1. Overview

The Airlines App is designed to display a list of airlines, allow users to mark airlines as favorites, view detailed information about each airline, and make phone calls or visit websites related to the airlines.

2. Key Decisions & Assumptions

Architecture:

- **MVP (Model-View-Presenter) Pattern**: Used to ensure a clear separation of concerns:
    - **Model**: The **Airline** class represents the data model for an airline. It manages the airline's data and state, including its properties and how it is encoded/decoded for persistence and network requests.
    - **View**: ***AirlinesVC*** and ***AirlineDetailVC*** handle the presentation of data and user interactions.
    - **Presenter**: ***AirlinesPresenter*** and ***AirlineDetailPresenter*** manage the business logic, interact with the model, and update the view.

UI Components:

- **TableView:** Utilized for listing airlines, with custom cells (***AirlineCell***) to display airline information.
- **UISegmentedControl:** Used for filtering between "***All Airlines***" and "***Favorites***."
- **Activity Indicator:** Implemented for loading states during data fetch operations.
- **UIKit:** The app uses UIKit for its user interface components, leveraging standard UI elements to create a native look and feel:
    - **UITableView:** Displays a list of airlines with custom cell configurations.
    - **UILabel, UIImageView, UIButton:** Standard UIKit components used within custom table view cells for displaying airline details and handling user interactions.
    - **UIView:** Custom views and layout configurations are handled using UIKit's layout system.

Data Handling:

- **Remote Data Fetching**: Airlines data is fetched from a remote API if the local database is empty.
- **Local Data Management**: Airlines data is stored in **Realm**. This helps in offline access and faster performance.

User Interaction:

- **Favorite Management**: Users can mark airlines as favorites. The favorite status is managed locally in **Realm** and updated dynamically.

- **CallKit Integration**: CallManager handles phone calls using **CallKit** to manage incoming calls and interactions. It is designed to provide seamless call handling and reporting within the app.

- **Web Interaction:** Allows users to open airline website directly from the app**.**

Error Handling:

- **Error Reporting**: Errors during data fetching are displayed in the console. The UI is updated based on the success or failure of data fetch operations.

Extensibility and Code Reusability:

- **Notification Handling**: Used to update the airline data across different views *(Notification.Name.airlineDidUpdate)*.

- **UIImageView Extension**: *loadImage* method uses SDWebImage for image loading and caching.

- **UIView Extension**: Provides methods to show and hide loading indicators.