

Prompt: I want to modify the search page - the location should be disabled and cleared when “auto-detect location” is enabled, the Keywords field should be a dropdown select with search and suggest functionality so that When a user starts typing, an autosuggest request should be made and corresponding items from the response should be shown using the ticketmaster api. When a request is made the component should show a spinner on the right and once a value is selected a cross icon should be visible that clears the input value

Code Generated:

```
import { useEffect, useMemo, useRef, useState } from "react";
import api from "../api/client";

type SuggestItem = { id: string; name: string };

interface Props {
  value: string;
  onChange: (v: string) => void; // fires on typing and on select
  placeholder?: string;
}

export default function KeywordAutocomplete({ value, onChange, placeholder }: Props) {
  const [open, setOpen] = useState(false);
  const [loading, setLoading] = useState(false);
  const [items, setItems] = useState<SuggestItem[]>([]);
  const boxRef = useRef<HTMLDivElement>(null);

  // debounce input
  const [q, setQ] = useState(value);
  useEffect(() => setQ(value), [value]);

  const debouncedQ = useDebounced(q, 250);

  // fetch suggestions
  useEffect(() => {
    let active = true;
    async function go() {
      if (!debouncedQ.trim()) {
        setItems([]);
        setLoading(false);
        return;
      }
      setLoading(true);
      try {
        const res = await api.get("/events/suggest", { params: { keyword: debouncedQ } });
        setItems(res.data);
      } catch (err) {
        console.error(err);
      }
    }
    go();
  }, [debouncedQ]);
}
```

```

// Ticketmaster returns attractions, venues, etc. We'll prefer attractions first.
const attractions = res.data?._embedded?.attractions ?? [];
const names: SuggestItem[] = attractions
  .map((a: any) => ({ id: a.id as string, name: a.name as string }))
  .filter((x) => !x.name);
if (active) setItems(names.slice(0, 8));
} catch (e) {
  if (active) setItems([]);
} finally {
  if (active) setLoading(false);
}
}
go();
return () => { active = false; };
}, [debouncedQ]);

// close on outside click
useEffect(() => {
  function onDocClick(e: MouseEvent) {
    if (!boxRef.current) return;
    if (!boxRef.current.contains(e.target as Node)) setOpen(false);
  }
  document.addEventListener("mousedown", onDocClick);
  return () => document.removeEventListener("mousedown", onDocClick);
}, []);

const showClear = value.trim().length > 0 && !loading;

return (
  <div className="relative" ref={boxRef}>
    <div className="relative">
      <input
        className="w-full border rounded px-3 py-2 pr-16 text-black"
        placeholder={placeholder ?? "Keyword"}
        value={value}
        onChange={(e) => {
          onChange(e.target.value);
          setQ(e.target.value);
          setOpen(true);
        }}
        onFocus={() => setOpen(true)}
      />
      /* Right-side controls */
      <div className="absolute inset-y-0 right-2 flex items-center gap-2">

```

```
{loading && (
  <div className="w-5 h-5 border-2 border-gray-400 border-t-transparent rounded-full
animate-spin" aria-label="Loading" />
)
{showClear && (
  <button
    type="button"
    onClick={() => {
      onChange("");
      setQ("");
      setItems([]);
      setOpen(false);
    }}
    className="w-6 h-6 rounded-full hover:bg-gray-200 flex items-center justify-center"
    aria-label="Clear"
    title="Clear"
  >
    {/* x icon */}
    <svg width="14" height="14" viewBox="0 0 20 20" fill="none">
      <path d="M5 5l10 10M15 5L5 15" stroke="black" strokeWidth="2"
strokeLinecap="round"/>
    </svg>
  </button>
)
</div>
</div>

{open && items.length > 0 && (
  <ul className="absolute z-20 mt-1 w-full max-h-64 overflow-auto rounded border
border-slate-300 bg-white shadow">
  {items.map(it) => (
    <li
      key={it.id + it.name}
      className="px-3 py-2 text-sm hover:bg-slate-100 cursor-pointer"
      onMouseDown={(e) => e.preventDefault()}
      onClick={() => {
        onChange(it.name);
        setOpen(false);
      }}
    >
      {it.name}
    </li>
  )))
</ul>
```

```

        )}
      </div>
    );
}
}

// simple debounce hook
function useDebounced<T>(value: T, delay = 300) {
  const [v, setV] = useState(value);
  useEffect(() => {
    const t = setTimeout(() => setV(value), delay);
    return () => clearTimeout(t);
  }, [value, delay]);
  return v;
}

```

Second Prompt: there is no dropdown happening with autocomplete. when i type only an x shows up below that does clear the box but there is no autocomplete happening. when the user starts typing the app should use the Ticketmaster suggest API to suggest autocomplete options

```

import { useEffect, useRef, useState } from "react";
import api from "../api/client";

type SuggestItem = { id: string; name: string };

interface Props {
  value: string;
  onChange: (v: string) => void;
  placeholder?: string;
  minChars?: number; // default 2
}

export default function KeywordAutocomplete({
  value,
  onChange,
  placeholder,
  minChars = 2
}: Props) {
  const [open, setOpen] = useState(false);
  const [loading, setLoading] = useState(false);
  const [items, setItems] = useState<SuggestItem[]>([]);
  const boxRef = useRef<HTMLDivElement>(null);

  // debounce

```

```

const [q, setQ] = useState(value);
useEffect(() => setQ(value), [value]);
const debouncedQ = useDebounced(q, 250);

// fetch suggestions
useEffect(() => {
  let active = true;

  async function fetchSuggest() {
    // don't query until minChars
    if (!debouncedQ || debouncedQ.trim().length < minChars) {
      if (active) setItems([]);
      setLoading(false);
      return;
    }
    setLoading(true);
    try {
      const res = await api.get("/events/suggest", { params: { keyword: debouncedQ } });

      const attractions = res.data?._embedded?.attractions ?? [];
      const venues = res.data?._embedded?.venues ?? [];

      const attNames: SuggestItem[] = attractions
        .map((a: any) => ({ id: a.id as string, name: a.name as string }))
        .filter((x) => !x.name);
      const venNames: SuggestItem[] = venues
        .map((v: any) => ({ id: v.id as string, name: v.name as string }))
        .filter((x) => !x.name);

      const merged = dedupeByName([...attNames, ...venNames]).slice(0, 10);
      if (active) setItems(merged);
    } catch (e) {
      if (active) setItems([]);
      console.error("Suggest failed", e);
    } finally {
      if (active) setLoading(false);
    }
  }
}

fetchSuggest();
return () => { active = false; };
}, [debouncedQ, minChars]);

// close on outside click

```

```

useEffect(() => {
  function onDocClick(e: MouseEvent) {
    if (!boxRef.current) return;
    if (!boxRef.current.contains(e.target as Node)) setOpen(false);
  }
  document.addEventListener("mousedown", onDocClick);
  return () => document.removeEventListener("mousedown", onDocClick);
}, []);

```

```

const showClear = value.trim().length > 0 && !loading;

```

```

return (
  <div className="relative" ref={boxRef}>
    <div className="relative">
      <input
        className="w-full border rounded px-3 py-2 pr-16 text-black"
        placeholder={placeholder ?? "Keyword"}
        value={value}
        onChange={(e) => {
          onChange(e.target.value);
          setQ(e.target.value);
          setOpen(true);
        }}
        onFocus={() => setOpen(true)}
      />
      {/* Right-side spinner & clear */}
      <div className="absolute inset-y-0 right-2 flex items-center gap-2">
        {loading && (
          <div
            className="w-5 h-5 border-2 border-gray-400 border-t-transparent rounded-full
animate-spin"
            aria-label="loading"
          />
        )}
        {showClear && (
          <button
            type="button"
            onClick={() => {
              onChange("");
              setQ("");
              setItems([]);
              setOpen(false);
            }}
          className="w-6 h-6 rounded-full hover:bg-gray-200 flex items-center justify-center"
        )}
      

```

```

        aria-label="Clear"
        title="Clear"
      >
        <svg width="14" height="14" viewBox="0 0 20 20" fill="none">
          <path d="M5 5l10 10M15 5L5 15" stroke="black" strokeWidth="2"
strokeLinecap="round"/>
        </svg>
      </button>
    )}
</div>
</div>

{open && items.length > 0 && (
  <ul className="absolute z-50 mt-1 w-full max-h-64 overflow-auto rounded border border-slate-300 bg-white shadow">
    {items.map((it) => (
      <li
        key={it.id + it.name}
        className="px-3 py-2 text-sm hover:bg-slate-100 cursor-pointer"
        onMouseDown={(e) => e.preventDefault()}
        onClick={() => {
          onChange(it.name);
          setOpen(false);
        }}
      >
        {it.name}
      </li>
    )));
  </ul>
)}
</div>
);
}

```

```

function dedupeByName(items: SuggestItem[]): SuggestItem[] {
  const seen = new Set<string>();
  const out: SuggestItem[] = [];
  for (const it of items) {
    const k = it.name.toLowerCase();
    if (!seen.has(k)) {
      seen.add(k);
      out.push(it);
    }
  }
}
```

```
    return out;
}

function useDebounced<T>(value: T, delay = 300) {
  const [v, setV] = useState(value);
  useEffect(() => {
    const t = setTimeout(() => setV(value), delay);
    return () => clearTimeout(t);
  }, [value, delay]);
  return v;
}
```

I used AI to vibe code the search page. I asked it to modify the existing page to start doing auto suggestions and a few other aspects like disabling the location. The autocomplete didn't work the first time, so I went back and specified using the Ticketmaster suggest API for the autocomplete and tried to explain what was happening. This fixed it and allowed me to set up the auto suggestion part of the form