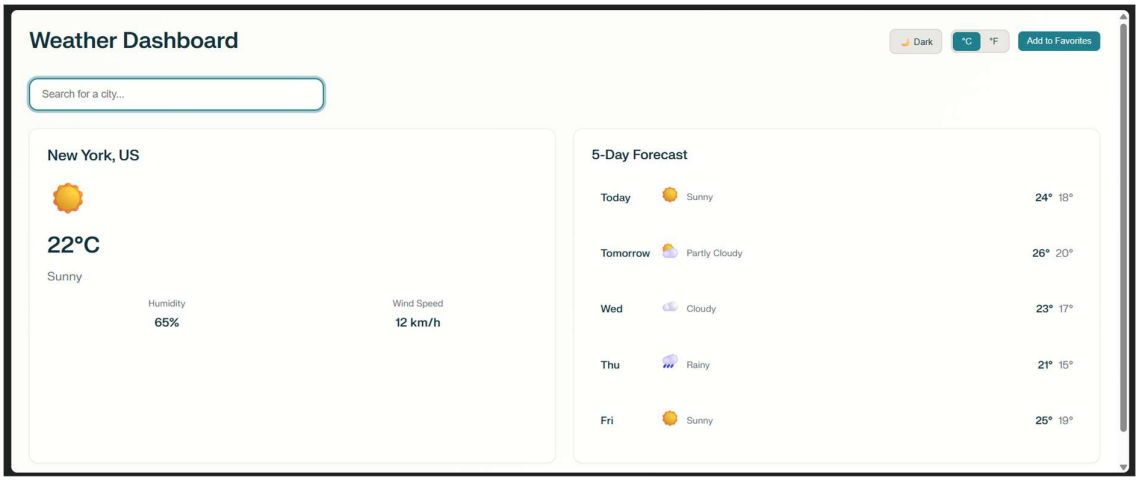
Weather Dashboard Application



Key Features

- **City Weather Search: Search functionality for weather data across major global cities**
- **Current Weather Display: Comprehensive current conditions including temperature, humidity, and wind speed**
- **5-Day Forecast: Extended weather predictions with visual indicators**
- **Favorite Cities Management: Save and quickly access preferred locations**
- **Temperature Unit Toggle: Switch between Celsius and Fahrenheit**
- **Theme System: Dark and light mode with persistent user preferences**
- **Responsive Design: Optimized for desktop, tablet, and mobile devices**
- **Error Handling: Graceful error states and user feedback**
- **Loading States: Smooth loading indicators for better user experience**

Component Structure

The project is organized into reusable, focused components:

- **App Component**: Main application container with global state management
- **WeatherSearch**: Intelligent search component with debounced input
- **CurrentWeather**: Displays detailed current weather information
- **WeatherForecast**: Shows 5-day forecast with visual weather indicators
- **FavoriteCities**: Manages user's saved locations with localStorage persistence
- **ThemeToggle**: Handles light/dark mode switching
- **LoadingSpinner**: Reusable loading state component
- **ErrorBoundary**: Catches and displays errors gracefully

Custom Hooks Implementation

The project demonstrates advanced React patterns through custom hooks:

- **useWeatherData**: Handles weather data fetching and caching
- **useLocalStorage**: Manages persistent data storage
- **useTheme**: Handles theme state and CSS variable management
- **useDebounce**: Optimizes search input performance

State Management Strategy

The application uses React's Context API for global state management, demonstrating:

- Weather data state
- User preferences (temperature units, theme)
- Favorite cities management
- Error and loading states

Skills Demonstrated

This project showcases proficiency in:

Core React Concepts

- Functional components with hooks
- State management with useState and useReducer
- Side effects with useEffect
- Context API for global state

- Component composition and props drilling avoidance

Advanced Patterns

- Custom hook creation and reuse
- Higher-order components for error boundaries
- Conditional rendering and list rendering
- Event handling and form management
- CSS-in-JS and CSS custom properties

Professional Development Practices

- Clean code organization and file structure
- Consistent naming conventions
- Proper error handling and user feedback
- Performance considerations
- Responsive design principles