**Airbox Systems Mobile Application Documentation**

**Overview**

This document outlines the design, architecture, and implementation of the mobile application developed for Airbox Systems. The app allows users to view images categorized into three groups: Cars, Helicopters, and Boats. It is built using .NET 8 MAUI, ensuring cross-platform compatibility with Android, iOS, and optionally Universal Windows Platform (UWP).

**Table of Contents**

[**Airbox Systems Mobile Application Documentation** 1](#_Toc176903789)

[**Overview** 1](#_Toc176903790)

[**Table of Contents** 1](#_Toc176903791)

[**Objective** 2](#_Toc176903792)

[**Technology Stack** 2](#_Toc176903793)

[**App Architecture** 2](#_Toc176903794)

[**MVVM Pattern** 2](#_Toc176903795)

[**Features** 2](#_Toc176903796)

[**UI Components** 3](#_Toc176903797)

[**Image Handling** 3](#_Toc176903798)

[**Enhancements** 3](#_Toc176903799)

[**Installation Guide** 3](#_Toc176903800)

[**Prerequisites** 3](#_Toc176903801)

[**Steps to Build and Run** 4](#_Toc176903802)

[**Testing and CI/CD** 4](#_Toc176903803)

[**Conclusion** 5](#_Toc176903804)

**Objective**

The primary goal of this application is to provide a streamlined interface that enables users to view images from three categories: **Cars**, **Helicopters**, and **Boats**. Each category can display up to nine images in a responsive, dynamic layout. The app allows users to tap any image for full-screen viewing, with an option to return to the main gallery view.

**Technology Stack**

* **Framework**: .NET 8 MAUI (Multi-platform App UI)
* **Programming Languages**: C#, XAML
* **Platforms**: Android, iOS, optionally UWP
* **Design Pattern**: Model-View-ViewModel (MVVM)
* **Tools**: Microsoft Visual Studio or JetBrains Rider

**App Architecture**

The application follows the **MVVM (Model-View-ViewModel)** architecture, which ensures a clean separation between the user interface and business logic, facilitating code maintainability and readability.

**MVVM Pattern**

* **Model**: Handles the core data logic
* **ViewModel**: Manages the communication between the UI and Model. It handles user commands (e.g., selecting an image) and ensures that the View remains platform-agnostic.
* **View**: XAML pages that represent the UI. The dynamic display of images and buttons for category selection are defined here.

**Features**

1. **Category Selection**:
   * The app provides three buttons, one for each category: Cars, Helicopters, and Boats.
   * Upon tapping a button, the app loads and displays up to nine images from the corresponding category.
2. **Dynamic Layout**:
   * The gallery view dynamically adjusts based on the number of images available for each category, ensuring optimal display.
3. **Full-Screen Mode**:
   * Users can tap any image to view it in full-screen mode. An option to return to the main gallery view is provided.
4. **Cross-Platform Support**:
   * The app is designed to work on Android and iOS. Support for UWP is optional and can be extended.

**UI Components**

1. **PresenterPage View**: Contains three buttons for category selection (Cars, Helicopters, and Boats).

It also displays a grid of images from the selected category, adjusting to the number of available images (up to nine).

1. **FullScreenImage View**: Displays a selected image in full-screen mode with a back button to return to the gallery.

**Image Handling**

* Images are stored in folders categorized as Cars, Helicopters, and Boats.
* The app uses synchronous programming to load images from these folders efficiently.
* A dynamic grid layout is used to display up to nine images per category.
* The full-screen image view is implemented using a popup page, ensuring a clean transition from the gallery to the full-screen display.

**Enhancements**

In addition to the core functionality, the app can include optional enhancements such as:

* **Animations**: Applied smooth transition between full-screen views and gallery.
* **UWP Visual Layer Animations**: If UWP support is included, additional animations can enhance user interactions.

**Installation Guide**

**Prerequisites**

* Ensure that **.NET 8 SDK** is installed.
* Install **Visual Studio** or **JetBrains Rider** with Xamarin and MAUI support.
* Ensure the required Android and iOS emulators are configured for testing.

**Steps to Build and Run**

1. Clone the repository from the provided GitHub link:

bash

Copy code

git clone https://github.com/AndreiZachi/AirboxApp

1. Open the solution in **Visual Studio** or **JetBrains Rider**.
2. Restore NuGet packages by running:

bash

Copy code

dotnet restore

1. Deploy the app to an emulator or connected device:
   * For Android:

bash

Copy code

dotnet build -t:Run -f net8.0-android

* + For iOS:

bash

Copy code

dotnet build -t:Run -f net8.0-ios

**Testing and CI/CD**

1. **Unit Testing**:
   * The app includes unit tests for image loading and navigation logic using **XUnit**.
   * Use the following command to run tests:

bash

Copy code

dotnet test

1. **Continuous Integration (CI)**:
   * A basic GitHub Actions pipeline is configured for automated builds and tests on each commit.
   * The pipeline also ensures compatibility across platforms (Android/iOS/Windows).

**Conclusion**

This application delivers a clean, user-friendly interface for browsing categorized images. Leveraging modern cross-platform development practices, it offers a maintainable codebase with an emphasis on user experience. Further enhancements, such as animations and UWP support, can be added based on project requirements.