SkyCast Android App – Features Documentation

---

1. TalkBack for Accessibility (Content Description)

Description:
Implements screen reader accessibility (e.g., TalkBack) by assigning `contentDescription` to UI elements. This ensures that visually impaired users can navigate and understand the app.

Implementation:

- **MainActivity.kt**
  Uses Jetpack Compose's `semantics` modifier to add `contentDescription` to the top app bar, navigation drawer items, and buttons in `LocationScreen` and `SettingsScreen`.
  Example:

```
 modifier = Modifier.semantics { contentDescription =
context.getString(R.string.app_name) + " App Bar" }
```

- **WeatherSection.kt**
  Applies `contentDescription` to the weather column, title, stats (e.g., humidity), and icons.
  Example:

```
 contentDescription =
context.getString(R.string.current_weather_description,
displayLocationName)
```

- **ForecastSection.kt**
  Adds `contentDescription` to the forecast section, `LazyRow`, and tiles.
  Example:

```
 contentDescription =
context.getString(R.string.weather_forecast_description)
```

- **LanguageSelectionDialog.kt**
  Describes dialog and radio buttons using localized strings.
  Example:

```
contentDescription = context.getString(R.string.language_selection_dialog)
```

- **FunActivity.kt**
  Adds `contentDescription` to dynamic elements like the progress bar and status text.
  Example:

```
contentDescription = context.getString(R.string.progress_bar,
(animatedProgress * 100).toInt())
```

Uses `liveRegion = LiveRegionMode.Polite` for updates.

Tools Used:
Jetpack Compose (`semantics`), `R.string` for localization, `LiveRegionMode` for dynamic updates.

---

2. Automatic Refresh Based on Battery Level

Description:
Automatically refreshes weather data based on battery status:

- 60%: every 30 seconds
- 30–60%: every 2 minutes
- <30%: manual refresh via a button

Implementation:

- **BatteryUtils.kt**
  Monitors battery via `BroadcastReceiver`.
  Example:
  `batteryPercentage = (level * 100 / scale)`

- **MainActivity.kt**
  Uses `observeBatteryLevel` with `LaunchedEffect` to schedule refreshes.
  Manual Refresh Example:
  `if (batteryPercentage < 30) { Button(onClick = { ... }) { ... } }`

- **MainViewModel.kt**
  Calls API via `viewModelScope.launch`.
  Example:

```
weatherResponse = apiService.getWeather(latLng.lat, latLng.lng)
```

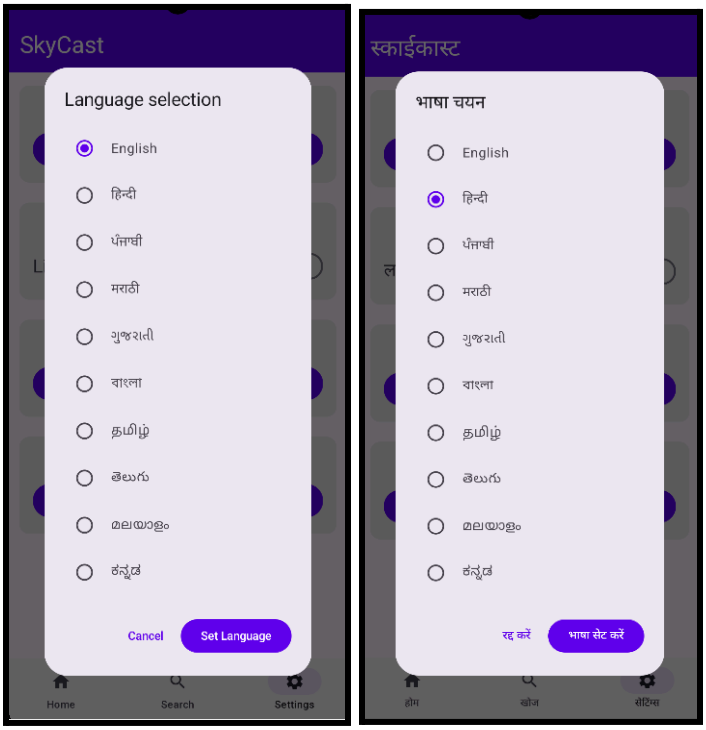
Tools Used:
Jetpack Compose (`LaunchedEffect`), Android `BroadcastReceiver`, Retrofit, Kotlin Coroutines.

---

3. Language Change (Accessibility/Localization)

Description:
Enables language selection (e.g., English, Hindi, Tamil). UI strings and location names update accordingly.

Visualization:



Implementation:

- **LanguageUtils.kt**
  Implements `setLocale` and `applySavedLocale`.
  Example:
  ```
  config.setLocale(Locale(languageCode))
  ```

- **LanguageSelectionDialog.kt**
  Shows language selection with `AlertDialog` and `RadioButton`.
  Example:
  ```
  RadioButton(selected = selectedLanguageCode == code, ...)
  ```

- **MainViewModel.kt**
  Stores language state and updates locale.
  Example:
  ```
  val targetGeocoder = Geocoder(context, Locale("hi", "IN"))
  ```

- **MainActivity.kt / FunActivity.kt**
  Applies saved locale on startup.

Tools Used:
Jetpack Compose (UI), `Locale`, `SharedPreferences`, `Geocoder`.

---

4. Weather from Location Coordinates (Sensing Hub)

Description:
Fetches weather using device GPS or network-based coordinates.

Visualization:

Implementation:

- **LocationUtils.kt**
  Checks if location services are enabled.

Example:

```
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
```

- **MainActivity.kt**
  Uses `FusedLocationProviderClient` to fetch location and call `getWeatherByLocation`.
  Example:

```
fusedLocationClient.getCurrentLocation(Priority.PRIORITY_HIGH_ACCURACY, ...)
```

- **MainViewModel.kt**
  Calls API and updates UI with `weatherResponse`.

- **ApiService.kt**
  Defines Retrofit calls for weather and forecast.

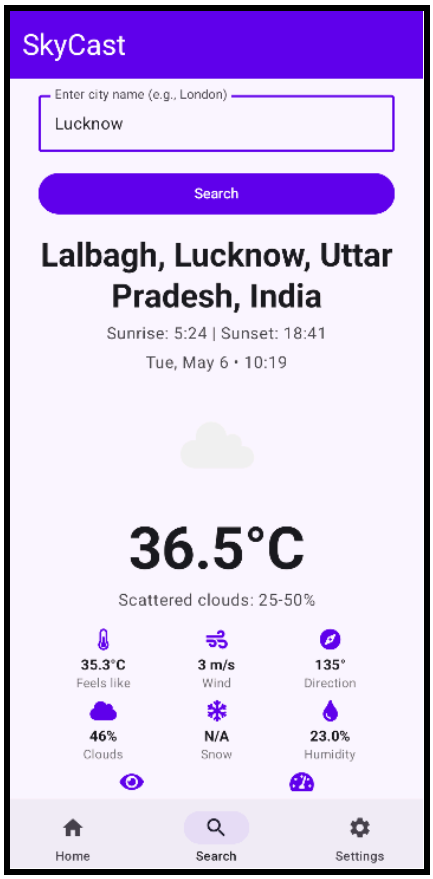
Tools Used:
Google Play Services, Retrofit, Geocoder, Jetpack Compose.

---

5. Weather Fetch by City Name

Description:
Lets users fetch weather data by entering a city name.

Visualization:



Implementation:

- **MainActivity.kt**
  Uses `TextField` and `Button` in `SearchScreen`.
  Example:
  `Button(onClick = { viewModel.getWeatherByLocationName(context, searchQuery) }) { ... }`

- **MainViewModel.kt**
  Uses `Geocoder` to convert the city name to coordinates.

- **ApiService.kt**
  Reuses weather endpoints with converted coordinates.

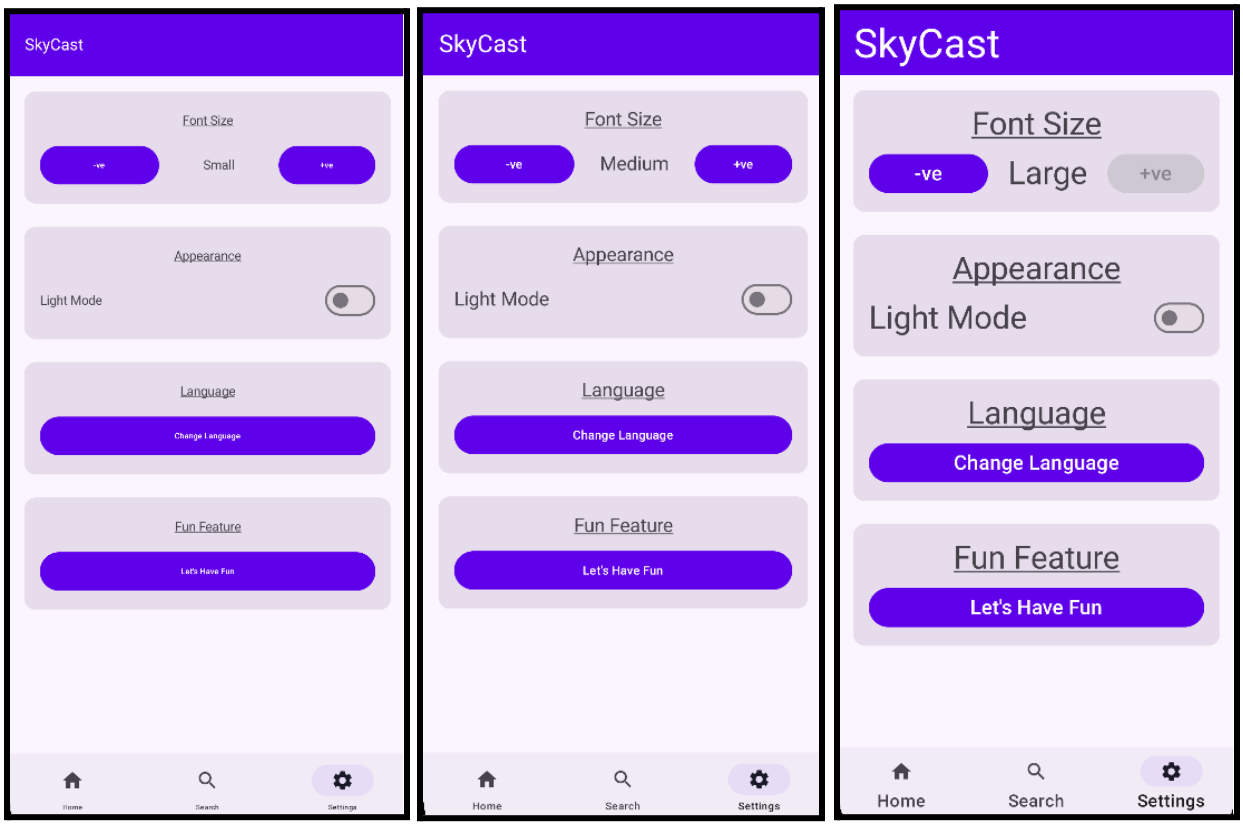
Tools Used:
Jetpack Compose, Retrofit, Geocoder.

---

6. Font Size Adjustment (Accessibility)

Description:
Enables font size scaling from 0.5x to 1.5x for readability.

Visualization:



Implementation:

- **MainViewModel.kt**
  Uses `mutableStateOf` for `fontSizeScale`.
  Example:

```
fontSizeScale = (fontSizeScale + 0.1f).coerceAtMost(1.5f)
```

- **Theme.kt**
  Scales fonts using `CompositionLocalProvider`.

- **MainActivity.kt**
  `SettingsScreen` provides a `Slider` or buttons for control.
  Example:

```
Slider(value = viewModel.fontSizeScale, onValueChange = { ... })
```

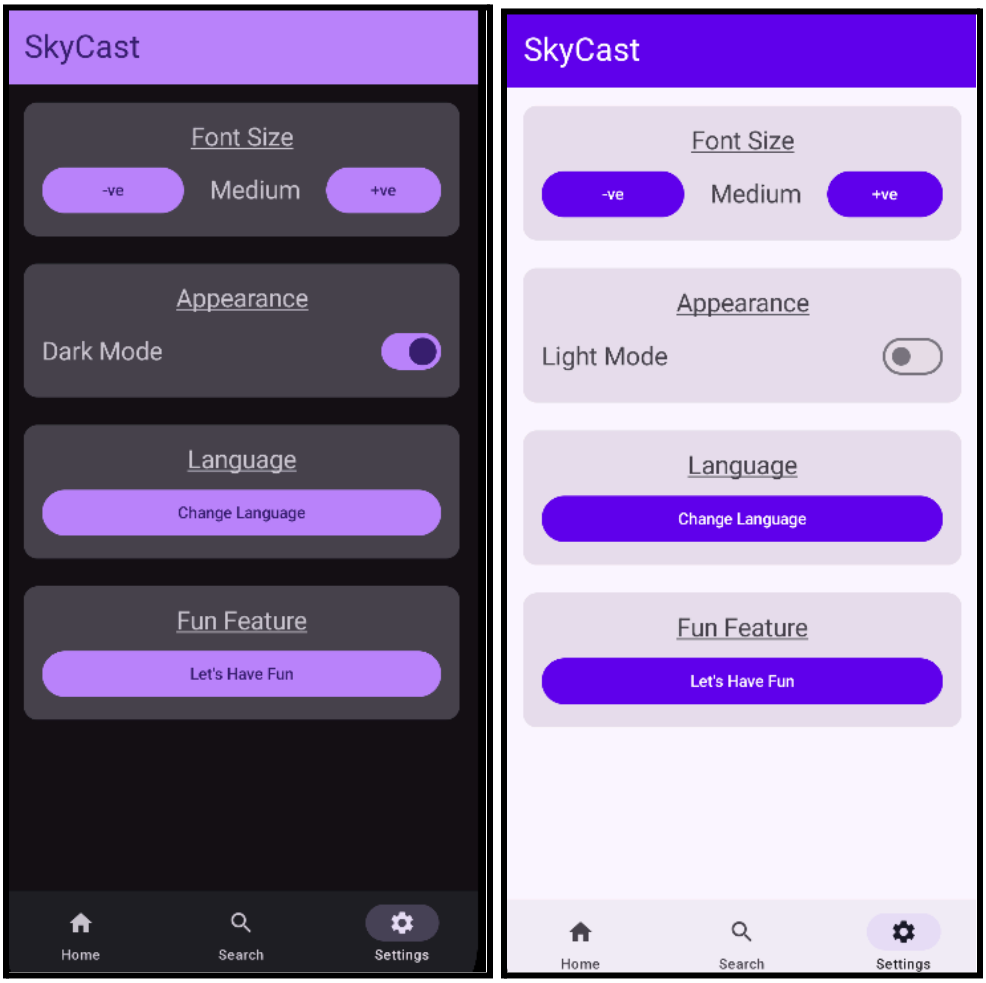
Tools Used:
Jetpack Compose (`Slider`, `Typography`, `mutableStateOf`).

---

7. Light Mode and Dark Mode (Accessibility/Battery)

Description:
Provides light and dark theme toggle. Dark mode saves battery and enhances readability.

Visualization:



Implementation:

- **MainViewModel.kt**
  Uses `mutableStateOf` for `darkMode`.
  Example:
  `darkMode = !darkMode`

- **Theme.kt**
  Applies `lightColorScheme` or `darkColorScheme`.

- **MainActivity.kt / FunActivity.kt**
  Applies selected themes using `SkyCastTheme`.
  Example:

```
SkyCastTheme(darkTheme = darkMode) { ... }
```

Tools Used:
Jetpack Compose (`isSystemInDarkTheme`, `MaterialTheme`, `Switch`).

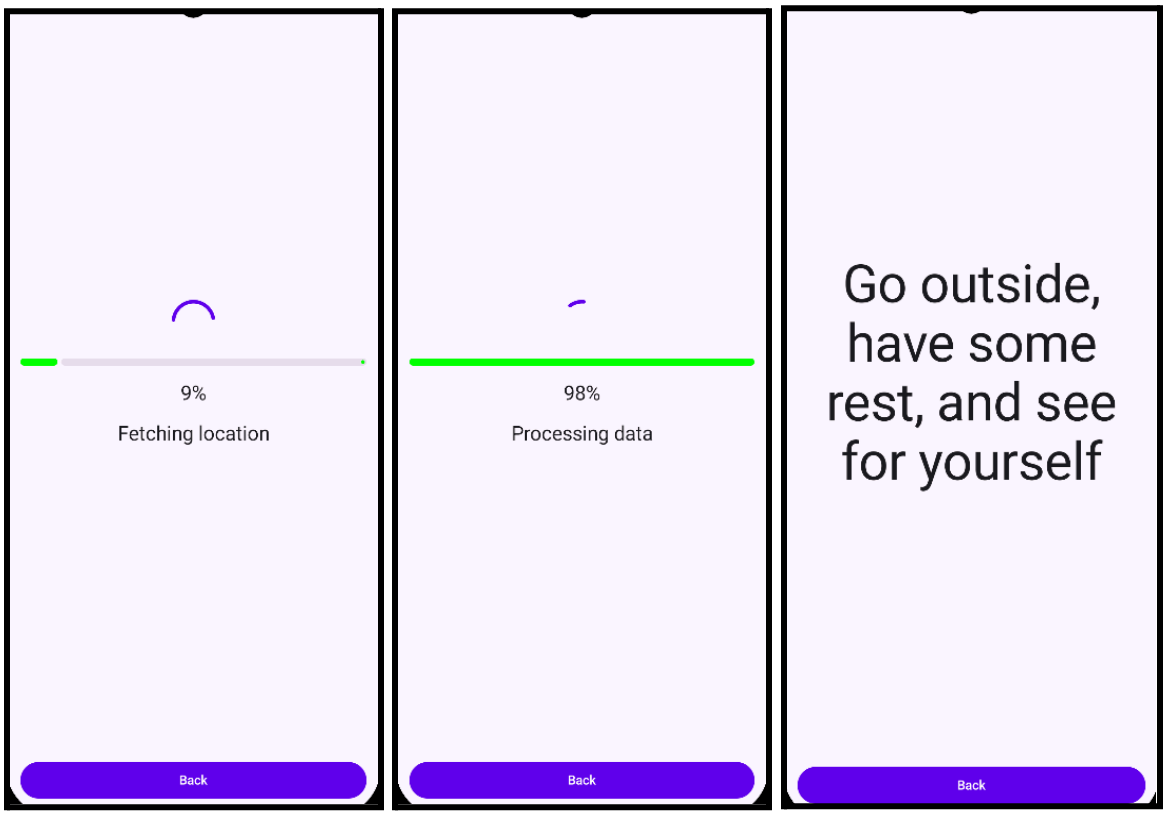
---

8. Fun Weather Simulation Feature

Description:

SkyCast features a playful *Fun Weather Simulation* screen (`FunActivity`) that mimics weather data retrieval through four animated stages: *Fetching Location (15%)*, *Getting Results (65%)*, *Processing Data (99%)*, and a final *"Go outside!" (100%)* message. It enhances engagement while remaining accessible.

Visualization:



Implementation:

Uses `LinearProgressIndicator` and `CircularProgressIndicator` with `animateFloatAsState` for smooth progress, timed delays with `LaunchedEffect`, and `semantics` for TalkBack support.

Tools Used:
Jetpack Compose, `animateFloatAsState`, `LaunchedEffect`, `semantics`, `LiveRegionMode`, and Material Design indicators.