

Project: Glasscast — A Minimal Weather App

Overview

- A SwiftUI weather app using MVVM, dependency injection via Environment, async/await, CoreLocation, MapKit, AppStorage, and glass-styled UI with animated WeatherBackground. Authentication flows reference Supabase/Auth. Haptics provide tactile feedback, and URLSession helpers for HTTPTypes.

Screens and Their Functionality

1. LoginView (LoginView.swift)

- Purpose: Authentication entry point.
- Features:
 - Email and password inputs with validation and show/hide password toggle.
 - Loading state and error alert on sign-in failure.
 - Navigation:
 - To SignupView for account creation.
 - To TabContainerView on successful sign-in (via LoginViewModel navigateToHome).
 - Styling: WeatherBackground (.coldSnowy), glassEffect and liquidGlass usage, adaptive styling, haptics.
 - DI: Uses AppContainer via @Environment(container); ViewModel resolved in init.

2. SignupView (SignupView.swift)

- Purpose: Account creation.
- Features:
 - Full name, email, password, confirm password fields with validation; terms toggle.
 - Create Account button with loading state and success alert.
 - Navigation:
 - Back to LoginView via link.
 - To LoginView automatically after success (navigateToLogin).
 - Styling: WeatherBackground (.hotHumid), liquidGlass card, glassEffect buttons.
 - DI: Uses AppContainer and SignupViewModel.

3. HomeView (HomeView.swift)

- Purpose: Main weather dashboard.
- Features:
 - Header with title/subtitle.
 - Notifications banner reflecting Settings toggles (Severe Alerts, Daily Summary).
 - Current weather card:
 - City, condition, temperature (C/F), high/low, adaptive symbol styling.
 - Manual refresh button and pull-to-refresh with haptics.

- 5-Day Forecast strip (horizontal): day name, symbol, high/low (unit-aware).
- Sunrise/Sunset card: compact SunriseArc visualization.
- Today's Highlights grid:
 - Feels like, wind, humidity, pressure, visibility, UV index, gust, wind direction/degree, dew point, heat index.
- Air Quality card: EPA/DEFRA if available.
- Precipitation chance card.
- Location-aware:
 - Requests when-in-use authorization.
 - Reverse geocodes current coordinate to city and refreshes.
- City selection integration:
 - Observes SelectedCityStore for external city changes and seeds UI from cachedWeather for instant feedback.
- Units via AppStorage: temperature (C/F), wind (km/h or mph), pressure (hPa or inHg).
- Styling: WeatherBackground, glassEffect/liquidGlass, adaptive foreground color.
- DI: @Environment(.container). Model: HomeViewModel injected in init.

4. RadarView (RadarView.swift)

- Purpose: Map-based radar with current and favorite city markers.
- Features:
 - SwiftUI Map (MapKit) with MapCameraPosition.
 - Centers on user location (LocationProvider) or fits region to favorites.
 - User marker:
 - Custom glass marker.
 - Popup showing city and current weather (temp/high/low/condition); fetch on tap or recenter button.
 - Favorite markers:
 - Loaded from FavoritesStore; coordinates resolved via CLGeocoder or stored lat/lon.
 - Popup fetches weather for the tapped favorite city.
 - Floating current-location glass button with animation and haptics.
 - Styling: WeatherBackground, custom glass markers, adaptive foreground.
 - DI: @Environment(.container) WeatherService for fetching weather.

5. SettingsView (SettingView.swift)

- Purpose: Preferences and profile management.
- Features:
 - Weather Units via AppStorage:
 - Temperature (C/F), Wind (km/h or mph), Pressure (hPa or inHg).
 - Appearance: Color scheme selection via ColorSchemeManager (light/dark/system).
 - Notifications: Severe Alerts and Daily Summary toggles via AppStorage.

- Profile: name, email, premium badge; sign-out with confirmation and progress state.
- Navigation:
 - Invisible NavigationLink to LoginView when SettingsViewModel.navigateToLogin toggles.
 - Styling: WeatherBackground (.sunny), glassEffect, adaptive foreground.

Architecture and Key Components

• **MVVM**

- ViewModels: HomeViewModel (provided), LoginViewModel (referenced), SettingsViewModel (referenced), SignupViewModel (referenced).
- ObservableObject + @Published state; Views use @StateObject/@ObservedObject/@EnvironmentObject.

• **Dependency Injection**

- AppContainer (provided) exposed via @Environment(.container).
- Factories like makeHomeViewModel() used in navigation from LoginView.
- WeatherService protocol with MockWeatherService default for previews/dev.

• **Asynchronous Concurrency**

- async/await for networking and geocoding.
- Task and .task modifiers; .task(id:) to react to changing inputs (e.g., location updates).
- MainActor.run for UI updates from background tasks.
- CancellationError handling and monotonic task IDs in HomeViewModel to prevent stale updates.

• **Location and Geocoding**

- LocationProvider wraps CLLocationManager; publishes authorizationStatus and coordinate.
- CLGeocoder used for reverse geocoding (coordinate -> city) and forward geocoding (city -> coordinate for favorites).
- Delegate methods are nonisolated and hop back to MainActor.

• **MapKit (SwiftUI Map)**

- Map, MapCameraPosition, MKCoordinateRegion, Annotation with custom content.
- Region fitting logic across multiple coordinates.

• **Preferences, Theming, and Effects**

- AppStorage for persistent settings (temperature, wind, pressure, notifications, scheme).

- WeatherBackground provides animated, themed layers (Sunny, Rain, Storm, Snow, Wind, Fog, Hot/Humid) using Canvas, TimelineView, gradients, paths, and animations.
- ColorSchemeManager (referenced) to select scheme and adaptive foreground colors.
- Custom glassEffect and liquidGlass modifiers used throughout.
- Networking Helpers
 - URLSession+HTTPTypes: async convenience methods bridging HTTPTypes to URLSession (data/upload/download/bytes), with availability annotations and conditional compilation.
- Constants and Accessibility
 - AppConstants: app info, storage keys, Supabase and Weather API keys/paths/headers, UI strings, symbols, accessibility labels, map defaults, logging prefixes, and mock data.

Detailed Concepts Used (by Swift/SwiftUI topic and file)

- SwiftUI view composition and modifiers
 - Files: LoginView.swift, SignupView.swift, HomeView.swift, RadarView.swift, SettingView.swift, WeatherTheming.swift
 - Concepts: View result builders; NavigationLink; ScrollView; VStack/HStack/LazyVGrid; GeometryReader; ProgressView; Button; Label; Image; Map; Annotation; custom subviews; previews.
- **Property wrappers**
 - @State, @StateObject, @ObservedObject, @EnvironmentObject, @Environment, @AppStorage, @Binding, @Published, @FocusState, @MainActor
 - Files: All major views; HomeViewModel; LocationProvider; SelectedCityStore; UI tests use @MainActor on test methods.
- **Concurrency and task management**
 - async/await; Task; .task (and .task(id:)); MainActor.run; CancellationError; refreshable
 - Files: HomeView, RadarView, SettingsView, HomeViewModel, LocationProvider, UI tests.
- **Protocols and mocking**
 - WeatherService protocol with MockWeatherService implementation
 - Files: HomeViewModel.swift
- **Data models**
 - CurrentWeather, ForecastDay (Equatable, Sendable; Identifiable for ForecastDay)

- Files: HomeModel.swift

- **MVVM and ObservableObject**

- HomeViewModel (state + refresh lifecycle), other ViewModels referenced
- Files: HomeViewModel.swift; referenced in LoginView, SettingsView, SignupView.

- **Dependency Injection via Environment**

- AppContainer via @Environment(.container)
- Files: LoginView.swift, HomeView.swift, RadarView.swift, SignupView.swift, AppContainer.swift.

- **CoreLocation and geocoding**

- LocationProvider (authorization + coordinate), CLGeocoder in HomeView and RadarView
- Files: LocationProvider.swift, HomeView.swift, RadarView.swift.

- **MapKit (SwiftUI Map)**

- Map, MapCameraPosition, MKCoordinateRegion, custom annotations and popups
- Files: RadarView.swift.

- **Custom rendering and animation**

- WeatherBackground and thematic layers with Canvas/TimelineView/gradients/paths/animations
- Files: WeatherTheming.swift.

- **Formatting and units**

- TemperatureUnit conversions and unit label; local helpers for wind/pressure/visibility formatting
- Files: TemperatureUnit.swift; HomeView.swift.

- **Haptics**

- HapticFeedback utility encapsulates UIKit feedback generators (impact, notification, selection).
- Files: HapticFeedback.swift; called in LoginView, HomeView, RadarView, SettingsView.

- **Networking helpers and conditional compilation**

- URLSession+HTTPTypes bridging HTTPTypes with URLSession; availability attributes; FoundationNetworking import
- Files: URLSession+HTTPTypes.swift.

File-by-File Functionality Notes

- LoginView.swift: Auth UI, DI, navigation to tabs, glass styling, haptics, alert handling.
- SignupView.swift: Account creation UI, DI, success alert and navigation, glass styling.
- HomeView.swift: Weather dashboard—current, forecast, sunrise/sunset, highlights, AQI, precipitation; pull-to-refresh; location-aware; unit-aware; adaptive theme.
- RadarView.swift: Map with user and favorite markers/popups; geocoding and weather fetch; region fitting; glass markers; recenter button.
- SettingView.swift: Units, appearance, notifications, profile, sign-out; link to login; glass styling.
- HomeViewModel.swift: WeatherService protocol; MockWeatherService; async concurrent fetch of current + forecast; cancellable refresh with task IDs; LoadingState lifecycle.
- HomeModel.swift: CurrentWeather and ForecastDay models; Equatable/Sendable; Identifiable for ForecastDay.
- LocationProvider.swift: CoreLocation wrapper; publishes authorizationStatus and coordinate; manages start/stop; delegate callbacks.
- SelectedCityStore.swift: Selected city and coordinate; cachedWeather to pre-seed HomeView.
- AppConstants.swift: Central constants (app info, storage keys, Supabase, Weather API, UI strings, symbols, accessibility, map defaults, logging, mock data).
- WeatherTheming.swift: WeatherTheme and WeatherBackground with animated layers.
- TemperatureUnit.swift: AppStorage-backed temperature unit conversion helpers and unit label.
- HapticFeedback.swift: UIKit-based haptic utilities.
- URLSession+HTTPTypes.swift: Async URLSession helpers bridging HTTPTypes.
- AppContainer.swift: DI container providing session, services, FavoritesStore, and factories; EnvironmentKey for .container.