

Dr. Divyanshu – 5D Integrative Health App Architecture

Technology Stack

Frontend Framework

- **Flutter:** Cross-platform mobile development framework
- **Dart:** Programming language for Flutter
- **Material Design 3:** UI design system with dark theme support

State Management

- **Flutter BLoC:** Business Logic Component pattern for state management
- **Equatable:** Value equality for Dart objects
- **Hydrated BLoC:** Persistent state management

Navigation

- **Go Router:** Declarative routing for Flutter
- **Auto Route:** Code generation for type-safe navigation

Local Storage

- **Hive:** Lightweight and fast key-value database
- **Shared Preferences:** Simple key-value storage for settings
- **SQLite:** Relational database for complex data

Network & API

- **Dio:** HTTP client for REST API calls
- **Retrofit:** Type-safe HTTP client generator
- **JSON Annotation:** JSON serialization/deserialization

UI Components

- **Cached Network Image:** Efficient image loading and caching
- **Flutter SVG:** SVG rendering support

- **Lottie:** Animation support
- **Shimmer:** Loading placeholder animations

External Integrations

- **URL Launcher:** WhatsApp bot integration
- **Google Sheets API:** Data synchronization
- **Firebase:** Push notifications and analytics
- **WebView Flutter:** In-app web content

Development Tools

- **Flutter Launcher Icons:** App icon generation
- **Flutter Native Splash:** Splash screen generation
- **Build Runner:** Code generation
- **Flutter Lints:** Code quality and style

Project Structure

```
lib/
├── core/
│   ├── constants/
│   │   ├── app_colors.dart
│   │   ├── app_strings.dart
│   │   └── api_endpoints.dart
│   ├── errors/
│   │   ├── exceptions.dart
│   │   └── failures.dart
│   ├── network/
│   │   ├── api_client.dart
│   │   └── network_info.dart
│   ├── theme/
│   │   ├── app_theme.dart
│   │   └── dark_theme.dart
│   └── utils/
│       ├── validators.dart
│       └── helpers.dart
├── features/
│   ├── consultation/
│   │   ├── data/
│   │   ├── domain/
│   │   └── presentation/
│   ├── ayurveda/
│   │   ├── data/
│   │   └── domain/
```

```
| | | presentation/
| | | naturopathy/
| | | | data/
| | | | domain/
| | | | presentation/
| | | acupuncture/
| | | | data/
| | | | domain/
| | | | presentation/
| | | nutrition/
| | | | data/
| | | | domain/
| | | | presentation/
| | | yoga/
| | | | data/
| | | | domain/
| | | | presentation/
| | | beauty/
| | | | data/
| | | | domain/
| | | | presentation/
| | | surgery/
| | | | data/
| | | | domain/
| | | | presentation/
| | | ebooks/
| | | | data/
| | | | domain/
| | | | presentation/
| | | health_tips/
| | | | data/
| | | | domain/
| | | | presentation/
| | shared/
| | | widgets/
| | | | custom_app_bar.dart
| | | | module_card.dart
| | | | consultation_form.dart
| | | | loading_widget.dart
| | | models/
| | | | user_model.dart
| | | | consultation_model.dart
| | | | health_tip_model.dart
| | main.dart
```

Clean Architecture Layers

1. Presentation Layer

- **Widgets:** UI components and screens
- **BLoC:** Business logic and state management
- **Pages:** Screen implementations

2. Domain Layer

- **Entities:** Core business objects
- **Use Cases:** Business logic operations
- **Repositories:** Abstract data access interfaces

3. Data Layer

- **Models:** Data transfer objects
- **Data Sources:** Remote and local data access
- **Repository Implementations:** Concrete data access

Module Architecture

Each of the 10 modules follows the same architectural pattern:

Data Flow

1. **UI Event** → BLoC receives user interaction
2. **BLoC** → Calls appropriate use case
3. **Use Case** → Executes business logic
4. **Repository** → Fetches data from data source
5. **Data Source** → Returns data to repository
6. **Repository** → Returns data to use case
7. **Use Case** → Returns result to BLoC
8. **BLoC** → Emits new state to UI

State Management Pattern

```
// BLoC Events
abstract class ConsultationEvent extends Equatable {}

class LoadConsultationForm extends ConsultationEvent {}
class SubmitConsultation extends ConsultationEvent {}
```

// BLoC States

```
abstract class ConsultationState extends Equatable {}
```

```
class ConsultationInitial extends ConsultationState {}
```

```
class ConsultationLoading extends ConsultationState {}
```

```
class ConsultationLoaded extends ConsultationState {}
```

```
class ConsultationError extends ConsultationState {}
```

External Service Integration

WhatsApp Bot Integration

- **URL Launcher:** Opens WhatsApp with pre-filled message
- **Deep Linking:** Direct integration with WhatsApp Business API
- **Phone Numbers:**
 - Main Health Bot: +91 9695570344
 - OPD Booking Bot: +91 7019620344

Google Sheets Integration

- **Google Sheets API:** Real-time data synchronization
- **Webhook Support:** Automatic data updates
- **Authentication:** OAuth 2.0 for secure access

Firebase Integration

- **Cloud Messaging:** Push notifications for health tips
- **Analytics:** User behavior tracking
- **Crashlytics:** Error reporting and monitoring

Security & Privacy

Data Protection

- **Local Encryption:** Sensitive data encrypted using Hive encryption
- **API Security:** HTTPS only, token-based authentication
- **User Privacy:** Minimal data collection, GDPR compliant

Authentication

- **Phone Number Verification:** OTP-based authentication

- **Biometric Authentication:** Fingerprint/Face ID support
- **Session Management:** Secure token handling

Performance Optimization

Image Optimization

- **Cached Network Image:** Efficient image loading
- **Image Compression:** Automatic image optimization
- **Lazy Loading:** Images loaded on demand

Database Optimization

- **Hive:** Fast local storage for frequently accessed data
- **SQLite:** Efficient queries with proper indexing
- **Data Caching:** Smart caching strategy

Network Optimization

- **Dio Interceptors:** Request/response logging and caching
- **Retry Logic:** Automatic retry for failed requests
- **Offline Support:** Local data fallback

Testing Strategy

Unit Tests

- **BLoC Testing:** State management logic
- **Use Case Testing:** Business logic validation
- **Repository Testing:** Data access layer

Widget Tests

- **UI Component Testing:** Individual widget behavior
- **Integration Testing:** Screen-level interactions
- **Golden Tests:** Visual regression testing

Integration Tests

- **End-to-End Testing:** Complete user flows
- **API Testing:** External service integration

- **Performance Testing:** App performance metrics

Build & Deployment

Development Environment

- **Flutter SDK:** Latest stable version
- **Android Studio:** IDE with Flutter plugin
- **VS Code:** Alternative IDE with Flutter extension

Build Configuration

- **Flavors:** Development, staging, production
- **Environment Variables:** API keys and configuration
- **Code Signing:** Android keystore management

CI/CD Pipeline

- **GitHub Actions:** Automated testing and building
- **Fastlane:** Deployment automation
- **Code Quality:** Automated code analysis

Scalability Considerations

Modular Architecture

- **Feature Modules:** Independent, reusable modules
- **Shared Components:** Common UI elements
- **Plugin Architecture:** Easy feature addition

Performance Scaling

- **Lazy Loading:** Load modules on demand
- **Code Splitting:** Reduce initial bundle size
- **Memory Management:** Efficient resource usage

Data Scaling

- **Pagination:** Large data set handling
- **Caching Strategy:** Multi-level caching
- **Database Optimization:** Query optimization