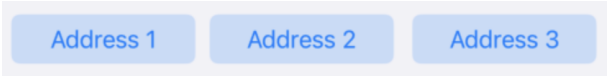
# Requirement

https://github.com/ynieSalesforce/iOS_Assessment

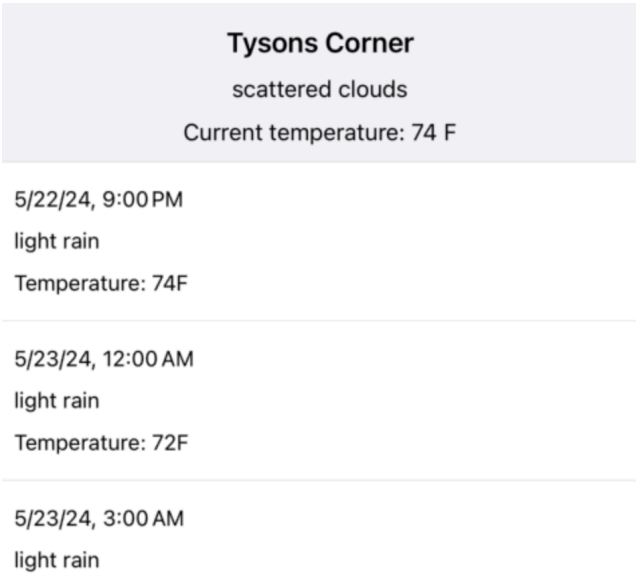
# Requirement Clarification

1. Please let me know if my understanding is correct:
   In UI, “Address 1”, “Address 2”, and “Address 3” are pre-defined locations. For example
   Address 1 : HITEC City, Hyderabad, Telangana 500081, India
   Address 2 : ITPL Main Rd, Pattandur Agrahara, Whitefield, Bengaluru, Karnataka 560066, India
   Address 3 : Some Address

2. As per the requirement document, I need to use the AddressService to obtain a CLLocation for the selected address, which will then be used to fetch the weather report for that latitude and longitude.

3. Please let me know if my understanding is correct:
   When user taps "Address 1" button, app will fetch weather report for address 1 and display it in the app like below screen shot. When user taps “Address 2” button then it will show report for Address 2.

4. Please let me know if my understanding is correct:
   If app is unable to get CLLocation in step 2 or unable to fetch weather report in step 3, it will display alert message to user on the **same screen**.

# Weather API JSON:


```
{
  "coord":{
    "lon":10.99,
    "lat":44.34
  },
  "weather":[
    {
      "id":804,
      "main":"Clouds",
      "description":"overcast clouds",
      "icon":"04d"
    }
  ],
  "base":"stations",
  "main":{
    "temp":291.36,
    "feels_like":291.53,
    "temp_min":287.41,
    "temp_max":295.61,
    "pressure":1012,
    "humidity":88,
    "sea_level":1012,
    "grnd_level":946
  },
  "visibility":7795,
  "wind":{
    "speed":0.51,
    "deg":333,
    "gust":1.68
  },
  "clouds":{
    "all":95
  },
  "dt":1719318486,
  "sys":{
    "type":2,
    "id":2004688,
    "country":"IT",
    "sunrise":1719286385,
    "sunset":1719342254
  },
  "timezone":7200,
  "id":3163858,
  "name":"Zocca",
  "cod":200
}
```

# Network Service

Network Service Implement NetworkService for API Requests:

- Created a NetworkService class to handle API requests.
- Implemented fetchData method to fetch data from a specified URL.
- Added error handling for various network-related issues.
- Utilized URLSession for asynchronous network operations.

# Weather Service

The `WeatherService` struct facilitates retrieval of current weather and weather forecast data using network requests. It adheres to the `WeatherServiceProtocol` to ensure consistency in method signatures across implementations.

- `retrieveCurrentWeather`: Fetches current weather data based on location.
- `retrieveWeatherForecast`: Fetches weather forecast data based on location.
- Error Handling: Handles scenarios such as invalid URLs, network errors, and JSON decoding failures robustly.
- `WeatherService.live`: Offers a default instance configured with a standard `NetworkService`.

# MVVM Design Pattern

## 1. Model

Based on the API JSON model is already present in the project. Following are model class files for weather and forecast related feature:

WeatherDisplayData
CurrentWeatherDisplayData
ForecastDisplayData

## 2. View

**SwiftUIView:** Composes various subviews into a single vertical stack.

- AddressListView: Displays a horizontal list of clickable address names.
- WeatherReportView: Shows the current weather report.
- ForecastListView: Lists the weather forecast.

- Sets background color using Color(UIColor.systemGray6).
- Displays an alert for error messages from the WeatherViewModel.

**AddressListView:** Custom view for each AddressButtonView, which triggers a weather data fetch for the selected address. AddressButtonView displays an address name and triggers an action when tapped.

**WeatherReportView:** Shows the location name, current weather, and temperature. Handles cases where weather data is unavailable by displaying a message.

**ForecastListView:** Displays a list of forecast items. Uses ForecastRowView to render each forecast item. Ensures the list is aligned and styled properly with padding and background color. ForecastRowView shows time, temperature, and weather description for each forecast item.

# 3. View Model

**WeatherViewModel:** ViewModel serves as a bridge between presentation layer and data layer. It retrieves and transforms data from the data layer, performs business logic, and exposes observable properties that view can bind to.

**Properties:**

**addressService:** Handles address-to-coordinates conversion.
**weatherService:** Fetches weather data.
**addresses:** Placeholder for address data.
**errorMessage:** Publishes error messages for UI alerts.
**weatherDisplayData:** Publishes current weather data for display.
**forecast:** Publishes weather forecast data for display.

**Initialization**:
Default initializer with live instances of AddressService and WeatherServiceProtocol.

**Methods:**

- retrieveCurrentWeatherAndForecast(address:):
    - Converts address to coordinates.
    - Fetches current weather and forecast data using the obtained location.
    - Updates errorMessage on failure.
- retrieveCurrentWeather(location:):
    - Fetches current weather data for the given location.
    - Updates weatherDisplayData on success.
    - Updates errorMessage on failure.
- retrieveWeatherForecast(location:):
    - Fetches weather forecast data for the given location.
    - Updates forecast on success.
    - Updates errorMessage on failure.

# Error Handling

1. **NetworkError Enumeration:**
   - Defines various network-related errors such as .invalidURL, .invalidData, .invalidResponse, .noInternetConnection, and .message(Error?).
   - Provides user-friendly error messages for each case via the errorMessage property.
2. **Error Mapping in NetworkService:**
   - Uses the mapError(_:) method to convert general Error instances into specific NetworkError instances.
   - Handles specific URLError codes like .notConnectedToInternet, .networkConnectionLost, .badURL, and .unsupportedURL.
3. WeatherService Implementation:
   - Fetches weather data using NetworkService and decodes JSON responses.
   - Handles decoding errors by wrapping them in the .message case of NetworkError.
4. WeatherViewModel Error Handling:
   - Updates errorMessage property on failure to fetch weather data, triggering alerts in the UI.
   - Manages errors during address-to-coordinates conversion in retrieveCurrentWeatherAndForecast(address:).
5. SwiftUIView Alert Handling:
   - Displays an alert with an appropriate error message when errorMessage in WeatherViewModel is set.
   - Resets errorMessage to nil when the alert is dismissed.

# Unit Test Cases

1. Test for Successful Current Weather Retrieval:
2. Test for Failed Current Weather Retrieval:
3. Test for Successful Weather Forecast Retrieval:
4. Test for Failed Weather Forecast Retrieval:

# String Constant

To enhance localization and manage string constants effectively across the weather feature, I have introduced WeatherString and NetworkErrorMessage enums. This approach simplifies the process of localization by centralizing and organizing all relevant strings used within the weather functionality.