**Lists and Keys**

In React, "lists" and "keys" are essential concepts when rendering collections of elements.

Here’s an overview of both:

**Lists**

**Lists** in React refer to rendering multiple elements based on an array of data.

React allows you to create a list of components by mapping over an array and generating a set of components from it.

**Key Points about Lists:**

1. **Dynamic Rendering**: Lists are used to render multiple components or elements dynamically based on data.

const numbers = [1, 2, 3, 4, 5];

const listItems = numbers.map(number =>

<li key={number.toString()}>{number}</li>

);

1. **Array Mapping**: You typically use the map() method to iterate over an array and return a new array of JSX elements.

const items = ['Apple', 'Banana', 'Cherry'];

const list = items.map(item =>

<li key={item}>{item}</li>

);

1. **Component Lists**: You can also create lists of components:

const users = [{ id: 1, name: 'John' }, { id: 2, name: 'Jane' }];

const userList = users.map(user =>

<UserProfile key={user.id} name={user.name} />

);

1. **Flexibility**: Lists can be used to render anything from a list of numbers, strings, objects, or even other React components.
2. **Dynamic Content**: Lists are particularly useful for rendering data from an API or state.

**Keys**

**Keys** are a special attribute you should include when rendering lists of elements.

They help React identify which items have changed, been added, or been removed, which improves performance and helps React manage the list efficiently.

**Key Points about Keys:**

1. **Uniqueness**: Each element in a list must have a **unique key**. This helps React identify and track changes in the list.

const items = ['Apple', 'Banana', 'Cherry'];

const list = items.map((item, index) =>

<li key={index}>{item}</li>

);

1. **Stable Identifier**: Ideally, use a stable and unique identifier, like an ID from your data, rather than the array index, to avoid potential issues with reordering or filtering lists.

const users = [{ id: 1, name: 'John' }, { id: 2, name: 'Jane' }];

const userList = users.map(user =>

<UserProfile key={user.id} name={user.name} />

);

1. **Performance**: Properly using keys allows React to efficiently update the user interface. Without keys or with duplicate keys, React may not correctly manage element updates, leading to performance issues and rendering bugs.
2. **Key Prop**: Keys should be provided directly on the elements that are being generated in the list.

<ul>

{list.map(item => (

<li key={item.id}>{item.text}</li>

))}

</ul>

1. **Not Visible**: The key prop is used internally by React and does not get rendered to the DOM. It's solely for React’s use to track and manage elements.

**Summary**

* **Lists**: Used to render multiple elements or components based on an array of data. Achieved with array methods like map().
* **Keys**: Unique identifiers used to help React efficiently manage and update lists. Should be stable and unique for each element in the list.

Using lists and keys effectively ensures smooth and efficient rendering of dynamic content in your React applications.

**Example 01**

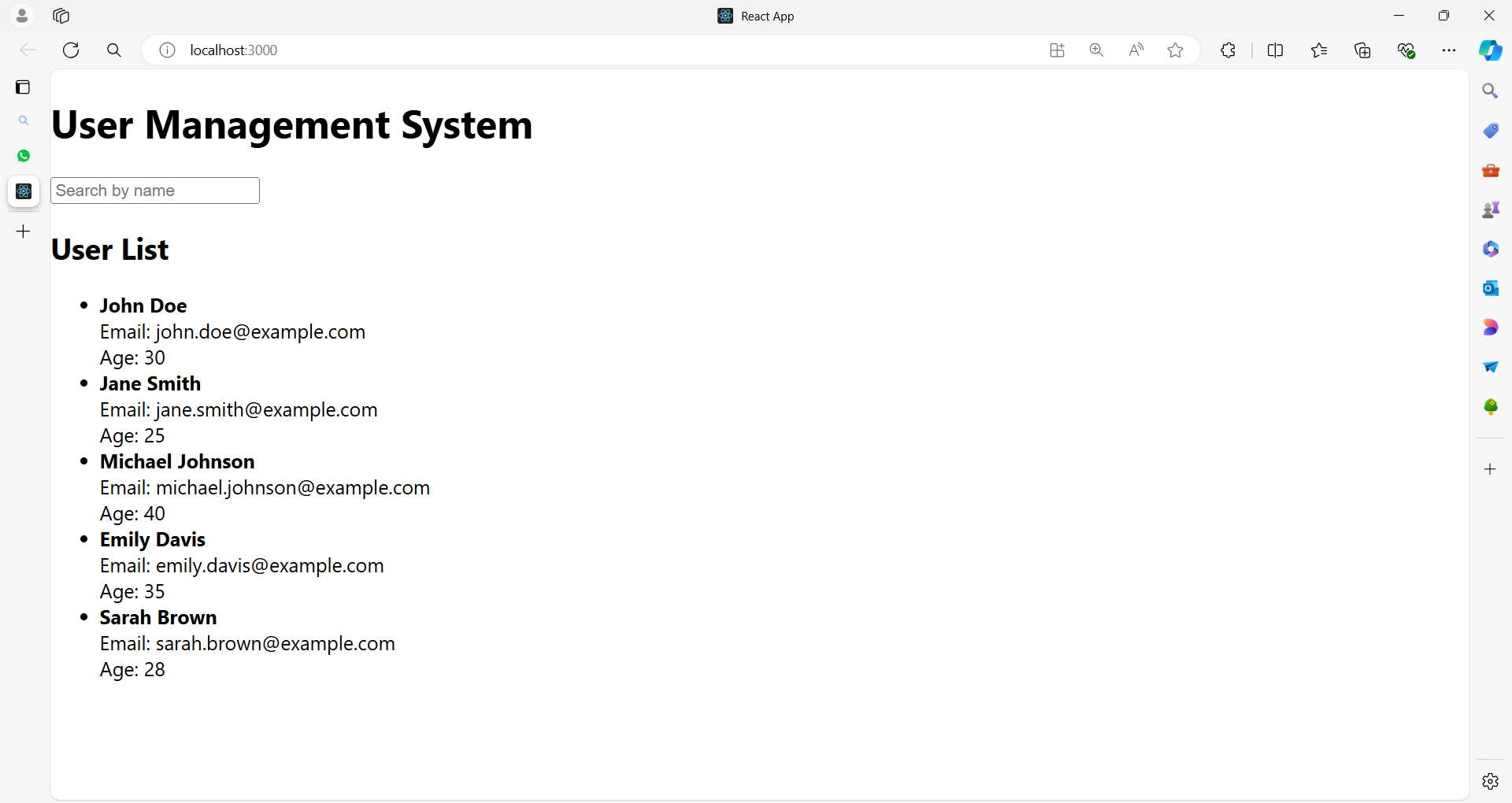
Let's create a comprehensive example that demonstrates the use of lists and keys in React.

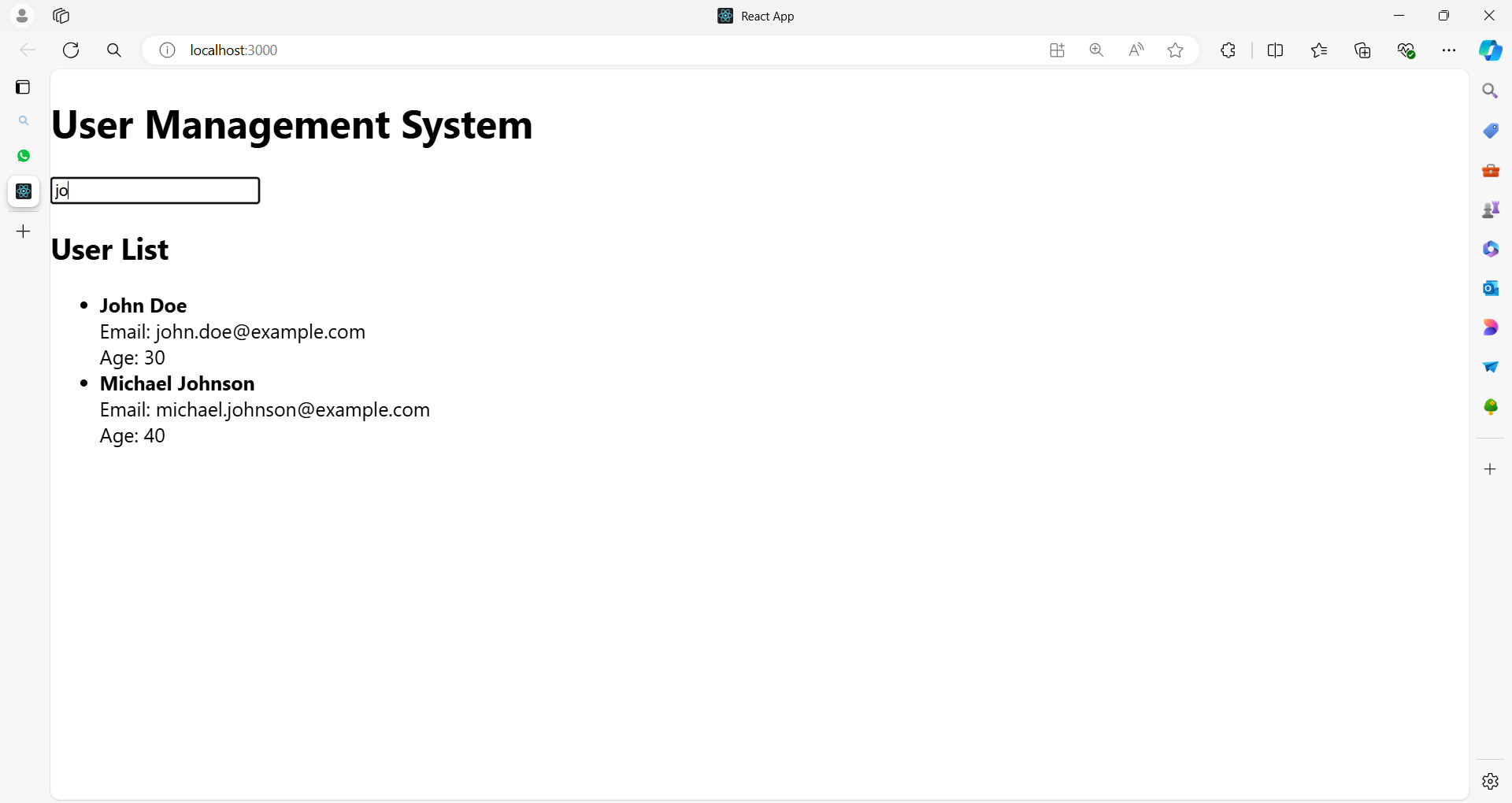
We'll build a simple application that renders a list of users, each with some details and the ability to filter the list based on user input.

**User Management System**

In this example, we'll create:

1. **A user list** with dynamic rendering.
2. **Keys** for each list item.
3. **A filter** to show users based on a search input.





Here's the complete code:

**App.js**

import React, { useState } from 'react';

// Sample user data

const initialUsers = [

{ id: 1, name: 'John Doe', email: 'john.doe@example.com', age: 30 },

{ id: 2, name: 'Jane Smith', email: 'jane.smith@example.com', age: 25 },

{ id: 3, name: 'Michael Johnson', email: 'michael.johnson@example.com', age: 40 },

{ id: 4, name: 'Emily Davis', email: 'emily.davis@example.com', age: 35 },

{ id: 5, name: 'Sarah Brown', email: 'sarah.brown@example.com', age: 28 },

];

// UserList component to display list of users

function UserList({ users }) {

return (

<ul>

{users.map(user => (

<li key={user.id}>

<strong>{user.name}</strong><br />

Email: {user.email}<br />

Age: {user.age}

</li>

))}

</ul>

);

}

// FilterForm component to handle search input

function FilterForm({ filter, setFilter }) {

return (

<input

type="text"

value={filter}

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

placeholder="Search by name"

/>

);

}

// Main App component

function App() {

const [users, setUsers] = useState(initialUsers);

const [filter, setFilter] = useState('');

// Filter users based on the search input

const filteredUsers = users.filter(user =>

user.name.toLowerCase().includes(filter.toLowerCase())

);

return (

<div>

<h1>User Management System</h1>

<FilterForm filter={filter} setFilter={setFilter} />

<h2>User List</h2>

<UserList users={filteredUsers} />

</div>

);

}

export default App;

**Explanation**

**1. Sample Data**

**const initialUsers = [**

**// User objects with id, name, email, and age**

**];**

* **Purpose**: Provides initial data for the list of users.
* **Keys**: Each user object has a unique id which will be used as a key.

**2. UserList Component**

**function UserList({ users }) {**

**return (**

**<ul>**

**{users.map(user => (**

**<li key={user.id}>**

**<strong>{user.name}</strong><br />**

**Email: {user.email}<br />**

**Age: {user.age}**

**</li>**

**))}**

**</ul>**

**);**

**}**

* **Rendering List**: Maps over the users array and returns a list item (<li>) for each user.
* **Keys**: Uses user.id as the unique key for each list item to help React identify each item uniquely.

**users.map(user => (**

**<li key={user.id}>**

**<strong>{user.name}</strong><br />**

**Email: {user.email}<br />**

**Age: {user.age}**

**</li>**

**))**

Let’s break it down line by line:

1. **users.map(user => (**
   * This line uses the map function on the users array.
   * **map** is an array method that creates a new array with the results of calling a provided function on every element in the array.
   * user is a parameter for the callback function, representing each individual item in the users array as it is processed by map.
2. **<li key={user.id}>**
   * This line returns an <li> element (a list item in HTML) for each user.
   * The key attribute is set to user.id. In React, key helps identify which items have changed, are added, or are removed. It should be unique for each item in the list to help React efficiently update the UI.
3. **<strong>{user.name}</strong><br />**
   * This line renders the user's name inside a <strong> tag, which makes the text bold.
   * {user.name} is a JavaScript expression that outputs the name of the user from the user object.
   * <br /> is an HTML line break element that moves the following content to a new line.
4. **Email: {user.email}<br />**
   * This line displays the label Email: followed by the user’s email address, extracted from the user.email property.
   * {user.email} outputs the email value for the current user.
   * <br /> again creates a line break to ensure that the following content starts on a new line.
5. **Age: {user.age}**
   * This line displays the label Age: followed by the user’s age, obtained from user.age.
   * {user.age} is a JavaScript expression that outputs the age of the user.
6. **))**
   * This closes the JSX expression opened by the map function. The map function will iterate over each user in the users array, returning an array of <li> elements, each populated with the corresponding user data.

**Summary**

This code snippet is rendering a list of users in HTML <li> elements.

Each list item displays the user's name (in bold), email, and age.

The key attribute ensures that each list item can be uniquely identified by React for efficient rendering and updating.

**3. FilterForm Component**

**function FilterForm({ filter, setFilter }) {**

**return (**

**<input**

**type="text"**

**value={filter}**

**onChange={(e) => setFilter(e.target.value)}**

**placeholder="Search by name"**

**/>**

**);**

**}**

* **Search Input**: Handles user input to filter the list of users.
* **Props**: Receives filter (search text) and setFilter (function to update the filter state).

**4. App Component**

**function App() {**

**const [users, setUsers] = useState(initialUsers);**

**const [filter, setFilter] = useState('');**

**const filteredUsers = users.filter(user =>**

**user.name.toLowerCase().includes(filter.toLowerCase())**

**);**

**return (**

**<div>**

**<h1>User Management System</h1>**

**<FilterForm filter={filter} setFilter={setFilter} />**

**<h2>User List</h2>**

**<UserList users={filteredUsers} />**

**</div>**

**);**

**}**

* **State Management**: Uses useState to manage users and filter.
* **Filtering**: Filters the users array based on the search input before passing it to the UserList component.
* **Rendering**: Renders the FilterForm and UserList components, passing necessary props.

**Summary**

* **Lists**: Rendered using the map() function and displayed dynamically based on data.
* **Keys**: Ensure each list item is uniquely identified using the key prop, which improves performance and helps React manage updates.
* **Filtering**: Demonstrates how to handle user input to filter and display a subset of data.

This example illustrates how to effectively use lists and keys in React, along with managing state and user input to create a dynamic and interactive UI.

**Example 02**

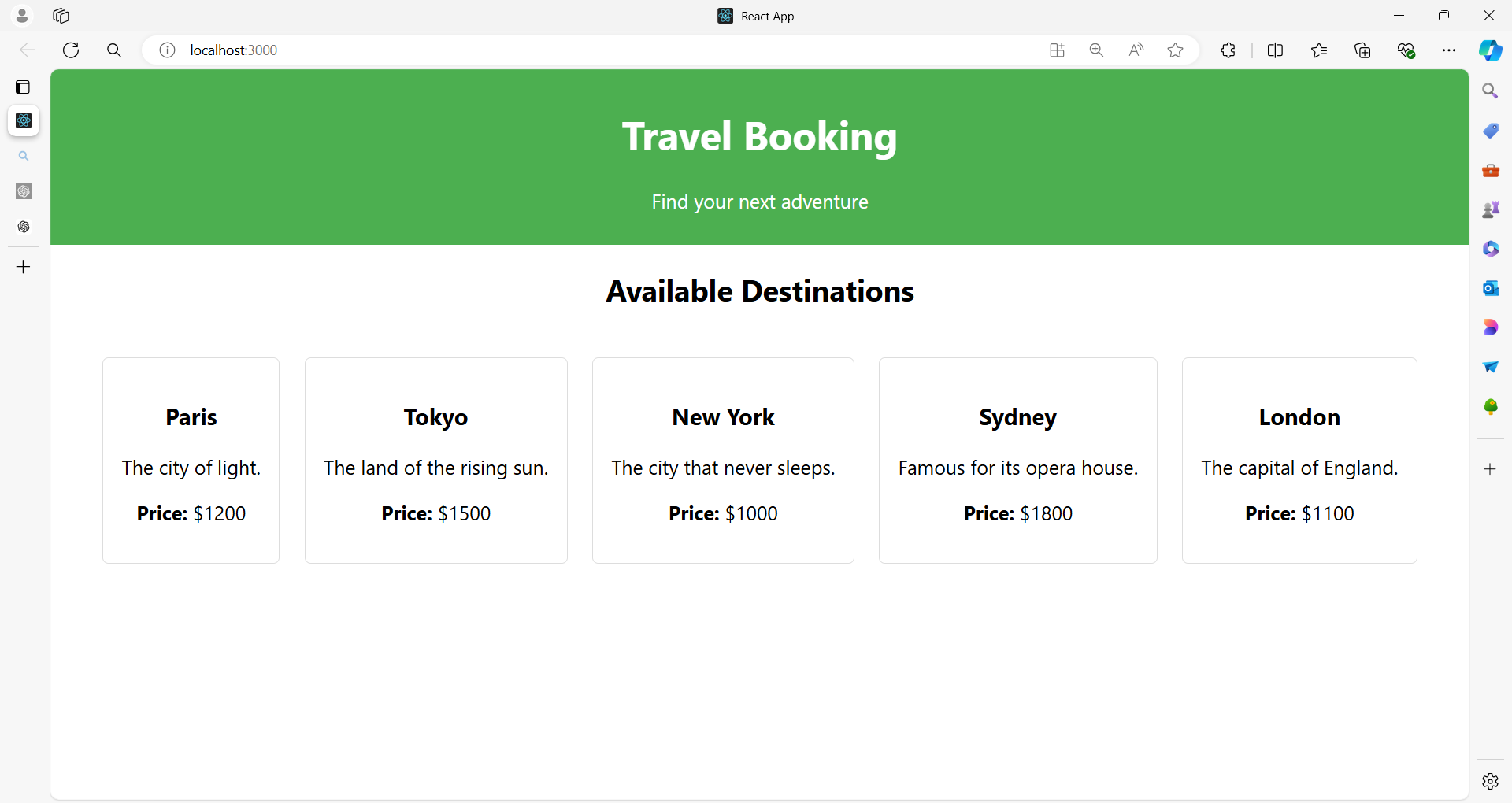
Here's a comprehensive example of a Travel Booking application that demonstrates the use of lists and keys.

The application will consist of several components and separate CSS files for each component.

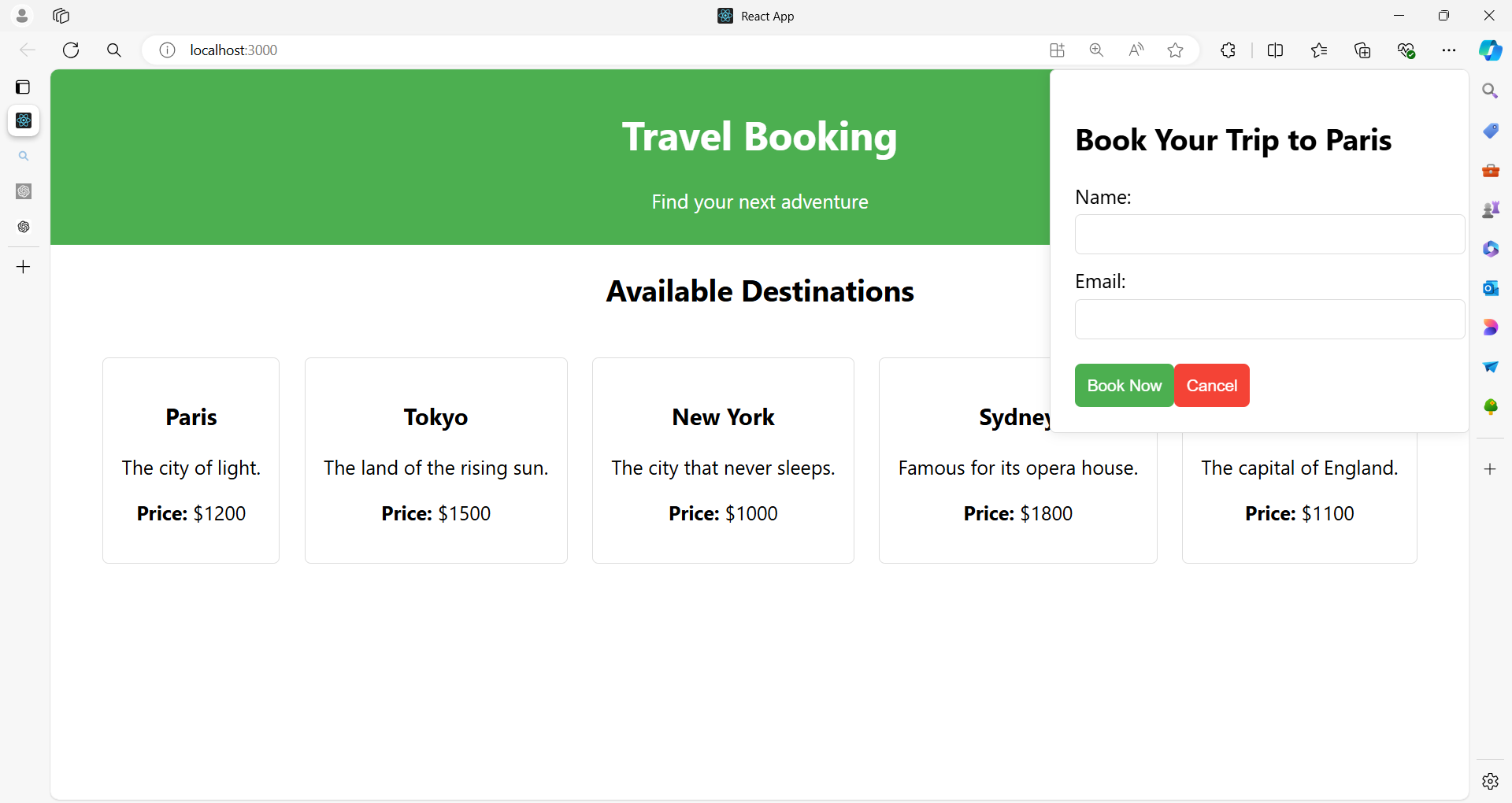
**Travel Booking Application**

**Components:**

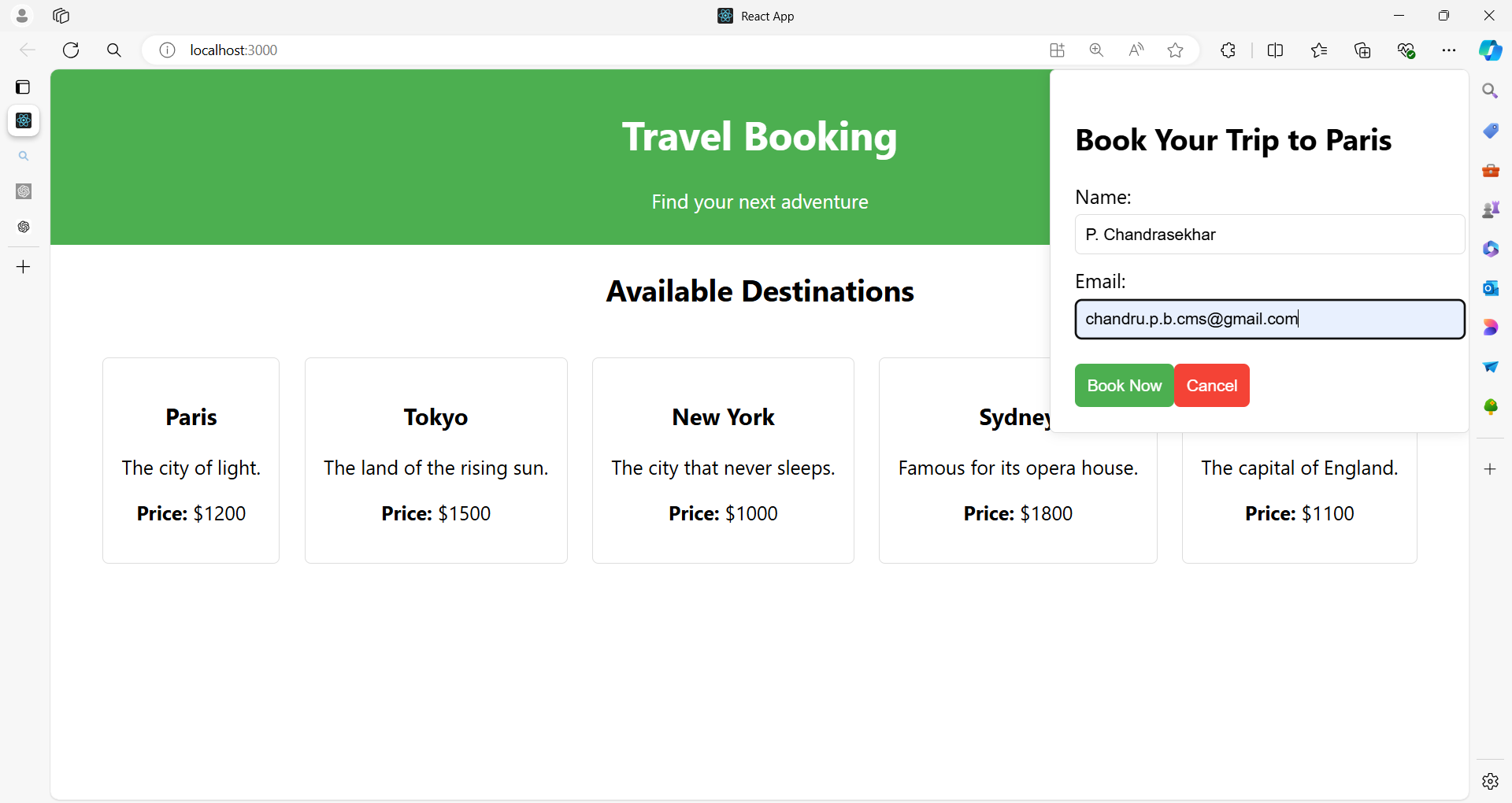
1. **App**: Main component.
2. **Header**: Displays the header of the application.
3. **DestinationList**: Renders a list of travel destinations.
4. **DestinationCard**: Displays details of a single destination.
5. **BookingForm**: Allows users to book a destination.



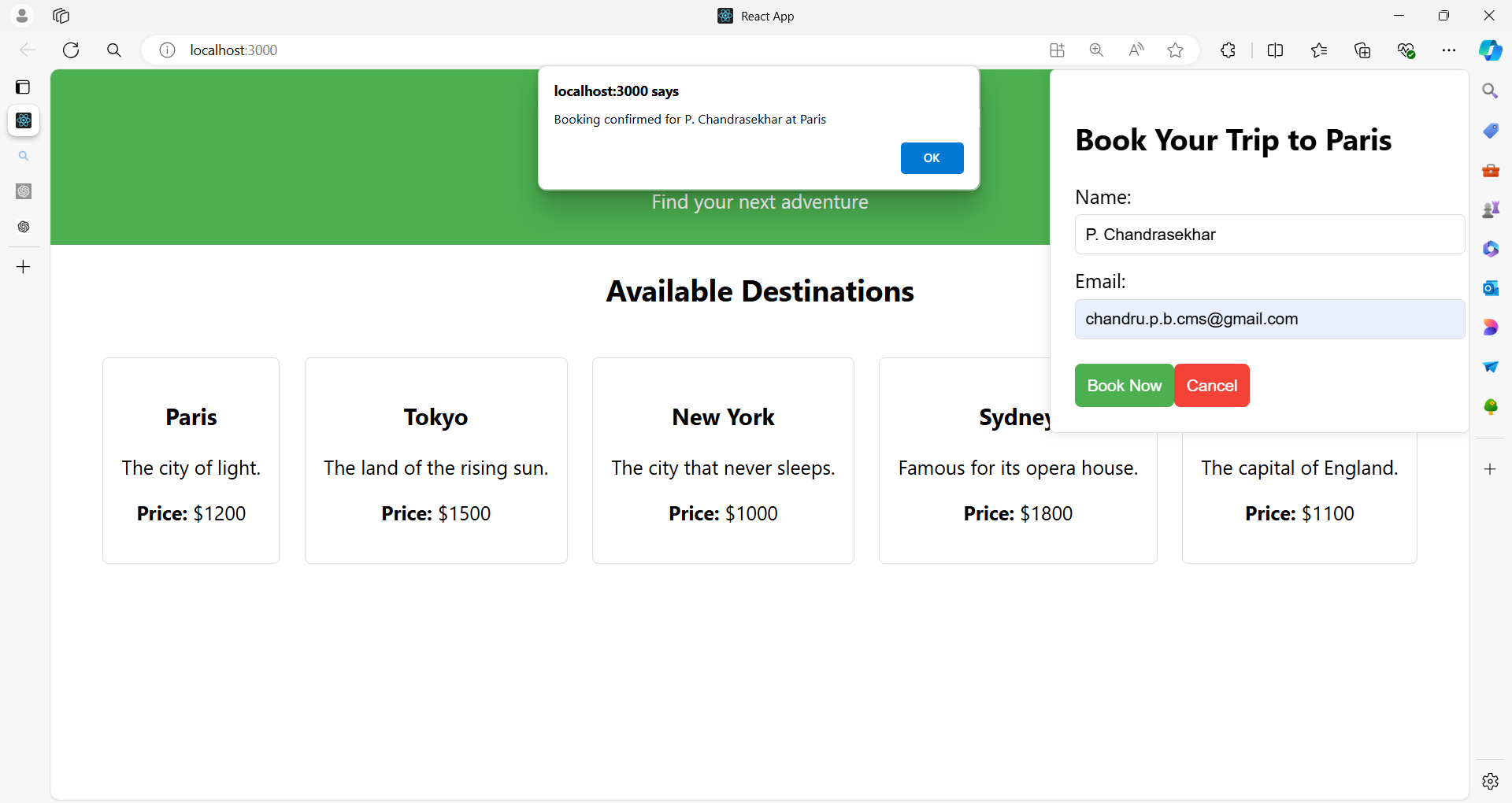
**After clicking Any City**



**Fill the form as shown below**



**After clicking Book Now button**



**Project Structure**

/travel-booking

/components

Header.js

DestinationList.js

DestinationCard.js

BookingForm.js

/styles

App.css

Header.css

DestinationList.css

DestinationCard.css

BookingForm.css

App.js

index.js

**1. App.js**

import React, { useState } from 'react';

import Header from './components/Header';

import DestinationList from './components/DestinationList';

import BookingForm from './components/BookingForm';

import './styles/App.css'; // Global styles

const destinations = [

{ id: 1, name: 'Paris', description: 'The city of light.', price: 1200 },

{ id: 2, name: 'Tokyo', description: 'The land of the rising sun.', price: 1500 },

{ id: 3, name: 'New York', description: 'The city that never sleeps.', price: 1000 },

{ id: 4, name: 'Sydney', description: 'Famous for its opera house.', price: 1800 },

{ id: 5, name: 'London', description: 'The capital of England.', price: 1100 },

{ id: 6, name: 'Rome', description: 'The eternal city with rich history.', price: 1300 },

{ id: 7, name: 'Barcelona', description: 'Famous for its architecture and beaches.', price: 1400 },

{ id: 8, name: 'Dubai', description: 'A modern city with luxurious experiences.', price: 2000 },

{ id: 9, name: 'Istanbul', description: 'Where East meets West with vibrant culture.', price: 1100 },

{ id: 10, name: 'Rio de Janeiro', description: 'Known for its carnival and beautiful beaches.', price: 1600 }

];

function App() {

  const [selectedDestination, setSelectedDestination] = useState(null);

  return (

    <div className="app">

      <Header />

      <DestinationList

        destinations={destinations}

        setSelectedDestination={setSelectedDestination}

      />

      {selectedDestination && (

        <BookingForm

          destination={selectedDestination}

          onClose={() => setSelectedDestination(null)}

        />

      )}

    </div>

  );

}

export default App;

styles/App.css

.destination-container {

  text-align: center;

}

.destination-title {

  margin-bottom: 20px;

  font-size: 24px;

}

.destination-list {

  display: block; /\* Makes sure items break into new lines \*/

}

.destination-item {

  display: inline-block; /\* Displays items in a line \*/

  width: 200px; /\* Adjust width as needed \*/

  margin: 10px;

  vertical-align: top; /\* Aligns items at the top \*/

  text-align: left; /\* Aligns text to the left \*/

}

button {

  padding: 10px;

  border: none;

  background-color: #61dafb;

  color: white;

  cursor: pointer;

}

button:hover {

  background-color: #4fa3d0;

}

**2. Header.js**

import React from 'react';

import '../styles/Header.css';

function Header() {

return (

<header className="header">

<h1>Travel Booking</h1>

<p>Find your next adventure</p>

</header>

);

}

export default Header;

**Header.css**

.header {

background-color: #4CAF50;

color: white;

padding: 10px;

text-align: center;

}

**3. DestinationList.js**

import React from 'react';

import DestinationCard from './DestinationCard';

import '../styles/DestinationList.css';

function DestinationList({ destinations, setSelectedDestination }) {

  return (

    <div className="destination-list">

      <h2>Available Destinations</h2>

      <div className="destination-items">

        {destinations.map(destination => (

          <DestinationCard

            key={destination.id}

            destination={destination}

            onSelect={() => setSelectedDestination(destination)}

          />

        ))}

      </div>

    </div>

  );

}

export default DestinationList;

**DestinationList.css**

.destination-list {

  text-align: center; /\* Centers the text and content \*/

}

.destination-list h2 {

  margin: 20px 0; /\* Adds space above and below the title \*/

  font-size: 24px;

  font-weight: bold;

}

.destination-items {

  display: flex; /\* Align items horizontally \*/

  flex-wrap: wrap; /\* Allow items to wrap to the next line \*/

  justify-content: center; /\* Center items horizontally \*/

  gap: 20px; /\* Space between items \*/

}

**4. DestinationCard.js**

import React from 'react';

import '../styles/DestinationCard.css';

function DestinationCard({ destination, onSelect }) {

return (

<div className="destination-card" onClick={onSelect}>

<h3>{destination.name}</h3>

<p>{destination.description}</p>

<p><strong>Price:</strong> ${destination.price}</p>

</div>

);

}

export default DestinationCard;

**DestinationCard.css**

.destination-card {

border: 1px solid #ddd;

border-radius: 5px;

padding: 15px;

cursor: pointer;

transition: background-color 0.3s;

}

.destination-card:hover {

background-color: #f0f0f0;

}

**5. BookingForm.js**

import React, { useState } from 'react';

import '../styles/BookingForm.css';

function BookingForm({ destination, onClose }) {

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

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

const handleSubmit = (e) => {

e.preventDefault();

alert(`Booking confirmed for ${name} at ${destination.name}`);

onClose();

};

return (

<div className="booking-form">

<h2>Book Your Trip to {destination.name}</h2>

<form onSubmit={handleSubmit}>

<div className="form-group">

<label>Name:</label>

<input

type="text"

value={name}

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

required

/>

</div>

<div className="form-group">

<label>Email:</label>

<input

type="email"

value={email}

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

required

/>

</div>

<button type="submit">Book Now</button>

<button type="button" onClick={onClose}>Cancel</button>

</form>

</div>

);

}

export default BookingForm;

**BookingForm.css**

.booking-form {

position: fixed;

top: 0;

right: 0;

width: 300px;

padding: 20px;

background-color: #fff;

border: 1px solid #ddd;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.booking-form .form-group {

margin-bottom: 10px;

}

.booking-form label {

display: block;

margin-bottom: 5px;

}

.booking-form input {

width: 100%;

padding: 8px;

border: 1px solid #ddd;

border-radius: 5px;

}

.booking-form button {

margin-top: 10px;

padding: 10px;

border: none;

border-radius: 5px;

cursor: pointer;

}

.booking-form button[type="submit"] {

background-color: #4CAF50;

color: white;

}

.booking-form button[type="button"] {

background-color: #f44336;

color: white;

}

**Summary**

* **App Component**: Manages the overall state and rendering of other components.
* **Header Component**: Displays the header with a global style.
* **DestinationList Component**: Renders a list of DestinationCard components using map() and keys.
* **DestinationCard Component**: Represents individual travel destinations.
* **BookingForm Component**: Provides a form for booking a selected destination.

Each component has its own CSS file for styling, demonstrating how to manage styles in a React application. The DestinationList uses the key prop to ensure each DestinationCard is uniquely identified, enhancing performance and accuracy in updates.

**Example 03**

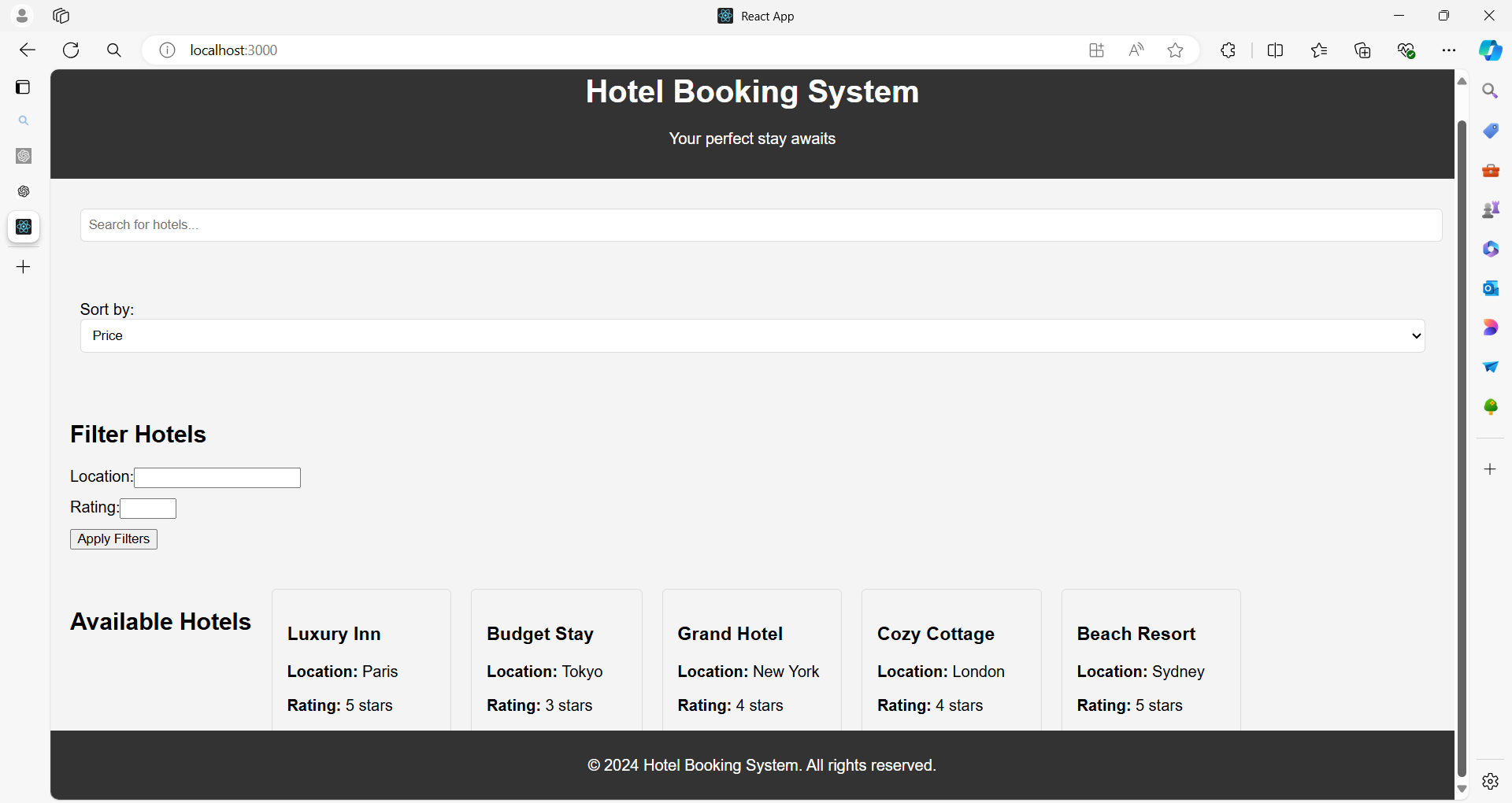
Here's a detailed example of a large React application to explain lists and keys.

The application will simulate a Hotel Booking System with various components, including a list of hotels, filtering options, and booking functionality.

**Hotel Booking System**

**Components:**

1. **App**: Main component.
2. **Header**: Displays the header of the application.
3. **HotelList**: Renders a list of hotels.
4. **HotelCard**: Displays details of a single hotel.
5. **HotelFilter**: Allows users to filter hotels by location and rating.
6. **HotelDetails**: Shows details of a selected hotel.
7. **BookingForm**: Form to book a selected hotel.
8. **Footer**: Displays footer information.
9. **SearchBar**: Allows searching for hotels by name.
10. **SortOptions**: Allows sorting hotels by price or rating.



**Project Structure**

/hotel-booking

/components

Header.js

HotelList.js

HotelCard.js

HotelFilter.js

HotelDetails.js

BookingForm.js

Footer.js

SearchBar.js

SortOptions.js

/styles

Header.css

HotelList.css

HotelCard.css

HotelFilter.css

HotelDetails.css

BookingForm.css

Footer.css

SearchBar.css

SortOptions.css

App.js

index.js

**1. App.js**

import React, { useState } from 'react';

import Header from './components/Header';

import HotelList from './components/HotelList';

import HotelFilter from './components/HotelFilter';

import HotelDetails from './components/HotelDetails';

import BookingForm from './components/BookingForm';

import Footer from './components/Footer';

import SearchBar from './components/SearchBar';

import SortOptions from './components/SortOptions';

import './styles/App.css'; // Global styles

// Sample hotel data

const hotels = [

  { id: 1, name: 'Luxury Inn', location: 'Paris', rating: 5, price: 200 },

  { id: 2, name: 'Budget Stay', location: 'Tokyo', rating: 3, price: 50 },

  { id: 3, name: 'Grand Hotel', location: 'New York', rating: 4, price: 150 },

  { id: 4, name: 'Cozy Cottage', location: 'London', rating: 4, price: 120 },

  { id: 5, name: 'Beach Resort', location: 'Sydney', rating: 5, price: 300 },

  { id: 6, name: 'Mountain Lodge', location: 'Zurich', rating: 4, price: 180 },

  { id: 7, name: 'Urban Retreat', location: 'San Francisco', rating: 3, price: 80 },

  { id: 8, name: 'Historic Manor', location: 'Rome', rating: 4, price: 160 },

  { id: 9, name: 'Sea View Hotel', location: 'Barcelona', rating: 5, price: 250 },

  { id: 10, name: 'Desert Oasis', location: 'Dubai', rating: 5, price: 350 }

];

function App() {

const [filteredHotels, setFilteredHotels] = useState(hotels);

const [selectedHotel, setSelectedHotel] = useState(null);

const [searchTerm, setSearchTerm] = useState('');

const [sortOption, setSortOption] = useState('price');

// Filter and sort hotels

const handleFilterChange = (filters) => {

const { location, rating } = filters;

const filtered = hotels.filter(hotel => {

return (

(!location || hotel.location === location) &&

(!rating || hotel.rating === rating)

);

});

setFilteredHotels(filtered);

};

const handleSearchChange = (term) => {

setSearchTerm(term);

const searched = hotels.filter(hotel =>

hotel.name.toLowerCase().includes(term.toLowerCase())

);

setFilteredHotels(searched);

};

const handleSortChange = (option) => {

setSortOption(option);

const sorted = [...filteredHotels].sort((a, b) => a[option] - b[option]);

setFilteredHotels(sorted);

};

return (

<div className="app">

<Header />

<SearchBar searchTerm={searchTerm} onSearchChange={handleSearchChange} />

<SortOptions sortOption={sortOption} onSortChange={handleSortChange} />

<HotelFilter onFilterChange={handleFilterChange} />

<HotelList

hotels={filteredHotels}

onSelectHotel={setSelectedHotel}

/>

{selectedHotel && (

<HotelDetails

hotel={selectedHotel}

onClose={() => setSelectedHotel(null)}

/>

)}

{selectedHotel && (

<BookingForm

hotel={selectedHotel}

onClose={() => setSelectedHotel(null)}

/>

)}

<Footer />

</div>

);

}

export default App;

**2. Header.js**

import React from 'react';

import './styles/Header.css';

function Header() {

return (

<header className="header">

<h1>Hotel Booking System</h1>

<p>Your perfect stay awaits</p>

</header>

);

}

export default Header;

**Header.css**

.header {

background-color: #333;

color: white;

padding: 15px;

text-align: center;

}

**3. HotelList.js**

import React from 'react';

import HotelCard from './HotelCard';

import '../styles/HotelList.css';

function HotelList({ hotels, onSelectHotel }) {

return (

<div className="hotel-list">

<h2>Available Hotels</h2>

<div className="hotel-card-container">

{hotels.map(hotel => (

<HotelCard

key={hotel.id}

hotel={hotel}

onSelect={() => onSelectHotel(hotel)}

/>

))}

</div>

</div>

);

}

export default HotelList;

**HotelList.css**

.hotel-list {

text-align: center; /\* Center-aligns the text \*/

}

.hotel-list h2 {

margin: 20px 0; /\* Adds space above and below the title \*/

font-size: 24px; /\* Font size for the title \*/

font-weight: bold; /\* Makes the title bold \*/

}

.hotel-list .hotel-card-container {

display: flex; /\* Aligns items horizontally \*/

flex-wrap: wrap; /\* Allows items to wrap to the next line \*/

justify-content: center; /\* Centers items horizontally \*/

gap: 20px; /\* Space between items \*/

}

**4. HotelCard.js**

import React from 'react';

import './styles/HotelCard.css';

function HotelCard({ hotel, onSelect }) {

return (

<div className="hotel-card" onClick={onSelect}>

<h3>{hotel.name}</h3>

<p><strong>Location:</strong> {hotel.location}</p>

<p><strong>Rating:</strong> {hotel.rating} stars</p>

<p><strong>Price:</strong> ${hotel.price} per night</p>

</div>

);

}

export default HotelCard;

**HotelCard.css**

.hotel-card {

border: 1px solid #ddd;

border-radius: 5px;

padding: 15px;

cursor: pointer;

transition: background-color 0.3s;

}

.hotel-card:hover {

background-color: #f9f9f9;

}

**5. HotelFilter.js**

import React, { useState } from 'react';

import './styles/HotelFilter.css';

function HotelFilter({ onFilterChange }) {

const [location, setLocation] = useState('');

const [rating, setRating] = useState('');

const handleApplyFilters = () => {

onFilterChange({ location, rating });

};

return (

<div className="hotel-filter">

<h2>Filter Hotels</h2>

<label>

Location:

<input

type="text"

value={location}

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

/>

</label>

<label>

Rating:

<input

type="number"

min="1"

max="5"

value={rating}

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

/>

</label>

<button onClick={handleApplyFilters}>Apply Filters</button>

</div>

);

}

export default HotelFilter;

**HotelFilter.css**

.hotel-filter {

padding: 20px;

background-color: #f4f4f4;

border-radius: 5px;

}

.hotel-filter label {

display: block;

margin-bottom: 10px;

}

**6. HotelDetails.js**

import React from 'react';

import './styles/HotelDetails.css';

function HotelDetails({ hotel, onClose }) {

return (

<div className="hotel-details">

<button className="close-btn" onClick={onClose}>Close</button>

<h2>{hotel.name}</h2>

<p><strong>Location:</strong> {hotel.location}</p>

<p><strong>Rating:</strong> {hotel.rating} stars</p>

<p><strong>Description:</strong> A wonderful place to stay with top-notch amenities.</p>

<p><strong>Price:</strong> ${hotel.price} per night</p>

</div>

);

}

export default HotelDetails;

**HotelDetails.css**

.hotel-details {

padding: 20px;

background-color: #fff;

border: 1px solid #ddd;

border-radius: 5px;

position: relative;

}

.close-btn {

position: absolute;

top: 10px;

right: 10px;

background: #f44336;

color: white;

border: none;

padding: 5px 10px;

cursor: pointer;

}

**7. BookingForm.js**

import React, { useState } from 'react';

import './styles/BookingForm.css';

function BookingForm({ hotel, onClose }) {

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

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

const handleSubmit = (e) => {

e.preventDefault();

alert(`Booking confirmed for ${name} at ${hotel.name}`);

onClose();

};

return (

<div className="booking-form">

<h2>Book Your Stay at {hotel.name}</h2>

<form onSubmit={handleSubmit}>

<label>

Name:

<input

type="text"

value={name}

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

required

/>

</label>

<label>

Email:

<input

type="email"

value={email}

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

required

/>

</label>

<button type="submit">Book Now</button>

<button type="button" onClick={onClose}>Cancel</button>

</form>

</div>

);

}

export default BookingForm;

**BookingForm.css**

.booking-form {

padding: 20px;

background-color: #fff;

border: 1px solid #ddd;

border-radius: 5px;

}

.booking-form label {

display: block;

margin-bottom: 10px;

}

.booking-form input {

width: 100%;

padding: 8px;

border: 1px solid #ddd;

border-radius: 5px;

}

.booking-form button {

margin-top: 10px;

padding: 10px;

border: none;

border-radius: 5px;

cursor: pointer;

}

.booking-form button[type="submit"] {

background-color: #4CAF50;

color: white;

}

.booking-form button[type="button"] {

background-color: #f44336;

color: white;

}

**8. Footer.js**

import React from 'react';

import './styles/Footer.css';

function Footer() {

return (

<footer className="footer">

<p>&copy; 2024 Hotel Booking System. All rights reserved.</p>

</footer>

);

}

export default Footer;

**Footer.css**

.footer {

background-color: #333;

color: white;

text-align: center;

padding: 10px;

position: fixed;

bottom: 0;

width: 100%;

}

**9. SearchBar.js**

import React from 'react';

import './styles/SearchBar.css';

function SearchBar({ searchTerm, onSearchChange }) {

return (

<div className="search-bar">

<input

type="text"

placeholder="Search for hotels..."

value={searchTerm}

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

/>

</div>

);

}

export default SearchBar;

**SearchBar.css**

.search-bar {

padding: 10px;

background-color: #f4f4f4;

border-radius: 5px;

margin: 20px;

}

.search-bar input {

width: 100%;

padding: 8px;

border: 1px solid #ddd;

border-radius: 5px;

}

**10. SortOptions.js**

import React from 'react';

import './styles/SortOptions.css';

function SortOptions({ sortOption, onSortChange }) {

return (

<div className="sort-options">

<label>

Sort by:

<select

value={sortOption}

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

>

<option value="price">Price</option>

<option value="rating">Rating</option>

</select>

</label>

</div>

);

}

export default SortOptions;

**SortOptions.css**

.sort-options {

padding: 10px;

background-color: #f4f4f4;

border-radius: 5px;

margin: 20px;

}

.sort-options label {

display: block;

}

.sort-options select {

width: 100%;

padding: 8px;

border: 1px solid #ddd;

border-radius: 5px;

}

**Summary**

* **App Component**: Manages state and handles the overall logic of filtering, searching, and sorting hotels. It renders all other components and manages their state.
* **Header Component**: Displays a static header.
* **HotelList Component**: Displays a list of HotelCard components using map() and keys.
* **HotelCard Component**: Represents individual hotels with a click event to select the hotel.
* **HotelFilter Component**: Provides input fields to filter hotels by location and rating.
* **HotelDetails Component**: Displays details of a selected hotel.
* **BookingForm Component**: Allows users to book a selected hotel.
* **Footer Component**: Displays footer information.
* **SearchBar Component**: Allows searching for hotels by name.
* **SortOptions Component**: Allows sorting hotels by price or rating.

This example demonstrates how to manage complex state and user interactions in a React application, using lists and keys effectively to render and update components.