Tier-Based Event Showcase

Great! Here's a breakdown and starter guide to help you successfully implement your **Tier-Based Event Showcase** project using **Next.js 14**, **Clerk.dev**, **Supabase**, and **Tailwind CSS** — following all of the provided requirements and bonus suggestions.

✅ **Project Structure Overview**

```
/tier-events-app
 ├── app/
 |   ├── page.tsx
 |   ├── events/
 |   |   └── page.tsx
 |   └── layout.tsx
 ├── components/
 |   ├── EventCard.tsx
 |   ├── TierBadge.tsx
 |   └── UpgradeTierButton.tsx (Bonus)
 ├── lib/
 |   ├── supabase.ts
 |   └── tierUtils.ts
 ├── styles/
 ├── .env.local
 ├── tailwind.config.js
 ├── next.config.js
 └── README.md
```

📦 Step-by-Step Implementation Guide

1. Setup the Project

```
npx create-next-app@latest tier-events-app --app --ts --tailwind

cd tier-events-app

npm install @clerk/nextjs @supabase/supabase-js
```

2. Configure Clerk.dev

- Go to [Clerk.dev](https://clerk.dev), create an account and project.
- Add your frontend and API keys to .env.local:

```
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=your_publishable_key

CLERK_SECRET_KEY=your_secret_key

NEXT_PUBLIC_CLERK_SIGN_IN_URL=/sign-in

NEXT_PUBLIC_CLERK_SIGN_UP_URL=/sign-up
```

- Add Clerk middleware to middleware.ts:

```
import { authMiddleware } from "@clerk/nextjs";


export default authMiddleware({
  publicRoutes: ["/"],
});


export const config = {
  matcher: ["/((?!_next|favicon.ico).*)"],
};
```

- Wrap your app in Clerk Provider (app/layout.tsx):

```
import { ClerkProvider } from '@clerk/nextjs';


export default function RootLayout({ children }: { children: React.ReactNode }) {
  return (
    <ClerkProvider>
      <html lang="en">
        <body>{children}</body>
      </html>
    </ClerkProvider>
  );
}
```

---

3. Create Supabase Events Table

Use Supabase.io:

Schema (SQL):

```
create type user_tier as enum ('free', 'silver', 'gold', 'platinum');


create table events (
  id uuid primary key default uuid_generate_v4(),
  title text not null,
  description text,
  event_date timestamp not null,
  image_url text,
  tier user_tier not null
);
```

Seed Sample Data:

```
insert into events (title, description, event_date, image_url, tier) values
('Free Event 1', 'Open to all', now() + interval '2 days', 'https://via.placeholder.com/150', 'free'),
('Free Event 2', 'Another open event', now() + interval '3 days',
'https://via.placeholder.com/150', 'free'),

('Silver Event 1', 'Exclusive to Silver and up', now() + interval '5 days',
'https://via.placeholder.com/150', 'silver'),

('Gold Event 1', 'Premium for Gold', now() + interval '7 days', 'https://via.placeholder.com/150',
'gold'),

('Platinum Event 1', 'Top tier only', now() + interval '10 days', 'https://via.placeholder.com/150',
'platinum'),

('Platinum Event 2', 'VIP Experience', now() + interval '12 days',
'https://via.placeholder.com/150', 'platinum');
```

---

4. Configure Supabase Client

Create lib/supabase.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);
```

Add to .env.local:

```
NEXT_PUBLIC_SUPABASE_URL=your_url

NEXT_PUBLIC_SUPABASE_ANON_KEY=your_anon_key
```

---

5. Tier Mapping Logic

Create lib/tierUtils.ts:

```
const tierHierarchy = ['free', 'silver', 'gold', 'platinum'];
```

```
export function getAccessibleTiers(userTier: string) {
  const index = tierHierarchy.indexOf(userTier);
  return tierHierarchy.slice(0, index + 1);
}

export function isTierHigher(eventTier: string, userTier: string) {
  return tierHierarchy.indexOf(eventTier) > tierHierarchy.indexOf(userTier);
}
```

---

6. Events Page Implementation

Create app/events/page.tsx:

```
import { currentUser } from '@clerk/nextjs/server';
import { supabase } from '@/lib/supabase';
import { getAccessibleTiers, isTierHigher } from '@/lib/tierUtils';
import EventCard from '@/components/EventCard';

export default async function EventsPage() {
  const user = await currentUser();
  const userTier = user?.publicMetadata?.tier as string || 'free';

  const { data: events, error } = await supabase
    .from('events')
    .select('*')
    .order('event_date', { ascending: true });

  const accessibleTiers = getAccessibleTiers(userTier);
  return (
    <div className="p-6">
      <h1 className="text-2xl font-bold mb-4">Your Events</h1>
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
        {events?.map((event) => (
          <EventCard
            key={event.id}
            event={event}
            userTier={userTier}
            locked={isTierHigher(event.tier, userTier)}
          />
        ))}
      </div>
    </div>
  );
}
```

---

7. Event Card Component

components/EventCard.tsx:

```
import TierBadge from './TierBadge';


export default function EventCard({ event, userTier, locked }: any) {
  return (
    <div className={`border rounded-lg p-4 shadow ${locked ? 'opacity-50' : 'bg-white'}`}>
```

```
      <img src={event.image_url} alt={event.title} className="w-full h-40 object-cover mb-2
rounded" />
      <h2 className="text-lg font-semibold">{event.title}</h2>
      <p className="text-sm">{event.description}</p>
      <p className="text-sm text-gray-500">Date: {new
Date(event.event_date).toLocaleDateString()}</p>
      <TierBadge tier={event.tier} />
      {locked && (
        <p className="text-red-500 text-sm mt-2">Upgrade to {event.tier} to access this event</p>
      )}
    </div>
  );
}
```

---

8. Tier Badge Component

components/TierBadge.tsx:

```
const tierColors: any = {
  free: 'bg-gray-400',
  silver: 'bg-silver-500',
  gold: 'bg-yellow-500',
  platinum: 'bg-purple-600',
};

export default function TierBadge({ tier }: { tier: string }) {
  return (
    <span className={`inline-block mt-2 px-2 py-1 text-xs text-white rounded
${tierColors[tier]}`}>
```

```
      {tier.toUpperCase()}
    </span>
  );
}
```

---

9. Bonus: Tier Upgrade Button (Mock)

```
'use client'
import { useUser } from '@clerk/nextjs';
import { useState } from 'react';

export default function UpgradeTierButton() {
  const { user } = useUser();
  const [loading, setLoading] = useState(false);

  const upgrade = async () => {
    setLoading(true);
    const newTier = 'platinum'; // Simulate upgrade
    await user?.update({ publicMetadata: { tier: newTier } });
    window.location.reload();
  };

  return (
    <button
      onClick={upgrade}
      disabled={loading}
      className="bg-blue-600 text-white px-4 py-2 rounded mt-4"
    >
      {loading ? 'Upgrading...' : 'Upgrade to Platinum'}
    </button>
  );
}
```

---

10. Deploy on Vercel

- Push your code to GitHub.
- Connect to [Vercel](), import your repo, and add environment variables.
- Deploy 🚀

📄 **README.md Template**

```
# Tier-Based Event Showcase 🎉

View and unlock exclusive events based on your membership tier.

## 🌐 Live Demo

[View Live on Vercel](https://your-vercel-link)

## 👥 Demo Accounts

| Tier     | Email                | Password |
|----------|----------------------|----------|
| Free     | free@example.com     | test123  |
| Silver   | silver@example.com   | test123  |
```

| Gold | gold@example.com | test123 |

| Platinum