LocationInput Component Explained for .NET Developers

A custom autocomplete input component for integrating LocationIQ suggestions into a React app using React Hook Form, Material UI, and TypeScript generics.

Component Signature & Props

```
type Props<T extends FieldValues> = {
    label: string;
} & UseControllerProps<T>;
```

- T is a generic form type (T extends FieldValues)
- UseControllerProps<T> provides React Hook Form integration props like name, control, and rules
- Custom label prop for the text field label

Form State Integration

```
const { field, fieldState } = useController({ ...props });
```

- field: includes value, onChange, onBlur connects to React Hook Form state
- fieldState: provides validation error state

Local Component State

```
const [loading, setLoading] = useState(false);
const [suggestions, setSuggestions] = useState<LocationIQSuggestion[]>([]);
const [inputValue, setInputValue] = useState(field.value || '');
```

- loading: shows loading indicator while fetching
- suggestions: stores fetched suggestions
- inputValue: current input box value

Sync Form Value with Input Value

```
useEffect(() => {
    if (field.value && typeof field.value === 'object') {
        setInputValue(field.value.venue || '');
    } else {
        setInputValue(field.value || '');
    }
}, [field.value]);
```

- Supports form value as either a string or an object
- Keeps inputValue in sync with form's state

LocationIQ API Setup

```
const locationUrl = 'https://api.locationiq.com/v1/autocomplete?...';
```

- Base URL with API key for fetching address suggestions

Fetch Suggestions with Debounce

```
const fetchSuggestions = useMemo(() => debounce(async (query: string) => {
    if (!query || query.length < 3) {
        setSuggestions([]);
        return;
    }
    setLoading(true);
    try {
        const res = await axios.get<LocationIQSuggestion[]>(`${locationUrl}q=${query}`);
        setSuggestions(res.data);
    } catch (error) {
        console.log(error);
    } finally {
        setLoading(false);
    }
}, 500), [locationUrl]);
```

- Uses debounce() to delay API call by 500ms
- Avoids sending requests on every keystroke
- useMemo() keeps it stable across renders

Handle Typing & Input Change

```
const handleChange = async (value: string) => {
    field.onChange(value);
    await fetchSuggestions(value);
};
```

- Updates form value
- Fetches suggestions from LocationIQ

Handle Selection from Suggestion List

```
const handleSelect = (location: LocationIQSuggestion) => {
    const city = location.address?.city || location.address?.town || location.address?.village;
    const venue = location.display_name;
    const latitude = location.lat;
    const longitude = location.lon;

    setInputValue(venue);
    field.onChange({ city, venue, latitude, longitude });
    setSuggestions([]);
};
```

- Sets full input value
- Passes selected location as an object to form state

UI Rendering with Material UI

```
<TextField
  {...props}
  value={inputValue}
  onChange={e => handleChange(e.target.value)}
  fullWidth
  variant="outlined"
  error={!!fieldState.error}
  helperText={fieldState.error?.message}
/>
```

Suggestion List:

```
{loading && <Typography>Loading...</Typography>}
{suggestions.length > 0 && (
  <List sx={{ border: 1 }}>
    {suggestions.map(suggestion => (
      <ListItemButton
        key={suggestion.place_id}
        onClick={() => handleSelect(suggestion)}
      >
        {suggestion.display_name}
      </ListItemButton>
    ))}
  </List>
)}
```

Form Usage Example

```
type EventForm = {
  location: {
    city: string;
    venue: string;
    latitude: string;
    longitude: string;
  };
};

<LocationInput<EventForm>
  name="location"
  control={control}
  label="Event Venue"
/>
```

You get autocompletion, form state syncing, validation, and clean API integration.

Summary

Feature	Purpose
useController	Connects component to React Hook Form
debounce + axios	Smart API calls for input suggestions
setInputValue	Local UI state management
field.onChange()	Pushes data back to the form
Material UI Components	Clean UI rendering and validation support