Technical Test – Implementation Notes

Overview

This prototype focuses on a reusable UI framework for a mobile game: screen management, reusable pop-up/windows, a persistent top bar and bottom navigation. The goal was to deliver a production-style structure (prefabs + code) that scales to new features quickly.

Project Structure

UIManager (prefab)

`UIManager` centralises all UI. It contains four empty anchors:

- **Scenes** – Container for future screens (Home, Shop, Map…). Screens are instantiated/destroyed on demand.

- **TopBar** – Persistent top bar prefab. Kept always loaded to avoid re-instantiation and to allow quick show/hide per screen.

- **NavBar** – Persistent bottom navigation prefab with the same rationale as the TopBar.

- **Windows** – Container for modal UI (pop-ups, full-screen overlays, toasts).

Scripting Layer

Two lightweight controller scripts provide a consistent API:

- **UIManagerView** – Entry point that owns references to the anchors above and exposes methods to push/pop screens and windows.

- **UIObjectView** – Base behaviour for screen/window prefabs (lifecycle hooks, events, and helpers).

This split keeps screen/window logic focused while centralising navigation in one place.

Base Prefabs (reusability)

Located at `Assets/Tripledot/Common/Prefabs/Base/`:

- **ScreenBase** – Template for feature screens (used by *Home*).
- **WindowBase** – Template for modal pop-ups (used by *Settings*).
- **FullScreenWindowBase** – Template for full-screen overlays (used by *Level Completed*).
- **TXT_TextBase** – TextMeshPro-based prefab to simplify localisation.

Common UI elements (buttons in two colour variants and a toggle) live under `Common/Prefabs`. Buttons share a single 9-slice white sprite and are tinted via the Image component and UIGradient script for optimal reuse.

Tooling & Packages

- **DOTween** – Simple, performant tweens for UI transitions.
- **UIGradient** – Two-colour gradients for UI styling (kept simple for the test; can be extended later).
- **Sprite Atlases** – Atlases created for Common, NavBar and TopBar assets (high-reuse elements). Feature-specific sprites remain in their folders and can join an atlas if/when needed.
- **SafeArea:** I used the safe area plugin to make sure all the screens look good in all the possible devices.

Task 1 – Home Screen

- The game looks like the home screen is a 3D scene with the NavBar and the TopBar, so my Home screen uses a placeholder image as background.

- **TopBar** and **NavBar** are *outside* the `HomeScreen` prefab by design (so they can be persistent across multiple screens and need fast show/hide without re-instantiation).

- Two debug buttons demonstrate `NavBar` show/hide animation and the Level Completed flow.

Task 2 – Settings Popup

- Implemented from **WindowBase** for maximum reuse: title area, content slot, and action buttons.

- **Buttons**: base prefab + two colour variants. Each button supports *text only*, *icon only*, or *text* + *icon*; `ContentSizeFitter` handles variable string lengths.

- **Toggle**: reusable toggle with feedback animation.

- **SettingValue** prefab (icon + label + toggle) to quickly add new settings.

Task 3 – Level Completed

- Implemented as a **FullScreenWindowBase** variant with **open/idle/close** animations and supportive particle systems.

Optimisation Notes

- **Sprite Atlases**: applied to high-reuse UI (Common/NavBar/TopBar) to reduce texture swaps and improve batching. Feature atlases can be added per module if memory or loading granularity requires it.

- **9-slice + tint**: a single white sprite tinted in materials/images avoids extra textures.

- **Mobile UI**: safe areas handled at Canvas/layout level (supports notches and home indicators).

Final Observations

I would like to clarify that I did this test having a crunch phase in my actual job so I tried to dedicate as much time as I could. But for example, the last task would be much more complete if I had more time. That is why I am adding two videos to the deliverable that show my skills with animation and particles so you can have a good picture about it.

Closing

The project showcases strong UI architecture (reusable bases and managers), clean animation/VFX integration, and practical optimisation choices. I'm confident the structure will scale smoothly to additional screens and features.