**Documentation of Flutter project code that provide explanations for design choices and trade-offs:**

**1. Main.dart**

**Get Package Usage**: The app utilizes the Get package for state management and navigation, providing a simple and reactive approach.

**Theme Configuration:** Light and dark themes are set up for the application.

**Initial Route and Binding**: The initial route is set to the SplashScreen, and bindings for controllers are defined using Get.put in the initialBinding.

**2. HomeScreen.dart**

**Bottom Navigation Bar:** A BottomNavigationBar is used for navigation between different screens.

**PageView for Screen Navigation:** A PageView is implemented for screen navigation, allowing users to swipe between different sections.

**AppBar:** The app features a common AppBar with action buttons for search, casting, and more.

**Drawer:** There is a placeholder Drawer widget, which can be customized for additional navigation or settings.

**3. GatxController.dart**

**MovieController:** Manages the state for the list of movies fetched from the API.

**4. SplashScreen.dart**

**Timer Usage:** A Timer is used to wait for 3 seconds before navigating to the home screen, providing a splash screen experience.

**FlareActor and Icon Combination**: A combination of FlareActor (from the Flare Flutter package) and Icon is used for the animated movie icon on the splash screen.

**Navigation:** The Navigator is used to push the HomeScreen after the timer expires.

**5. FavouriteController.dart**

**FavoriteController:** Manages the state of favorite movies using GetX, including storing and retrieving favorites from shared preferences.

**6. MovieListWidget.dart**

**FutureBuilder:** Uses a FutureBuilder to fetch and display a list of movies from the MovieController.In the FutureBuilder widget, a CircularProgressIndicator is displayed when the data is being fetched. This provides visual feedback to the user that the app is working on loading the movie data.

**MoviesGridView:** Displays movies in a GridView with a dynamic number of columns based on screen orientation.

**Expanded Widget:** Each MovieItemWidget is wrapped in an Expanded widget to avoid overflow issues.

**7. MovieItemWidget.dart**

GestureDetector: Uses a GestureDetector for tapping on movies to navigate to the details page.

Obx Widget: Utilizes the Obx widget to reactively update UI based on changes in favorite status.

Column and SingleChildScrollView: Columns are wrapped in SingleChildScrollView to handle overflow issues.

**8. MovieDetailsPage.dart**

Details Page: Displays details of a selected movie, including title, release date, overview, and an image.

ClipRRect and Image.network: Uses ClipRRect for rounded corners and Image.network to display the movie poster.

Padding and Column: Implements structured layout using Padding and Column widgets.

**9. MovieModel.dart and MovieService.dart**

**Movie Model:** Defines the Movie class to represent movie data.

**Movie Service:** Fetches movies from the API using the http package.

**10. Categories.dart and Channels.dart**

**Placeholder Widgets**: These are placeholder widgets, and you can replace them with the actual content for categories and channels.

**Overall Design Choices:**

**GetX State Management:** The project uses GetX for its simplicity and reactivity, making it easy to manage the state and update the UI.

**Navigation:** The app uses the Get.to method for navigation, which is part of the Get package.

**Responsive Design:** The use of Expanded, SingleChildScrollView, and dynamic column count in MoviesGridView contributes to a responsive design.

**Trade-offs:**

**Placeholder Widgets:** The categories and channels screens are implemented as placeholder widgets. Actual implementations would depend on the application requirements.

**Splash Screen Duration:** The 3-second timer on the splash screen might be adjusted based on user experience considerations.

This documentation provides an overview of the project structure, design choices, and trade-offs.

**Overall The Flutter Movie and Shows App features a dynamic interface with a responsive grid layout, allowing users to explore and mark their favorite movies. Utilizing GetX for efficient state management, the app seamlessly navigates between home, favorites, categories, and channels screens. A visually engaging splash screen with an animated movie icon precedes the app, enhancing the user experience.** **The app retrieves movie data from "The Movie Database (TMDb)" API using an asynchronous network request. The MovieService class handles the API communication, sending a request to TMDb's movie discovery endpoint. The response, containing a list of movies in JSON format, is then parsed and converted into a list of Movie objects. This approach ensures real-time and up-to-date movie information within the app.**