# 1. Architecture Document

## 1.1. Introduction

This document describes the architectural approach for the Flutter prototype app containing 9 modular sub-apps, such as health tracking, games, and communication features.

The purpose of this architecture is to ensure separation of concerns, scalability, and maintainability while keeping the prototype lightweight and easy to navigate. The app follows Flutter’s recommended layered architecture with unidirectional data flow.

## 1.2. High-Level Architecture

## 1.2.1. Layered Structure

The app is organized into three main layers:

- **UI Layer (Presentation):** Contains widgets and screens. Responsible for displaying data and handling user interactions.

- **Logic Layer (Application/Domain Logic):** Includes view models that manage the state and business logic.

- **Data Layer:** Responsible for data sources such as local storage or services. In this prototype, it uses in-memory repositories.

Note: In this prototype, logic and state are handled with in-memory state (via providers), and no real data layer is implemented.

## 1.2.2. Unidirectional Data Flow

- UI → Logic → Data → Logic → UI.

This approach decouples UI from logic and simplifies debugging.

## 1.3. Navigation & Routing

The app uses Flutter’s built-in `Navigator` with named routes for simple navigation between pages. Since the prototype is small, advanced routing is not required.

- Map each major module to a Route path (/health-wellness, /games, /video-call, etc.).

## 1.4. Data Persistence

This prototype does not include persistent storage (e.g., local database or shared preferences). All state is managed in-memory using providers from the riverpod package (or provider, based on what's actually used).

If persistence were needed later, options include:

Backend written in Java Spring Boot, with database used with:

- MySQL

- PostgreSQL

## 1.5. GitHub Delivery

Repository Layout:

/assets - Extra resources and tools

/fonts - Custom fonts

/images - UI images and icons

/lib - App source code

/pages - Different screens

/providers - Simple state management

/widgets - Reusable UI components

main.dart - App entry point

README.md - overview

- CI/CD: optional GitHub Actions to run flutter analyze on pushes.

- Branches: main

## 1.6. INFO:

For the purpose of writing well structured and professional deliverable we used the help of AI to write it.