**Weather App (Flutter task)**

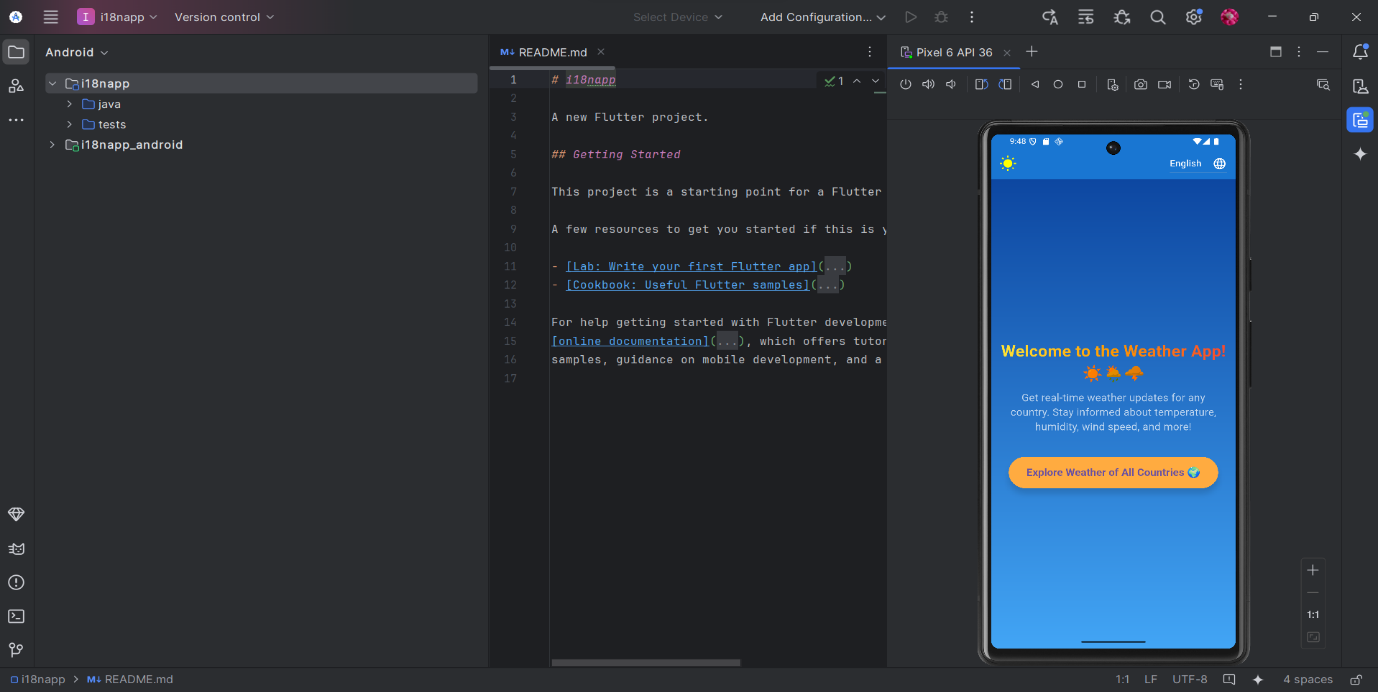
**Home Page**

**Functionality:**

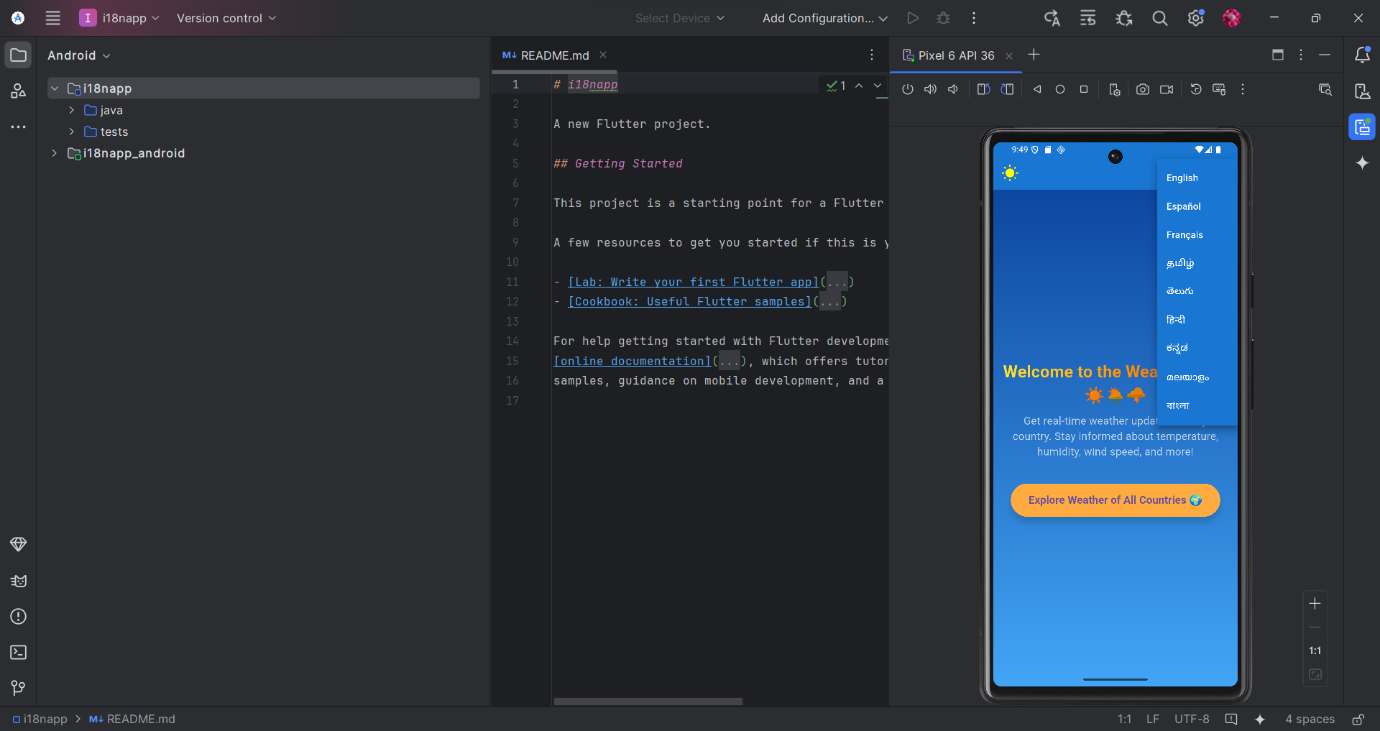
The Home Page serves as the entry point of the application. It provides a welcome message and an option to explore the weather of various countries. Users can also select their preferred language from a dropdown menu.

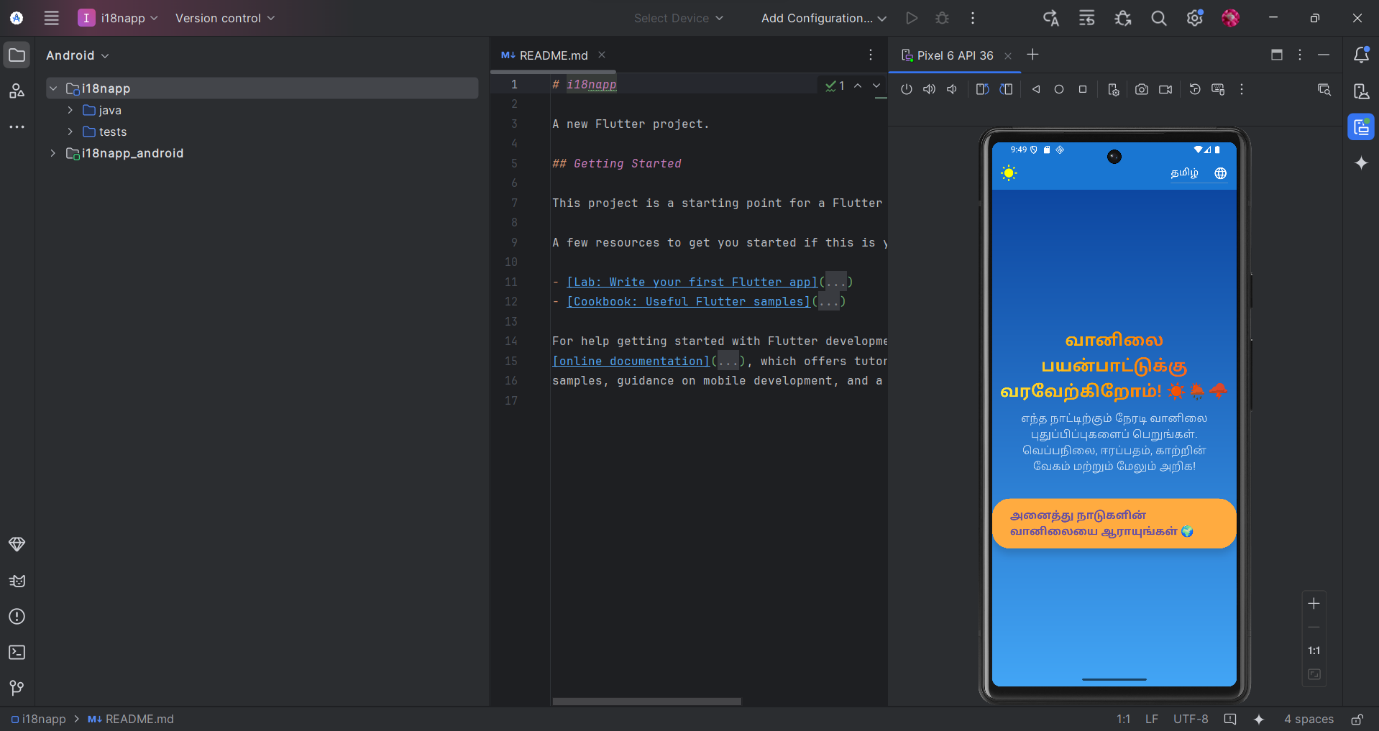
**Implementation:**

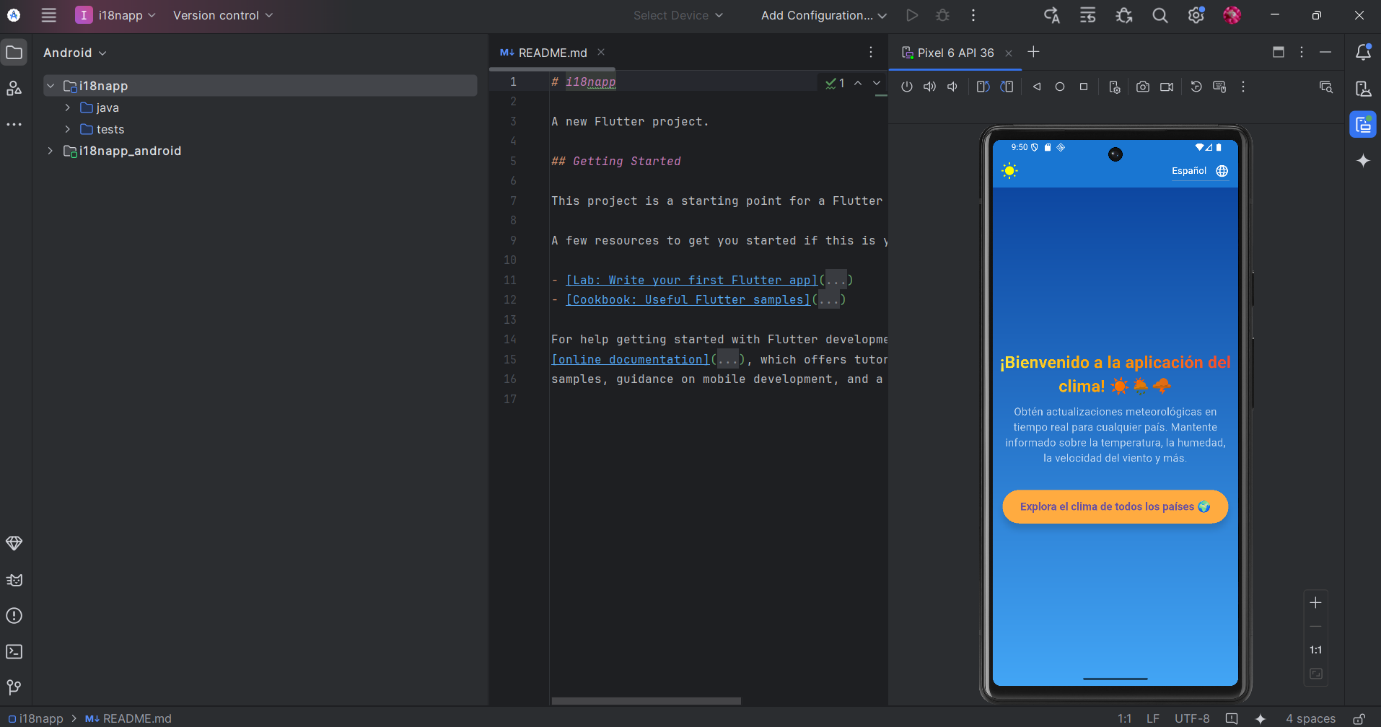
* Designed using a Scaffold widget with an AppBar containing a weather icon.
* Gradient blue background for a visually appealing UI.

****

* Localization support using a language selection dropdown.
* "Explore Weather" button navigates users to the Weather Page.

****

****

****

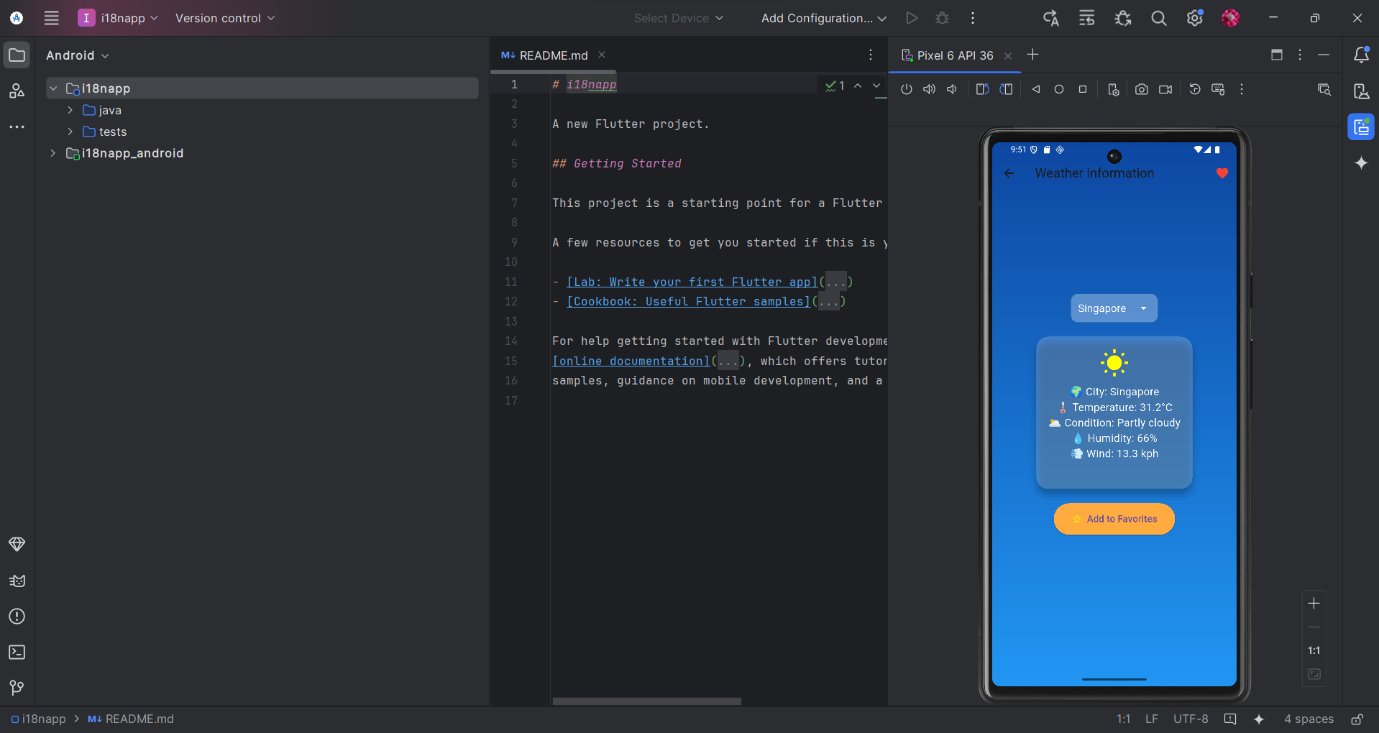
**Weather Page**

**Functionality:**

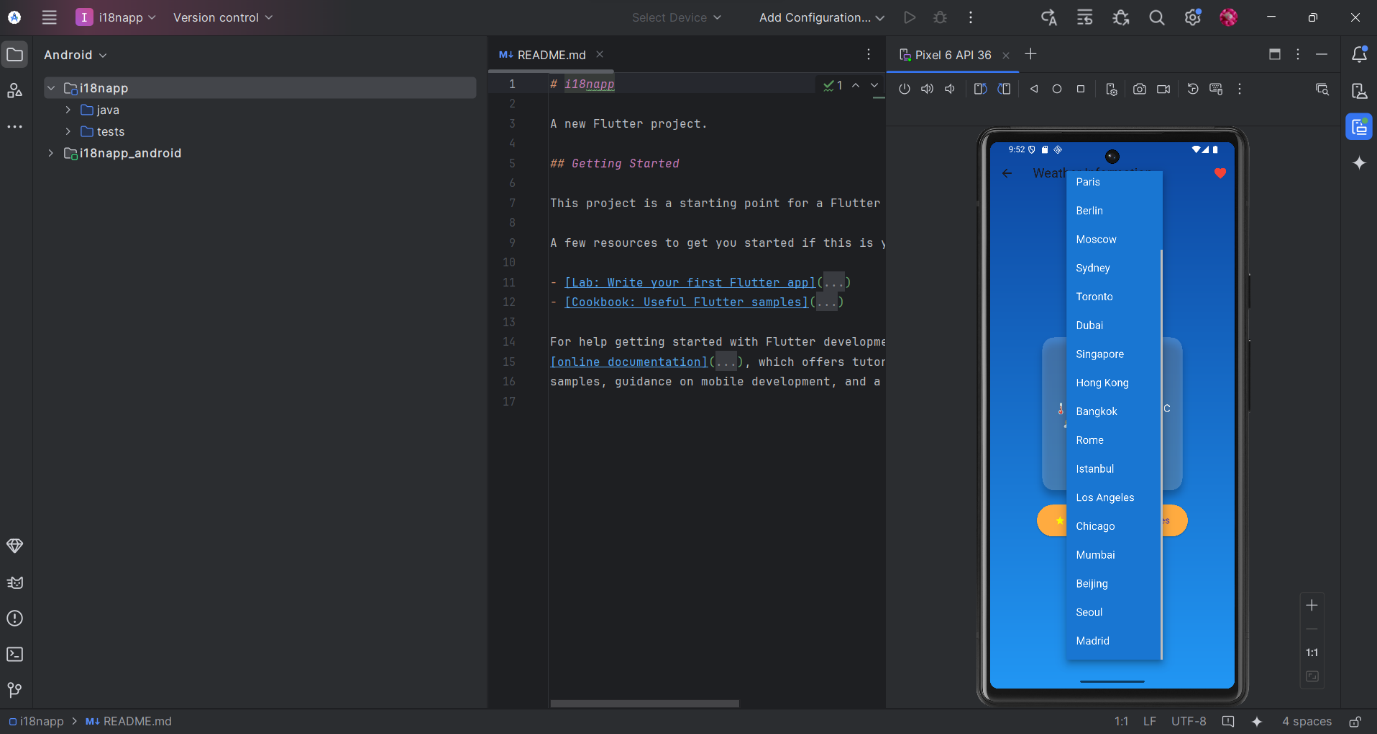
The Weather Page enables users to fetch real-time weather data for different cities. A dropdown menu lets users select a city, and the app retrieves and displays weather details such as temperature, humidity, wind speed, and conditions.

**Implementation:**

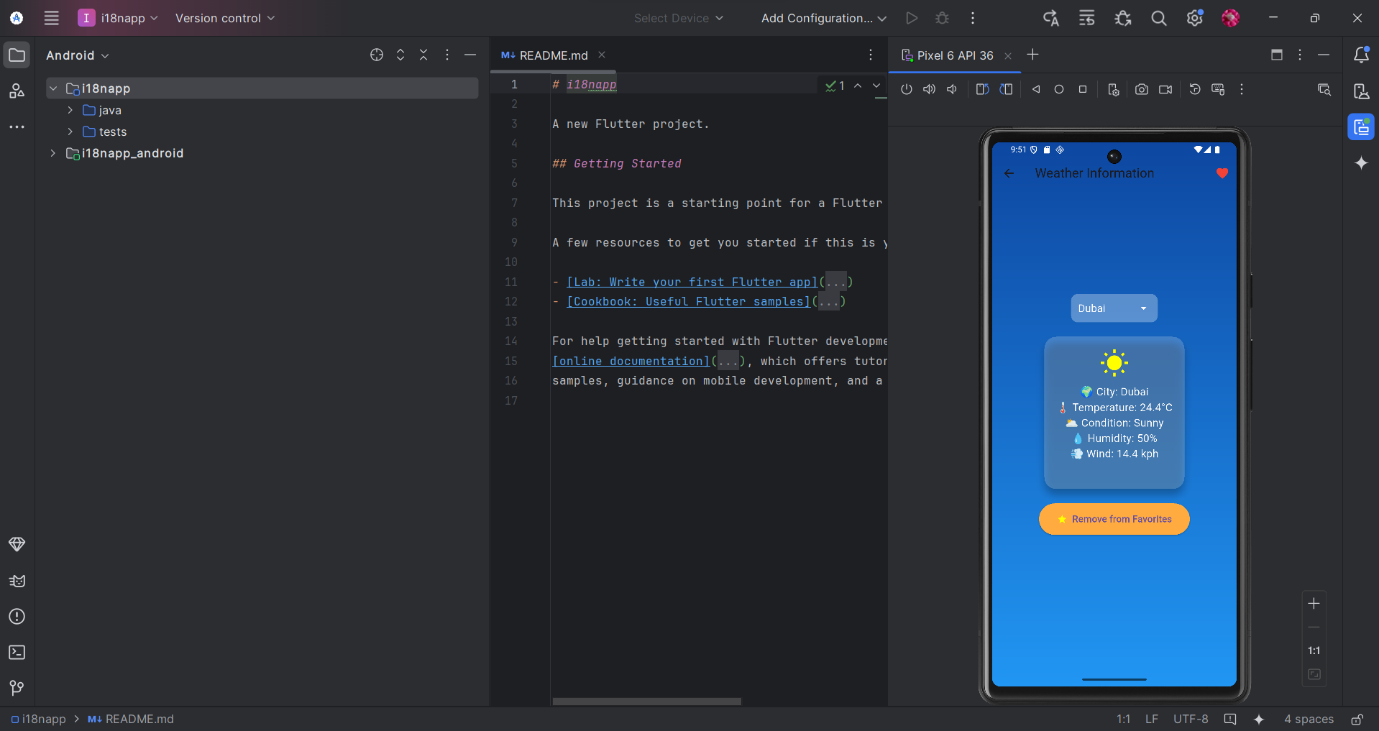
* API call to fetch weather data from WeatherAPI.
* Updates UI dynamically using setState().
* "Add to Favorites" button allows users to save preferred cities.
* Background color set to blue for consistency with the theme.

****

* DropdownButton for selecting cities.

****

* "Remove from Favorites" button allows users to remove cities from favorites.

****

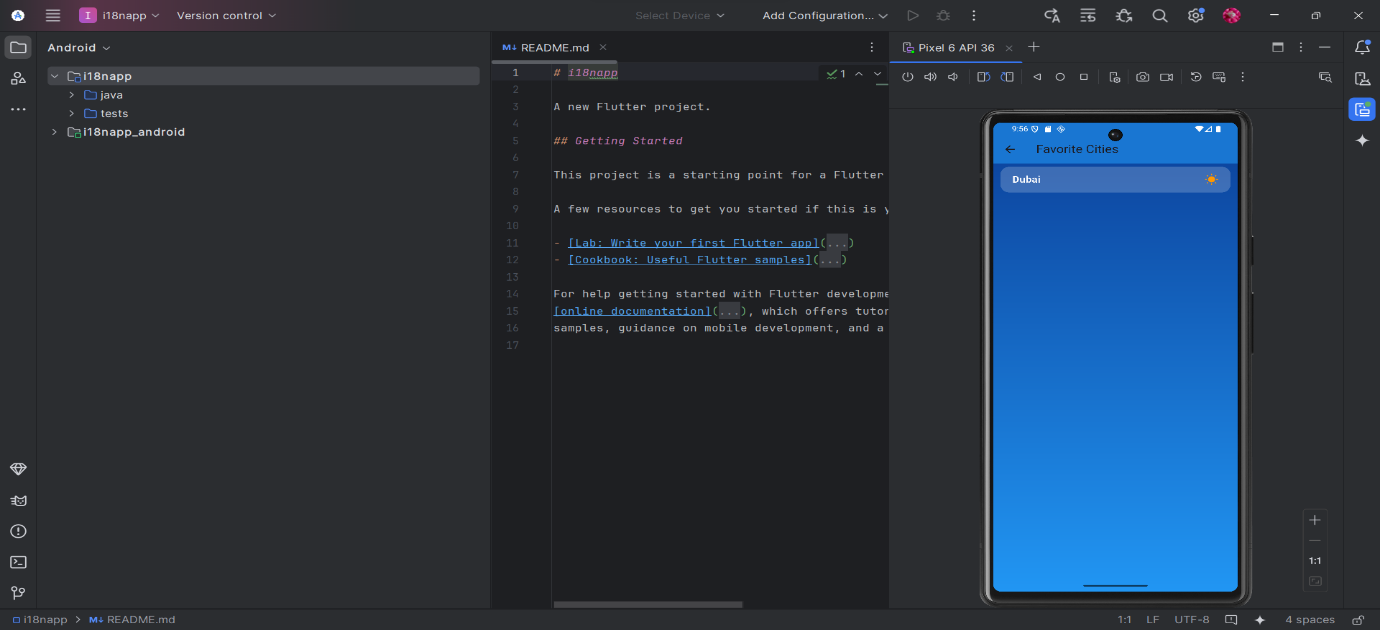
**Favorite Cities Page**

**Functionality:**

The Favorite Cities Page displays a list of cities marked as favorites by the user. Users can manage their list and navigate back to the weather page.

**Implementation:**

* Uses a ListView.builder to display the saved cities.
* Cards with a small height to prevent large white blocks.
* Full blue background for consistency.

****

**Implementation Details**

**1. API Integration**

* The app fetches real-time weather data from WeatherAPI using an HTTP GET request.
* The response JSON is parsed, and relevant details like temperature, humidity, and condition are displayed.

**2. State Management**

* setState updates the UI dynamically when the user selects a city or modifies the favorite list.
* The weather data updates instantly when a new city is chosen.
* The app maintains a list of favorite cities, allowing users to add or remove cities easily.

**4. Localization**

* The app supports multiple languages using AppLocalizations, allowing users to switch languages dynamically.
* Strings are stored in JSON files, and locale changes are handled via setState.

**5. Navigation**

* Navigator.push is used to transition between the Home Page, Weather Page, and Favorite Cities Page smoothly.
* The app structure ensures that users can easily switch between different sections.