[iOS] Coding Challenge: Real-Time Price Tracker App

Objective

Build an **iOS app using SwiftUI** that displays real-time price updates for multiple stock symbols (e.g., NVDA, AAPL, GOOG, etc.) and supports a second screen for symbol details using **SwiftUI NavigationStack**.

Requirements

Core Features

Live Price Tracking for Multiple Symbols

- Use **25 stock symbols** to ensure the list is long and scrollable (e.g., AAPL, GOOG, TSLA, AMZN, MSFT, NVDA, etc.).

WebSocket Echo Integration

- Connect to: `wss://ws.postman-echo.com/raw`
- Every 2 seconds, for each symbol:
    - Generate a random price update.
    - Send it to the WebSocket server.
    - Receive the echoed message.
    - Update the symbol's data in the UI accordingly.

Real-Time Price Display (Feed Screen)

- Use a **List** or **LazyVStack** to display one row per symbol.
- Each row shows:
    - Symbol name (e.g., AAPL)
    - Current price
    - Price change indicator (green ↑ / red ↓)
- List is **sorted by price value** (highest at the top).
- Tap a row to open the **Symbol Details screen** for that symbol.

Top Bar UI (Feed Screen)

- **Left**: Connection status indicator (🟢 connected / 🔴 disconnected).
- **Right**: Toggle button to Start / Stop the price feed.

Symbol Details Screen

- Shows:
    - Selected symbol as title.
    - Current price with ↑/↓ indicator (same logic as feed).
    - Some description about the symbol.

Technical Expectations

- 100% **SwiftUI** UI.
- Architecture: **MVVM** or **Unidirectional Data Flow**.
- Use **NavigationStack** with two destinations:
    - `feed` (start destination)
    - `symbol details`
- Use **Combine** to handle WebSocket data streams.
- Maintain **immutable UI state** and clean separation of concerns.
- Proper use of:
    - `ObservableObject` and `@Published`
    - `@StateObject` / `@ObservedObject`
    - `@EnvironmentObject` or `@State` for shared state
- The WebSocket stream should be managed so **both screens can observe updates without duplicate connections**.

Bonus (Optional)

- Price flashes green for 1s on increase and red for 1s on decrease.
- SwiftUI UI tests or unit tests.
- Light and dark themes.
- Deep link: `stocks://symbol/{symbol}` that opens the details screen.

Deliverables

Create a **public GitHub repository** that includes:

- Source code with **progressive commit history**.
- A clean, well-structured **README.md**.

Submission Details

Submit your solution via a **GitHub repository link**.