1. Project Overview

- **Project Name**: AstroSagga
- **Architecture**: Architecture used MVVM.
- This app follows the MVVM (Model-View-ViewModel) architecture pattern.
- State management is handled using GetX for ease of navigation.
- Dio for API Calls: Dio is used for making HTTP requests, handling retries, and intercepting API responses.

2. Folder Structure

-lib/
—binding [Global Binding]
    all_astrologer_binding.dart
    recharge_voucher_binding.dart

-—controller [Global Controllers]
    all_astrologer_controller.dart
    app_update_controller.dart
    auth_controller.dart
    connectivity_service.dart
    razorpay_controller.dart

—model [Global Models]

—util
    app_constants.dart [Contains base url, API Endpoints and Constant Variables]
    app_events.dart [Firebase analytics and facebook events]
    app_strings.dart [Constant Strings]
    color_resources.dart
    common_methods.dart
    dimensions.dart
    images.dart
    language_translation.dart
    shared_preference.dart

```
—view
        -screens
                -Agora
                        -Live_stream
                                -Controller
                                -Model
                                -Wiget
                                 live_stream_screen.dart

                        -Videocall
                                -controller
                                -model
                                service_video_call_screen.dart
                                video_call_screen.dart

                        -voicecall
                                -controller
                                voice_call_screen.dart


                -Astro_service
                        -binding
                        -controller
                        -model
                        Service_astrologer_list_screen.dart
                        Service_category_list_screen.dart
                        Service_detail_screen.dart
                        service_list_screen.dart

                -Astro_shop
                        -binding
                        -controller
                        -model
                        add_address_screen.dart
                        payment_summary_screen.dart
                        product_detail_screen.dart
                        products_category_list_screen.dart
                        products_list_screen.dart
                        Save_address_screen.dart
```

-Astrologer_detail
    -Binding
    -controller
    astrologer_profile_detail_screen.dart

-Blog
    -binding
    -controller
    Blog_detail_screen.dart
    blog_screen.dart

-Chat
    -binding
    -controller
    -model
    -widget
    Chat_channel_list_screen.dart
    Chat_history_screen.dart
    chat_screen.dart

-Contact_us
    Assistant_screen.dart

-Courses
    Dash_courses_screen.dart

-Dashboard
    -binding
    -controller
    Dash_call_chat_astrologer_screen.dart
    Dash_home_screen.dart
    Dash_live_screen.dart
    Dash_profile_screen.dart
    dashboard_screen.dart

-Deeplink
    Deeplink_astrologer_detail.dart
    deeplink_product_detail.dart

-Feedback
    -controller
    feedback_screen.dart

-Followings

-Intake_form
-Language
-Login
-Orders
-Queue_list
-Search
-searchPlace
-Splash
-Update_profile
-Vedic_astro
-Video_section
-Wallet
-webview

-widgets [Global Widgets]

main.dart

bindings/: Handles the binding of controllers and services.

controllers/: Contains the business logic controllers.

models/: Contains the data models.

views/: All UI components for different screens.

utils/: Common utilities, helper functions, and constants.

3. Design Patterns

The app uses GetX to manage state. Controllers are used to manage business logic, and the views are reactive to the changes in the controller's state.

4. Navigation Pattern

The app uses GetX for navigation, leveraging named routes for better code structure.
[route names are defined in main.dart file]

5. Error Handling

Error handling is done through try-catch blocks and GetX's in-built snack bars for user notifications.

6. Conclusion

The app is structured following clean code principles, with separation of concerns and reusability in mind. Future developers can extend the app easily by following the modular structure.

Synilogic Tech Pvt Ltd