ChatGPT

健身助手 Flutter 应用项目结构说明

1. Flutter 代码目录结构

以下是项目主要代码的目录结构（省略了 Android、iOS、.dart_tool 等原生或构建目录），展示了 Flutter 相关文件和资源的组织方式：

```
lib/
├── main.dart
├── mock_data.dart
├── theme.dart
├── models/
│   ├── article.dart
│   ├── weight_entry.dart
│   └── workout_entry.dart
├── screens/
│   ├── dashboard_screen.dart
│   ├── body_screen.dart
│   ├── workout_screen.dart
│   ├── nutrition_screen.dart
│   └── library_screen.dart
└── widgets/
    ├── glass_card.dart
    ├── weight_line_chart.dart
    ├── ring_progress.dart
    └── primary_button.dart

assets/
└── articles/
    ├── muscle_gain.md
    ├── fat_burn.md
    ├── diet.md
    └── recovery.md
```

2. 各文件和目录的功能说明

- **lib/main.dart**：应用程序入口。创建 `FitnessMiniApp` 根部件，内部使用 `MaterialApp` 设置应用标题和主题，并定义主页为 `MainScreen`（带底部导航栏的主界面） [1] 。同时通过 `routes` 映射定义了几个命名路由（如 `/body`，`/workout`，`/nutrition`，`/library`）到对应的页面部件，便于在应用内通过路由名跳转 [2] 。`MainScreen` 是一个 `StatefulWidget`，包含底部导航栏（`BottomNavigationBar`）切换五个主页面（首页、体测、训练、饮食、资源）。`MainScreen` 内维护 `_selectedIndex` 状态，点击导航项时通过 `setState` 更新索引，主界面正文区域根据索引显示对应的页面部件 [3] 。

- **lib/theme.dart**：应用主题配置。提供 `buildAppTheme()` 函数返回一个全局 `ThemeData` 对象，使用预设的种子颜色创建了 Material 3 的颜色方案（主色调为蓝色系，辅助色为绿色系） [4]。同时定制了部分组件主题：比如卡片（`CardTheme`）统一圆角和阴影，提升视觉一致性 [5]；ElevatedButton 统一圆角半径 [6]；AppBar 背景色使用主色并去除阴影 [7]；BottomNavigationBar 设置选中项的颜色等 [8]。这些主题配置确保应用各处的样式一致，符合设计规范。

- **lib/mock_data.dart**：模拟数据管理模块。定义了一个 `MockData` 类，用于存储应用中的所有数据状态并提供更新方法。`MockData` 继承自 `ChangeNotifier`，实现**单例模式**（用工厂构造返回 `_instance` 实例），确保全局只有一个数据源供各界面使用 [9]。主要包含：

- **体重数据**：`weights7d` 列表存储最近7天的体重记录（WeightEntry对象列表），`currentWeight` 当前体重，用于绘制体重折线图等 [10]。初始化时生成模拟的近7天体重数据 [11]。提供 `addWeight(double)` 方法添加新体重记录（保持列表长度不超过7，并更新当前体重），然后调用 `notifyListeners()` 通知界面刷新 [12]。
- **热量数据**：`calorieIntake` 当日摄入卡路里、`caloriesBurned` 当日消耗卡路里、`calorieGoal` 目标卡路里，用于计算热量盈亏。提供 `updateCalorieIntake(int)` 和 `updateCaloriesBurned(int)` 方法更新摄入/消耗值，并 `notifyListeners()` [13]。`calorieBalance` 通过 getter 计算当前摄入-消耗的差值，正为盈余、负为亏空 [14]。
- **训练计划数据**：`workoutToday` 列表保存当天的训练项目（WorkoutEntry对象列表），每项包括名称、组数、是否完成等 [15] [16]。初始化时模拟添加了几项训练 [17]。提供 `addWorkout(String, int)` 增加新训练项、`toggleWorkoutCompleted(int)` 切换某项训练的完成状态（修改对应项的 `isCompleted` 并通知刷新） [18]。`workoutCompletionPercent` getter 计算当天训练完成百分比 [19] 用于展示环形进度。
- **文章资源数据**：`articles` 列表保存若干文章信息（Article对象列表），包括标题、封面图片URL、Markdown文件路径、分类等 [20] [21]。初始化时添加了模拟的训练、饮食、康复类别的文章各若干篇 [21] [22]。这些文章用于资源库界面的列表展示。

`MockData` 的所有修改方法最后都会调用 `notifyListeners()`，这使得已注册监听该数据的界面能够自动收到通知并刷新UI，实现简单的全局状态管理。

- **lib/models/**：数据模型类目录，包含应用中几种数据结构的定义：
- **article.dart**：定义 `Article` 类，用于描述一篇健身文章资源，含 `title` 标题、`coverUrl` 封面图片链接、`mdPath` Markdown文件路径、`category` 分类等属性 [23]。主要供资源库页面使用。
- **weight_entry.dart**：定义 `WeightEntry` 类，表示一条体重记录，包含日期 `date` 和体重值 `value` [24]。提供 `copyWith()` 方法创建修改了某些字段的新实例 [25]。`MockData.weights7d` 列表用该类存储体重数据点。

- **workout_entry.dart**：定义 `WorkoutEntry` 类，表示一次训练记录，包含日期 `date`、动作名称 `name`、组数 `sets`、是否已完成 `isCompleted` 等属性 [16]。同样提供 `copyWith()` 方法方便状态切换 [26]。`MockData.workoutToday` 列表中的元素即为此类实例。

- **lib/screens/**：应用主要功能界面（页面）文件所在目录，每个 Dart 文件定义一个页面对应的 `StatefulWidget`：

- **dashboard_screen.dart**：**首页**（Dashboard）界面。展示汇总的今日健身概览数据，包括三个主要卡片：**今日体重、训练完成度、热量盈亏**。界面使用多个自定义组件组合：比如 GlassCard 作为卡片容器、WeightLineChart 绘制最近7天体重折线图、RingProgress 显示训练完成百分比的环形图等 [27] [28]。页面顶部有 AppBar 标题“健身助手”和设置按钮（目前未实现功能，详见第4节） [29]。此外，Dashboard 页面提供一个“+”号**浮动操作按钮菜单**（FAB menu）：点击主按钮展开3个子按钮，用于快速导航/录入——跳转到训练页面、跳转到饮食页面，以及添加今日体重记录 [30] [31]。其中添加体重按钮会弹出一个对话框让用户输入新体重，并调用 `_mockData.addWeight()` 更新数据 [32]。

Dashboard 页面通过在 `initState` 中对 `MockData` 添加监听器来实现数据刷新：当全局数据变化时，调用本地 `_updateUI()` 执行 `setState` 刷新界面 [33]。在 `dispose` 中移除监听器以防内存泄漏 [34]。这种模式使得例如在其他页面更新了体重后，本页会自动收到通知，显示最新数据。

- **body_screen.dart**：**体测**（身体数据）页面。用于展示和编辑用户的基本身体指标。界面包括：个人资料卡片（显示身高和当前体重，可点击编辑）、BMI指数卡片（显示根据身高体重计算的BMI值，并用颜色区分范围）、体重趋势卡片（折线图显示7日体重变化及统计）、历史体重记录列表等。点击个人资料卡片上的编辑按钮，会弹出对话框允许用户修改身高和当前体重 [35] [36]。保存时更新 `_mockData.height` 和调用 `_mockData.addWeight()`（从而更新当前体重并通知监听者） [37]。页面底部有一个添加体重的浮动按钮，功能与Dashboard类似，打开输入对话框添加新记录 [38]。该页面同样通过监听全局 `MockData` 数据变化来实时更新UI（例如BMI值和体重列表会随数据改变自动刷新） [39] [40]。
- **workout_screen.dart**：**训练计划**页面。用于展示当天的训练任务清单，并提供管理功能。界面顶部卡片显示“今日训练状态”，包括当天训练完成百分比的环形图和完成/总次数 [41]。下方是“训练计划”卡片，列出当天所有训练项目：使用 `ListView.builder` 动态生成列表项，每项显示训练名称和组数，右侧有完成勾选按钮，可切换完成状态，底层通过调用 `_mockData.toggleWorkoutCompleted(index)` 更新数据 [42]。列表项支持向左滑动删除功能（使用 `Dismissible`），删除后从全局列表移除该项目并弹出提示 [43]。在“训练计划”卡片右上有一个编辑按钮，但目前仅作为占位，`onPressed` 尚未实现任何逻辑 [44]。第三个卡片是“快速添加训练”，提供几个常见训练的快捷按钮，横向滚动列表显示预设的动作（俯卧撑、深蹲等），点击某个图标立即调用 `_quickAddWorkout(name, sets)` 将一项默认3组的新训练添加到计划中，并通过 SnackBar 提示 [45] [46] [47]。页面底部还有一个浮动添加按钮“+”，点按弹出底部抽屉对话框让用户自定义添加训练项目（输入名称和组数后保存） [48] [49]。Workout 页面在 `initState` 添加对 `MockData` 的监听，当训练列表或完成情况变化时刷新UI [50] [51]。
- **nutrition_screen.dart**：**饮食记录**页面。用于记录当天饮食摄入并追踪卡路里。页面顶部是“今日热量摘要”卡片，显示摄入、消耗和剩余热量三项数据，并用不同颜色标识（摄入蓝色、消耗绿色、盈余为红/绿色） [52]。卡片下方有一个进度条表示摄入量相对于每日目标的进度 [53] [54]。第二个板块是“快速录入”卡片，包括三个快捷按钮（+100、+250、+500 kcal）用于快速增加摄入热量 [55]；下方列出若干常见食物（鸡蛋、香蕉等）的按钮列表，横向滚动显示，每个食物按钮点击时会直接把对应热量加入总摄入并通知SnackBar提示 [56] [57]。第三个板块是“食物搜索”卡片，里面是一个带搜索和扫码图标的输入框 [58]；目前这只是界面元素，没有实现实际搜索或扫码功能（仅供展示界面，详见第4节)。页面下半部分是按照**早餐、午餐、晚餐**分段的饮食记录列表：通过 `_buildMealSections()` 动态生成三个餐次卡片 [59] [60]。每个餐次卡片列出所属的食物条目（名称、分量、热量），底部有“添加到X餐”按钮，点击可弹出底部对话框，在相应餐次下新增一条记录 [61] [62]。对话框允许填写食物名称、分量和热量，并选择所属餐次，保存后将该记录加入 `_meals` 数据并更新总热量 [63] [64]。列表中的食物项也支持滑动删除，删除后将对应记录移除并更新热量汇总 [65] [66]。Nutrition 页面同样在初始化时监听 `MockData`，主要是为了当消耗量（caloriesBurned）等数据更新时可以刷新剩余热量等显示 [67] [68]（此外初始时调用 `_updateCaloriesFromMeals()` 根据当前餐食列表计算总摄入热量并同步到 `MockData` [69] [70]）。浮动按钮“+”用于快速添加食物（等同于添加到某餐的操作，但默认选择早餐，可在对话框中切换餐次） [71] [72]。

- **library_screen.dart**：**学习资源库**页面。用于浏览健身相关文章的离线资源。界面顶部有两部分筛选控件：一个横向滚动的分类筛选栏（“全部/训练/饮食/康复”），使用 `ChoiceChip` 列出类别标签，支持点击选择某分类来过滤文章列表 [73] [74]；其下是一个搜索框，用户在此输入关键字可以实时过滤文章标题（通过 `onChanged` 动态更新 `_searchQuery` 状态并触发界面刷新） [75]。主内容分为两块：当未输入搜索词且选择分类为“全部”时，会显示一个“推荐文章”板块，横向滚动列出所有文章的精选卡片 [76] [77]；每个卡片包括文章封面图和标题、分类标签，点击卡片将打开文章详情对话框 [78] [79]。推荐区下方（或在有搜索条件/筛选时直接）显示**文章列表**：使用网格布局（`GridView`）两列排列文章卡片 [80]。普通文章卡片较简洁，包含封面缩略图和标题、分类等简要信息 [81] [82]，点击也会打开详情对话框 [83]。详情对话框中，显示文章的标题、封面大图和正文内容 [84] [85]。目前正文内容使用的是示例文本占位，实际并未从 Markdown 文件加载（说明应用尚未集成Markdown解析，离线文章只是设想) [86] [87]。对话框底部提供“收藏”和“分享”两个按钮，但目前 `onPressed` 无具体实现，只是静

态按钮 [88]。Library 页面没有使用全局监听器，因为文章数据在启动时固定，筛选和搜索通过本地状态 `_selectedCategory` 和 `_searchQuery` 即可完成。通过组合筛选条件，对 `_mockData.articles` 列表使用 `where` 方法过滤出 `filteredArticles` 后渲染 [89] [90]。如果过滤结果为空则显示提示文本 [90]。点击某篇文章卡片打开详情对话框 `_showArticleDialog()`，对话框实现为 `showDialog` 弹出自定义内容，其中还包括一个关闭按钮以及收藏/分享操作（未实现） [91] [92] [88]。

- **lib/widgets/**：公共组件目录，存放可以重复使用的UI组件，提升代码复用性：

- **glass_card.dart**：自定义卡片组件 `GlassCard`。其实质是对 `Card` 小部件的封装，统一应用了内容内边距（16像素） [93]。命名为 “Glass” 可能暗示毛玻璃效果，但目前只是一个普通卡片容器，作用是简化界面中卡片背景+内边距的重复代码。
- **weight_line_chart.dart**：体重折线图组件 `WeightLineChart`，继承自 `StatelessWidget`。使用第三方库 fl_chart 绘制折线图 [94]。构造需要传入最近若干天的体重数据列表 `data`（List<WeightEntry>）。组件在 `build` 中先判断数据是否为空，如为空则显示简短提示文字 [95]。如果有数据，则计算y轴最小/最大值范围用于图表刻度 [96]，并将体重数据转换为 FlSpot 点序列。随后构造 `LineChart` 组件：配置曲线平滑、线条颜色为主题色、节点圆点显示，以及在折线下方填充淡色区域 [97] [98]。图表坐标轴定制为：下方x轴不显示具体值而显示对应日期（仅在整数索引处标注月/日） [99]；左侧y轴显示数值刻度（保留一位小数） [100]；隐藏顶部和右侧的刻度标题 [101]。该组件用于Dashboard和Body页面的体重趋势展示，将 `MockData.weights7d` 传入即可绘制最近7天的体重变化曲线。
- **ring_progress.dart**：环形进度组件 `RingProgress`，用于显示完成率等百分比数据的圆环图。同样使用 fl_chart 库的 `PieChart` 实现 [102]：传入参数 `percent`（0~1的完成率）和 `label`（中心文字标签）。`build` 方法返回一个固定150x150大小的堆叠组件，底层是 `PieChart` 绘制两个扇形（已完成部分使用主题主色，未完成部分灰色），通过调整 `sections` 数据使其形成环形 [103] [104]。上层居中叠放一个列组件，显示白色居中的百分比数值和标签文字 [105] [106]。该组件用于Dashboard和Workout页面，分别展示训练完成百分比等信息。
- **primary_button.dart**：主要按钮组件 `PrimaryButton`。对 `ElevatedButton` 的简单封装，统一应用了主题主色背景、白色文字、圆角矩形边框（24dp圆角）等样式 [107]。接受一个子组件 `child` 作为按钮内容（通常是文字），以及 `onPressed` 回调。该组件用于替代常规的 `ElevatedButton` 以提供一致的主按钮样式，比如在各个对话框的“保存/添加”按钮中都使用了PrimaryButton [108] [49]。

3. 页面逻辑与组件关系说明

3.1 页面导航逻辑

本应用采用了**主页 + 底部导航栏**的框架，将主要功能划分为五个 `Screen` 页面，并通过一个底部导航栏（`BottomNavigationBar`）切换显示。这五个页面（Dashboard、体测、训练、饮食、资源）在应用启动时由 `MainScreen` 创建一个列表保存 [109]，通过索引控制展示 [3]。用户点击底部导航的不同标签，会调用 `onTap` 回调将 `_selectedIndex` 更新为对应索引，从而触发界面刷新显示新的子页面 [110]。由于这几个页面都是通过 `const` 构造创建的，它们会一直保存在内存，切换时无需重新初始化。

除了主导航切换外，应用还定义了一些**命名路由**（`MaterialApp.routes`） [2]，例如在 main.dart 中将字符串路径 `'/workout'` 映射到 `WorkoutScreen`，`'/nutrition'` 映射到 `NutritionScreen` 等。这些路由主要用于从Dashboard页的浮动按钮菜单进行页面跳转。例如，Dashboard页展开的FAB子菜单中，“记录训练”按钮的点击事件通过 `Navigator.pushNamed(context, '/workout')` 打开训练页面 [31]；“记录饮食”按钮类似地跳转到饮食页面 [111]。这种方式实现了从首页快速进入其他功能页面。不过需要注意，由于这些页面本身也可以通过底部导航直接访问，使用路由跳转会导致页面堆栈中出现重复的页面实例（比如通过FAB打开的训练页和通过导航栏切换的训练页不是同一个Widget实例）。当前应用并未对这种导航一致性进行特殊处理，因此建议初学者注意保持导航方式的一致，或者在实际项目中采用 `PageView` 等方式支持手势滑动和导航栏同步，以避免多实例问题。

应用中**没有采用二级子页面**（除了上述路由跳转外），所有新增/编辑操作都没有通过Navigator跳转到新页面，而是使用了对话框（`showDialog`）或底部抽屉（`showModalBottomSheet`）在当前页面弹出表单。这符合项目的设计目标："所有新增/编辑均使用弹框或底部抽屉，尽可能避免二级页面跳转" [112]。比如，添加体重、编辑个人资料、添加训练、添加食物等交互，都是在原页面弹出浮层完成输入，保存后直接刷新当前界面数据。这种设计提升了操作效率，也简化了导航结构，适合离线工具类App的用户体验。

3.2 状态管理与数据通信

本项目采用了**集中式的简单状态管理**模式：即由 `MockData` 单例类充当全局的数据仓库，所有页面共享这一个数据源。`MockData` 通过继承 `ChangeNotifier` 提供通知机制，各界面在初始化时通过 `MockData()` 获取同一实例，并调用 `addListener()` 订阅数据变化 [113]。当用户在某个页面执行操作导致数据修改时（例如添加体重、完成训练、删除食物等），对应的 `MockData` 方法会更新数据并调用 `notifyListeners()` [114] [115]。随后，所有监听了该 `MockData` 实例的页面都会收到通知，触发各自定义的监听回调。在本项目中，各页面的监听回调通常很简单 —— 即调用 `setState()` 重建UI [116]，从而使界面上的数据和全局状态保持同步。

例如：用户在体测页修改了当前体重并点击保存，`BodyScreen` 会调用 `_mockData.addWeight()` 更新体重 [37]。`addWeight` 内部修改了数据并通知监听者 [12]。Dashboard页和Body页本身都监听了 `MockData`，因此它们的 `_updateUI()` 被触发，执行 `setState` 刷新界面。结果是Dashboard首页上的"今日体重"卡片会自动显示更新后的新体重，体测页的BMI值等也同步更新，而无需额外的页面间通信代码。

这种通过全局单例+监听的方案非常直观，适合小型应用：**所有页面通过读写同一份数据来实现通信**。比如在Workout页添加/删除训练，Dashboard页的训练完成度环形图和列表会随之更新，因为两者看的是同一份 `workoutToday` 列表；饮食页增加了摄入热量，Dashboard页的热量盈亏卡片也会变化。需要注意的是，目前 `MockData` 数据仅存在于内存，应用关闭后不会持久保存，也未与任何后台服务交互（完全离线运行）。这意味着"后台逻辑"由 `MockData` 模拟，所有数据重启应用即重置。初学者在实践中可以考虑将数据存储到本地（如使用 SQLite/SharedPreferences）或者远程服务器，以保持状态持久。

3.3 组件之间的协作

UI组件之间主要通过**父子关系和回调**进行协作，没有使用复杂的状态提升或全局事件总线。各页面（Screen）主要承担将全局数据按需传递给子组件和处理用户交互的职责。比如Dashboard页在build时，将 `_mockData.weights7d` 列表传给 `WeightLineChart` 来绘制折线图 [117]，将 `_mockData.workoutCompletionPercent` 传给 `RingProgress` 绘制完成环 [118]。这些子组件本身不维护状态，只根据父组件给的数据渲染。因此数据流向是**单向的**：全局 `MockData` -> 页面 StatefulWidget -> 无状态子组件，确保了数据源的一致。

当用户与界面交互时，通常由页面部件的回调来更新全局数据，再依靠监听机制分发更新。例如： - 用户点击Workout 页某项训练的复选图标，触发 `onPressed` 回调调用 `_mockData.toggleWorkoutCompleted(index)` 切换完成状态 [119]。数据改变后，该页自动重建列表，更新图标的勾选状态；Dashboard页由于监听数据，也自动更新了环形进度百分比。 - 用户在 Nutrition 页的快捷按钮上点击"+100"，触发 `_mockData.updateCalorieIntake(_mockData.calorieIntake + 100)` 增加摄入 [120]。因为Dashboard页也显示热量盈亏，所以Dashboard收到通知后重新计算盈亏并更新颜色显示。 - 用户点击某个自定义组件（如PrimaryButton）执行保存操作时，通常在其 `onPressed` 中由页面逻辑去调用相应的 `_mockData` 方法更新数据，然后关闭对话框。组件本身不直接修改数据，只通过回调通知父层处理。

此外，一些页面内的组件通信通过闭包或参数完成。例如Library页的搜索框，在 `onChanged` 中调用 `setState` 修改了父组件的 `_searchQuery` 状态 [121]，从而让列表组件重建实现过滤；Library页的文章卡片通过 `GestureDetector` 包裹，点击时调用父状态的 `_showArticleDialog(article)` 方法，进而弹出详情对话框 [122]。这些做法体现了Flutter常见的组件通信模式：**子组件通过回调将事件通知父组件处理**，父组件决定状态变化并传递新数据给子组件。

综上，项目整体架构清晰：**MainScreen** 管理导航，**MockData** 管理状态，各 **Screen 页面** 读取和操作数据，**Widgets** 封装可重复UI。页面之间不直接互动，而是通过共享的 `MockData` 间接联系。这种模式对于入门者来说易于理解，实现了功能模块的解耦和数据单一来源。

4. 界面中仅供展示的模块和功能

最后，需要指出本项目中哪些界面元素目前**仅作为静态演示**，未连接实际逻辑或数据（即“前端界面有，但无后台/功能响应”）：

- **Dashboard 首页**：顶部 AppBar 的“设置”按钮（齿轮图标）目前没有实现点击功能，按下不会跳转或打开任何设置界面，仅为预留占位 123 。应用中也未实现任何设置页，因此该按钮是静态的。

- **Workout 训练计划页**：在“训练计划”卡片右上角，有一个“编辑”按钮（带铅笔图标）。当前此按钮的 `onPressed` 回调为空，仅包含一条 `TODO` 注释，表示计划未来添加编辑训练列表的功能 44 。因此点击“编辑”不会有任何实际效果，属于UI占位。

- **Nutrition 饮食记录页**：其中“食物搜索”卡片内的搜索输入框和旁边的二维码扫描图标目前只是界面展示。用户输入搜索关键词并不会过滤下方列表（实际过滤逻辑未实现），扫码图标点击也没有任何响应（未集成扫码功能） 58 。这个模块主要起提示作用，实际功能尚未开发。

- **Library 资源库页**：文章详情对话框中的内容和操作部分存在静态占位情况：

- 文章正文目前使用写死的示例文本显示，并没有真正根据所点文章去加载其 Markdown 文件内容 86 。因此，无论点开哪篇文章，看到的正文都是相同的占位符说明文字。实际的Markdown解析与显示功能未连接。
- 对话框底部的“收藏”和“分享”按钮当前没有任何逻辑，`onPressed` 方法为空实现 124 。点击这些按钮不会改变收藏状态或触发分享操作，只是UI上展示了按钮。项目也没有实现收藏列表等后续功能。

以上模块的存在说明该应用有些功能还处于原型阶段，仅提供界面展示以供演示或测试。对于初学者而言，这些部分可以作为扩展练习：比如实现设置页面、训练计划编辑功能、食物搜索功能、文章内容加载与收藏分享等。如果需要将应用完善，这些占位的按钮和模块将是下一步可以着手的方向。当前版本中，它们不会影响应用主体功能的使用，但用户点击时没有反馈，应在实际应用发布前予以完善。

---

1 2 3 109 110 main.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/main.dart

4 5 6 7 8 theme.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/theme.dart

9 10 11 12 13 14 15 17 18 19 20 21 22 114 115 mock_data.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/mock_data.dart

16 26 workout_entry.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/models/workout_entry.dart

23 article.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/models/article.dart

24 25 weight_entry.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/models/weight_entry.dart

27 28 29 30 31 32 33 34 111 113 116 117 118 123 dashboard_screen.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/screens/dashboard_screen.dart

35 36 37 38 39 40 108 body_screen.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/screens/body_screen.dart

41 42 43 44 45 46 47 48 49 50 51 119 workout_screen.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/screens/workout_screen.dart

52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 120 nutrition_screen.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/screens/nutrition_screen.dart

73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 121 122 124 library_screen.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/screens/library_screen.dart

93 glass_card.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/widgets/glass_card.dart

94 95 96 97 98 99 100 101 weight_line_chart.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/widgets/weight_line_chart.dart

102 103 104 105 106 ring_progress.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/widgets/ring_progress.dart

107 primary_button.dart
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/lib/widgets/primary_button.dart

112 README.md
https://github.com/DR9K69AI79/DMT312_MobileAPP_ASGM1/blob/654d01eae704bf7377857e121ba7428aa6b2b134/README.md