### 

### ****ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY****

### **Mobile Application Design and Development**

### **COURSE CODE:-SEng3206**

### **Mobile App Project Documentation**

**Project Title : FitLife Mobile App**

### **Name Id No**

### **Leta Dejene Ugr/30823/15**

### **Segni Tsega Ugr/31204/15**

### **Betselot Ertumo Ugr/30297/15**

### **Remedan Jemal Ugr/31138/15**

### **Bayisa Balcha Ugr/30232/15**

### **Kenenisa Gemechu Ugr/30774/15**

**submitted to: Yared Tekalegn Negash**

**submission date: Fri, 2 May 2025**

****Abstract****

FitLife is a mobile fitness application developed using Flutter to help users maintain a healthy lifestyle through structured home and gym workout routines. The app features a rich database of over 2000 exercises, each accompanied by detailed instructions and visual GIFs for guidance. Users can browse exercises by body part, muscle group, or equipment, and use a keyword-based search to find specific workouts. During onboarding, the app collects user data such as height, weight, age, and gender to calculate BMI and provide personalized workout recommendations. The app supports offline access to preloaded workout sets and allows users to save their favorite exercises locally using Hive. FitLife is built using modern technologies like Provider for state management, Dio for API integration, SharedPreferences for user data, and Figma for UI design. The app is designed to be responsive across devices and aims to expand in future versions with features such as workout timers, set counters, and personalized workout plans.

### ****Table of Contents****

1. **Introduction**
2. **Key Features**
3. **Technologies Used**
4. **App Architecture**
5. **App Pages and Future Scope**
6. **Use Case Diagram**
7. **System Requirements**
8. **System Architecture**
9. **UI Design**
10. **Technical Implementation and Testing**
11. **Conclusion**

### **Introduction**

FitLife is a cross-platform mobile application developed using Flutter, aimed at helping users achieve their fitness goals through both gym-based and home workouts. Designed with accessibility and personalization in mind, FitLife offers a curated library of over 2000 exercises, each accompanied by detailed step-by-step instructions and animated GIFs for visual assistance. The app enables users to filter exercises by body part, muscle group, or equipment, making it easy to target specific fitness objectives.

Upon first launch, users input personal details such as height, weight, age, and gender, which are used to calculate Body Mass Index (BMI) and offer customized workout recommendations—whether the goal is weight loss, muscle gain, or overall fitness maintenance. These details remain editable through the Settings page, allowing for adaptive recommendations over time.

FitLife supports offline workouts with a selection of preloaded routines, ensuring users can stay active without an internet connection. The app also features a keyword-based search for quick access to specific exercises, a favorite section powered by Hive for storing go-to workouts, and a clean, responsive user interface designed in Figma.

Built using the latest versions of Flutter and Dart, and leveraging tools like Provider (ChangeNotifier), SharedPreferences, Hive, Dio, and the ExerciseDB API, FitLife is both robust and user-friendly. With future plans to include workout timers, set counters, scheduled plans, and custom notifications, FitLife aims to be a comprehensive fitness companion for users at any level.

### ****Key Features****

1. **2000+ Guided Exercises**
   * Extensive library of exercises with step-by-step instructions and animated GIFs for visual guidance.
2. **Smart Categorization**
   * Browse exercises by **body part**, **muscle group**, or **equipment** for targeted training.
3. **Keyword-Based Search**
   * Easily find workouts using a responsive and intuitive search function.
4. **Personalized Recommendations**
   * Collects user data (height, weight, age, gender) during onboarding to calculate BMI and suggest relevant workouts (e.g., weight loss, muscle gain).
5. **Offline Workout Access**
   * Includes a preloaded set of offline workouts that can be used without an internet connection.
6. **Favorites Section**
   * Users can tap a heart icon to save preferred exercises, stored locally using Hive for quick access.
7. **Editable User Profile**
   * User data can be modified later through the Settings page, ensuring recommendations remain up-to-date.
8. **Modern UI/UX Design**
   * Clean, responsive interface designed in Figma, optimized for both Android and iOS devices.
9. **Latest Flutter Technologies**
   * Built using Provider (ChangeNotifier), Dio, Hive, and SharedPreferences for a robust and smooth experience.
10. **Future Enhancements (Planned)**
    * Integration of workout timers, set counters, custom notifications, and personalized workout plans

### ****Technologies Used****

#### **Development Framework**

* **Flutter**: Cross-platform framework used to build the app for both Android and iOS using a single codebase.
* **Dart**: The programming language used in Flutter for writing application logic.

#### **State Management**

* **Provider** (ChangeNotifier): Manages app state and allows efficient data flow between widgets.

#### **API and Networking**

* **Dio**: A powerful HTTP client for handling API requests and network communication (used to fetch data from ExerciseDB API).
* **ExerciseDB API**: Provides a large database of exercises with names, steps, target muscles, and GIFs.

#### **Local Storage**

* **Hive**: Lightweight and fast NoSQL database used to store user favorites and other structured data.
* **SharedPreferences**: Used for storing simple key-value pairs like user height, weight, age, and gender.

#### **UI/UX Design**

* **Figma**: Used for prototyping and designing the user interface before implementation.

#### **Responsive Design**

* Ensures the app adjusts smoothly across various screen sizes and device orientations.

#### **Offline Support**

* Preloaded workout routines are stored locally within the app to enable offline access.

### ****App Architecture****

FitLife uses a layered architecture based on Flutter’s widget tree and the Provider package for state management. It separates the user interface, business logic, and data access layers to ensure scalability, maintainability, and a responsive user experience.

#### **1. Presentation Layer (UI)**

* Developed using **Flutter widgets**.
* Responsive layout compatible with various screen sizes.
* Uses Figma-designed components.
* Main Pages:
  + **Home** – Featured workouts and suggestions.
  + **Discovery** – Browse by body part, muscle, or equipment.
  + **Favorite** – View saved workouts (stored via Hive).
  + **Settings** – Update personal info for BMI and suggestions.

#### **2. Business Logic Layer**

* Built with **Provider (ChangeNotifier)**.
* Handles:
  + API interactions
  + State updates
  + User input (height, weight, age, etc.)
  + Search and filtering
  + Favorites management
  + BMI calculation and workout recommendations

#### **3. Data Layer**

* **Remote Data**: Exercises are fetched on demand using **Dio** from **ExerciseDB API**.
* **Local Data**:
  + **Hive** for storing favorites and structured user data.
  + **SharedPreferences** for lightweight user settings (name, age, height, weight, gender).

#### **4. Offline Access Layer**

* A curated set of workouts is stored locally as **JSON assets**.
* These allow access to exercises without requiring an internet connection.

#### **5. Utilities**

* Helper functions for:
  + Parsing and filtering exercise data
  + Body Mass Ratio calculation
  + JSON handling
  + Search keyword matching

### **Architecture Overview**

+-------------------------+

| Presentation Layer | <-- Flutter Widgets (UI)

+-------------------------+

| Business Logic Layer | <-- Provider / ChangeNotifier

+-------------------------+

| Data Layer |

| - Dio (API fetch) |

| - Hive (Favorites) |

| - SharedPreferences |

+-------------------------+

| Offline Assets Layer | <-- Preloaded workout JSONs

+-------------------------+

| Utility Layer | <-- Helpers: BMI, Filters, JSON

+---------------------------------------------+

**App Pages and Future Scope**

### **Home Page:**

* **Offline Workouts**: Provide a list of exercises that can be accessed without an internet connection.
* **Popular Exercises**: A section that showcases trending or highly rated exercises from the Discovery page, allowing users to easily jump to them.

### **Discovery Page:**

* **Quick Search**: A search bar for users to quickly find exercises by name or keyword.
* **Category-Based Navigation**: Options to filter exercises based on body part, muscle type, or equipment, giving users a more structured way to explore.

### **Favorite Page:**

* **Simple List**: A straightforward list of all exercises the user has saved as favorites for easy access later.

### **Settings Page:**

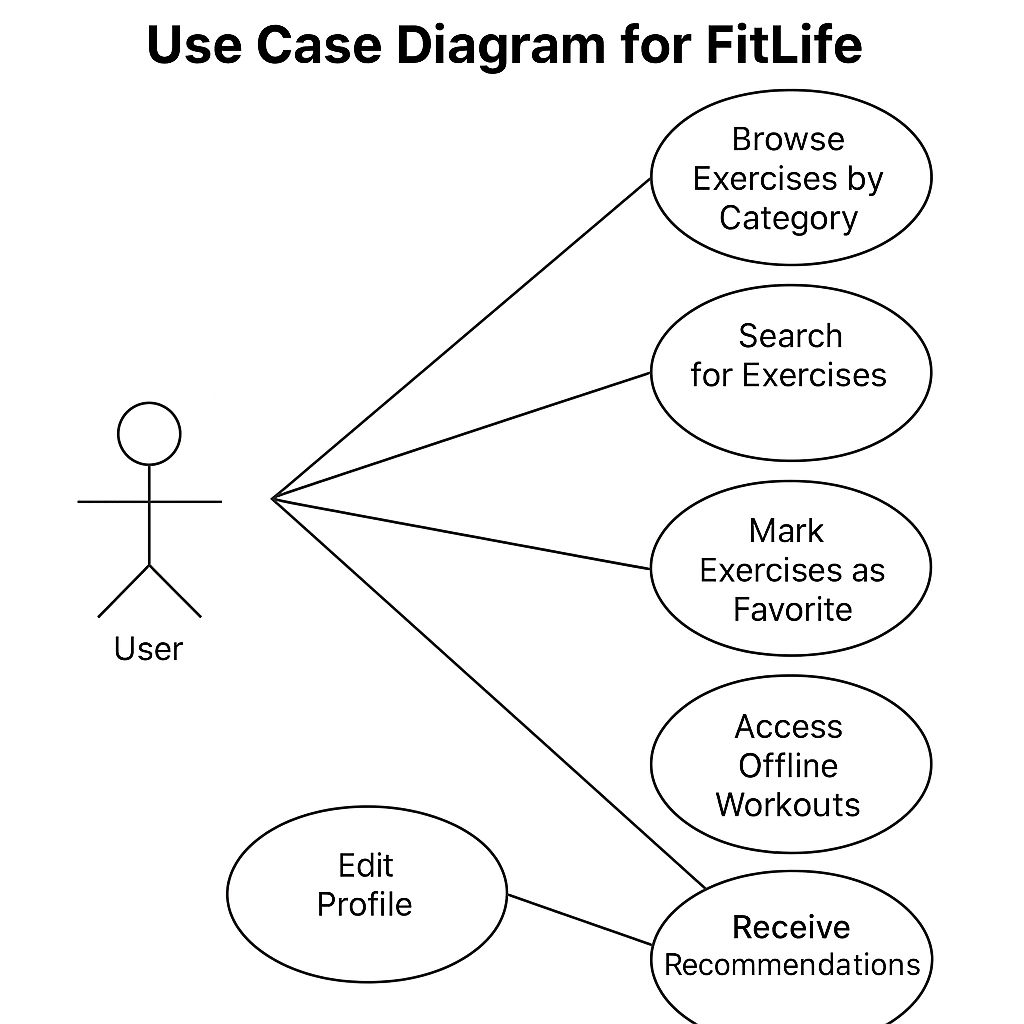
* No specific details yet, but this could include workout preferences and notification settings in the future.

### **Future Features:**

1. **Timers for Each Exercise**: Each exercise can have a timer for rest periods, set counts, or specific duration.
2. **User-Defined Workout Plans**: Users can create their personalized workout plans, alongside any predefined plans.
3. **Custom Notifications**: Reminders for upcoming workouts, or specific exercise reminders within the workout plan.

****Use Case Diagram****

**A simple use case diagram to show how the app is used by user visually**



### ****System Requirements****

#### **Supported Platforms**

* **Mobile Operating Systems**: The app will be available on both **Android** and **iOS** platforms.
* **Target Versions**:
  + **Android**: Android 6 and above
  + **iOS**: iOS 8 and above

#### **Hardware Requirements**

* **Device Compatibility**: The app supports a wide range of mobile devices, ensuring compatibility with various screen sizes and hardware capabilities.
* **Offline Functionality**: The app includes pre-loaded offline workouts that do not require an internet connection to access.

#### **Network Requirements**

* **Network Connectivity**:
  + The app performs optimally with **good network connectivity** but is designed to function well with **low bandwidth** as well.
  + Minimal data usage is required when accessing online content, but offline features ensure users can continue without an active internet connection.

#### **Backend Infrastructure**

* **Cloud Services**: The app does not rely on any cloud storage.
* **Authentication System**: There is no user authentication system in place; users can access the app as guests, ensuring a seamless experience.

#### **User Load and Scalability**

* **Simultaneous Users**: The app operates independently on the device, without the need for cloud services. Therefore, scalability concerns related to simultaneous users are minimal.

#### **Performance Expectations**

* **App Speed**:
  + The app is optimized for fast load times, particularly for offline content.
  + Navigating between pages and performing searches should be swift, even on devices with lower specifications.
* **Battery Usage**:
  + The app is designed with battery efficiency in mind. Background processes like timers and notifications will be optimized to minimize battery drain.

### ****System Architecture****

The **FitLife** app follows a client-centric architecture with a focus on local storage, offline functionality, and minimal reliance on external services. Below is an overview of the app’s architecture components:

#### **1. Client-Side (Mobile Application)**

* **Platforms**: The app is built for both **Android** and **iOS**, using **Flutter** to ensure cross-platform compatibility.
* **UI Layer**:
  + Built using **Flutter** widgets and **Figma** for design.
  + Consists of the following pages:
    - **Home Page**: Displays offline workouts and popular exercises.
    - **Discovery Page**: Offers both quick search and category-based search for exercises.
    - **Favorite Page**: Displays a list of saved exercises.
    - **Settings Page**: Allows customization of user preferences (notifications, workout plans, etc.).

#### **2. Data Storage**

* **Local Storage**:
  + **Hive**: Used for storing user data such as **favorites** (saved exercises). Hive is an efficient, lightweight key-value database that provides quick access to stored data.
  + **SharedPreferences**: Used for small-scale, simple storage such as user preferences and settings.

#### **3. Data Flow & Interactions**

* **Offline Workouts**:
  + All exercises are pre-loaded and stored on the device, enabling users to access them even without an internet connection.
* **ExerciseDB API**:
  + The app integrates with the **ExerciseDB API** to fetch exercises when the device is online, supplementing the offline content with new data or updates as needed.
* **Notification System**:
  + **Custom Notifications**: The app utilizes notifications to remind users of upcoming workouts, based on their workout plans or preferences. Notifications are triggered using Flutter’s **Local Notifications** package.

#### **4. User Interaction and Workflow**

* **User Onboarding**:
  + During the first launch, users input basic details such as height, weight, age, and gender, which are stored locally and used for BMI calculation and exercise recommendations.
* **Exercise Recommendations**:
  + Based on the user data (height, weight, etc.), exercise recommendations are dynamically filtered and suggested. These exercises can be accessed through the **Discovery** or **Home** pages.
* **Custom Workout Plans**:
  + Users can define their own workout plans by selecting exercises from the library. These plans are saved locally using **Hive**.
* **Timers & Set Counters**:
  + Each exercise comes with a timer (for rest periods and exercise duration). The timer is set for individual exercises and is triggered as the user goes through their workout.

#### **5. System Flow**

1. **Initial Launch**: The app loads pre-loaded exercises from local storage (Hive) for offline use.
2. **Discovery**: When the user searches for exercises via the quick search or category filter, the app will pull data from **ExerciseDB API** (when online).
3. **Exercise Playback**: Timers for each exercise are triggered as the user progresses through their workout, with notifications being sent for upcoming exercises.
4. **Favorite Management**: Users can save exercises to their favorites list. These are stored locally in **Hive** for fast access.
5. **Workout Plans**: Users create custom workout plans. These plans are stored locally and can be modified at any time.

#### **6. Error Handling & Offline Support**

* **Error Handling**:
  + When the app encounters connectivity issues, it will gracefully fall back on offline data (pre-loaded workouts).
  + **Local Error Handling**: For any internal errors, such as invalid user input or data retrieval failures, appropriate error messages are displayed.
* **Offline Functionality**:
  + The app prioritizes offline usability by storing all core exercise content on the device, ensuring that users can continue their workouts even without network connectivity.

### ****UI Design****

The **FitLife** app is designed to provide a seamless, intuitive, and visually appealing experience for users. The app follows a modern and minimalistic design, ensuring that all key features are easily accessible and the overall layout is clean and simple. Below is an overview of the core design elements and UI components.

#### **1. Design Philosophy**

* **Minimalistic and Clean**: The design focuses on simplicity, ensuring that users can easily navigate through the app without unnecessary distractions.
* **Consistency**: Consistent use of colors, icons, and typography ensures a unified experience across all pages of the app.
* **Accessibility**: The app ensures that text is readable and UI elements are easily accessible for a wide range of users.

#### **2. UI Components**

The app contains several key UI components, each designed to provide a smooth user experience.

##### **Home Page**

* **Offline Workouts**: Displayed as cards with workout titles, short descriptions, and thumbnail images. The cards use green for key text and icons to emphasize important information.
* **Popular Exercises**: A scrollable section that displays trending exercises, which can be tapped to view more details.
* **Navigation**: A **Bottom Navigation Bar** with rounded corners, using gradient backgrounds. The bar includes icons for navigating to the **Discovery**, **Favorite**, and **Settings** pages.

##### **Discovery Page**

* **Search Bar**: Positioned at the top, with a mix of green and black text for contrast. Users can search exercises by name or category.
* **Category Filters**: Displayed as clickable buttons or tags in a combination of green and black.
* **Exercise Thumbnails**: Exercises are shown in a grid with icons and thumbnails, ensuring easy browsing.

##### **Favorite Page**

* **Favorite List**: A vertically scrollable list of saved exercises, each displayed with an image, title, and an option to remove from favorites.
* **Empty State**: When no favorites are saved, a message is displayed encouraging users to add exercises to their favorites list.

##### **Settings Page**

* **User Preferences**: Toggles and sliders are used for adjusting settings like notification preferences, workout plans, and more. These elements are designed with rounded corners and a mix of green and black for buttons and text.
* **Notifications Settings**: Users can configure when and how often they receive reminders or workout notifications.

#### **3. Color Scheme**

* **Primary Colors**:
  + **Green**: Used for buttons, some text, and key elements to grab attention

e.g: "Start Workout" buttons, active icons).

* + **Black**: Used for background elements, text, and boxes to provide contrast against the green.
* **Gradient Background**: The **navigation bar** and main **four pages** have a gradient background that enhances the visual appeal, with a smooth transition from one color to another.
* **Fit Man Background**: A static fit man image is used as the background on the **start page** and **user data-taking page**, creating an engaging atmosphere for users at the beginning of their fitness journey.

#### **4. Typography**

* **Primary Font**: A modern sans-serif font Roboto is used for easy readability.
* **Heading Fonts**: Larger, bold fonts are used for titles and headings, helping to create a clear visual hierarchy.
* **Body Text**: Simple and legible font weight for descriptions and labels.

#### **5. Iconography**

* **Custom Icons**: The app uses **custom icons** designed for exercise categories (e.g., cardio, strength training) to visually represent different types of exercises. These icons are consistent with the green and black color scheme.
* **Navigation Icons**: The navigation icons in the **Bottom Navigation Bar** are designed with a minimalist approach, using simple shapes and consistent styling.

#### **6. Interaction and Transitions**

* **Button Feedback**: Buttons, such as the green **Start Workout** button, change color slightly when pressed, providing feedback to the user that their action has been acknowledged.
* **Rounded Corners**: Buttons, cards, and UI elements feature **rounded corners**, creating a modern and beautiful interfaces.

#### **7. Wireframes & Mockups**

* **Home Page**: A scrollable list of offline workouts and popular exercises, with easily tappable cards and a bottom navigation bar for seamless transitions.
* **Discovery Page**: A clean layout with a search bar at the top, followed by category-based filtering options and exercise thumbnails in a grid format.
* **Favorite Page**: A simple list of favorite exercises with the option to remove them, displayed in a scrollable format.
* **Settings Page**: A settings page with toggles, sliders, and a clear structure for managing user preferences.

#### **8. Responsiveness**

* **Small Screens (Phones)**: The layout adapts to smaller screens, ensuring that buttons and text are readable and properly sized.
* **Large Screens (Tablets)**: The app utilizes the larger screen space by distributing content into multiple columns and optimizing the layout for more comfortable viewing.

### ****Technical Implementation and Testing****

#### **1. Technical Stack**

* **Programming Languages and Frameworks**:
  + The app is developed using **Dart** with the **Flutter** framework, ensuring cross-platform compatibility for both **Android** and **iOS**.
* **State Management**:
  + The app uses **Provider** with Change-Notifier for efficient state management, ensuring smooth data flow and reactivity across the app.
  + **Future-Builder** is employed to asynchronously load and display API data in the UI, ensuring responsive UI updates as the data is fetched.
* **API Integration**:
  + The app integrates with the **ExerciseDB API** for exercise data, and this data is displayed to users through various UI components.
* **Database and Offline Support**:
  + **Offline workouts** are bundled within the app and stored **locally** for seamless user experience even without network connectivity.
* **Local Notifications**:
  + The app will use **Flutter local notifications** to remind users of upcoming workouts or exercises.

#### **2. Core Functionalities**

* **Offline Workouts**:
  + Offline workouts are bundled with the app, ensuring users can access workout data even without internet connectivity. This feature is critical for users in low or no-network environments.
* **Exercise Search**:
  + The search functionality allows users to search exercises based on categories or keywords. The data is dynamically loaded and displayed using **FutureBuilder** to keep the UI responsive.
* **Notifications (Future)**:
  + Custom **local notifications** will be implemented to remind users of their upcoming workouts. These notifications can be configured based on user preferences.
* **Workout Plans (Future)**:
  + Users can define custom workout plans, and this data is stored locally on their device. This feature gives users full control over their fitness journey.
* **Timers (Future)**:
  + **Timers** for each exercise will be implemented to ensure users can time their workouts efficiently. These timers run in the background while the app is active.

#### **3. Testing**

* **Unit Testing**:
  + The app includes **unit tests** for key business logic components, ensuring that data handling and calculations such as exercise filtering or workout timing are correct.
* **Widget Testing**:
  + **Widget testing** is implemented to test individual UI elements, such as buttons, text fields, and navigation elements, ensuring they function as expected.
* **Integration Testing**:
  + **Integration tests** ensure that multiple components of the app work together seamlessly

example: adding an exercise to the favorites list or syncing workout data.

* **Performance Testing**:
  + The app’s performance has been tested using **Chrome debugging tools** to ensure smooth operation, particularly when handling data from the ExerciseDB API. The app performs well under various conditions.
* **Error Handling**:
  + Errors such as API call failures or network issues are gracefully handled. In case of an error, the app displays a **GIF with an error message** to inform users about the issue.
* **Cross-Platform Testing**:
  + The app has been thoroughly tested on **Android** devices, ensuring compatibility and consistent behavior across different devices and screen sizes.

#### **4. Version Control and Collaboration**

* **Version Control**:
  + **GitHub** is used for version control, and the team works collaboratively on the codebase. Changes are tracked using **Git**, allowing for efficient code management and collaboration.
* **CI/CD**:
  + The app’s development workflow follows best practices, with regular code reviews, pull requests, and updates to ensure smooth releases.

### ****Conclusion****

The **FitLife** app offers a comprehensive solution for fitness enthusiasts, providing access to over 2000 exercises, customizable workout plans, and offline capabilities. Developed using **Flutter** and **Dart**, the app ensures a seamless user experience across both **Android** and **iOS** platforms. With a strong focus on performance, usability, and flexibility, the app leverages **Provider** for efficient state management and **FutureBuilder** for asynchronous data loading.

The app is designed with a clean, minimalist UI, utilizing green and black tones, rounded corners, and a gradient background for an appealing aesthetic. It supports offline functionality, enabling users to access workouts even without network connectivity, and sends timely **local notifications** to remind users of their workouts.

Through thorough testing—ranging from **unit tests** and **widget testing** to **integration tests** and **performance monitoring**—the app ensures high reliability, responsiveness, and smooth operation. The use of **GitHub** for version control ensures collaborative development and efficient code management.

Looking forward, the **FitLife** app is positioned for future enhancements, including potential third-party integrations and scalability improvements to support additional features like cloud synchronization or user accounts. It is a robust solution for users aiming to maintain a healthy lifestyle, whether at home or in the gym.