Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

ReactJS

Domaći zadatak 2

|  |  |  |
| --- | --- | --- |
| Ime | Prezime | Broj indeksa |
|  |  |  |
|  |  |  |
| Link ka Github-u |  | |

Sadržaj

[1 Korisnički zahtev 3](#_Toc209825236)

[2 Implementacija 4](#_Toc209825237)

[3 Korisničko uputstvo 48](#_Toc209825238)

# Korisnički zahtev

Eventim je SPA web aplikacija namenjena otkrivanju koncerata i festivala, praćenju izvođača i detalja događaja, uz mogućnost ostavljanja recenzija i pravljenja rezervacija koje se čuvaju lokalno. Ideja je da korisnik na jednom mestu pretražuje događaje po zemlji, ključnoj reči, tipu (concert/festival) i izvođaču, brzo dođe do detaljne stranice događaja sa vremenom, lokacijom i listom izvođača (svaki je linkovan na svoju stranicu), kao i da doda sopstvenu recenziju ili ostavi rezervaciju.

Na početnoj se vide Featured events (tri najbliža po datumu) i Top artists. Na stranici Events korisnik unosi pojmove u pretragu i/ili podešava filtere (tip događaja i izvođač), rezultati se prikazuju u karticama sa paginacijom (20 po stranici). Klik na karticu vodi na Event detail gde se vide slika, naziv, datum, mesto i izvođači, ispod čega su sekcije za Reviews (lista postojećih + forma za dodavanje) i Reservations (lista postojećih + forma za dodavanje) – sve što korisnik unese ostaje sačuvano u njegovom pregledaču. Stranica Artists nudi pretragu izvođača sa filterima po žanru i državi, a broj događaja po izvođaču se računa iz lokalnog kataloga, klik vodi na Artist detail sa pregledom svih njegovih nastupa.

Podaci dolaze iz Ticketmaster Discovery API-ja. U frontendu koristimo ključ iz .env (VITE\_TICKETMASTER\_KEY) i pozivamo GET /events.json da uz parametre poput countryCode, keyword i size preuzmemo događaje, kao i GET /attractions.json da pribavimo izvođače (attractions). Nakon mapiranja podatke spakujemo u naš lokalni store i povezujemo događaje sa izvođačima preko artistIds.

Stanje aplikacije čuva se u globalnom store-u baziranom na Zustand-u, tako da se artists, events, reviews i reservations upisuju u localStorage i čuvaju i nakon osvežavanja stranice. U store-u držimo i filtere za Events (tekst pretraga, tip, izvođač), meta informacije (izvor i vreme), kao i praktične akcije: sinhronizaciju sa Ticketmaster-om, setovanje i reset filtera, dodavanje i dodavanje.

# Implementacija

Aplikacija je kreirana korišćenjem JavaScript okvira ReactJS-a. Za lakše kreiranje aplikacije iskorišćena je bilioteka Vite, koja instalira samo neophodne biblioteke i kreira inicijalnu strukturu React TypeScript aplikacije. Dodate su i biblioteke TailwindCSS (za stilizaciju), React router dom (za rutiranje na stranici), zustand (za kreiranje localnog store-a) i React icons (za određene ikonice na sajtu).

{

  "name": "eventim",

  "private": true,

  "version": "0.0.0",

  "type": "module",

  "scripts": {

    "dev": "vite",

    "build": "tsc -b && vite build",

    "lint": "eslint .",

    "preview": "vite preview"

  },

  "dependencies": {

    "@tailwindcss/vite": "^4.1.12",

    "react": "^19.1.1",

    "react-dom": "^19.1.1",

    "react-icons": "^5.5.0",

    "react-router-dom": "^7.8.2",

    "tailwindcss": "^4.1.12",

    "zustand": "^5.0.8"

  },

  "devDependencies": {

    "@eslint/js": "^9.33.0",

    "@types/react": "^19.1.10",

    "@types/react-dom": "^19.1.7",

    "@vitejs/plugin-react": "^5.0.0",

    "eslint": "^9.33.0",

    "eslint-plugin-react-hooks": "^5.2.0",

    "eslint-plugin-react-refresh": "^0.4.20",

    "globals": "^16.3.0",

    "typescript": "~5.8.3",

    "typescript-eslint": "^8.39.1",

    "vite": "^7.1.2"

  }

}

main.tsx:

import { StrictMode } from 'react';

import { createRoot } from 'react-dom/client';

import './index.css';

import App from './App.tsx';

createRoot(document.getElementById('root')!).render(

  <StrictMode>

    <App />

  </StrictMode>

);

App.tsx:

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import Layout from './components/Layout';

import Home from './pages/Home';

import Events from './pages/Events';

import EventDetails from './pages/EventDetails';

import Artists from './pages/Artists';

import ArtistDetails from './pages/ArtistDetails';

function App() {

  return (

    <>

      <Router>

        <Routes>

          <Route element={<Layout />}>

            <Route path='/' element={<Home />} />

            <Route path='/events' element={<Events />} />

            <Route path='/events/:id' element={<EventDetails />} />

            <Route path='/artists' element={<Artists />} />

            <Route path='/artists/:id' element={<ArtistDetails />} />

          </Route>

        </Routes>

      </Router>

    </>

  );

}

export default App;

Layout.tsx:

import { Outlet } from 'react-router-dom';

import Navbar from './Navbar';

import Footer from './Footer';

export default function Layout() {

  return (

    <div className='min-h-dvh bg-[#0b0b0d] text-gray-100'>

      <Navbar />

      <main id='main' className='mx-auto max-w-7xl px-4 sm:px-6 lg:px-8'>

        <Outlet />

      </main>

      <Footer />

    </div>

  );

}

Navbar.tsx:  
import { useState } from 'react';

import { NavLink, Link } from 'react-router-dom';

const navItems = [

  { to: '/', label: 'Home' },

  { to: '/events', label: 'Events' },

  { to: '/artists', label: 'Artists' },

];

export default function Navbar() {

  const [open, setOpen] = useState(false);

  return (

    <header className='sticky top-0 z-50 border-b border-white/10 bg-black/80 backdrop-blur'>

      <nav className='mx-auto flex h-16 max-w-7xl items-center justify-between px-4 sm:px-6 lg:px-8'>

        {/\* Logo \*/}

        <Link to='/' className='flex items-center gap-2'>

          <span className='grid h-8 w-8 place-items-center rounded-lg bg-red-600 text-black font-black'>

            E

          </span>

          <span className='text-lg font-semibold tracking-wide text-white'>

            Eventim

          </span>

        </Link>

        {/\* Desktop links \*/}

        <ul className='hidden gap-1 md:flex'>

          {navItems.map((item) => (

            <li key={item.to}>

              <NavLink

                to={item.to}

                end={item.to === '/'}

                className={({ isActive }) =>

                  isActive

                    ? 'inline-flex items-center rounded-xl px-3 py-2 text-sm font-medium bg-red-600 text-white shadow-lg shadow-red-600/30'

                    : 'inline-flex items-center rounded-xl px-3 py-2 text-sm font-medium text-gray-200 hover:text-white hover:bg-red-600/20'

                }

              >

                {item.label}

              </NavLink>

            </li>

          ))}

        </ul>

        {/\* CTA button \*/}

        <div className='hidden md:block'>

          <Link

            to='/events'

            className='rounded-xl bg-red-600 px-4 py-2 text-sm font-semibold text-white shadow-md shadow-red-600/30 transition hover:bg-red-500'

          >

            Browse Events

          </Link>

        </div>

        {/\* Mobile menu button \*/}

        <button

          onClick={() => setOpen(!open)}

          className='inline-flex items-center justify-center rounded-xl p-2 text-gray-200 hover:bg-red-600/20 md:hidden'

        >

          {open ? (

            <svg

              className='h-6 w-6'

              viewBox='0 0 24 24'

              fill='none'

              stroke='currentColor'

              strokeWidth={2}

            >

              <path d='M6 6l12 12M6 18L18 6' />

            </svg>

          ) : (

            <svg

              className='h-6 w-6'

              viewBox='0 0 24 24'

              fill='none'

              stroke='currentColor'

              strokeWidth={2}

            >

              <path d='M4 6h16M4 12h16M4 18h16' />

            </svg>

          )}

        </button>

      </nav>

      {/\* Mobile menu \*/}

      {open && (

        <div className='md:hidden border-t border-white/10 bg-black/95'>

          <ul className='mx-auto max-w-7xl px-4 py-3 sm:px-6 lg:px-8'>

            {navItems.map((item) => (

              <li key={item.to} className='py-1'>

                <NavLink

                  to={item.to}

                  end={item.to === '/'}

                  onClick={() => setOpen(false)}

                  className={({ isActive }) =>

                    isActive

                      ? 'block rounded-xl px-3 py-2 text-base bg-red-600 text-white shadow-lg shadow-red-600/30'

                      : 'block rounded-xl px-3 py-2 text-base text-gray-200 hover:text-white hover:bg-red-600/20'

                  }

                >

                  {item.label}

                </NavLink>

              </li>

            ))}

            <li className='pt-2'>

              <Link

                to='/events'

                onClick={() => setOpen(false)}

                className='block rounded-xl bg-red-600 px-3 py-2 text-center text-base font-semibold text-white shadow-md shadow-red-600/30 transition hover:bg-red-500'

              >

                Browse Events

              </Link>

            </li>

          </ul>

        </div>

      )}

    </header>

  );

}

Footer.tsx:

import { Link } from 'react-router-dom';

import { FaTwitter, FaInstagram, FaYoutube, FaHeart } from 'react-icons/fa';

export default function Footer() {

  return (

    <footer className='mt-12 border-t border-white/10 bg-black'>

      <div className='mx-auto max-w-7xl px-4 py-10 sm:px-6 lg:px-8'>

        <div className='grid gap-8 sm:grid-cols-2 lg:grid-cols-4'>

          <div className='space-y-3'>

            <div className='flex items-center gap-2'>

              <span className='grid h-8 w-8 place-items-center rounded-lg bg-red-600 text-black font-black'>

                E

              </span>

              <span className='text-lg font-semibold text-white'>Eventim</span>

            </div>

            <p className='text-sm text-gray-400'>

              Discover, follow, and review concerts and festivals. A black and

              red rhythm for music lovers.

            </p>

          </div>

          <div>

            <h3 className='text-sm font-semibold uppercase tracking-wider text-gray-300'>

              Navigation

            </h3>

            <ul className='mt-3 space-y-2 text-sm'>

              <li>

                <Link className='text-gray-400 hover:text-white' to='/'>

                  Home

                </Link>

              </li>

              <li>

                <Link className='text-gray-400 hover:text-white' to='/events'>

                  Events

                </Link>

              </li>

              <li>

                <Link className='text-gray-400 hover:text-white' to='/artists'>

                  Artists

                </Link>

              </li>

            </ul>

          </div>

          <div>

            <h3 className='text-sm font-semibold uppercase tracking-wider text-gray-300'>

              Resources

            </h3>

            <ul className='mt-3 space-y-2 text-sm'>

              <li>

                <a

                  className='text-gray-400 hover:text-white'

                  href='#'

                  onClick={(e) => e.preventDefault()}

                >

                  Terms of Service

                </a>

              </li>

              <li>

                <a

                  className='text-gray-400 hover:text-white'

                  href='#'

                  onClick={(e) => e.preventDefault()}

                >

                  Privacy Policy

                </a>

              </li>

            </ul>

          </div>

          <div>

            <h3 className='text-sm font-semibold uppercase tracking-wider text-gray-300'>

              Follow us

            </h3>

            <div className='mt-3 flex gap-3'>

              <a

                href='#'

                aria-label='Twitter'

                onClick={(e) => e.preventDefault()}

                className='inline-flex h-9 w-9 items-center justify-center rounded-lg border border-white/10 text-gray-300 hover:border-red-600 hover:bg-red-600/20 hover:text-white transition'

              >

                <FaTwitter className='h-5 w-5' />

              </a>

              <a

                href='#'

                aria-label='Instagram'

                onClick={(e) => e.preventDefault()}

                className='inline-flex h-9 w-9 items-center justify-center rounded-lg border border-white/10 text-gray-300 hover:border-red-600 hover:bg-red-600/20 hover:text-white transition'

              >

                <FaInstagram className='h-5 w-5' />

              </a>

              <a

                href='#'

                aria-label='YouTube'

                onClick={(e) => e.preventDefault()}

                className='inline-flex h-9 w-9 items-center justify-center rounded-lg border border-white/10 text-gray-300 hover:border-red-600 hover:bg-red-600/20 hover:text-white transition'

              >

                <FaYoutube className='h-5 w-5' />

              </a>

            </div>

          </div>

        </div>

        <div className='mt-10 flex flex-col items-center justify-between gap-4 border-t border-white/10 pt-6 sm:flex-row'>

          <p className='text-xs text-gray-500'>

            © {new Date().getFullYear()} Eventim. All rights reserved.

          </p>

          <p className='text-xs text-gray-500 flex items-center'>

            Made with{' '}

            <span className='text-red-500'>

              <FaHeart className='text-red-500 mx-1' />

            </span>{' '}

            using React & Tailwind.

          </p>

        </div>

      </div>

    </footer>

  );

}

Home.tsx:  
import { Link } from 'react-router-dom';

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

import { useEventim } from '../store/eventim';

import { loadFromTicketmasterAndSet } from '../api/sync';

import EventCard, { EventCardSkeleton } from '../components/EventCard';

export default function Home() {

  const { events, artists, lastSyncSource, lastSyncAt } = useEventim();

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

  const [err, setErr] = useState<string | null>(null);

  useEffect(() => {

    const hasKey = Boolean(import.meta.env.VITE\_TICKETMASTER\_KEY);

    if (!hasKey || lastSyncSource) return;

    (async () => {

      try {

        setLoading(true);

        await loadFromTicketmasterAndSet({

          countryCode: 'US',

          keyword: 'music',

          size: 24,

        });

      } catch (e: any) {

        setErr(e?.message ?? 'Failed to fetch Ticketmaster data.');

      } finally {

        setLoading(false);

      }

    })();

  }, [lastSyncSource]);

  // Map artistId -> artist (for EventCard to show names)

  const artistById = useMemo(

    () => Object.fromEntries(artists.map((a) => [a.id, a])),

    [artists]

  );

  // Pick 3 upcoming events (or earliest if none upcoming)

  const featuredEvents = useMemo(() => {

    const now = Date.now();

    const byTime = [...events].sort(

      (a, b) => new Date(a.datetime).getTime() - new Date(b.datetime).getTime()

    );

    const upcoming = byTime.filter(

      (e) => new Date(e.datetime).getTime() >= now

    );

    return (upcoming.length ? upcoming : byTime).slice(0, 3);

  }, [events]);

  // Top 3 artists by frequency in events (fallback to first 3)

  const topArtists = useMemo(() => {

    if (!events.length || !artists.length) return artists.slice(0, 3);

    const counts: Record<string, number> = {};

    for (const ev of events)

      for (const id of ev.artistIds) counts[id] = (counts[id] ?? 0) + 1;

    const sorted = Object.entries(counts)

      .sort((a, b) => b[1] - a[1])

      .map(([id]) => artists.find((a) => a.id === id))

      .filter(Boolean) as typeof artists;

    const fill = [...sorted, ...artists.filter((a) => !sorted.includes(a))];

    return fill.slice(0, 3);

  }, [events, artists]);

  return (

    <div className='pb-12'>

      {/\* HERO \*/}

      <section className='relative isolate overflow-hidden rounded-2xl border border-white/10 bg-gradient-to-b from-black to-[#0b0b0d] px-6 py-16 sm:px-10 sm:py-20 lg:px-16'>

        <div className='pointer-events-none absolute inset-0 -z-10'>

          <div className='absolute -top-24 right-[-10%] h-72 w-72 rounded-full bg-red-600/20 blur-3xl' />

          <div className='absolute -bottom-24 left-[-10%] h-72 w-72 rounded-full bg-red-600/10 blur-3xl' />

        </div>

        <div className='mx-auto max-w-3xl text-center'>

          <span className='inline-flex items-center gap-2 rounded-full border border-white/10 bg-black/50 px-3 py-1 text-xs text-gray-300'>

            <span className='inline-block h-1.5 w-1.5 rounded-full bg-red-500' />

            Live music tracker

          </span>

          <h1 className='mt-4 text-4xl font-extrabold tracking-tight text-white sm:text-5xl'>

            Discover concerts & festivals.{' '}

            <span className='text-red-500'>Review</span>,{' '}

            <span className='text-red-500'>reserve</span>, and follow your

            favorite artists.

          </h1>

          <p className='mt-4 text-base text-gray-300 sm:text-lg'>

            Eventim keeps your music calendar in one place — browse events, read

            reviews, and save your seat in a few clicks.

          </p>

          <div className='mt-6 flex flex-col items-center justify-center gap-3 text-xs text-gray-400 sm:flex-row'>

            {loading && (

              <span className='text-red-300'>Loading live data…</span>

            )}

            {err && <span className='text-red-300'>{err}</span>}

            {lastSyncAt && (

              <span className='opacity-80'>

                Synced from {lastSyncSource} ·{' '}

                {new Date(lastSyncAt).toLocaleString()}

              </span>

            )}

          </div>

          <div className='mt-8 flex flex-col items-center justify-center gap-3 sm:flex-row'>

            <Link

              to='/events'

              className='w-full rounded-xl bg-red-600 px-5 py-3 text-center text-sm font-semibold text-white shadow-md shadow-red-600/30 transition hover:bg-red-500 sm:w-auto'

            >

              Browse Events

            </Link>

            <Link

              to='/artists'

              className='w-full rounded-xl border border-white/15 bg-white/5 px-5 py-3 text-center text-sm font-semibold text-gray-200 transition hover:bg-red-600/20 hover:text-white sm:w-auto'

            >

              Explore Artists

            </Link>

          </div>

        </div>

      </section>

      {/\* FEATURED EVENTS \*/}

      <section className='mt-12'>

        <div className='mb-6 flex items-end justify-between'>

          <div>

            <h2 className='text-2xl font-semibold text-white'>

              Featured Events

            </h2>

            <p className='text-sm text-gray-400'>

              Handpicked picks happening soon around the region.

            </p>

          </div>

          <Link

            to='/events'

            className='hidden rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white sm:inline-block'

          >

            View all

          </Link>

        </div>

        <div className='grid gap-6 sm:grid-cols-2 lg:grid-cols-3'>

          {loading && featuredEvents.length === 0

            ? [0, 1, 2].map((i) => <EventCardSkeleton key={i} />)

            : featuredEvents.map((ev) => (

                <EventCard key={ev.id} event={ev} artistById={artistById} />

              ))}

        </div>

        <div className='mt-6 text-center sm:hidden'>

          <Link

            to='/events'

            className='inline-block rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white'

          >

            View all

          </Link>

        </div>

      </section>

      {/\* TOP ARTISTS (unchanged for now) \*/}

      <section className='mt-12'>

        <div className='mb-6 flex items-end justify-between'>

          <div>

            <h2 className='text-2xl font-semibold text-white'>Top Artists</h2>

            <p className='text-sm text-gray-400'>

              Trending performers you might like.

            </p>

          </div>

          <Link

            to='/artists'

            className='hidden rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white sm:inline-block'

          >

            View all

          </Link>

        </div>

        <div className='grid gap-6 sm:grid-cols-2 lg:grid-cols-3'>

          {topArtists.map((a) => (

            <Link

              key={a.id}

              to={`/artists/${a.id}`}

              className='group overflow-hidden rounded-2xl border border-white/10 bg-[#0f0f13] transition hover:-translate-y-0.5 hover:border-red-600/50'

            >

              <div className='aspect-[4/3] w-full overflow-hidden'>

                <img

                  src={a.imageUrl || '/avatar-placeholder.jpg'}

                  alt={a.name}

                  className='h-full w-full object-cover transition duration-300 group-hover:scale-105'

                  loading='lazy'

                />

              </div>

              <div className='space-y-1 p-4'>

                <h3 className='line-clamp-1 text-base font-semibold text-white'>

                  {a.name}

                </h3>

                <p className='line-clamp-1 text-sm text-gray-400'>

                  {a.genre?.join(', ') ?? 'Artist'}

                </p>

              </div>

            </Link>

          ))}

        </div>

        <div className='mt-6 text-center sm:hidden'>

          <Link

            to='/artists'

            className='inline-block rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white'

          >

            View all

          </Link>

        </div>

      </section>

    </div>

  );

}

EventCard.tsx:

import { Link } from 'react-router-dom';

import type { IArtist, IEvent } from '../data/types';

import { EventEntity } from '../data/domain/EventEntity';

type EventLike = IEvent | EventEntity;

type EventCardProps = {

  event: EventLike;

  artistById?: Record<string, Pick<IArtist, 'name'> | undefined>;

  to?: string;

  className?: string;

  imageClassName?: string;

  showArtists?: boolean;

  placeholderSrc?: string;

};

export default function EventCard({

  event,

  artistById,

  to,

  className,

  imageClassName,

  showArtists = true,

  placeholderSrc = '/placeholder.png',

}: EventCardProps) {

  const href = to ?? `/events/${event.id}`;

  const dateStr =

    (event as any).formattedDate?.() ||

    new Intl.DateTimeFormat(navigator.language, {

      dateStyle: 'medium',

      timeStyle: 'short',

    }).format(new Date(event.datetime));

  const artists =

    event.artistIds

      ?.map((id) => artistById?.[id]?.name)

      .filter(Boolean)

      .join(' · ') || 'Various artists';

  return (

    <Link

      to={href}

      aria-label={event.title}

      className={

        'group overflow-hidden rounded-2xl border border-white/10 bg-[#0f0f13] transition hover:-translate-y-0.5 hover:border-red-600/40 ' +

        (className ?? '')

      }

    >

      <div

        className={

          'aspect-[16/9] w-full overflow-hidden ' + (imageClassName ?? '')

        }

      >

        <img

          src={event.imageUrl || placeholderSrc}

          alt={event.title}

          className='h-full w-full object-cover transition duration-300 group-hover:scale-105'

          loading='lazy'

        />

      </div>

      <div className='space-y-2 p-4'>

        <div className='inline-flex items-center gap-2'>

          <span className='rounded-md bg-red-600/20 px-2 py-0.5 text-xs font-medium text-red-300'>

            {event.type}

          </span>

          <span className='text-xs text-gray-400'>{dateStr}</span>

        </div>

        <h3 className='line-clamp-1 text-base font-semibold text-white'>

          {event.title}

        </h3>

        <p className='line-clamp-1 text-sm text-gray-400'>

          {event.venue.name} · {event.venue.city}, {event.venue.country}

        </p>

        {showArtists && (

          <p className='line-clamp-1 text-xs text-gray-500'>{artists}</p>

        )}

      </div>

    </Link>

  );

}

export function EventCardSkeleton() {

  return (

    <div className='overflow-hidden rounded-2xl border border-white/10 bg-[#0f0f13]'>

      <div className='aspect-[16/9] w-full animate-pulse bg-white/5' />

      <div className='space-y-3 p-4'>

        <div className='h-3 w-24 animate-pulse rounded bg-white/10' />

        <div className='h-4 w-3/4 animate-pulse rounded bg-white/10' />

        <div className='h-3 w-1/2 animate-pulse rounded bg-white/10' />

      </div>

    </div>

  );

}

ArtistCard.tsx:

import { Link } from 'react-router-dom';

import type { IArtist } from '../data/types';

type Props = {

  artist: IArtist;

  to?: string;

  eventCount?: number;

  className?: string;

  placeholderSrc?: string;

};

export default function ArtistCard({

  artist,

  to,

  eventCount,

  className,

  placeholderSrc = '/avatar-placeholder.jpg',

}: Props) {

  const href = to ?? `/artists/${artist.id}`;

  return (

    <Link

      to={href}

      aria-label={artist.name}

      className={

        'group overflow-hidden rounded-2xl border border-white/10 bg-[#0f0f13] transition hover:-translate-y-0.5 hover:border-red-600/40 ' +

        (className ?? '')

      }

    >

      <div className='aspect-[4/3] w-full overflow-hidden'>

        <img

          src={artist.imageUrl || placeholderSrc}

          alt={artist.name}

          className='h-full w-full object-cover transition duration-300 group-hover:scale-105'

          loading='lazy'

        />

      </div>

      <div className='space-y-1 p-4'>

        <h3 className='line-clamp-1 text-base font-semibold text-white'>

          {artist.name}

        </h3>

        <p className='line-clamp-1 text-sm text-gray-400'>

          {artist.genre?.join(', ') ?? 'Artist'}

          {artist.country ? ` · ${artist.country}` : ''}

        </p>

        {typeof eventCount === 'number' && (

          <p className='text-xs text-gray-500'>{eventCount} event(s)</p>

        )}

      </div>

    </Link>

  );

}

export function ArtistCardSkeleton() {

  return (

    <div className='overflow-hidden rounded-2xl border border-white/10 bg-[#0f0f13]'>

      <div className='aspect-[4/3] w-full animate-pulse bg-white/5' />

      <div className='space-y-3 p-4'>

        <div className='h-4 w-3/4 animate-pulse rounded bg-white/10' />

        <div className='h-3 w-1/2 animate-pulse rounded bg-white/10' />

      </div>

    </div>

  );

}

Pagination.tsx:

import React from 'react';

type Props = {

  page: number;

  pageSize: number;

  total: number;

  onPageChange: (next: number) => void;

  className?: string;

};

export default function Pagination({

  page,

  pageSize,

  total,

  onPageChange,

  className,

}: Props) {

  const totalPages = Math.max(1, Math.ceil(total / pageSize));

  if (totalPages <= 1) return null;

  const start = (page - 1) \* pageSize + 1;

  const end = Math.min(total, page \* pageSize);

  const makeBtn = (p: number, label?: React.ReactNode, isActive?: boolean) => (

    <button

      key={`p-${label ?? p}`}

      onClick={() => onPageChange(p)}

      disabled={p === page || p < 1 || p > totalPages}

      className={[

        'inline-flex items-center justify-center rounded-lg border px-3 py-1.5 text-sm transition',

        isActive

          ? 'border-red-600 bg-red-600 text-white'

          : 'border-white/10 bg-white/5 text-gray-200 hover:bg-red-600/20 hover:text-white disabled:opacity-60',

      ].join(' ')}

      aria-current={isActive ? 'page' : undefined}

      aria-label={typeof label === 'string' ? label : `Page ${p}`}

    >

      {label ?? p}

    </button>

  );

  const nums: (number | string)[] = [];

  const push = (v: number | string) => nums.push(v);

  const window = 1;

  const pages = new Set<number>([

    1,

    totalPages,

    page - window,

    page,

    page + window,

  ]);

  const list = Array.from(pages)

    .filter((p) => p >= 1 && p <= totalPages)

    .sort((a, b) => a - b);

  for (let i = 0; i < list.length; i++) {

    const cur = list[i];

    const prev = list[i - 1];

    if (i && prev !== undefined && cur - prev > 1) push('…');

    push(cur);

  }

  return (

    <div

      className={

        'mt-6 flex flex-col items-center justify-between gap-3 sm:flex-row ' +

        (className ?? '')

      }

    >

      <div className='text-xs text-gray-400'>

        Showing <span className='text-gray-200'>{start}</span>–

        <span className='text-gray-200'>{end}</span> of{' '}

        <span className='text-gray-200'>{total}</span>

      </div>

      <nav className='flex items-center gap-2' aria-label='Pagination'>

        {makeBtn(page - 1, 'Prev')}

        {nums.map((n, i) =>

          typeof n === 'number' ? (

            makeBtn(n, undefined, n === page)

          ) : (

            <span key={`dots-${i}`} className='px-2 text-sm text-gray-500'>

              {n}

            </span>

          )

        )}

        {makeBtn(page + 1, 'Next')}

      </nav>

    </div>

  );

}

Events.tsx:

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

import { useEventim } from '../store/eventim';

import { loadFromTicketmasterAndSet } from '../api/sync';

import EventCard from '../components/EventCard';

import Pagination from '../components/Pagination';

const PAGE\_SIZE = 20;

export default function Events() {

  const { artists, filters, setFilters, getEventEntities } = useEventim();

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

  const [err, setErr] = useState<string | null>(null);

  const [country, setCountry] = useState('US');

  const [kw, setKw] = useState('music');

  const [page, setPage] = useState(1);

  const artistById = useMemo(

    () => Object.fromEntries(artists.map((a) => [a.id, a])),

    [artists]

  );

  async function onFetch() {

    try {

      setLoading(true);

      setErr(null);

      await loadFromTicketmasterAndSet({

        countryCode: country,

        keyword: kw,

        size: 50,

      });

      setPage(1);

    } catch (e: any) {

      setErr(e?.message ?? 'Failed to fetch data.');

    } finally {

      setLoading(false);

    }

  }

  const entities = getEventEntities();

  const filtered = useMemo(() => {

    const res = entities

      .filter((ev) => {

        if (filters.typeFilter !== 'all' && ev.type !== filters.typeFilter)

          return false;

        if (

          filters.artistFilter &&

          !ev.artistIds.includes(filters.artistFilter)

        )

          return false;

        const hay = `${ev.title} ${ev.venue.city} ${ev.venue.country} ${

          ev.venue.name

        } ${ev.artistIds

          .map((id) => artistById[id]?.name ?? '')

          .join(' ')}`.toLowerCase();

        return hay.includes(filters.search.toLowerCase());

      })

      .sort(

        (a, b) =>

          new Date(a.datetime).getTime() - new Date(b.datetime).getTime()

      );

    return res;

  }, [entities, filters, artistById]);

  useEffect(() => {

    setPage(1);

  }, [filters, filtered.length]);

  const total = filtered.length;

  const start = (page - 1) \* PAGE\_SIZE;

  const paginated = filtered.slice(start, start + PAGE\_SIZE);

  return (

    <div className='py-8'>

      <div className='mb-4 grid gap-3 sm:grid-cols-2 lg:grid-cols-4'>

        <div className='flex gap-2'>

          <input

            value={filters.search}

            onChange={(e) => setFilters({ search: e.target.value })}

            placeholder='Search events, cities, artists...'

            className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 placeholder:text-gray-500 focus:outline-none focus:ring-2 focus:ring-red-500/60'

          />

        </div>

        <select

          value={filters.typeFilter}

          onChange={(e) => setFilters({ typeFilter: e.target.value as any })}

          className='rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 focus:outline-none focus:ring-2 focus:ring-red-500/60'

        >

          <option value='all'>All types</option>

          <option value='concert'>Concert</option>

          <option value='festival'>Festival</option>

        </select>

        <select

          value={filters.artistFilter ?? ''}

          onChange={(e) => setFilters({ artistFilter: e.target.value || null })}

          className='rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 focus:outline-none focus:ring-2 focus:ring-red-500/60'

        >

          <option value=''>All artists</option>

          {artists.map((a) => (

            <option key={a.id} value={a.id}>

              {a.name}

            </option>

          ))}

        </select>

        <div className='flex items-center gap-2'>

          <select

            value={country}

            onChange={(e) => setCountry(e.target.value)}

            className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 focus:outline-none focus:ring-2 focus:ring-red-500/60'

            title='Country (ISO 2)'

          >

            <option value='DE'>Germany</option>

            <option value='US'>USA</option>

            <option value='GB'>UK</option>

            <option value='IT'>Italy</option>

            <option value='FR'>France</option>

          </select>

          <input

            value={kw}

            onChange={(e) => setKw(e.target.value)}

            placeholder='Keyword (e.g., music, concert, artist)'

            className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 placeholder:text-gray-500 focus:outline-none focus:ring-2 focus:ring-red-500/60'

          />

          <button

            onClick={onFetch}

            disabled={loading}

            className='whitespace-nowrap rounded-lg bg-red-600 px-3 py-2 text-sm font-semibold text-white shadow-md shadow-red-600/30 transition hover:bg-red-500 disabled:opacity-60'

          >

            {loading ? 'Loading...' : 'Load from Ticketmaster'}

          </button>

        </div>

      </div>

      {err && (

        <div className='mb-4 rounded-lg border border-red-500/30 bg-red-500/10 px-3 py-2 text-sm text-red-200'>

          {err}

        </div>

      )}

      <div className='mb-4 text-sm text-gray-400'>

        Showing <span className='text-gray-200'>{paginated.length}</span> on

        this page of <span className='text-gray-200'>{total}</span> events

      </div>

      <div className='grid gap-6 sm:grid-cols-2 lg:grid-cols-3'>

        {paginated.map((ev) => (

          <EventCard key={ev.id} event={ev} artistById={artistById} />

        ))}

      </div>

      {total === 0 && (

        <div className='mt-10 rounded-xl border border-white/10 bg-black/40 p-6 text-center text-gray-400'>

          No events match your filters. Try changing the country or keyword.

        </div>

      )}

      <Pagination

        page={page}

        pageSize={PAGE\_SIZE}

        total={total}

        onPageChange={setPage}

      />

    </div>

  );

}

EventDetails.tsx:

import { Link, useParams } from 'react-router-dom';

import { useMemo, useState } from 'react';

import { useEventim } from '../store/eventim';

import { EventEntity } from '../data/domain/EventEntity';

import EventCard from '../components/EventCard';

export default function EventDetail() {

  const { id: eventIdParam, eventId: eventIdAlt } = useParams();

  const eventId = eventIdParam ?? eventIdAlt ?? '';

  const { events, artists, reviews } = useEventim();

  const raw = events.find((e) => e.id === eventId);

  const event = raw ? new EventEntity(raw) : null;

  const artistById = useMemo(

    () => Object.fromEntries(artists.map((a) => [a.id, a])),

    [artists]

  );

  const eventReviews = useMemo(

    () =>

      reviews.filter((r) => r.targetType === 'event' && r.targetId === eventId),

    [reviews, eventId]

  );

  const avgRating =

    eventReviews.length > 0

      ? (

          eventReviews.reduce((s, r) => s + r.rating, 0) / eventReviews.length

        ).toFixed(1)

      : null;

  // Related events: first same city, else same artist(s)

  const related = useMemo(() => {

    if (!event) return [];

    const sameCity = events

      .filter((e) => e.id !== event.id && e.venue.city === event.venue.city)

      .slice(0, 3);

    if (sameCity.length) return sameCity;

    const byArtist = events

      .filter(

        (e) =>

          e.id !== event.id &&

          e.artistIds.some((id) => event.artistIds.includes(id))

      )

      .slice(0, 3);

    return byArtist;

  }, [events, event]);

  if (!event) {

    return (

      <div className='py-16 text-center'>

        <h1 className='text-2xl font-semibold text-white'>Event not found</h1>

        <p className='mt-2 text-sm text-gray-400'>

          The event you’re looking for doesn’t exist or hasn’t been loaded yet.

        </p>

        <Link

          to='/events'

          className='mt-6 inline-block rounded-lg bg-red-600 px-4 py-2 text-sm font-semibold text-white hover:bg-red-500'

        >

          Back to Events

        </Link>

      </div>

    );

  }

  return (

    <div className='py-8'>

      {/\* Header / Summary \*/}

      <section className='relative overflow-hidden rounded-2xl border border-white/10 bg-[#0f0f13]'>

        <div className='grid gap-6 p-6 md:grid-cols-[minmax(280px,42%)\_1fr] md:p-8'>

          {/\* Image \*/}

          <div className='overflow-hidden rounded-xl border border-white/10 bg-white/5'>

            <img

              src={event.imageUrl || '/placeholder.png'}

              alt={event.title}

              className='h-full w-full object-cover'

              loading='lazy'

            />

          </div>

          {/\* Info \*/}

          <div className='space-y-4'>

            <div className='inline-flex items-center gap-2'>

              <span className='rounded-md bg-red-600/20 px-2 py-0.5 text-xs font-medium text-red-300'>

                {event.type}

              </span>

              {avgRating && (

                <span className='rounded-md bg-white/5 px-2 py-0.5 text-xs text-yellow-300'>

                  ★ {avgRating} ({eventReviews.length})

                </span>

              )}

            </div>

            <h1 className='text-2xl font-bold text-white sm:text-3xl'>

              {event.title}

            </h1>

            <p className='text-sm text-gray-300'>

              <span className='font-medium'>{event.formattedDate()}</span>

              <span className='mx-2 text-gray-500'>•</span>

              {event.venue.name} · {event.venue.city}, {event.venue.country}

            </p>

            {/\* Artists \*/}

            <div className='flex flex-wrap gap-2'>

              {event.artistIds.length ? (

                event.artistIds.map((id: string) => {

                  const a = artistById[id];

                  return a ? (

                    <Link

                      key={id}

                      to={`/artists/${id}`}

                      className='rounded-full border border-white/10 bg-white/5 px-3 py-1 text-xs text-gray-200 transition hover:bg-red-600/20 hover:text-white'

                    >

                      {a.name}

                    </Link>

                  ) : (

                    <span

                      key={id}

                      className='rounded-full border border-white/10 bg-white/5 px-3 py-1 text-xs text-gray-400'

                    >

                      Unknown artist

                    </span>

                  );

                })

              ) : (

                <span className='text-xs text-gray-400'>Artists: —</span>

              )}

            </div>

            {/\* Description \*/}

            {event.description ? (

              <p className='whitespace-pre-line text-sm leading-relaxed text-gray-300'>

                {event.description}

              </p>

            ) : (

              <p className='text-sm text-gray-400'>

                No description provided for this event.

              </p>

            )}

            {/\* CTAs \*/}

            <div className='flex flex-wrap gap-2 pt-2'>

              <Link

                to='/events'

                className='rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white'

              >

                Back to Events

              </Link>

            </div>

          </div>

        </div>

      </section>

      {/\* RESERVATIONS \*/}

      <ReservationSection eventId={event.id} />

      {/\* REVIEWS \*/}

      <ReviewsSection eventId={event.id} />

      {/\* Related events \*/}

      <section className='mt-12'>

        <div className='mb-4 flex items-end justify-between'>

          <div>

            <h2 className='text-xl font-semibold text-white'>

              You may also like

            </h2>

            <p className='text-sm text-gray-400'>

              More shows {event.venue.city ? `in ${event.venue.city}` : ''}{' '}

              {event.venue.city ? 'or with similar artists.' : ''}

            </p>

          </div>

          <Link

            to='/events'

            className='hidden rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white md:inline-block'

          >

            Browse all events

          </Link>

        </div>

        <RelatedGrid events={related} artistById={artistById} />

      </section>

    </div>

  );

}

/\* ---------- Reservations (list + form) ---------- \*/

function ReservationSection({ eventId }: { eventId: string }) {

  const { reservations, addReservation, events } = useEventim();

  const event = events.find((e) => e.id === eventId);

  const myReservations = reservations.filter((r) => r.eventId === eventId);

  const [name, setName] = useState('');

  const [email, setEmail] = useState('');

  const [qty, setQty] = useState(1);

  const [msg, setMsg] = useState<{ ok?: string; err?: string } | null>(null);

  function maskEmail(v: string) {

    const [u, d] = v.split('@');

    if (!u || !d) return v;

    const safeUser =

      u.length <= 2

        ? u[0] + '\*'

        : u[0] + '\*'.repeat(Math.max(1, u.length - 2)) + u[u.length - 1];

    return `${safeUser}@${d}`;

  }

  function onSubmit(e: React.FormEvent) {

    e.preventDefault();

    setMsg(null);

    if (!name.trim()) return setMsg({ err: 'Please enter your name.' });

    if (!/^\S+@\S+\.\S+$/.test(email))

      return setMsg({ err: 'Please enter a valid email.' });

    if (qty < 1 || qty > 10)

      return setMsg({ err: 'Quantity must be between 1 and 10.' });

    addReservation({ eventId, name: name.trim(), email: email.trim(), qty });

    setName('');

    setEmail('');

    setQty(1);

    setMsg({ ok: 'Reservation saved locally.' });

  }

  return (

    <section className='mt-12' id='reserve'>

      <div className='mb-3 flex items-center justify-between'>

        <h2 className='text-xl font-semibold text-white'>Reservations</h2>

        <div className='text-sm text-gray-400'>

          {myReservations.length} total

          {event ? (

            <>

              {' '}

              <span className='text-gray-600'>•</span>{' '}

              <span className='text-gray-500'>{event.title}</span>

            </>

          ) : null}

        </div>

      </div>

      <div className='grid gap-6 md:grid-cols-2'>

        {/\* List \*/}

        <div className='space-y-3'>

          {myReservations.length === 0 ? (

            <div className='rounded-xl border border-white/10 bg-black/40 p-4 text-sm text-gray-400'>

              No reservations yet — reserve your seat now.

            </div>

          ) : (

            myReservations.map((r) => (

              <div

                key={r.id}

                className='rounded-xl border border-white/10 bg-[#0f0f13] p-4'

              >

                <div className='text-sm text-gray-300'>

                  <span className='font-semibold text-white'>{r.name}</span>{' '}

                  <span className='mx-1 text-gray-500'>·</span>

                  <span className='text-gray-400'>

                    {maskEmail(r.email)}

                  </span>{' '}

                  <span className='mx-1 text-gray-500'>·</span>

                  <span className='text-gray-300'>{r.qty} ticket(s)</span>

                </div>

                <div className='mt-1 text-xs text-gray-500'>

                  {new Date(r.createdAt).toLocaleString()}

                </div>

              </div>

            ))

          )}

        </div>

        {/\* Form \*/}

        <form

          onSubmit={onSubmit}

          className='rounded-xl border border-white/10 bg-[#0f0f13] p-4'

        >

          <h3 className='mb-3 text-base font-semibold text-white'>

            Reserve tickets

          </h3>

          <p className='mb-3 text-xs text-gray-500'>

            Demo only — reservations are stored in your browser (no payment).

          </p>

          <div className='grid gap-3'>

            <div className='grid gap-2 sm:grid-cols-2'>

              <input

                value={name}

                onChange={(e) => setName(e.target.value)}

                placeholder='Your name'

                className='rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 placeholder:text-gray-500 focus:outline-none focus:ring-2 focus:ring-red-500/60'

              />

              <input

                value={email}

                onChange={(e) => setEmail(e.target.value)}

                placeholder='Email'

                type='email'

                className='rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 placeholder:text-gray-500 focus:outline-none focus:ring-2 focus:ring-red-500/60'

              />

            </div>

            <div className='flex items-center gap-2'>

              <label className='w-24 text-sm text-gray-400'>Quantity</label>

              <input

                value={qty}

                onChange={(e) => setQty(Number(e.target.value))}

                type='number'

                min={1}

                max={10}

                className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 focus:outline-none focus:ring-2 focus:ring-red-500/60'

              />

            </div>

            {msg?.err && (

              <div className='rounded-lg border border-red-500/30 bg-red-500/10 px-3 py-2 text-sm text-red-200'>

                {msg.err}

              </div>

            )}

            {msg?.ok && (

              <div className='rounded-lg border border-emerald-500/30 bg-emerald-500/10 px-3 py-2 text-sm text-emerald-200'>

                {msg.ok}

              </div>

            )}

            <div className='flex justify-end'>

              <button

                type='submit'

                className='rounded-lg bg-red-600 px-3 py-2 text-sm font-semibold text-white shadow-md shadow-red-600/30 transition hover:bg-red-500'

              >

                Save reservation

              </button>

            </div>

          </div>

        </form>

      </div>

    </section>

  );

}

/\* ---------- Reviews (list + form) ---------- \*/

function ReviewsSection({ eventId }: { eventId: string }) {

  const { reviews } = useEventim();

  const myReviews = reviews.filter(

    (r) => r.targetType === 'event' && r.targetId === eventId

  );

  const avg =

    myReviews.length > 0

      ? myReviews.reduce((s, r) => s + r.rating, 0) / myReviews.length

      : null;

  return (

    <section className='mt-12'>

      <div className='mb-3 flex items-center justify-between'>

        <h2 className='text-xl font-semibold text-white'>Reviews</h2>

        <div className='text-sm text-gray-400'>

          {avg ? (

            <>

              <span className='text-yellow-300'>★ {avg.toFixed(1)}</span>{' '}

              <span className='text-gray-500'>({myReviews.length})</span>

            </>

          ) : (

            <span className='text-gray-500'>No ratings yet</span>

          )}

        </div>

      </div>

      <div className='grid gap-6 md:grid-cols-2'>

        <ReviewsList reviews={myReviews} />

        <ReviewForm eventId={eventId} />

      </div>

    </section>

  );

}

function ReviewsList({

  reviews,

}: {

  reviews: {

    id: string;

    author: string;

    rating: number;

    comment: string;

    createdAt: string;

  }[];

}) {

  if (reviews.length === 0) {

    return (

      <div className='rounded-xl border border-white/10 bg-black/40 p-4 text-sm text-gray-400'>

        No reviews yet — be the first to leave one!

      </div>

    );

  }

  return (

    <div className='space-y-3'>

      {reviews.map((r) => (

        <div

          key={r.id}

          className='rounded-2xl border border-white/10 bg-[#0f0f13] p-4'

        >

          <div className='mb-1 text-sm text-gray-400'>

            <span className='font-semibold text-gray-200'>{r.author}</span>{' '}

            <span className='mx-1'>·</span>

            <span className='text-yellow-300'>

              {Array.from({ length: r.rating })

                .map(() => '★')

                .join('')}

            </span>

            <span className='mx-1'>·</span>

            <span className='text-xs text-gray-500'>

              {new Date(r.createdAt).toLocaleString()}

            </span>

          </div>

          <p className='text-sm text-gray-300'>{r.comment}</p>

        </div>

      ))}

    </div>

  );

}

function ReviewForm({ eventId }: { eventId: string }) {

  const { addReview } = useEventim();

  const [rating, setRating] = useState<number>(5);

  const [author, setAuthor] = useState<string>('');

  const [comment, setComment] = useState<string>('');

  const [msg, setMsg] = useState<{ ok?: string; err?: string } | null>(null);

  function onSubmit(e: React.FormEvent) {

    e.preventDefault();

    setMsg(null);

    if (!author.trim()) return setMsg({ err: 'Please enter your name.' });

    if (comment.trim().length < 5)

      return setMsg({ err: 'Comment must be at least 5 characters.' });

    addReview({

      targetType: 'event',

      targetId: eventId,

      rating: Math.max(1, Math.min(5, rating)) as 1 | 2 | 3 | 4 | 5,

      author: author.trim(),

      comment: comment.trim(),

    });

    setAuthor('');

    setComment('');

    setRating(5);

    setMsg({ ok: 'Thanks! Your review was saved locally.' });

  }

  return (

    <form

      onSubmit={onSubmit}

      className='rounded-xl border border-white/10 bg-[#0f0f13] p-4'

    >

      <h3 className='mb-3 text-base font-semibold text-white'>

        Leave a review

      </h3>

      <div className='grid gap-3'>

        <div className='flex items-center gap-2'>

          <label className='w-24 text-sm text-gray-400'>Rating</label>

          <select

            value={rating}

            onChange={(e) => setRating(Number(e.target.value))}

            className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 focus:outline-none focus:ring-2 focus:ring-red-500/60'

          >

            {[5, 4, 3, 2, 1].map((n) => (

              <option key={n} value={n}>

                {n} ★

              </option>

            ))}

          </select>

        </div>

        <div className='flex items-center gap-2'>

          <label className='w-24 text-sm text-gray-400'>Name</label>

          <input

            value={author}

            onChange={(e) => setAuthor(e.target.value)}

            placeholder='Your name'

            className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 placeholder:text-gray-500 focus:outline-none focus:ring-2 focus:ring-red-500/60'

          />

        </div>

        <div>

          <label className='mb-1 block text-sm text-gray-400'>Comment</label>

          <textarea

            value={comment}

            onChange={(e) => setComment(e.target.value)}

            placeholder='Share your experience…'

            rows={4}

            className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 placeholder:text-gray-500 focus:outline-none focus:ring-2 focus:ring-red-500/60'

          />

        </div>

        {msg?.err && (

          <div className='rounded-lg border border-red-500/30 bg-red-500/10 px-3 py-2 text-sm text-red-200'>

            {msg.err}

          </div>

        )}

        {msg?.ok && (

          <div className='rounded-lg border border-emerald-500/30 bg-emerald-500/10 px-3 py-2 text-sm text-emerald-200'>

            {msg.ok}

          </div>

        )}

        <div className='flex justify-end'>

          <button

            type='submit'

            className='rounded-lg bg-red-600 px-3 py-2 text-sm font-semibold text-white shadow-md shadow-red-600/30 transition hover:bg-red-500'

          >

            Submit review

          </button>

        </div>

      </div>

    </form>

  );

}

/\* ---------- Related grid helper ---------- \*/

function RelatedGrid({

  events,

  artistById,

}: {

  events: { id: string }[];

  artistById: Record<string, { name: string } | undefined>;

}) {

  return (

    <div className='grid gap-6 sm:grid-cols-2 lg:grid-cols-3'>

      {events.length ? (

        events.map((e: any) => (

          <EventCard key={e.id} event={e} artistById={artistById} />

        ))

      ) : (

        <div className='rounded-2xl border border-white/10 bg-[#0f0f13] p-6 text-center text-gray-400'>

          No related events found.

        </div>

      )}

    </div>

  );

}

Artists.tsx:

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

import { useEventim } from '../store/eventim';

import { loadFromTicketmasterAndSet } from '../api/sync';

import ArtistCard, { ArtistCardSkeleton } from '../components/ArtistCard';

import Pagination from '../components/Pagination';

const PAGE\_SIZE = 20;

export default function Artists() {

  const { artists, events } = useEventim();

  const [q, setQ] = useState('');

  const [genre, setGenre] = useState('');

  const [country, setCountry] = useState('');

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

  const [err, setErr] = useState<string | null>(null);

  const [page, setPage] = useState(1);

  const eventCountByArtist = useMemo(() => {

    const m: Record<string, number> = {};

    for (const ev of events)

      for (const id of ev.artistIds) m[id] = (m[id] ?? 0) + 1;

    return m;

  }, [events]);

  const genreOptions = useMemo(() => {

    const set = new Set<string>();

    for (const a of artists) (a.genre ?? []).forEach((g) => g && set.add(g));

    return Array.from(set).sort((a, b) => a.localeCompare(b));

  }, [artists]);

  const countryOptions = useMemo(() => {

    const set = new Set<string>();

    for (const a of artists) if (a.country) set.add(a.country);

    return Array.from(set).sort((a, b) => a.localeCompare(b));

  }, [artists]);

  const filtered = useMemo(() => {

    const query = q.trim().toLowerCase();

    return artists

      .filter((a) => {

        if (genre && !(a.genre ?? []).some((g) => g === genre)) return false;

        if (country && a.country !== country) return false;

        if (!query) return true;

        const hay = `${a.name} ${(a.genre ?? []).join(' ')} ${

          a.country ?? ''

        }`.toLowerCase();

        return hay.includes(query);

      })

      .sort((a, b) => a.name.localeCompare(b.name));

  }, [artists, q, genre, country]);

  useEffect(() => {

    setPage(1);

  }, [q, genre, country, filtered.length]);

  const total = filtered.length;

  const start = (page - 1) \* PAGE\_SIZE;

  const paginated = filtered.slice(start, start + PAGE\_SIZE);

  async function onRefresh() {

    try {

      setLoading(true);

      setErr(null);

      await loadFromTicketmasterAndSet({

        countryCode: 'US',

        keyword: 'music',

        size: 50,

      });

      setPage(1);

    } catch (e: any) {

      setErr(e?.message ?? 'Failed to fetch data.');

    } finally {

      setLoading(false);

    }

  }

  return (

    <div className='py-8'>

      <div className='mb-4 grid gap-3 sm:grid-cols-2 lg:grid-cols-4'>

        <input

          value={q}

          onChange={(e) => setQ(e.target.value)}

          placeholder='Search artists, genres, countries…'

          className='w-full rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 placeholder:text-gray-500 focus:outline-none focus:ring-2 focus:ring-red-500/60'

        />

        <select

          value={genre}

          onChange={(e) => setGenre(e.target.value)}

          className='rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 focus:outline-none focus:ring-2 focus:ring-red-500/60'

        >

          <option value=''>All genres</option>

          {genreOptions.map((g) => (

            <option key={g} value={g}>

              {g}

            </option>

          ))}

        </select>

        <select

          value={country}

          onChange={(e) => setCountry(e.target.value)}

          className='rounded-lg border border-white/10 bg-black/40 px-3 py-2 text-sm text-gray-200 focus:outline-none focus:ring-2 focus:ring-red-500/60'

        >

          <option value=''>All countries</option>

          {countryOptions.map((c) => (

            <option key={c} value={c}>

              {c}

            </option>

          ))}

        </select>

        <div className='flex items-center justify-end'>

          <button

            onClick={onRefresh}

            disabled={loading}

            className='rounded-lg bg-red-600 px-3 py-2 text-sm font-semibold text-white shadow-md shadow-red-600/30 transition hover:bg-red-500 disabled:opacity-60'

          >

            {loading ? 'Loading…' : 'Refresh from Ticketmaster'}

          </button>

        </div>

      </div>

      {err && (

        <div className='mb-4 rounded-lg border border-red-500/30 bg-red-500/10 px-3 py-2 text-sm text-red-200'>

          {err}

        </div>

      )}

      <div className='mb-4 text-sm text-gray-400'>

        Showing <span className='text-gray-200'>{paginated.length}</span> on

        this page of <span className='text-gray-200'>{total}</span> artists

      </div>

      <div className='grid gap-6 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4'>

        {loading && artists.length === 0

          ? [0, 1, 2, 3, 4, 5, 6, 7].map((i) => <ArtistCardSkeleton key={i} />)

          : paginated.map((a) => (

              <ArtistCard

                key={a.id}

                artist={a}

                eventCount={eventCountByArtist[a.id] ?? 0}

              />

            ))}

      </div>

      {total === 0 && (

        <div className='mt-10 rounded-xl border border-white/10 bg-black/40 p-6 text-center text-gray-400'>

          No artists match your filters. Try clearing genre or country filters.

        </div>

      )}

      <Pagination

        page={page}

        pageSize={PAGE\_SIZE}

        total={total}

        onPageChange={setPage}

      />

    </div>

  );

}

ArtistDetailts.tsx:

import { useMemo } from 'react';

import { Link, useParams } from 'react-router-dom';

import { useEventim } from '../store/eventim';

import EventCard from '../components/EventCard';

export default function ArtistDetail() {

  const { id: artistId = '' } = useParams();

  const { artists, events } = useEventim();

  const artist = artists.find((a) => a.id === artistId) || null;

  const artistEvents = useMemo(() => {

    const list = events.filter((e) => e.artistIds.includes(artistId));

    return list.sort(

      (a, b) => new Date(a.datetime).getTime() - new Date(b.datetime).getTime()

    );

  }, [events, artistId]);

  const artistById = useMemo(

    () => Object.fromEntries(artists.map((a) => [a.id, a])),

    [artists]

  );

  if (!artist) {

    return (

      <div className='py-16 text-center'>

        <h1 className='text-2xl font-semibold text-white'>Artist not found</h1>

        <p className='mt-2 text-sm text-gray-400'>

          The artist you’re looking for doesn’t exist or hasn’t been loaded yet.

        </p>

        <Link

          to='/artists'

          className='mt-6 inline-block rounded-lg bg-red-600 px-4 py-2 text-sm font-semibold text-white hover:bg-red-500'

        >

          Back to Artists

        </Link>

      </div>

    );

  }

  return (

    <div className='py-8'>

      {/\* Header \*/}

      <section className='relative overflow-hidden rounded-2xl border border-white/10 bg-[#0f0f13]'>

        <div className='grid gap-6 p-6 sm:grid-cols-[220px\_1fr] sm:p-8'>

          {/\* Image \*/}

          <div className='overflow-hidden rounded-xl border border-white/10 bg-white/5'>

            <img

              src={artist.imageUrl || '/avatar-placeholder.jpg'}

              alt={artist.name}

              className='h-full w-full object-cover'

              loading='lazy'

            />

          </div>

          {/\* Info \*/}

          <div className='space-y-3'>

            <h1 className='text-2xl font-bold text-white sm:text-3xl'>

              {artist.name}

            </h1>

            <div className='flex flex-wrap items-center gap-2 text-sm'>

              <span className='rounded-md bg-white/5 px-2 py-1 text-gray-300'>

                {artist.genre?.join(', ') || 'Genre: —'}

              </span>

              <span className='rounded-md bg-white/5 px-2 py-1 text-gray-300'>

                Country: {artist.country ?? '—'}

              </span>

              <span className='rounded-md bg-white/5 px-2 py-1 text-gray-300'>

                Events: {artistEvents.length}

              </span>

            </div>

            {artist.bio ? (

              <p className='max-w-3xl text-sm leading-relaxed text-gray-300'>

                {artist.bio}

              </p>

            ) : (

              <p className='max-w-3xl text-sm leading-relaxed text-gray-400'>

                No bio yet.

              </p>

            )}

            <div className='flex flex-wrap gap-2 pt-2'>

              <Link

                to='/artists'

                className='rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white'

              >

                Back to Artists

              </Link>

              <Link

                to='/events'

                className='rounded-lg border border-white/15 bg-white/5 px-3 py-2 text-sm text-gray-200 transition hover:bg-red-600/20 hover:text-white'

              >

                Browse Events

              </Link>

            </div>

          </div>

        </div>

      </section>

      {/\* Artist's Events \*/}

      <section className='mt-10'>

        <div className='mb-4 flex items-end justify-between'>

          <div>

            <h2 className='text-xl font-semibold text-white'>Shows</h2>

            <p className='text-sm text-gray-400'>

              Events featuring {artist.name}.

            </p>

          </div>

          <span className='text-sm text-gray-400'>

            {artistEvents.length} total

          </span>

        </div>

        <div className='grid gap-6 sm:grid-cols-2 lg:grid-cols-3'>

          {artistEvents.length > 0 ? (

            artistEvents.map((ev) => (

              <EventCard key={ev.id} event={ev} artistById={artistById} />

            ))

          ) : (

            <div className='rounded-2xl border border-white/10 bg-[#0f0f13] p-6 text-center text-gray-400'>

              No events for this artist yet. Try fetching data on the Events

              page.

            </div>

          )}

        </div>

      </section>

    </div>

  );

}

# Korisničko uputstvo

A screenshot of a music website

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a music festival

AI-generated content may be incorrect.

A screenshot of a music album

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.