPropsType) {

  const [parentCategoryId, setParentCategoryId] = useState<string | ''>('');

  const [childCategoryId, setChildCategoryId] = useState<string | ''>('');

  const [imagePreviews, setImagePreviews] = useState<string[]>([]);

  const [images, setImages] = useState<File[]>([]);

  const fileInputRef = useRef<HTMLInputElement>(null);



  const handleParentCategoryChange = (value: string) => {

    setParentCategoryId(value);

    setChildCategoryId(''); // ?? ????? ???? ?? ???? ???

  };



  const handleChildCategoryChange = (value: string) => {

    setChildCategoryId(value);

  };



  const handleImageChange = (e: ChangeEvent<HTMLInputElement>) => {

    const files = e.target.files;

    if (files && files.length > 0) {

      const fileArray = Array.from(files).slice(0, 5); // ?? 5? ???

      const validImages = fileArray.filter(

        (file) => file.size > 0 && file.type.startsWith('image/')

      );

      setImages((prevImages) => [...prevImages, ...validImages]);
      setImagePreviews((prevPreviews) => [
        ...prevPreviews,
        ...validImages.map((file) => URL.createObjectURL(file)),
      ]);

      // ?? ??? ???? ?? ??? ?????.
      if (fileInputRef.current) {
        fileInputRef.current.value = '';
      }
    }
  };

  const handleImageRemove = (index: number) => {
    setImages((prevImages) => prevImages.filter((_, i) => i !== index));
    setImagePreviews((prevPreviews) => prevPreviews.filter((_, i) => i !== index));
  };

  const totalFormData = async (e: FormEvent) => {
    e.preventDefault();
    const form = e.currentTarget as HTMLFormElement;
    const formData = new FormData(form);

    // ?? FormData? ??? ?? ?? ???? ? ? ?? ??
    images.forEach((image) => {
      if (
        image.size > 0 &&
        !Array.from(formData.entries()).some(
      ([key, value]) =>
        key === 'images' &&
        (value as File).name === image.name &&
        (value as File).size === image.size
    )
   ) {
    formData.append('images', image);
   }
  });

  // User ID ?? / User name ??
  formData.append('user_id', user_id);
  if (user_name) {
   formData.append('user_name', user_name);
  }
  if (user_avatar_url) {
   formData.append('user_avatar_url', user_avatar_url);
  }
  if (user_email) {
   formData.append('user_email', user_email);
  }

  // ????? ???? ???? null? ??
  formData.append('parent_category_id', parentCategoryId || '');
  formData.append('child_category_id', childCategoryId || '');

  await createPost(formData);
```

```
};


const parentCategories = parentCategoriesArray;

const childCategories = childCategoriesArray;


return (

  <div className="mt-4 mx-auto px-6">

    <h1 className="text-2xl font-bold mb-4 text-gray-800">??? ?? ???</h1>

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

      {/* ?? */}

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

        <Label htmlFor="title">Title</Label>

        <Input type="text" name="title" id="title" required />

      </div>

      {/* ?? */}

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

        <Label htmlFor="content">Content</Label>

        <Textarea name="content" id="content" required />

      </div>

      {/* ????1? */}

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

        <Label htmlFor="parent_category">Parent Category</Label>

        <Select value={parentCategoryId} onValueChange={handleParentCategoryChange}>

          <SelectTrigger className="w-full">

            <SelectValue placeholder="Select Parent Category" />

          </SelectTrigger>

          <SelectContent className="max-h-48 overflow-y-auto">
      {parentCategories.map((category) => (
        <SelectItem key={category.id} value={category.id}>
          {category.name}
        </SelectItem>
      ))}
    </SelectContent>
  </Select>
</div>
<div className="space-y-2">
  <Label htmlFor="child_category">Child Category</Label>
  <Select value={childCategoryId} onValueChange={handleChildCategoryChange}>
    <SelectTrigger className="w-full">
      <SelectValue placeholder="Select Child Category" />
    </SelectTrigger>
    <SelectContent className="max-h-48 overflow-y-auto">
      {childCategories.map((category) => (
        <SelectItem key={category.id} value={category.id}>
          {category.name}
        </SelectItem>
      ))}
    </SelectContent>
  </Select>
</div>
{/* ??? */}
<div className="space-y-2">
  <Label htmlFor="images">Images (?? 5?)</Label>
  <Input
```

```
      type="file"
      name="images"
      id="images"
      multiple
      accept="image/*"
      onChange={handleImageChange}
      ref={fileInputRef}
    />
  </div>
  <div className="flex space-x-2">
    {imagePreviews.map((src, index) => (
      <div key={index} className="relative w-24 h-24">
        <img src={src} alt={`Preview ${index + 1}`} className="w-full h-full object-cover" />
        <button
          type="button"
          onClick={() => handleImageRemove(index)}
          className="absolute top-0 right-0 w-6 h-6 bg-red-500 text-white rounded-full flex items-center justify-center"
        >
          ×
        </button>
      </div>
    ))}
  </div>
  {/* url */}
  <div className="space-y-2">
    <Label htmlFor="linkUrl1">?? Url #1</Label>
    <Input type="text" name="linkUrl1" id="linkUrl1" />
```

```
          <Label htmlFor="linkUrl2">?? Url #2</Label>

          <Input type="text" name="linkUrl2" id="linkUrl2" />

        </div>

        {/* ?? */}

        <Button type="submit">Create Post</Button>

      </form>

    </div>

  );

}
```

app\post_component\PostHeader.tsx

```
import { postformatDate } from '@/lib/utils/formDate';

import { BiDotsVerticalRounded } from 'react-icons/bi';

import AuthorHoverCard from './AuthorHoverCard';



const PostHeader = ({

  title,

  author,

  author_name,

  author_email,

  author_avatar_url,

  author_id,

  views,

  comments,

  created_at,

  point,

}: {

  title: string;

  author: string;

  author_name: string;

  author_avatar_url: string;

  author_email: string;

  author_id: string;

  views: number;

  comments: number;

  created_at: string;

  point: number;
```

```
}) => (
  <div className="flex flex-col justify-center gap-2 py-2 ">
    {/* ??? ?? */}
    <div className="flex items-center author">
      <AuthorHoverCard
        author_name={author_name || author_email || 'unknown'}
        author_avatar_url={author_avatar_url || '/money-3d-main.png'}
        author_id={author_id}
        point={point}
        triggerElement={
          <img
            src={author_avatar_url || '/money-3d-main.png'}
            alt=""
            className="w-12 h-12 ring-1 rounded-full object-cover"
          />
        }
      />
      <div className="flex-1 h-10 ml-2 bg-white flex flex-col justify-center ">
        <AuthorHoverCard
          author_name={author_name || author_email || 'unknown'}
          author_avatar_url={author_avatar_url || '/money-3d-main.png'}
          author_id={author_id}
          point={point || 0}
          triggerElement={
            <p className="text-sm text-gray-600 font-bold leading-tight tracking-tighter line-clamp-1">
              {author_name || author_email || 'unknown'}
            </p>
          }
        />
      </div>
      <BiDotsVerticalRounded size={24} />
    </div>

    <hr />
    {/* ?? ? ??? ?? */}
    <div className="flex flex-col">
      <h2 className=" font-semibold mb-2 ">{title}</h2>
      <div className="flex gap-2">
        <p className="text-gray-600 text-xs font-semibold">??? {views || 0}</p>
        <p className="text-gray-600 text-xs font-semibold">?? {comments || 0}</p>
        <p className="text-gray-600 text-xs font-light">{postformatDate(created_at) || 0}</p>
      </div>
    </div>

    <hr />
  </div>
);

export default PostHeader;
```

app\post_component\PostLikeDislike.tsx

```
import { useState, useEffect } from 'react';

import { BiDislike, BiLike } from 'react-icons/bi';

import { fetchLikesDislikes, toggleLike } from '../_action/like';



interface Props {

  post_id: string;

  user_id: string | null;

}



export default function PostLikeDislike({ post_id, user_id }: Props) {

  const [likes, setLikes] = useState(0);

  const [dislikes, setDislikes] = useState(0);

  const [userLike, setUserLike] = useState<boolean | null>(null); // null: no action, true: liked, false: disliked



  useEffect(() => {

    const fetchData = async () => {

      try {

        const { likes, dislikes, likeUsers } = await fetchLikesDislikes(post_id);

        setLikes(likes);

        setDislikes(dislikes);



        // ?? ???? ???? ???? ??

        if (likeUsers.some((likeUser) => likeUser.user_id === user_id)) {

          setUserLike(true);

        } else if (

          likeUsers.some((likeUser) => likeUser.user_id === user_id && likeUser.liked === false)
        ) {
          setUserLike(false);
        } else {
          setUserLike(null);
        }
      } catch (error) {
        console.error('Error fetching likes/dislikes:', error);
      }
    };

    fetchData();
  }, [post_id, user_id]);

  const handleLike = async () => {
    if (!user_id) {
      console.log('User is not logged in');
      return; // ????? ?? ???? ?? ??? ??
    }

    try {
      const newLikeState = userLike === true ? null : true;
      setUserLike(newLikeState);
      if (newLikeState === true) {
        setLikes((prev) => prev + 1);
        if (userLike === false) {
          setDislikes((prev) => prev - 1);
        }
      } else {
        setLikes((prev) => prev - 1);
      }
      if (newLikeState !== null) {
        // <== ??? ??
        await toggleLike(post_id, user_id, newLikeState);
      }
    } catch (error) {
      console.error('Error updating like:', error);
    }
  };

  const handleDislike = async () => {
    if (!user_id) {
      console.log('User is not logged in');
      return; // ????? ?? ???? ?? ??? ??
    }

    try {
      const newDislikeState = userLike === false ? null : false;
      setUserLike(newDislikeState);
      if (newDislikeState === false) {
        setDislikes((prev) => prev + 1);
        if (userLike === true) {
          setLikes((prev) => prev - 1);
        }
      } else {
        setDislikes((prev) => prev - 1);

      }

      if (newDislikeState !== null) {

        // <== ??? ??

        await toggleLike(post_id, user_id, newDislikeState);

      }

    } catch (error) {

      console.error('Error updating dislike:', error);

    }

  };



  return (

    <div className="flex w-4/5 h-12 items-center justify-center gap-4  mx-auto">

      <div className="flex w-full justify-center gap-3 items-center">

        <div

          className={`flex gap-2 items-center cursor-pointer ${

            userLike === true ? 'text-red-600' : ''

          }`}

          onClick={handleLike}

        >

          <BiLike size={24} />

          <p>???</p>

        </div>

        <p className={`flex-shrink-0 ${userLike === true ? 'text-red-600' : ''}`}>{likes}</p>

      </div>



      <p className="text-gray-400 font-semibold">|</p>
      <div className="flex w-full justify-center gap-3 items-center">
        <div
          className={`flex gap-2 items-center cursor-pointer ${
            userLike === false ? 'text-gray-600' : 'text-gray-400'
          }`}
          onClick={handleDislike}
        >
          <BiDislike size={24} />
          <p>???</p>
        </div>
        <p className={`flex-shrink-0 ${userLike === false ? 'text-gray-600' : 'text-gray-400'}`}>
          {dislikes}
        </p>
      </div>
    </div>
  );
}
```

app\post_component\SearchBar.tsx

```
'use client';


// components/SearchBar.tsx

import React, { useState } from 'react';

import { useRouter } from 'next/navigation';

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

import { BiSearch } from 'react-icons/bi';


type SearchBarProps = {

  initialQuery?: string;

};


const SearchBar: React.FC<SearchBarProps> = ({ initialQuery = '' }) => {

  const [searchTerm, setSearchTerm] = useState(initialQuery);

  const router = useRouter();


  const handleSearch = () => {

    if (searchTerm.trim() !== '') {

      router.push(`/post/search?query=${searchTerm}`);

    }

  };


  return (

    <div className="flex justify-center  mx-4">

      <div className="relative w-full max-w-lg">

        <BiSearch
```

```
          className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400"

          size={20}

        />

        <input

          type="text"

          placeholder="Search posts..."

          value={searchTerm}

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

          className="border border-gray-300 rounded-md p-2 pl-10 w-full"

        />

      </div>

      <Button onClick={handleSearch} className="ml-2">

        Search

      </Button>

    </div>

  );

};


export default SearchBar;
```

app\post_hooks\useFetchPostById.tsx

```
// hooks/useFetchPostById.ts

import { useState, useEffect } from 'react';

import { fetchPostById } from '../_action/post';



export const useFetchPostById = (id: string) => {

  const [post, setPost] = useState<any>(null);

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

  const [loading, setLoading] = useState<boolean>(true);



  useEffect(() => {

    const fetchPost = async () => {

      try {

        const data = await fetchPostById(id);

        setPost(data);

        if (!data) {

          setError('???? ??? ? ????.');

        }

      } catch (error: any) {

        console.log(error);

        setError(error.message);

      } finally {

        setLoading(false);

      }

    };



    fetchPost();
  }, [id]);



  return { post, error, loading };

};
```

app\profile\sticker.ts

```
export interface StickerType {

  id: string;

  url: string;

  position: { x: number; y: number };

  size: number;

}


export interface ProfileType {

  avatar_url?: string;

  email: string;

  id: string;

  username?: string;

  stickers?: StickerType[];

}
```

app\profile\[id]\page.tsx

```
import { Suspense } from 'react';

import { getcurrentUserBrowserFromCookie } from '@/lib/cookiesBrowser';

import { fetchProfileById } from '../_actions/profile';

import ProfileForm from '../_components/ProfileForm';



export default async function ProfilePage({ params: { id } }: { params: { id: string } }) {

  const profile = await fetchProfileById(id);

  //console.log('profile', profile);

  const currentUser = await getcurrentUserBrowserFromCookie();

  const isEditable = profile.id === currentUser?.id;



  //console.log('profile', profile);



  return (

    <div className="mt-8 mx-auto px-6 flex flex-col items-center gap-4">

      <h1 className="text-2xl font-bold mb-4 text-gray-800">

        {profile.username || profile.email || 'unknown'}

        <span className="text-base text-gray-600"> ?? ???</span>

      </h1>

      <Suspense fallback={<div>Loading...</div>}>

        <ProfileForm profile={profile} isEditable={isEditable} />

      </Suspense>

    </div>

  );

}
```

app\profile_actions\profile.ts

```
// server.ts



'use server';



import createSupabaseServerClient from '@/lib/supabse/server';

import { redirect } from 'next/navigation';

import { deleteAvatarFromSupabase, uploadAvatarToSupabase } from '@/app/post/_action/image';

import { cookies } from 'next/headers';

import { revalidatePath } from 'next/cache';

import { getcurrentUserFromCookies } from '@/lib/cookies';



export interface ProfileType {

  avatar_url?: string;

  email: string;

  id: string;

  username?: string;

  birthday?: string;

  gender?: 'male' | 'female' | 'Let me see';

  location?: string;

  current_points?: number;

}



export const fetchProfileById = async (id: string): Promise<ProfileType> => {

  const supabase = await createSupabaseServerClient();

  const { data, error } = await supabase

    .from('userdata')
    .select('id, email, username, avatar_url, birthday, gender, location, current_points')
    .eq('id', id)
    .single();

  if (error) {
    throw new Error(error.message);
  }

  return data;
};

export const updateProfile = async (formData: FormData) => {
  const currentUser = await getcurrentUserFromCookies();
  const currentUserId = currentUser?.id || '';

  const supabase = await createSupabaseServerClient();
  const userId = formData.get('userId') as string;
  const username = formData.get('username') as string;
  const email = formData.get('email') as string;
  const birthday = formData.get('birthday') as string;
  const gender = formData.get('gender') as string;
  const location = formData.get('location') as string;
  const avatarFile = formData.get('avatar') as File | null;

  const existingProfile = await fetchProfileById(userId);
  const oldAvatarUrl = existingProfile.avatar_url || '';
```

```
let newAvatarUrl = oldAvatarUrl;


if (!currentUserId || currentUserId !== userId) {

  throw new Error('Not authorized');

}


if (avatarFile) {

  newAvatarUrl = await updateProfileAvatar(avatarFile, userId, oldAvatarUrl);

}


const updatedFields: any = {};

if (username && username !== existingProfile.username) updatedFields.username = username;

if (email && email !== existingProfile.email) updatedFields.email = email;

if (newAvatarUrl !== existingProfile.avatar_url) updatedFields.avatar_url = newAvatarUrl;

if (birthday && birthday !== existingProfile.birthday) updatedFields.birthday = birthday;

if (gender && gender !== existingProfile.gender) updatedFields.gender = gender;

if (location && location !== existingProfile.location) updatedFields.location = location;


console.log('Updating userdata with:', updatedFields);


const { data, error } = await supabase

  .from('userdata')

  .update(updatedFields)

  .eq('id', userId)

  .select();


if (error) {
  console.error('Error updating userdata:', error);
  throw new Error(error.message);
} else {
  console.log('Update successful:', data);
}

const { error: postError } = await supabase
  .from('post')
  .update({ author_name: username, author_avatar_url: newAvatarUrl })
  .eq('author_id', userId);

if (postError) {
  throw new Error(postError.message);
}

const updatedUserData = {
  id: userId,
  username,
  email,
  avatar_url: newAvatarUrl,
  birthday,
  gender,
  location,
};

cookies().set('currentUser', JSON.stringify(updatedUserData), {
  httpOnly: true,
    secure: process.env.NODE_ENV === 'production',
    maxAge: 60 * 60 * 24 * 7,
    path: '/',
    sameSite: 'lax',
  });

  /* revalidatePath('/');
  redirect('/'); */

  return updatedUserData;
};

export const updateProfileAvatar = async (file: File, userId: string, oldAvatarUrl: string) => {
  const supabase = await createSupabaseServerClient();

  if (oldAvatarUrl) {
    await deleteAvatarFromSupabase(oldAvatarUrl);
  }

  const fileName = `avatars/${userId}/${Date.now()}_${file.name}`;
  const { error } = await supabase.storage.from('avatars').upload(fileName, file);

  if (error) {
    console.error('Error uploading avatars:', error);
    throw new Error(error.message);
  }
  const newAvatarUrl = supabase.storage.from('avatars').getPublicUrl(fileName).data.publicUrl;


  const { error: updateError } = await supabase

    .from('userdata')

    .update({ avatar_url: newAvatarUrl })

    .eq('id', userId);


  if (updateError) {

    throw new Error(updateError.message);

  }


  return newAvatarUrl;

};
```

app\profile_components\ProfileForm.tsx

```
'use client';


import { useState, FormEvent, ChangeEvent, useRef } from 'react';

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

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

import { ProfileType, updateProfile } from '../_actions/profile';

import { BiCamera, BiUser } from 'react-icons/bi';

import {

  Select,

  SelectTrigger,

  SelectContent,

  SelectItem,

  SelectLabel,

  SelectValue,

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

import StickerDisplayComponent from './StickerDisplay';

import { useRouter } from 'next/navigation';


type ProfileFormProps = {

  profile: ProfileType;

  isEditable: boolean;

};


export default function ProfileForm({ profile, isEditable }: ProfileFormProps) {

  // const router = useRouter();

  const [username, setUsername] = useState(profile.username || '');
const [email, setEmail] = useState(profile.email || '');

const [birthday, setBirthday] = useState(profile.birthday || '');

const [gender, setGender] = useState(profile.gender || 'Let me see');

const [location, setLocation] = useState(profile.location || '');

const [avatarUrl, setAvatarUrl] = useState(profile.avatar_url || '');

const [newAvatarFile, setNewAvatarFile] = useState<File | null>(null);

const point = profile.current_points || 0;

const fileInputRef = useRef<HTMLInputElement>(null);



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

  e.preventDefault();

  const formData = new FormData();

  formData.append('userId', profile.id);

  formData.append('username', username);

  formData.append('email', email);

  formData.append('birthday', birthday);

  formData.append('gender', gender);

  formData.append('location', location);



  if (newAvatarFile) {

    formData.append('avatar', newAvatarFile);

  }



  try {

    const result = await updateProfile(formData);

    if (result.avatar_url) {

      setAvatarUrl(result.avatar_url);
    }

    alert('Profile updated successfully');
    window.location.reload();
    //router.push('/post');
  } catch (error) {
    console.error('Error updating profile:', error);
    alert('Error updating profile');
  }
};

const handleImageClick = () => {
  if (fileInputRef.current) {
    fileInputRef.current.click();
  }
};

const handleImageChange = (e: ChangeEvent<HTMLInputElement>) => {
  const files = e.target.files;
  if (files && files.length > 0) {
    const file = files[0];
    setNewAvatarFile(file);
    const objectUrl = URL.createObjectURL(file);
    setAvatarUrl(objectUrl);
  }
};
```

```
  return (
    <form onSubmit={isEditable ? handleSubmit : undefined} className="w-full max-w-md space-y-4">
      <div className="flex flex-col ">
        <div className="grid grid-cols-2 ">
          <div className="col-span-1  ">
            <div
              className="relative w-40 h-40 bg-gray-200 rounded-full flex items-center justify-center cursor-pointer"
              onClick={handleImageClick}
            >
              {avatarUrl ? (
                <img
                  src={avatarUrl}
                  alt="Avatar"
                  className="w-full h-full object-cover rounded-full"
                />
              ) : (
                <BiUser size={120} color="gray" />
              )}
              {isEditable && (
                    <div className="absolute bottom-0 -right-0 z-10 w-12 h-12 rounded-full bg-gray-800 flex items-center
justify-center">
                  <BiCamera size={32} color="white" />
                </div>
              )}
            </div>
            <input
              type="file"
        ref={fileInputRef}
        className="hidden"
        accept="image/*"
        onChange={handleImageChange}
      />
    </div>
    <div className="col-span-1 px-2 py-2 flex flex-col justify-between border-x border-gray-200">
      <div>
        <h2 className="text-sm text-gray-400 leading-none mb-1">Point</h2>
        <p className="font-semibold leading-none">{point}</p>
      </div>
      <div>
        <h2 className="text-sm text-gray-400 leading-none mb-1">Trophy</h2>
        <p className="font-semibold leading-none">??? ?? ?</p>
        <p className="font-semibold leading-none">12,773</p>
      </div>
      <div>
        <h2 className="text-sm text-gray-400 leading-none mb-1">Support</h2>
        <p className="font-semibold leading-none">??? ? 72</p>
      </div>
    </div>
  </div>
  <div className="mt-8 flex flex-col space-y-4">
    <div className="username">
      <label className="block text-sm font-medium text-gray-600">Username</label>
      <Input
        type="text"
    value={username}

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

    required

    disabled={!isEditable}

    className="text-gray-800"

  />

</div>

<div className="birthday">

  <label className="block text-sm font-medium text-gray-600">Birthday</label>

  <Input

    type="date"

    value={birthday}

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

    disabled={!isEditable}

    className="text-gray-800"

  />

</div>

<div className="gender">

  <label className="block text-sm font-medium text-gray-600">Gender</label>

  <Select

    value={gender}

    onValueChange={(value: 'male' | 'female' | 'Let me see') => setGender(value)}

    disabled={!isEditable}

  >

    <SelectTrigger>

      <SelectValue placeholder="Select Gender" />

    </SelectTrigger>
```

```
      <SelectContent>

        <SelectItem value="male" className="text-gray-800">

          Male

        </SelectItem>

        <SelectItem value="female" className="text-gray-800">

          Female

        </SelectItem>

        <SelectItem value="Let me see" className="text-gray-800">

          Let me see

        </SelectItem>

      </SelectContent>

    </Select>

  </div>

  <div className="location">

    <label className="block text-sm font-medium text-gray-600">Location</label>

    <Input

      type="text"

      value={location}

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

      required

      disabled={!isEditable}

      className="text-gray-800"

    />

  </div>

  <div className="hidden">

    <label className="block text-sm font-medium text-gray-600">Email</label>

    <Input
```

```
              type="email"

              value={email}

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

              required

              disabled={!isEditable}

            />

          </div>

          {isEditable && (

            <Button className="mt-4" type="submit">

              Update Profile

            </Button>

          )}

          <StickerDisplayComponent />

        </div>

      </div>

    </form>

  );

}
```

app\profile_components\StickerDisplay.tsx

```
interface StickerGradeDetails {

  title: string;

  color: string;

}


type StickerData = Record<

  string,

  {

    bronze: StickerGradeDetails;

    silver: StickerGradeDetails;

    gold: StickerGradeDetails;

  }

>;


const sticker: StickerData = {

  dolphin: {

    bronze: {

      title: '????',

      color: '#CD7F32',

    },

    silver: {

      title: '????',

      color: '#C0C0C0',

    },

    gold: {

      title: '????',
    color: '#FFD700',
  },
},
grandmother: {
  bronze: {
    title: '????',
    color: '#CD7F32',
  },
  silver: {
    title: '????',
    color: '#C0C0C0',
  },
  gold: {
    title: '????',
    color: '#FFD700',
  },
},
fireman: {
  bronze: {
    title: '???',
    color: '#CD7F32',
  },
  silver: {
    title: '???',
    color: '#C0C0C0',
  },
  gold: {
      title: '???',

      color: '#FFD700',

    },

  },

  hunter: {

    bronze: {

      title: '??????',

      color: '#CD7F32',

    },

    silver: {

      title: '??????',

      color: '#C0C0C0',

    },

    gold: {

      title: '??????',

      color: '#FFD700',

    },

  },

  // ?? ???? ? ??? ????? ?? ??

};


type StickerDisplayProps = {

  type: keyof typeof sticker;

  level: number;

};


const getGrade = (level: number): keyof typeof sticker.dolphin => {
  if (level >= 1 && level <= 30) {
    return 'bronze';
  } else if (level >= 31 && level <= 70) {
    return 'silver';
  } else if (level >= 71 && level <= 99) {
    return 'gold';
  } else {
    throw new Error('Invalid level');
  }
};


const StickerDisplay = ({ type, level }: StickerDisplayProps) => {
  const grade = getGrade(level);
  const stickerType = sticker[type]?.[grade];
  const src = `/sticker/${type}.webp`;


  if (!stickerType) {
    return <div>??? ??? ?? ? ????.</div>;
  }


  return (
    <div className="mt-2 col-span-1 rounded-full flex flex-col gap-1 items-center justify-center">
      {/* ????? */}
      <div className="w-24 h-24 flex flex-col items-center justify-center relative border-b-2 border-gray-200">
        <img src={src} alt="" className="object-cover hover:animate-bounce" />
        <div
          className="absolute -bottom-1 -right-1 w-8 h-8 rounded-full flex  justify-center items-center"
          style={{ backgroundColor: stickerType.color }}
        >
          <p className="text-sm font-semibold tracking-tighter">
            <span className="text-xs text-gray-600 tracking-tighter">lv.</span>
            {level}
          </p>
        </div>
      </div>
      {/* ??? ?? */}
      <div className="flex flex-col">
        <p className="font-semibold">lv.{level}</p>
        <p className="font-semibold">{stickerType.title}</p>
      </div>
    </div>
  );
};


const StickerDisplayComponent = () => {
  return (
    <div className="grid grid-cols-3 gap-2 w-full">
      <StickerDisplay type="dolphin" level={24} />
      <StickerDisplay type="dolphin" level={32} />
      <StickerDisplay type="dolphin" level={72} />
      <StickerDisplay type="grandmother" level={24} />
      <StickerDisplay type="grandmother" level={35} />
      <StickerDisplay type="grandmother" level={75} />
      <StickerDisplay type="fireman" level={28} />
      <StickerDisplay type="fireman" level={40} />

      <StickerDisplay type="fireman" level={80} />

      <StickerDisplay type="hunter" level={26} />

      <StickerDisplay type="hunter" level={45} />

      <StickerDisplay type="hunter" level={68} />

      {/* ??? ?? ? ?? ??? ?? */}

    </div>

  );

};


export default StickerDisplayComponent;
```

app\utils\handlePostClick.ts

```
'use client';

// utils/handlePostClick.ts

import { useRouter } from 'next/navigation';



export const handlePostClick = (postId: string, userId: string | null) => {

  const router = useRouter();



  const readPostsKey = `readPosts_${userId}`;

  const storedReadPosts = JSON.parse(localStorage.getItem(readPostsKey) || '[]');



  // ?? ?? ?? ? ??? ??? ID ??

  const updatedReadPosts = Array.from(new Set([...storedReadPosts, postId]));



  localStorage.setItem(readPostsKey, JSON.stringify(updatedReadPosts));



  // ??? ?? ??

  const detailUrl = `/post/detail/${postId}`;

  router.push(detailUrl);

};
```

app_components\Ad_Bottom.tsx

```
// 'use client';


// import { useState, useEffect } from 'react';


// export default function AdFixedPage() {

//   const [isVisible, setIsVisible] = useState(true);


//   useEffect(() => {

//     const timer = setTimeout(() => {

//       setIsVisible(false);

//     }, 5000);


//     return () => clearTimeout(timer); // Cleanup the timer if the component is unmounted

//   }, []);


//   if (!isVisible) {

//     return <></>;

//   }


//   return (

//     <div className="mx-auto w-80 h-24 flex flex-col relative border-[1px] border-red-200 p-1">

//       <h2 className="text-xs">Ad 3?? ?????</h2>

//       <div

//         className="flex-1 bg-cover bg-no-repeat w-full h-full"

//         style={{ backgroundImage: 'url(/adBottom-unicef.jpg)' }}

//       >
//         {/* <img src="/adBottom-unicef.jpg" alt="" className="object-cover w-full h-full" /> */}

//       </div>

//     </div>

//   );

// }


/* ??? ??? ?? ?? ??  */

export default function AdFixedPage() {

  return (

    <div className="mx-auto w-80 h-24 flex flex-col relative border-[1px] border-red-200 p-1">

      <h2 className="text-xs">Ad ?????? ???????!!</h2>

      <div

        className="flex-1 bg-cover bg-no-repeat w-full h-full"

        style={{ backgroundImage: 'url(/adBottom-unicef.jpg)' }}

      >

        {/* <img src="/adBottom-unicef.jpg" alt="" className="object-cover w-full h-full" /> */}

      </div>

    </div>

  );

}
```

app_components\Footer.tsx

```
import Link from 'next/link';

import {

  BiDonateHeart,

  BiGroup,

  BiHomeAlt2,

  BiMenu,

  BiMessageDots,

  BiPlusCircle,

  BiShoppingBag,

} from 'react-icons/bi';



const Footer = () => {

  return (

      <footer className="fixed bottom-0 left-0 z-50 h-16 w-screen bg-white flex items-center justify-between px-9
rounded-t-2xl border-[1px] border-gray-200">

      <Link href="/post">

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

          <BiHomeAlt2 size={24} />

          <p className="text-[10px] text-gray-800 text-center">?</p>

        </div>

      </Link>

      <Link href="/goodluck">

        <div className="flex flex-col gap-[2px] justify-center items-center">

          <BiGroup size={24} />

          <p className="text-[10px] text-gray-800 text-center">????</p>

        </div>
      </Link>

      <Link href="/post/create">

        <div className="flex flex-col gap-[2px] justify-center items-center">

          <BiPlusCircle size={24} />

          <p className="text-[10px] text-gray-800 text-center">???</p>

        </div>

      </Link>

      <Link href="/message">

        <div className="flex flex-col gap-[2px] justify-center items-center">

          <BiMessageDots size={24} />

          <p className="text-[10px] text-gray-800 text-center">???</p>

        </div>

      </Link>

      <div className="flex flex-col gap-[2px] justify-center items-center">

        <BiMenu size={24} />

        <p className="text-[10px] text-gray-800 text-center">??</p>

      </div>

    </footer>

  );

};


export default Footer;
```

app_components\Header.tsx

```
//app>_components>Header.tsx



'use client';



import { useRouter } from 'next/navigation';

import { BiArrowBack } from 'react-icons/bi';

import { RxHamburgerMenu } from 'react-icons/rx';

import { useEffect, useState } from 'react';

import Link from 'next/link';

import UserHoverCard from './UserHoverCard';

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

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

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

import {

  Sheet,

  SheetClose,

  SheetContent,

  SheetDescription,

  SheetFooter,

  SheetHeader,

  SheetTitle,

  SheetTrigger,

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

import { childCategoriesArray, parentCategoriesArray } from '../post/_action/category';



interface CurrentUser {
  id: string;
  username: string | null;
  email: string | null;
  avatar_url: string | null;
  current_points: number;
}

const Header = () => {
  const router = useRouter();
  const [currentUser, setCurrentUser] = useState<CurrentUser | null>(null);

  const parentCategories = parentCategoriesArray;
  const childCategories = childCategoriesArray;

  useEffect(() => {
    const fetchUserInfo = async () => {
      try {
        const response = await fetch('/api/user');
        const userInfo = await response.json();
        setCurrentUser(userInfo);
      } catch (error) {
        console.error('Failed to fetch user info:', error);
      }
    };

    fetchUserInfo();
  }, []);
  const handleBackClick = () => {
    router.back();
  };

  const handleLoginClick = () => {
    router.push('/login'); // ??? ???? ??
  };

  return (
    /* top?? status? ?? ???? */
      <header className="fixed top-0  left-0 z-50 w-screen h-16 flex items-center justify-between px-6 border-b-[1px]
border-gray-200 bg-white">
      <BiArrowBack size={24} onClick={handleBackClick} className="cursor-pointer" />

      <img
        src="/logo-benefiboard.svg"
        alt=""
        className="absolute left-1/2 transform -translate-x-1/2"
      />

      <div className="flex items-center gap-3">
        {currentUser ? (
          <UserHoverCard
            avatar_url={currentUser.avatar_url}
            username={currentUser.username}
            email={currentUser.email}
            user_id={currentUser.id}

            point={currentUser.current_points}

            triggerElement={

              <img

                src={currentUser.avatar_url || '/money-3d-main.png'} // ??? URL? ????? ?? ??? ??

                alt="User Avatar"

                className="w-8 h-8 rounded-full cursor-pointer"

              />

            }

          />

        ) : (

          <Link href="/auth">

            <p className="text-xs tracking-tighter border-[1px] p-[2px] border-gray-200">???</p>

          </Link>

        )}

        <Sheet>

          <SheetTrigger asChild>

            <RxHamburgerMenu size={24} />

          </SheetTrigger>

          <SheetContent>

            <SheetHeader>

              <SheetTitle>????</SheetTitle>

            </SheetHeader>

            <div className="flex flex-col py-4  gap-4">

              <h2 className="text-lg font-semibold">???</h2>

              <div className="flex flex-col gap-2">

                {parentCategories.map((category) => (
                  <SheetClose asChild key={category.id}>
                    <Link href={`/post/${category.id}`}>
                      <p className="font-semibold pl-2">- {category.name}</p>
                    </Link>
                  </SheetClose>
                ))}
              </div>
            </div>
            <SheetFooter>
              <SheetClose asChild>
                <Button type="submit">Save changes</Button>
              </SheetClose>
            </SheetFooter>
          </SheetContent>
        </Sheet>
       </div>
      </header>
     );
    };

    export default Header;
```

app_components\MainBanner.tsx

```
const MainBanner = () => {
  return (
    <div className="bg-gray-200 py-20 text-center">
      <h1 className="text-4xl font-bold">Shop and Give Back</h1>
      <p className="mt-4 text-lg">
        Join us in making the world a better place through your purchases
      </p>
    </div>
  );
};

export default MainBanner;
```

app_components\PointTest2.tsx

```
'use client';


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

import { useState, useEffect } from 'react';


type PointsArray = number[];


const calculatePoints = (): PointsArray => {

  const rounds = 7 + Math.floor(Math.random() * 3); // 7 ~ 9 ??? ??

  const points: PointsArray = new Array(rounds).fill(0);

  let sum = 0;

  let bigPointIndex = Math.floor(Math.random() * rounds);


  for (let i = 0; i < rounds; i++) {

    if (i === bigPointIndex) {

      points[i] = 17 + Math.floor(Math.random() * 19); // 17 ~ 35 ??? ??

    } else {

      points[i] = Math.floor(Math.random() * 4); // 0 ~ 3 ??? ??

    }

    sum += points[i];

  }


  // ??? 35? ??? ?? ??

  while (sum !== 35) {

    sum = 0;

    bigPointIndex = Math.floor(Math.random() * rounds);
    for (let i = 0; i < rounds; i++) {

      if (i === bigPointIndex) {

        points[i] = 17 + Math.floor(Math.random() * 19);

      } else {

        points[i] = Math.floor(Math.random() * 4);

      }

      sum += points[i];

    }

  }


  return points;

};


const PointTester2 = () => {

  const [points, setPoints] = useState<PointsArray>([]);

  const [currentPointIndex, setCurrentPointIndex] = useState(0);

  const [history, setHistory] = useState<PointsArray[]>([]);


  const handleClick = () => {

    const newPoints = calculatePoints();

    console.log(newPoints);

    setPoints(newPoints);

    setHistory([...history, newPoints]); // ??? ???? ??

    setCurrentPointIndex(0); // ???? ????? ? ?? ????? ??

  };


  return (
    <div>

      <h1>Point Tester</h1>

      <Button onClick={handleClick}>Generate Points</Button>

      {points.length > 0 && <p>Current Point: {points[currentPointIndex]}</p>}

      <div>

        {points.map((point, index) => (

          <div key={index}>

            <p>

              Round {index + 1}: {point}

            </p>

          </div>

        ))}

      </div>

      <div>

        <h2>History</h2>

        {history.map((pointsSet, index) => (

          <div key={index}>

            <p>

              Attempt {index + 1}: {pointsSet.join(', ')} (Sum:{' '}

              {pointsSet.reduce((acc, val) => acc + val, 0)})

            </p>

          </div>

        ))}

      </div>

    </div>

  );

};
```

```
export default PointTester2;
```

app_components\PointTester1.tsx

```
'use client';


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

import { useState, useEffect } from 'react';


type PointsArray = number[];


const calculatePoints = (): PointsArray => {

  const points: PointsArray = [];

  let sum = 0;

  let bigPointIndex = Math.floor(Math.random() * 7);


  for (let i = 0; i < 7; i++) {

    if (i === bigPointIndex) {

      points[i] = 17 + Math.floor(Math.random() * 19); // 17 ~ 35 ??? ??

    } else {

      points[i] = Math.floor(Math.random() * 4); // 0 ~ 3 ??? ??

    }

    sum += points[i];

  }


  // ??? 32~35 ??? ??? ?? ??

  while (sum < 32 || sum > 35) {

    sum = 0;

    bigPointIndex = Math.floor(Math.random() * 7);

    for (let i = 0; i < 7; i++) {
      if (i === bigPointIndex) {

        points[i] = 17 + Math.floor(Math.random() * 19);

      } else {

        points[i] = Math.floor(Math.random() * 4);

      }

      sum += points[i];

    }

  }


  return points;

};


const PointTester1 = () => {

  const [points, setPoints] = useState<PointsArray>([]);

  const [currentPointIndex, setCurrentPointIndex] = useState(0);

  const [history, setHistory] = useState<PointsArray[]>([]);


  const handleClick = () => {

    const newPoints = calculatePoints();

    console.log(newPoints);

    setPoints(newPoints);

    setHistory([...history, newPoints]); // ??? ???? ??

    setCurrentPointIndex(0); // ???? ????? ? ?? ????? ??

  };


  return (

    <div>
```

```
      <h1>Point Tester</h1>

      <Button onClick={handleClick}>Generate Points</Button>

      {points.length > 0 && <p>Current Point: {points[currentPointIndex]}</p>}

      <div>

        {points.map((point, index) => (

          <div key={index}>

            <p>

              Round {index + 1}: {point}

            </p>

          </div>

        ))}

      </div>

      <div>

        <h2>History</h2>

        {history.map((pointsSet, index) => (

          <div key={index}>

            <p>

              Attempt {index + 1}: {pointsSet.join(', ')} (Sum:{' '}

              {pointsSet.reduce((acc, val) => acc + val, 0)})

            </p>

          </div>

        ))}

      </div>

    </div>

  );

};
```

```
export default PointTester1;
```

app_components\profile.tsx

```
// pages/profile.tsx



import { GetServerSidePropsContext } from 'next';



type Props = {

  userData: {

    username: string;

    avatar_url: string;

    email: string;

  } | null;

};



const ProfilePage: React.FC<Props> = ({ userData }) => {

  // userData? ???? ??? ???? ??????.

  return <p>{userData?.username}</p>;

};



export async function getServerSideProps(context: GetServerSidePropsContext) {

  const { req } = context;

  const cookies = req.cookies;



  let userData = null;

  if (cookies.userData) {

    userData = JSON.parse(cookies.userData);

  }
  return {
    props: {
      userData,
    },
  };
}

export default ProfilePage;
```

app_components\Status.tsx

```
import React from 'react';

import { BiBarChart, BiBattery, BiWifi } from 'react-icons/bi';



export default function Status() {

  return (

    <div className="fixed top-0 left-0 h-[42px] z-50 w-screen flex items-center justify-between px-6 bg-white">

      <p className="font-semibold text-sm text-gray-800">9:30</p>

      <div className="flex gap-[2px]">

        <BiWifi />

        <BiBarChart />

        <BiBattery />

      </div>

    </div>

  );

}
```

app_components\Topposts.tsx

```
// app/_components/TopPosts.tsx

'use client';



import { useRouter } from 'next/navigation';

import Link from 'next/link';

import { useState, useEffect } from 'react';

import { SlBubble, SlEye, SlHeart } from 'react-icons/sl';

import { listformatDate } from '@/lib/utils/formDate';

import { PostType } from '../post/types';



type TopPostsProps = {

  posts: PostType[];

  userId?: string | null;

  searchTerm?: string;

};



export default function TopPosts({ posts, userId }: TopPostsProps) {

  const router = useRouter();

  const [readPosts, setReadPosts] = useState<string[]>([]);



  useEffect(() => {

    if (userId) {

      const readPostsKey = `readPosts_${userId}`;

      const storedReadPosts = JSON.parse(localStorage.getItem(readPostsKey) || '[]');

      setReadPosts(storedReadPosts);

    }
  }, [userId]);


  const handlePostClick = (postId: string) => {
    const readPostsKey = `readPosts_${userId}`;
    const storedReadPosts = JSON.parse(localStorage.getItem(readPostsKey) || '[]');
    const updatedReadPosts = Array.from(new Set([...storedReadPosts, postId]));


    setReadPosts(updatedReadPosts);
    localStorage.setItem(readPostsKey, JSON.stringify(updatedReadPosts));


    const detailUrl = `/post/detail/${postId}`;
    router.push(detailUrl);
  };


  const isPostRead = (postId: string) => {
    return readPosts.includes(postId);
  };


  return (
    <div className="flex flex-col pt-4 w-full px-4 relative">
      <Link href="/post">
          <div className="absolute w-8 h-8 -bottom-4 left-1/2 transform -translate-x-1/2 bg-red-400 flex items-center
justify-center rounded-full">
          <p className="text-lg font-semibold text-white">+</p>
        </div>
      </Link>
      {posts.length ? (
posts.map((post) => {
  const profileUrl = {
    pathname: `/profile/${post.author_id}`,
  };


  return (
    <div
      key={post.id}
      className="flex flex-col py-2 bg-white border-b-[1px] border-gray-200"
      onClick={() => handlePostClick(post.id)}
    >
      <div className="flex justify-between items-center">
        <div className="categoryCreatorComments flex gap-2 flex-1 overflow-hidden">
          <div className="flex">
            <p className="text-xs leading-tight tracking-tight text-gray-600">
              {post.parent_category_name || '????'} &gt;
            </p>
            <p className="text-xs leading-tight tracking-tight text-gray-600 ml-1">
              {post.child_category_name || '????'}
            </p>
          </div>
        </div>
        <p className="text-xs text-gray-600">
          {listformatDate(post.created_at) || 'No time'}
        </p>
      </div>
      <div className="flex-1 pt-2 pb-1 cursor-pointer">
```

```
        <p
          className={`font-semibold line-clamp-1 leading-tight tracking-tighter ${isPostRead(post.id) ? 'text-gray-400'
: ''}`}
        >
          {post.title}
        </p>
      </div>
      <div className="flex gap-4 items-center overflow-hidden">
        <Link href={profileUrl} className="overflow-hidden flex-1">
          <p className="text-xs font-semibold leading-tight tracking-tight text-gray-600 truncate">
            {post.author_name || post.author_email || 'unknown'}
          </p>
        </Link>
        <div className="flex gap-1">
          <div className="flex items-center gap-[2px]">
            <SlHeart size={12} color="gray" />
            <p className="text-xs leading-tight tracking-tight text-gray-600">
              {post.views || '0'}
            </p>
          </div>
          <div className="flex items-center gap-[2px]">
            <SlEye size={14} color="gray" />
            <p className="text-xs leading-tight tracking-tight text-gray-600">
              {post.views || '0'}
            </p>
          </div>
          <div className="flex items-center gap-[2px]">
          <SlBubble size={12} color="gray" />
          <p className="text-xs leading-tight tracking-tight text-gray-600">
           {post.comments || '0'}
          </p>
         </div>
        </div>
       </div>
      </div>
     );
    })
   ) : (
    <p className="hover:text-red-200 text-blue-400">No posts</p>
   )}
  </div>
 );
}
```

app_components\UserHoverCard.tsx

```
'use client';


import { useState } from 'react';

import { HoverCard, HoverCardContent, HoverCardTrigger } from '@/components/ui/hover-card';

import { AiOutlineClose } from 'react-icons/ai';

import Link from 'next/link';


interface UserHoverCardProps {

  avatar_url: string | null;

  username: string | null;

  email: string | null;

  user_id: string;

  triggerElement: JSX.Element;

  point: number;

}


export default function UserHoverCard({

  avatar_url,

  username,

  email,

  user_id,

  point,

  triggerElement,

}: UserHoverCardProps) {

  const [isOpen, setIsOpen] = useState(false);
const handleTouch = () => {
  setIsOpen(!isOpen);
};

const handleClose = () => {
  setIsOpen(false);
};

return (
  <HoverCard open={isOpen} onOpenChange={setIsOpen}>
    <HoverCardTrigger asChild>
      <div onTouchStart={handleTouch}>{triggerElement}</div>
    </HoverCardTrigger>
    {isOpen && (
      <div className="fixed inset-0 flex items-center justify-center z-50">
        <div className="relative w-80 bg-white shadow-lg p-4 rounded-lg">
          <div className="absolute top-2 right-2">
            <button onClick={handleClose} className="text-gray-500 hover:text-gray-700">
              <AiOutlineClose size={20} />
            </button>
          </div>
          <div className="flex flex-col justify-between space-y-4">
            <img
              src={avatar_url || '/money-3d-main.png'}
              alt="avatar_image"
              className="w-60 h-60 ring-1 rounded-full object-cover mx-auto mt-4"
            />
            <div className="flex flex-col justify-center space-y-1 py-4 border-y-[1px] border-gray-200">

              <h4 className="text font-semibold pb-2 ">{username || email}</h4>

              <p className="text-sm">Point : {point}</p>

              <p className="text-sm">Trophy : ??? ?? ? 999,999,999</p>

              <p className="text-sm">Support : ??? ? 999</p>

            </div>

            <Link href={`/profile/${user_id}`}>

              <p className="text-sm font-semibold">???...</p>

            </Link>

          </div>

        </div>

      </div>

    )}

  </HoverCard>

 );

}
```

app_emoji-gather\emoji-gather.tsx

```
import { ReactNode } from 'react';

import './emoji-scss.scss'; // scss ??? import ???.



interface EmojiProps {

  size?: string;

  children?: ReactNode;

  className?: string;

}



const EmojiWrapper: React.FC<EmojiProps> = ({ size = '120px', children, className }) => {

  const scale = parseFloat(size) / 120;

  return (

    <div

      className={`emoji-wrapper ${className}`}

      style={{

        width: size,

        height: size,

        transform: `scale(${scale})`,

        transformOrigin: 'top left',

      }}

    >

      {children}

    </div>

  );

};
```

```
export const EmojiLike: React.FC<EmojiProps> = ({ size }) => (

  <EmojiWrapper size={size}>

    <div className="emoji emoji--like">

      <div className="emoji__hand">

        <div className="emoji__thumb" />

      </div>

    </div>

  </EmojiWrapper>

);



export const EmojiLove: React.FC<EmojiProps> = ({ size }) => (

  <EmojiWrapper size={size}>

    <div className="emoji emoji--love">

      <div className="emoji__heart" />

    </div>

  </EmojiWrapper>

);



export const EmojiHaha: React.FC<EmojiProps> = ({ size }) => (

  <EmojiWrapper size={size}>

    <div className="emoji emoji--haha">

      <div className="emoji__face">

        <div className="emoji__eyes" />

        <div className="emoji__mouth">

          <div className="emoji__tongue" />

        </div>

      </div>
    </div>
  </EmojiWrapper>
);

export const EmojiYay: React.FC<EmojiProps> = ({ size }) => (
  <EmojiWrapper size={size}>
    <div className="emoji emoji--yay">
      <div className="emoji__face">
        <div className="emoji__eyebrows" />
        <div className="emoji__mouth" />
      </div>
    </div>
  </EmojiWrapper>
);

export const EmojiWow: React.FC<EmojiProps> = ({ size }) => (
  <EmojiWrapper size={size}>
    <div className="emoji emoji--wow">
      <div className="emoji__face">
        <div className="emoji__eyebrows" />
        <div className="emoji__eyes" />
        <div className="emoji__mouth" />
      </div>
    </div>
  </EmojiWrapper>
);
```

```
export const EmojiSad: React.FC<EmojiProps> = ({ size }) => (
  <EmojiWrapper size={size}>
    <div className="emoji emoji--sad">
      <div className="emoji__face">
        <div className="emoji__eyebrows" />
        <div className="emoji__eyes" />
        <div className="emoji__mouth" />
      </div>
    </div>
  </EmojiWrapper>
);


export const EmojiAngry: React.FC<EmojiProps> = ({ size }) => (
  <EmojiWrapper size={size}>
    <div className="emoji emoji--angry">
      <div className="emoji__face">
        <div className="emoji__eyebrows" />
        <div className="emoji__eyes" />
        <div className="emoji__mouth" />
      </div>
    </div>
  </EmojiWrapper>
);
```

app_items\dolphin_item.tsx

```
export const DolphinBasic = ({ width, color }: { width: number; color: string }) => {

  // Original SVG aspect ratio calculation

  const originalWidth = 380;

  const originalHeight = 480;

  const aspectRatio = originalHeight / originalWidth;



  // Calculate height based on aspect ratio and provided width

  const height = width * aspectRatio;



  // Construct class name for dynamic color

  const className = `${color}`;



  console.log(className);



  return (

    <div className={className}>

      <svg

        width={width}

        height={height}

        viewBox="0 0 380 480"

        fill="none"

        xmlns="http://www.w3.org/2000/svg"

      >

        <path

          fillRule="evenodd"

          clipRule="evenodd"
          d="M114.323 74.1068C108.476 71.2302 100.646 74.1068 96.9915 80.5542C93.3373 87.0017 95.1122 94.6395
100.959 97.516C106.805 100.393 114.636 97.516 118.29 91.0686C121.944 84.6211 120.169 77.0825 114.323
74.1068ZM112.861 83.2324C111.608 85.0179 109.207 85.7122 107.536 84.7203C105.866 83.7284 105.657 81.447
106.91 79.5623C108.163 77.7769 110.564 77.0825 112.235 78.0744C113.905 79.0664 114.114 81.447 112.861
83.2324Z"
         className="fill-current"
       />
       <path
         fillRule="evenodd"
         clipRule="evenodd"
           d="M375.646 454.111C374.08 427.726 360.299 407.193 329.395 385.272C317.389 376.741 315.509 374.46
317.075 370.095C317.806 368.211 320.729 363.449 323.548 359.382C335.659 342.421 338.896 337.461 345.264
325.856C364.266 291.039 370.948 267.233 372.201 229.044C373.14 201.667 369.799 177.365 361.76
152.27C356.227 134.911 340.671 105.154 328.351 88.192C325.01 83.6292 325.01 71.0318 328.351
64.6835C331.796 58.0377 340.357 51.2926 350.693 47.4241C360.299 43.754 362.804 42.0678 365.206
37.7033C367.503 33.6365 367.085 29.8672 364.266 25.0068C362.387 21.8326 359.985 20.1464 353.721
17.2698C342.132 12.0127 333.989 10.3264 320.207 10.3264C305.278 10.2272 294.733 12.5086 273.539
20.444C264.873 23.6181 256.73 26.2963 255.268 26.2963C253.806 26.2963 250.361 25.0068 247.542
23.5189C229.48 13.8973 199.621 4.27569 178.949 1.39913C166.42 -0.287133 138.649 -0.485519 123.614
1.00236C80.7044 5.26761 44.2674 24.2133 21.2984 54.3676C1.67047 79.8599 -3.34093 114.974 8.97875
139.077L12.3197 145.426L8.03912 151.774C1.77487 161.098 0 166.157 0 174.291C0.104404 191.649 12.3197
197.601 35.6018 191.55C40.1956 190.26 51.2624 186.59 60.3456 183.218C79.1383 176.274 100.75 169.529 101.794
170.521C102.107 170.918 101.376 175.183 100.124 179.944C98.5575 185.896 97.7223 192.046 97.7223
198.89C97.7223 215.654 102.003 226.763 112.234 236.484C118.812 242.733 122.466 244.419 129.983
245.014C134.577 245.411 136.561 245.113 139.588 243.328L143.451 241.146L142.72 233.21C141.676 222.101
143.242 216.05 149.507 208.71C156.919 199.783 159.007 195.319 159.738 187.086C160.052 183.019 160.782
179.746 161.2 179.746C163.079 179.746 179.68 186.59 190.329 191.847C204.632 198.989 216.221 205.536 223.947
```

210.991C231.777 216.546 253.284 236.583 253.284 238.368C253.284 239.063 254.224 240.451 255.164
241.245C258.296 243.725 267.692 263.266 270.511 273.482C272.808 281.814 273.121 285.485 273.121
303.141C273.017 324.963 272.39 328.633 265.5 345.396C261.95 353.828 256.417 360.87 251.092 363.549C249.526
364.243 241.174 367.02 232.717 369.798C224.156 372.476 212.358 376.543 206.616 378.923C187.51 386.759
165.063 402.432 154.518 415.128C153.161 416.715 150.133 422.27 147.523 427.23C143.66 434.967 142.825
437.843 142.303 445.183C141.885 453.516 141.885 454.012 144.809 456.59C148.463 459.963 149.924 459.467
158.799 451.73C162.244 448.854 168.613 444.886 173.52 442.704L182.29 438.835L204.423 437.942C236.58
436.752 237.102 436.752 251.301 429.908L263.62 423.956L269.676 429.313C276.984 436.058 283.144 438.637
299.327 441.414C315.3 444.291 333.884 449.151 340.671 452.325C347.875 455.698 356.227 462.74 359.672
468.295C361.238 470.775 363.535 473.552 364.788 474.643C367.085 476.528 367.189 476.528 370.217
474.445C374.915 471.271 376.377 465.716 375.646 454.111ZM256.208 367.219C254.015 370.889 247.125 374.063
227.288 380.411C205.676 387.156 192.626 393.009 180.619 400.845C170.179 407.689 157.128 420.088 152.639
427.428C150.551 430.602 148.776 432.586 148.567 431.991C147.627 429.214 154.831 418.402 162.975
410.367C177.696 395.687 199.099 384.28 227.288 376.146C246.498 370.591 252.24 368.508 254.12
366.425C256.103 364.441 257.565 364.937 256.208 367.219ZM348.083 294.61C347.353 296.792 346.622 298.38
346.309 298.082C345.995 297.784 346.308 295.801 347.144 293.618C348.919 288.659 349.649 289.452 348.083
294.61ZM302.772 16.4763C309.245 15.3852 323.131 15.8811 330.126 17.1706C334.302 18.0634 340.253 19.7496
343.385 20.9399C348.919 23.1221 357.167 29.8672 355.914 31.0575C355.496 31.3551 354.243 30.7599 353.095
29.6688C346.935 23.9157 331.588 19.8488 316.867 19.8488C304.86 19.8488 292.123 22.6262 272.912
29.3712C261.741 33.2397 256.208 34.0332 256.208 31.5535C256.208 30.5615 295.464 17.6666 302.772
16.4763ZM6.68187 123.802C1.98368 103.17 7.93472 76.5865 21.2984 57.7401C31.1124 44.1508 48.8611 29.3712
65.4614 21.1383C78.6163 14.5916 102.525 8.14417 122.048 5.76357C130.818 4.77165 168.404 6.06114 178.844
7.7474C222.903 14.6908 275.522 43.2581 304.651 75.6938C323.757 97.0201 329.604 105.551 340.984
128.861C351.007 149.195 357.689 174.092 360.821 202.362C363.222 224.581 359.672 259.397 352.677
281.219C350.067 289.452 348.71 291.139 350.38 284.096C357.271 256.223 358.941 211.388 353.93
185.102C350.171 164.867 343.803 147.31 333.049 126.877C318.641 99.3015 299.535 74.6027 281.996
60.4183C264.873 46.5314 237.624 31.0575 216.43 23.1221C173.728 7.35064 128.73 5.76357 86.2378

18.5593C73.3961 22.4278 65.9834 26.0979 53.1417 35.1244C40.0912 44.25 29.8596 53.8716 23.3865
63.1956C15.1386 75.1979 8.45673 95.929 8.35233 110.312C8.35233 118.049 11.2756 131.34 14.0946
136.697C16.4959 141.061 16.9135 143.343 15.1386 142.351C13.0505 140.962 8.24793 130.844 6.68187
123.802ZM97.2002 76.0906C103.882 69.9407 113.905 69.2463 119.647 74.6027C125.389 79.9591 124.763 89.3823
118.081 95.5322C111.399 101.682 101.376 102.376 95.6342 97.0201C89.892 91.6637 90.5184 82.2405 97.2002
76.0906ZM5.53342 161.197C9.91839 152.072 16.1826 145.624 20.8808 145.029C21.194 144.93 20.3588 146.021
19.106 147.211C13.3637 153.063 4.38497 171.116 5.74223 174.191C6.05544 175.183 5.74223 175.58 4.90699
175.382C2.19249 174.588 2.5057 167.347 5.53342 161.197ZM119.543 236.781C104.926 227.457 98.0355 212.38
99.8104 193.038C100.646 184.507 104.509 171.712 107.223 168.934C109.938 166.058 110.042 167.545 107.536
173.199C102.629 184.21 101.063 194.823 102.838 206.032C105.344 220.811 110.146 229.342 120.169
236.087C125.389 239.559 124.867 240.253 119.543 236.781ZM284.084 339.743C281.16 341.726 276.253 345.297
273.33 347.38L267.901 351.447L271.764 339.445C277.193 322.682 278.446 312.366 276.88 295.106C276.149
287.369 274.687 277.946 273.748 274.177C268.945 257.017 257.147 236.087 245.037 223.093C238.355 215.951
221.024 203.255 208.599 196.212C194.609 188.277 162.453 173.795 147.105 168.438C128.626 161.991 124.867
159.61 129.983 157.924C131.027 157.527 132.28 156.833 132.593 156.337C132.906 155.841 134.368 155.246
135.621 155.146C136.978 155.047 138.127 154.551 138.127 153.956C138.127 153.46 138.44 153.361 138.962
153.659C139.275 154.055 140.215 153.956 140.841 153.46C141.572 152.964 143.973 151.972 146.375
151.278C148.776 150.683 150.551 149.79 150.238 149.592C150.029 149.393 144.182 150.782 137.396
152.766C130.61 154.75 118.916 157.924 111.399 159.709C94.3813 163.975 83.8365 167.248 51.7844
178.457C23.4909 188.376 16.287 189.764 9.81399 186.491C7.6215 185.4 5.1158 183.416 4.17617 182.027C2.5057
179.448 2.5057 179.448 5.74223 180.044C16.9135 182.027 36.5414 173.894 52.0977 160.9C58.8839 155.146
69.0111 142.946 74.4401 133.82C79.4515 125.389 83.2101 122.116 86.7598 123.207C89.0567 123.802 92.7109
121.421 92.7109 119.437C92.7109 118.941 91.6668 117.652 90.414 116.66C88.2215 114.974 88.5347 114.775
96.4694 110.709C101.063 108.526 104.926 106.146 105.031 105.65C105.239 105.154 107.327 104.658 109.416
104.658C112.339 104.658 115.575 103.567 118.603 101.583C120.691 100.194 122.675 98.4088 124.45
96.3257C127.373 92.9532 129.148 91.9613 135.621 90.5726C146.166 88.2912 168.404 88.8863 180.724
91.8621C191.895 94.6395 199.621 98.7063 207.451 106.146C214.446 112.792 214.864 112.494 199.621

```
113.684C194.923 113.982 188.032 115.073 184.273 115.867C177.696 117.454 167.777 122.116 167.777
123.603C167.777 124 169.448 126.282 171.536 128.563C176.234 133.622 185.213 145.327 191.686 154.353C194.4
158.122 198.263 163.28 200.456 165.76C214.237 181.631 243.575 194.426 267.483 194.922C276.775 195.121
277.089 195.022 280.012 192.046C284.397 187.285 283.144 183.019 270.824 160.404C262.681 145.723 260.384
140.565 261.741 140.565C264.038 140.565 287.529 162.784 294.837 171.712C305.278 184.606 316.345 210.793
319.685 230.83C321.043 238.567 321.356 246.105 320.938 259C320.521 277.252 319.477 282.608 313.108
299.867C307.888 313.457 293.689 333.196 284.084 339.743ZM362.7 453.516L361.029 445.58C357.062 426.238
344.534 410.565 318.015 391.223C312.064 386.859 308.514 382.792 308.514 380.213C308.514 379.816 310.185
380.907 312.273 382.792C314.152 384.478 319.999 388.842 325.01 392.314C348.292 408.582 363.326 430.701
362.909 448.159L362.7 453.516Z"
          className="fill-current"
        />
        <path
          fillRule="evenodd"
          clipRule="evenodd"
            d="M118.498 78.0737C112.652 75.1971 104.821 78.0737 101.167 84.5212C97.5131 90.9686 99.288 98.6064
105.135 101.483C110.981 104.36 118.812 101.483 122.466 95.0355C126.12 88.588 124.345 81.0494 118.498
78.0737ZM117.037 87.1993C115.784 88.9848 113.383 89.6791 111.712 88.6872C110.042 87.6953 109.833 85.4139
111.086 83.5292C112.339 81.7438 114.74 81.0494 116.41 82.0414C118.081 83.0333 118.29 85.4139 117.037
87.1993Z"
          className="fill-current opacity-20"
        />
        <path
          fillRule="evenodd"
          clipRule="evenodd"
            d="M379.822 458.078C378.256 431.693 364.474 411.16 333.571 389.239C321.564 380.708 319.685 378.427
321.251 374.062C321.982 372.178 324.905 367.416 327.724 363.349C339.835 346.388 343.072 341.428 349.44
```

329.823C368.442 295.006 375.124 271.2 376.376 233.011C377.316 205.634 373.975 181.332 365.936
156.237C360.403 138.878 344.846 109.121 332.527 92.1589C329.186 87.5961 329.186 74.9988 332.527
68.6505C335.972 62.0046 344.533 55.2596 354.869 51.3911C364.474 47.721 366.98 46.0347 369.381
41.6703C371.678 37.6034 371.261 33.8341 368.442 28.9737C366.563 25.7996 364.161 24.1133 357.897
21.2368C346.308 15.9796 338.165 14.2933 324.383 14.2933C309.453 14.1942 298.909 16.4756 277.715
24.4109C269.049 27.585 260.906 30.2632 259.444 30.2632C257.982 30.2632 254.537 28.9737 251.718
27.4859C233.656 17.8643 203.796 8.24264 183.124 5.36608C170.596 3.67982 142.824 3.48143 127.79
4.96931C84.8802 9.23456 48.4431 28.1802 25.4742 58.3345C5.84625 83.8268 0.834847 118.941 13.1545
143.044L16.4955 149.393L12.2149 155.741C5.95065 165.065 4.17578 170.124 4.17578 178.257C4.28019 195.616
16.4955 201.568 39.7776 195.517C44.3714 194.227 55.4382 190.557 64.5214 187.185C83.3141 180.241 104.926
173.496 105.97 174.488C106.283 174.885 105.552 179.15 104.299 183.911C102.733 189.863 101.898 196.013
101.898 202.857C101.898 219.62 106.179 230.73 116.41 240.451C122.988 246.7 126.642 248.386 134.159
248.981C138.753 249.378 140.736 249.08 143.764 247.295L147.627 245.113L146.896 237.177C145.852 226.068
147.418 220.017 153.682 212.677C161.095 203.75 163.183 199.286 163.914 191.053C164.227 186.986 164.958
183.713 165.376 183.713C167.255 183.713 183.855 190.557 194.505 195.814C208.808 202.956 220.397 209.503
228.123 214.958C235.953 220.513 257.46 240.55 257.46 242.335C257.46 243.03 258.4 244.418 259.339
245.212C262.472 247.692 271.868 267.233 274.687 277.449C276.984 285.781 277.297 289.452 277.297
307.108C277.193 328.93 276.566 332.6 269.675 349.363C266.126 357.795 260.592 364.837 255.268
367.516C253.702 368.21 245.349 370.987 236.893 373.765C228.331 376.443 216.534 380.51 210.792
382.89C191.686 390.726 169.239 406.399 158.694 419.095C157.337 420.682 154.309 426.237 151.699
431.197C147.836 438.934 147.001 441.81 146.479 449.15C146.061 457.482 146.061 457.978 148.984
460.557C152.638 463.93 154.1 463.434 162.974 455.697C166.42 452.82 172.788 448.853 177.695 446.671L186.465
442.802L208.599 441.909C240.756 440.719 241.278 440.719 255.477 433.875L267.796 427.923L273.852
433.28C281.16 440.025 287.32 442.604 303.502 445.381C319.476 448.258 338.06 453.118 344.846 456.292C352.05
459.665 360.403 466.707 363.848 472.262C365.414 474.742 367.711 477.519 368.964 478.61C371.261 480.495
371.365 480.495 374.393 478.412C379.091 475.238 380.553 469.683 379.822 458.078ZM260.384 371.186C258.191
374.856 251.3 378.03 231.464 384.378C209.852 391.123 196.801 396.975 184.795 404.812C174.355 411.656

161.304 424.055 156.815 431.395C154.727 434.569 152.952 436.553 152.743 435.958C151.803 433.181 159.007

422.369 167.151 414.334C181.872 399.654 203.274 388.247 231.464 380.113C250.674 374.558 256.416 372.475

258.295 370.392C260.279 368.408 261.741 368.904 260.384 371.186ZM352.259 298.577C351.528 300.759 350.797

302.346 350.484 302.049C350.171 301.751 350.484 299.767 351.32 297.585C353.094 292.626 353.825 293.419

352.259 298.577ZM306.948 20.4432C313.421 19.3521 327.307 19.8481 334.302 21.1376C338.478 22.0303 344.429

23.7166 347.561 24.9069C353.094 27.0891 361.342 33.8341 360.089 35.0244C359.672 35.322 358.419 34.7269

357.271 33.6358C351.111 27.8826 335.763 23.8158 321.042 23.8158C309.036 23.8158 296.299 26.5931 277.088

33.3382C265.917 37.2067 260.384 38.0002 260.384 35.5204C260.384 34.5285 299.639 21.6335 306.948

20.4432ZM10.8576 127.769C6.15946 107.137 12.1105 80.5535 25.4742 61.707C35.2882 48.1178 53.0369 33.3382

69.6372 25.1053C82.7921 18.5586 106.701 12.1111 126.224 9.73052C134.994 8.7386 172.58 10.0281 183.02

11.7144C227.079 18.6578 279.698 47.225 308.827 79.6608C327.933 100.987 333.78 109.518 345.16

132.828C355.182 153.162 361.864 178.059 364.996 206.329C367.398 228.548 363.848 263.364 356.853

285.186C354.243 293.419 352.886 295.105 354.556 288.063C361.447 260.19 363.117 215.355 358.106

189.069C354.347 168.834 347.979 151.277 337.225 130.844C322.817 103.268 303.711 78.5697 286.171

64.3852C269.049 50.4984 241.8 35.0244 220.606 27.0891C177.904 11.3176 132.906 9.73052 90.4136

22.5263C77.5719 26.3947 70.1592 30.0648 57.3175 39.0913C44.267 48.217 34.0354 57.8386 27.5623

67.1626C19.3144 79.1648 12.6325 99.8959 12.5281 114.279C12.5281 122.016 15.4514 135.307 18.2703

140.664C20.6716 145.028 21.0892 147.31 19.3144 146.318C17.2263 144.929 12.4237 134.811 10.8576

127.769ZM101.376 80.0575C108.058 73.9076 118.081 73.2133 123.823 78.5697C129.565 83.926 128.939 93.3492

122.257 99.4991C115.575 105.649 105.552 106.343 99.81 100.987C94.0677 95.6307 94.6942 86.2074 101.376

80.0575ZM9.7092 165.164C14.0942 156.038 20.3584 149.591 25.0566 148.996C25.3698 148.897 24.5346 149.988

23.2817 151.178C17.5395 157.03 8.56075 175.083 9.91801 178.158C10.2312 179.15 9.91801 179.547 9.08278

179.349C6.36827 178.555 6.68148 171.314 9.7092 165.164ZM123.719 240.748C109.102 231.424 102.211 216.347

103.986 197.005C104.821 188.474 108.684 175.678 111.399 172.901C114.113 170.025 114.218 171.512 111.712

177.166C106.805 188.177 105.239 198.79 107.014 209.999C109.52 224.778 114.322 233.309 124.345

240.054C129.565 243.526 129.043 244.22 123.719 240.748ZM288.259 343.709C285.336 345.693 280.429 349.264

277.506 351.347L272.077 355.414L275.94 343.412C281.369 326.648 282.622 316.333 281.056 299.073C280.325

```
291.336 278.863 281.913 277.923 278.144C273.121 260.983 261.323 240.054 249.212 227.06C242.53 219.918
225.199 207.221 212.775 200.179C198.785 192.244 166.629 177.762 151.281 172.405C132.802 165.958 129.043
163.577 134.159 161.891C135.203 161.494 136.456 160.8 136.769 160.304C137.082 159.808 138.544 159.213
139.797 159.113C141.154 159.014 142.302 158.518 142.302 157.923C142.302 157.427 142.616 157.328 143.138
157.626C143.451 158.022 144.391 157.923 145.017 157.427C145.748 156.931 148.149 155.939 150.55
155.245C152.952 154.65 154.727 153.757 154.413 153.559C154.205 153.36 148.358 154.749 141.572
156.733C134.785 158.717 123.092 161.891 115.575 163.676C98.5571 167.942 88.0123 171.215 55.9602
182.424C27.6667 192.343 20.4628 193.731 13.9898 190.458C11.7973 189.367 9.29158 187.383 8.35195
185.994C6.68148 183.415 6.68148 183.415 9.91801 184.011C21.0892 185.994 40.7172 177.861 56.2734
164.867C63.0597 159.113 73.1869 146.913 78.6159 137.787C83.6273 129.356 87.3859 126.083 90.9356
127.174C93.2325 127.769 96.8866 125.388 96.8866 123.404C96.8866 122.908 95.8426 121.619 94.5898
120.627C92.3973 118.941 92.7105 118.742 100.645 114.675C105.239 112.493 109.102 110.113 109.206
109.617C109.415 109.121 111.503 108.625 113.591 108.625C116.515 108.625 119.751 107.534 122.779
105.55C124.867 104.161 126.851 102.376 128.625 100.293C131.549 96.9202 133.324 95.9282 139.797
94.5395C150.342 92.2581 172.58 92.8533 184.899 95.829C196.071 98.6064 203.796 102.673 211.627
110.113C218.622 116.759 219.039 116.461 203.796 117.651C199.098 117.949 192.208 119.04 188.449
119.833C181.872 121.421 171.953 126.083 171.953 127.57C171.953 127.967 173.624 130.249 175.712
132.53C180.41 137.589 189.389 149.293 195.862 158.32C198.576 162.089 202.439 167.247 204.632
169.727C218.413 185.598 247.751 198.393 271.659 198.889C280.951 199.088 281.264 198.989 284.188
196.013C288.573 191.252 287.32 186.986 275 164.371C266.857 149.69 264.56 144.532 265.917 144.532C268.214
144.532 291.705 166.751 299.013 175.678C309.453 188.573 320.52 214.76 323.861 234.797C325.218 242.534
325.532 250.072 325.114 262.967C324.696 281.219 323.652 286.575 317.284 303.834C312.064 317.424 297.865
337.163 288.259 343.709ZM366.876 457.483L365.205 449.547C361.238 430.205 348.709 414.532 322.191
395.19C316.24 390.826 312.69 386.759 312.69 384.18C312.69 383.783 314.36 384.874 316.449 386.759C318.328
388.445 324.174 392.809 329.186 396.281C352.468 412.549 367.502 434.668 367.085 452.126L366.876 457.483Z"
          className="fill-current opacity-20"
        />
```

```
      </svg>
    </div>
  );
};
```

lib\cookies.ts

```
import { cookies } from 'next/headers';


export type CurrentUser = {
  current_points: number;
  id: string;
  username: string | null;
  email: string | null;
  avatar_url: string | null;
};


export function getcurrentUserFromCookies(): CurrentUser | null {
  const cookieStore = cookies();
  const currentUserCookie = cookieStore.get('currentUser');


  // ???? ?? ?? ??
  // console.log('All cookies:', cookieStore.getAll());
  // console.log('currentUserCookie:', currentUserCookie);


  if (currentUserCookie) {
    try {
      const decodedValue = decodeURIComponent(currentUserCookie.value);
      //console.log('decodedValue:', decodedValue); // ???? ? ??
      return JSON.parse(decodedValue);
    } catch (error) {
      console.error('Error parsing currentUser cookie:', error);
      return null;
    }
  } else {
    console.log('no cookie');
  }
  return null;
}


export function getCurrentUserInfo() {
  const currentUser = getcurrentUserFromCookies();


  if (!currentUser) {
    return null;
  }


  const userInfo = {
    id: currentUser.id ?? '',
    username: currentUser.username ?? '',
    email: currentUser.email ?? '',
    avatar_url: currentUser.avatar_url ?? '',
    point: currentUser.current_points ?? '',
    current_points: currentUser.current_points ?? '',
  };


  return userInfo;
}
```

lib\cookiesBrowser.ts

```
'use server';



import { getcurrentUserFromCookies } from '@/lib/cookies';



export async function getcurrentUserBrowserFromCookie() {

  return await getcurrentUserFromCookies();

}
```

lib\utils.ts

```
import { type ClassValue, clsx } from "clsx"

import { twMerge } from "tailwind-merge"



export function cn(...inputs: ClassValue[]) {

  return twMerge(clsx(inputs))

}
```

lib\actions\userSessionBrowser.ts

```
'use client';



import createSupabaseBrowserClient from '../supabse/client';



export default async function readUserSessionBrowser() {

  const supabase = await createSupabaseBrowserClient();



  return supabase.auth.getSession();

}
```

lib\actions\userSessionServer.ts

```
'use server';



import createSupabaseServerClient from '../supabse/server';



export default async function readUserSessionServer() {

  const supabase = await createSupabaseServerClient();



  return supabase.auth.getSession();

}
```

lib\supabse\client.ts

```
'use client';



import { createBrowserClient } from '@supabase/ssr';



export default function createSupabaseBrowserClient() {

  return createBrowserClient(

    process.env.NEXT_PUBLIC_SUPABASE_URL!,

    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!

  );

}
```

lib\supabse\server.ts

```
'use server';



import { createServerClient, type CookieOptions } from '@supabase/ssr';

import { cookies } from 'next/headers';



export default async function createSupabaseServerClient() {

  const cookieStore = cookies();



  return createServerClient(

    process.env.NEXT_PUBLIC_SUPABASE_URL!,

    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,

    {

      cookies: {

        get(name: string) {

          return cookieStore.get(name)?.value;

        },

        set(name: string, value: string, options: CookieOptions) {

          cookieStore.set({ name, value, ...options });

        },

        remove(name: string, options: CookieOptions) {

          cookieStore.set({ name, value: '', ...options });

        },

      },

    }

  );

}
```

lib\utils\formDate.ts

```
import dayjs from 'dayjs';

import timezone from 'dayjs/plugin/timezone';

import utc from 'dayjs/plugin/utc';



dayjs.extend(utc);

dayjs.extend(timezone);



export function listformatDate(created_at: string) {

  // ?? ??? ??? ?? ???? ??

  const now = dayjs().tz('Asia/Seoul');

  const today = now.format('YYYY-MM-DD'); // 'YYYY-MM-DD' ???? ?? ?? ??



  // ??? ?? ??? ?? ?? ????? ??

  const [datePart, timePart] = created_at.split('T'); // ??? ??? ??



  if (datePart === today) {

    // ?? ??? ?? ?? ??? ???? ??

    const [hours, minutes] = timePart.split(':');

    return `${hours}:${minutes}`;

  } else {

    // ?? ??? ?? ?? ?? ??? ???? ??

    const [, month, day] = datePart.split('-');

    return `${month}-${day}`;

  }

}
```

```
export function postformatDate(created_at: string) {

  const datePart = created_at.split('T')[0]; // '2024-06-05' ?? ??

  const timePart = created_at.split('T')[1]; // '06:58:23.404661+00' ?? ??

  const [, month, day] = datePart.split('-');

  const [hours, minutes] = timePart.split(':');



  return `${month}-${day} ${hours}:${minutes}`;

}
```

lib\utils\supabaseClient.ts

```
import { createClient } from '@supabase/supabase-js';



const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL!;

const supabaseAnonKey = process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!;



export const supabase = createClient(supabaseUrl, supabaseAnonKey);
```

middleware.ts

```
// middleware.ts

import { createServerClient, type CookieOptions } from '@supabase/ssr';

import { NextResponse, type NextRequest } from 'next/server';



export async function middleware(request: NextRequest) {

  let response = NextResponse.next({

    request: {

      headers: request.headers,

    },

  });



  const supabase = createServerClient(

    process.env.NEXT_PUBLIC_SUPABASE_URL!,

    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,

    {

      cookies: {

        get(name: string) {

          return request.cookies.get(name)?.value;

        },

        set(name: string, value: string, options: CookieOptions) {

          response.cookies.set(name, value, { ...options, sameSite: 'none', secure: true });

        },

        remove(name: string, options: CookieOptions) {

          response.cookies.set(name, '', { ...options, sameSite: 'none', secure: true });

        },

      },
    }
  );


  const {
    data: { user },
  } = await supabase.auth.getUser();


  if (user) {
    const { data: currentUser, error } = await supabase
      .from('userdata')
      .select('id, username, avatar_url, email, current_points')
      .eq('id', user.id)
      .single();


    if (currentUser) {
      //console.log('Setting currentUser cookie:', currentUser); // ?? ??
      response.cookies.set('currentUser', JSON.stringify(currentUser), {
        httpOnly: false, // ???????? ?? ????? ??
        secure: process.env.NODE_ENV === 'production',
        maxAge: 60 * 60 * 24 * 7, // 7 days
        path: '/',
        sameSite: 'lax', // none?? ???? ??? ??
      });
    } else {
      console.error('Error fetching currentUser:', error);
    }
  }
```

```
  return response;

}



export const config = {

  matcher: ['/((?!_next/static|_next/image|favicon.ico|.*\\.(?:svg|png|jpg|jpeg|gif|webp)$).*)'],

};
```