# Assignment 2: 项目功能升级与技术实现详述

## 1. 系统架构升级与核心设计理念

为了满足 Assignment 2 关于数据持久化和用户认证的要求，项目架构从 Asg1 的纯前端、内存数据状态，演进为 **以本地数据库为核心的服务驱动型架构**。

- 核心转变:
  - **引入数据持久化层:** 放弃了 Asg1 中应用关闭即丢失的内存数据（`in-memory data`），引入了 `SQLite` 本地数据库。这是本次升级的基石，确保了所有用户数据的长期性和安全性。
  - **封装数据库服务:** 创建了一个独立的 `DatabaseService`。该服务作为应用与数据库之间唯一的通信桥梁，封装了所有 SQL 操作（增删改查）。这种设计实现了**关注点分离 (Separation of Concerns)**，使得业务逻辑层（如 `DataManager` 和各个页面）无需关心具体的 SQL 实现，只需调用服务提供的方法即可，极大地提升了代码的可维护性和可扩展性。
  - **状态管理重构:** 重构了原有的 `DataManager`。它不再自行管理数据，而是作为数据库服务的客户端，负责调用 `DatabaseService` 的方法来获取或修改数据，并将数据提供给 UI 层进行展示。

## 2. 核心功能模块实现详述

以下是对 Asg2 要求新增的每一个核心功能的详细说明。

### 2.1 用户认证系统 (User Authentication)

- 功能目标: 为应用提供一个安全、独立的用户系统，确保每个用户的数据都得到隔离和保护。这是构建个性化体验的基础。
- 设计与实现:
  1. UI/UX 设计:
     - **注册页面（`registration_screen.dart`）:** 设计了一个独立的页面，包含“姓名”、“邮箱”和“密码”三个 `TextField`。为了提升用户体验，密码输入框采用了隐藏文本的属性。
     - **登录页面（`login_screen.dart`）:** 提供“邮箱”和“密码”输入框。界面简洁明了，专注于核心登录功能。通过 `Navigator` 实现与注册页面之间的跳转。
  2. 技术实现:
     - **数据库支持:** 在 `SQLite` 中创建了 `users` 表，其中 `email` 字段被设置为 `UNIQUE` 约束，从数据库层面保证了邮箱的唯一性，防止重复注册。
     - **后端逻辑（`DatabaseService`）:**
       - `Future<void> createUser(User user)`: 接收一个 `User` 模型对象，执行 `INSERT` SQL 语句，将新用户信息写入 `users` 表。
       - `Future<User?> getUser(String email, String password)`: 接收邮箱和密码，执行 `SELECT` 查询。如果找到匹配的用户，则返回 `User` 对象；否则返回 `null`。
     - **前端调用流程:**
       - 用户在注册页面点击“注册”，UI 层将输入数据封装成 `User` 对象，调用 `DatabaseService.createUser()`。
       - 用户在登录页面点击“登录”，UI 层调用 `DatabaseService.getUser()`。根据返回结果，若不为 `null`，则将用户信息保存到全局状态管理器中，并使用 `Navigator.pushReplacementNamed()` 跳转到主页，防止用户回退到登录页。若为 `null`，则显示错误提示（如 `SnackBar`）。

## 2.2 数据库集成与 CRUD 操作

- 功能目标: 实现所有应用核心数据（体重、训练、饮食等）的持久化存储，并提供完整的增、删、改、查 (CRUD) 操作接口。
- 设计与实现:
  1. 技术选型:
     - 根据课程课件 (`Week 08 Lecture.pdf`) 的指导，我们选用了 `sqflite` 包。它是在 Flutter 中使用 SQLite 的官方推荐和社区标准，性能可靠且功能强大。
     - 同时引入 `path` 和 `path_provider` 包，用于安全、跨平台地获取应用文档目录的路径，以存放 `.db` 数据库文件。
  2. 数据库表结构设计:
     - `users` 表: `id`, `name`, `email`, `password`
     - `weights` 表: `id`, `user_id` (外键), `date`, `value`
     - `workouts` 表: `id`, `user_id` (外键), `date`, `name`, `sets`, `is_completed`
     - `nutrition` 表: `id`, `user_id` (外键), `date`, `meal_type`, `name`, `calories`, `amount`
     - 外键约束 (`user_id`): 这是数据库设计的关键。通过将各业务表与 `users` 表的 `id` 相关联，我们建立了数据归属关系。在执行任何 `SELECT` 查询时，都必须附带 `WHERE user_id = ?` 条件，从而保证了登录用户只能访问和操作自己的数据。
  3. CRUD 实现 (`DatabaseService`):
     - Create (创建): `addWeight(WeightEntry entry)` 方法执行 `INSERT INTO weights...`。
     - Read (读取): `getWeights(int userId)` 方法执行 `SELECT * FROM weights WHERE user_id = ?`，返回指定用户的所有体重记录。
     - Update (更新): `updateWorkout(WorkoutEntry entry)` 方法执行 `UPDATE workouts SET is_completed = ? WHERE id = ?`，用于标记训练完成。
     - Delete (删除): `deleteWeight(int id)` 方法执行 `DELETE FROM weights WHERE id = ?`。
     - 通过将这些 SQL 语句封装在 `DatabaseService` 的方法中，我们为上层业务逻辑提供了清晰、语义化的接口。

## 2.3 个人资料页面 (Profile Screen)

- 功能目标: 提供一个中心化的界面，用于展示用户的个人信息，并允许用户对其进行更新。
- 设计与实现:
  - UI/UX 设计: 页面顶部显著位置显示用户的头像和姓名。下方以列表或卡片形式展示详细信息（身高、当前体重、BMI等）。提供一个明确的“编辑”按钮或可点击的字段，以进入编辑模式。
  - 技术实现:
    - 页面加载时，从全局状态管理器获取当前登录用户的 `id`。
    - 调用 `DatabaseService.getUserById(id)` （需在 `DatabaseService` 中新增此方法）来获取最新的用户信息并展示在界面上。
    - 当用户在编辑模式下保存信息时，调用 `DatabaseService.updateUser(user)` 方法，将更新后的 `User` 对象持久化到数据库。

## 2.4 音视频集成 (Audio/Video Integration)

- 功能目标: 丰富应用内容，提升用户体验，并满足课程的高分要求。为用户提供视觉或听觉上的指导和激励。
- 设计与实现:
  1. 技术选型:

- 选用 `video_player` 和 `chewie` 包的组合。`video_player` 是底层播放器，而 `chewie` 在其基础上封装了一套美观、完整的播放器 UI（包含播放/暂停按钮、进度条、全屏切换等），极大地简化了开发工作。

2. **集成方式:**
   - 在 `LibraryScreen` (学习资源库) 页面中创建了一个新的组件 `VideoPlayerWidget`。
   - **资源管理:** 将一个 `.mp4` 格式的健身教学视频文件放置在项目的 `assets/videos/` 目录下，并在 `pubspec.yaml` 文件中进行了声明。
   - **播放器初始化:** 在 `VideoPlayerWidget` 的 `initState` 方法中，初始化 `VideoPlayerController`，并将其与 `ChewieController` 绑定。`ChewieController` 负责管理 UI 的显示和行为。
   - **生命周期管理:** 在 `dispose` 方法中，必须释放 `VideoPlayerController` 和 `ChewieController` 资源，以防止内存泄漏。

## 2.5 帮助/支持页面 (Help/Support Screen)

- 功能目标: 完善应用的功能完整性，为用户提供一个获取帮助的渠道。
- **设计与实现:**
  - **UI/UX 设计:** 这是一个静态信息页面，设计上力求简洁、清晰。使用 `ListView` 或 `Column` 布局，将常见问题 (FAQ) 以问答形式排列，并在页面底部提供联系方式。
  - **技术实现:**
    - 该页面被实现为一个 `StatelessWidget`，因为其内容是静态的，不涉及状态变化。
    - 所有文本内容都硬编码在代码中。在 `SettingsScreen` 中添加一个列表项，通过 `Navigator.push()` 导航到该帮助页面。