Design Document

Asja Attanasio

Davide Barzan

Samuele Pozzani

Academy Year 2024/2025

Deliverable:	DD
Title:	Design Document
Authors:	Asja Attanasio, Davide Barzan, Samuele Pozzani
Version:	1.0
Date:	05-July-2025
Download page:	`https://github.com/sampozz/GymMe`
Copyright:	Copyright © 2025, Asja Attanasio, Davide Barzan, Samuele Pozzani – All rights reserved

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Purpose

The primary aim of this document is to aid the development team in bringing the envisioned system to life. It offers a comprehensive overview of the chosen system architecture and extensively details the various components, elucidating their interconnections. Additionally, the Design Document (DD) outlines implementation, integration, and testing strategies. These plans are formulated considering component priority, required effort, and their impact on stakeholders.

1.2 Scope

This application is designed to facilitate the management of fitness center reservations, providing users with a seamless experience to search, book, and manage gym sessions directly from their mobile devices or web browsers. The app caters to two distinct user types: regular users (customers) and administrators.

1.2.1 Core Features and Functionality

The core functionality of the app includes:

- **User Interaction:** Users can search for gyms in their vicinity, view available training slots, and make reservations for various activities such as gym workouts, group classes, or personal training sessions. The app also allows users to manage their bookings, access notifications, and store preferences like favorite gyms.
- **Admin Features:** Administrators can manage the app's content and functionalities by adding or modifying gym information, managing users' subscriptions and medical certificates, and viewing user bookings. Admins have the ability to monitor and control system operations for a smooth user experience.
- **Real-time Booking and Notifications:** The app integrates real-time data updates, allowing users to receive notifications about their bookings, gym events, and other personalized information. Firebase is used for real-time synchronization, user authentication, and push notifications.
- **Multi-platform Support:** The app is available for Android devices and web browsers, ensuring a broad reach across various platforms. This cross-platform approach allows users to manage their fitness bookings regardless of the device they use.

1.2.2 Target Audience

The target audience for this app includes fitness enthusiasts, gym members, and fitness centers' staff. Regular users can be anyone looking for a gym to train, while administrators will typically be gym managers or owners who wish to manage gym operations digitally.

1.3 Definitions, Acronyms

1.3.1 Definitions

The following are the key terms used throughout the application, along with their definitions:

User A person who uses the application to search for gyms, book fitness activities, and manage their personal profile, including viewing and modifying bookings, preferences, and notifications.

Admin A user with elevated permissions who manages the application's content and user data. Admins can add or modify gym information and slots, manage user subscriptions and certifications.

Gym A fitness center or gym facility that users can book sessions in. A gym has relevant information such as its name, address, available activities, and opening hours. It can also have different types of activities (e.g., weight room, courses, personal training).

Activity A specific fitness session or class available at a gym. Activities can include, for example, weight room workouts, group classes (e.g., yoga, spinning), or personal training sessions. Personal training sessions are paid activities, while all other sessions are included in the user's subscription. Each activity has a set time, price (only for personal training), and availability.

Slot A specific time slot during which an activity is scheduled to take place. A slot has a defined start time, end time, and a number of available spots, indicating the number of participants that can book a session. The availability of these spots determines whether users can reserve a place for the activity.

Booking A reservation made by a user for a specific activity at a gym during a defined slot. A booking includes details such as the gym, activity, date and time.

1.3.2 Acronyms

API Application Programming Interface

UI User Interface

UX User Experience

REST Representational State Transfer

BaaS Backend as a Service

APK Android Package Kit

MVVM Model-View-ViewModel

SDK Software Development Kit

GDPR General Data Protection Regulation

CCPA California Consumer Privacy Act

CDN Content Delivery Network

1.4 Reference Documents

The following documents were referenced during the design and development of the application:

- **Firebase Documentation**: Provides the official guide for Firebase Authentication and Firestore database services used in the app.
  Available at: `https://firebase.google.com/docs`
- **Google Material Design Guidelines**: The official user interface guidelines for Android apps.
  Available at: `https://material.io/design`
- **General Data Protection Regulation (GDPR)**: Legal requirements for user data protection and privacy.
  Available at: `https://gdpr.eu/`
- **Google Maps API Documentation**: Official documentation for integrating Google Maps and location-based services in the app.
  Available at: `https://developers.google.com/maps/documentation`
- **Google Calendar API Documentation**: Provides the official documentation for integrating Google Calendar services into the app, including adding, viewing, and modifying events.
  Available at: `https://developers.google.com/calendar`
- **Google Identity Services Documentation**: Provides the official guide for integrating Google Sign-In and handling OAuth 2.0 based authentication within web and mobile applications.
  Available at: `https://developers.google.com/identity`
- **Stripe API Documentation**: Official documentation for integrating payments services in the app.
  Available at: `https://docs.stripe.com/api`
- **Imgur API Documentation**: Provides the official guide for uploading, managing, and retrieving images via the Imgur image hosting service API.
  Available at: `https://apidocs.imgur.com/`

1.5 Document Structure Overview

This Design Document is organized into five main chapters, each addressing a key aspect of the application's development, from the initial conceptualization to the final testing and deployment plan. The structure is as follows:

1. **Introduction**
   This chapter provides an overview of the application, including its objectives and target users. It also defines key terms and acronyms used throughout the document and lists reference documents that guided the development of the application.

2. **Architectural Design**
   The second chapter provides a comprehensive overview of the system architecture, detailing the major components of the application and their interactions. The chapter begins with high-level and distributed views, progressing to component, deployment, and runtime perspectives. It also includes descriptions of component interfaces, selected architectural styles, and patterns, such as authentication, along with key design decisions related to availability and the notification service.

3. **UI Design**
   This chapter presents the design of the user interface, detailing the layout, user flows, and visual aesthetics, with support of mockups of key screens. This section aims to provide a clear understanding of how users will interact with the application.

4. **Requirements**
   The fourth chapter focuses on the functional, non-functional and system requirements of the application. It also outlies on the system requirements, distinguishing between the mobile application and the web application. This section ensures that the application meets all necessary criteria to fulfill its intended purpose.

5. **Implementation and Testing Plan**
   This chapter outlines the implementation, integration, and testing plan for the system. It emphasizes the importance of identifying and fixing errors during development to ensure the final product's quality. It includes details on work distribution, component integration via GitHub, and the unit tests implemented to validate the system's functionality.

Each chapter builds on the previous one, offering a detailed look at the application's design, development, and testing approach. This document acts as a roadmap for developers, designers, and stakeholders, helping keep everyone aligned with the project's objectives and requirements throughout its lifecycle.

Chapter 2

Architectural Design

2.1 Overview

This section provides an overview of the most relevant architectural components and interfaces, how these elements interact, and a comprehensive description of how properties such as scalability and redundancy for reliability are managed and reached.

2.1.1 High Level View

The application architecture is based on a two-layer approach that leverages the Flutter framework for the frontend and Firebase services as Backend-as-a-Service (BaaS). This choice simplifies development and maintenance by eliminating the need for dedicated server infrastructure while ensuring scalability and reliability through Google Cloud's infrastructure.

The application is accessible on Android devices via direct installation (APK) and on Web browsers through a WebApp. The primary users are gym customers and gym managers.

The Frontend (Flutter) is responsible of managing the user interface and presentation logic, providing a consistent experience on Android devices and web browsers. It implements the MVVM (Model-View-ViewModel) architectural pattern to separate business logic from the user interface.

Whereas, the Backend consists of two main services:

- Firebase Authentication: Manages the authentication process via email/password and Google Sign-In, automatically distinguishing between customer and manager roles.

- Firestore Database: Stores and manages data related to gyms, classes, reservations, and users, providing a scalable, real-time infrastructure to synchronize information across devices.

The application also integrates external APIs, including Google Maps Platform for displaying gym locations and searching addresses, as well as the Calendar API for event synchronization.

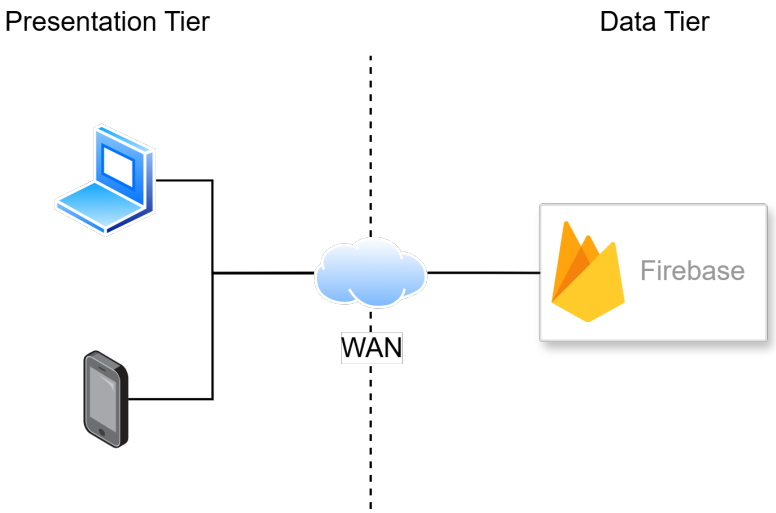


Figure 2.1: High-level architecture of the system

Communication between the frontend and backend occurs via the Firebase SDK APIs provided by Google, which handle secure connections and data synchronization automatically. Integration with the Google Maps Platform and Google Calendar is achieved through REST APIs and Flutter-specific widgets for map management Figure 2.1.
The system is designed to support automatic scalability and fault tolerance, thanks to the native characteristics of Firebase services, which distribute data globally to ensure high availability.

2.1.2 Distributed View

The distributed architecture of the application relies on Firebase's cloud-based services to handle data storage, user authentication, and communication between devices. This approach enables the system to scale effortlessly while maintaining reliability and performance.
Key characteristics of the distributed architecture include:

- Scalability: Firebase dynamically scales its backend resources to accommodate varying levels of user activity without requiring manual intervention. This allows the application to handle an increasing number of gym customers and managers without performance degradation.

- Fault Tolerance: Data gets replicated across multiple geographic regions, ensuring that user data remains accessible even in the event of hardware failures or regional outages. Firebase's infrastructure automatically manages failover processes to maintain service continuity.

- Load Balancing: User requests are distributed across Firebase's global infrastructure, which reduces latency and optimizes performance. This is particularly beneficial when handling high volumes of simultaneous requests for gym bookings and data retrieval.

The Android application is distributed via APK files, allowing direct installation on users' devices. The web version is hosted on a Raspberry Pi mini-PC. This local hosting solution allows users to access the web application through their browsers while maintaining control over deployment and updates.

The application benefits from Firebase's managed infrastructure, eliminating the need for custom backend servers. Data consistency and synchronization are handled by Firestore's real-time database capabilities, allowing users to view and update gym-related information in real time across multiple devices.

2.2 Frontend Architecture

The frontend of the application is developed using Flutter and follows a modular architecture that separates concerns across different components to improve maintainability, testability, and scalability. A high level view is shown in Figure 2.2.

2.2.1 Widgets

Widgets are responsible solely for rendering the user interface. They are stateless or stateful depending on the interaction requirements but do not directly manage application data or business logic. Widgets react to changes in the application state provided by external sources (e.g., providers) and update the UI accordingly.

2.2.2 Providers

Providers are used for state management and application data handling. They expose the current state of the app and notify listeners when changes occur. This ensures a reactive architecture where UI components automatically reflect the current application state. Providers encapsulate business logic and coordinate interactions between widgets, services, and models.

2.2.3 Services

Services are responsible for performing CRUD (Create, Read, Update, Delete) operations against Firebase Firestore and other Firebase modules. They act as the communication layer between the application and the backend, abstracting away the implementation details of Firebase APIs. Services are typically injected into providers to decouple data access from UI rendering and business logic.

2.2.4 Models

Models define the data structures used throughout the app and represent entities such as users, gyms, activities, and bookings. These models serve as Data Transfer Objects (DTOs), enabling type-safe communication between different layers of the app. They also include methods for serializing and deserializing data to and from Firestore-compatible formats (e.g., `fromFirestore`, `toFirestore`).

Summary

This layered approach promotes separation of concerns:

- **Widgets** handle UI rendering.
- **Providers** manage app state and business logic.
- **Services** handle communication with Firebase.
- **Models** define and transfer structured application data.

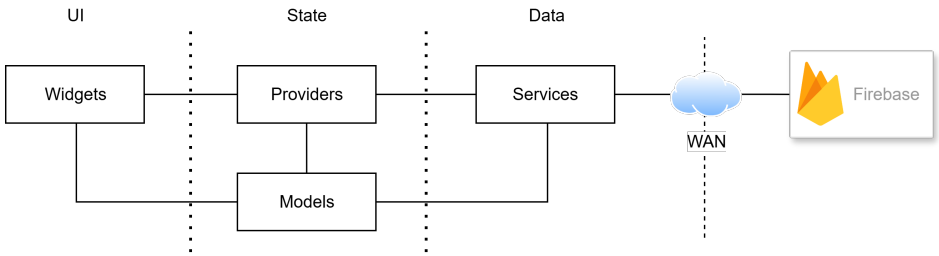


Figure 2.2: High-level architecture of the frontend

This structure enhances code reusability, simplifies testing, and aligns with Flutter's best practices for scalable app development.

2.3 Component View

The component view offers a visual breakdown of a software system, showcasing its building blocks (components), their relationships, interfaces, and how they interact to achieve the system's functionality. It provides an overview of the system's architecture, aiding communication and guiding development by illustrating the structure and interactions among different components.

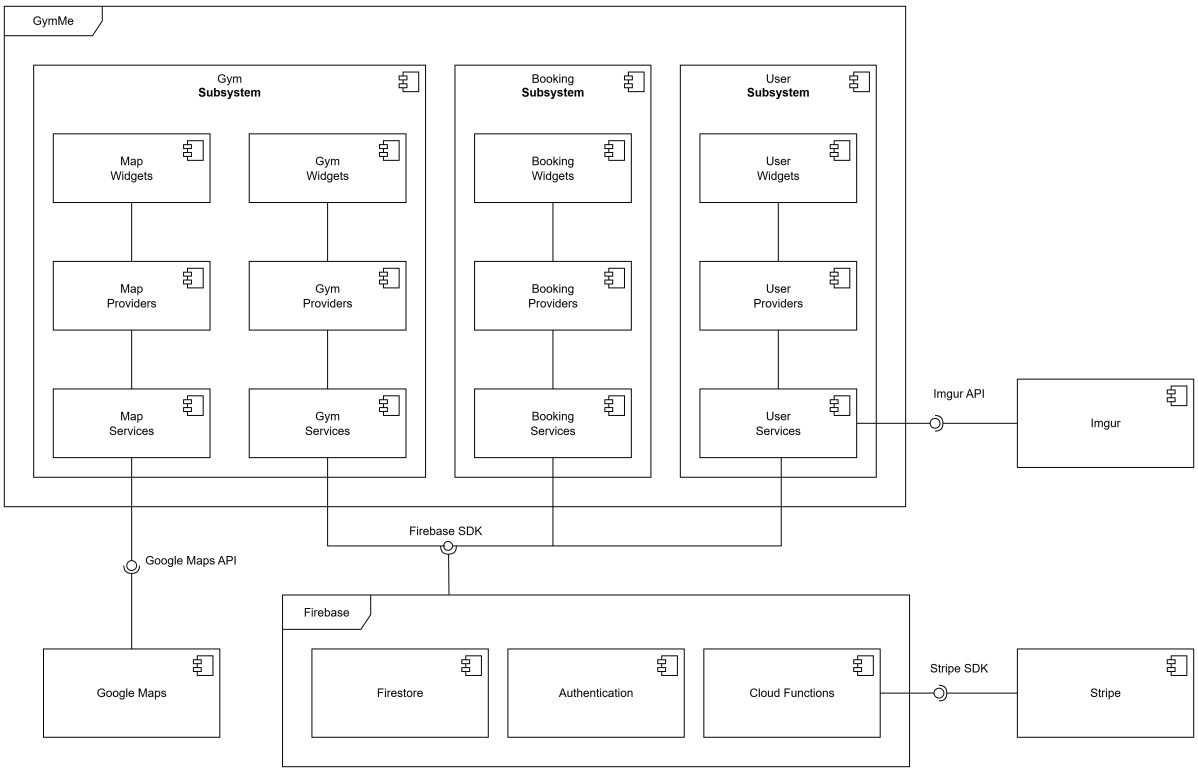


Figure 2.3: Component breakdown of the application

Main Components

1. *Gym Exploration Module*

- Displays gyms in a list or map format using Google Maps API.
- Allows users to search for gyms and filter results.
- Includes a section for favorite gyms selected by the user.

2. *Booking Module*

   - Handles reservations for gym activities, including courses, personal training sessions, and weight room slots.
   - Ensures real-time synchronization with Firestore.
   - Integrates with external calendar APIs to add bookings as calendar events if the user has enabled synchronization.

3. *User Authentication Module*

   - Manages login and registration via Firebase Authentication (email/password and Google Sign-In).
   - Differentiates between customers and managers based on roles stored in Firebase.

4. *Gym Management Module (for managers)*

   - Allows gym managers to create, update, and delete gyms, classes, and schedules.
   - Displays enrolled users for each session.

5. *Notification Module*

   - Supports push notifications via Firebase Cloud Messaging to remind users of upcoming reservations.
   - Provides an in-app notification center to view past and upcoming notifications.

6. *Search Module*

   - Enables users to search for gyms, activities, and trainers.
   - Uses Firestore queries and, where applicable, Google Maps location-based search.

7. *State Management*

   - Uses 'provider' for managing application state and ViewModel interactions.
   - Ensures efficient data handling and UI updates.

8. *Local Storage & Caching*

   - Implements caching for frequently accessed data (e.g., favorite gyms, recent searches) to improve performance and offline usability.

Each of these components interacts primarily with Firebase Firestore for data retrieval and persistence, Firebase Authentication for access control, and external APIs (Google Maps Platform, Calendar API) where necessary.

2.4 Deployment View

The deployment view illustrates the various components involved in the execution of the application, including the devices and services that interact with it. The application is designed to run on both mobile devices and web browsers, with Firebase providing backend services.

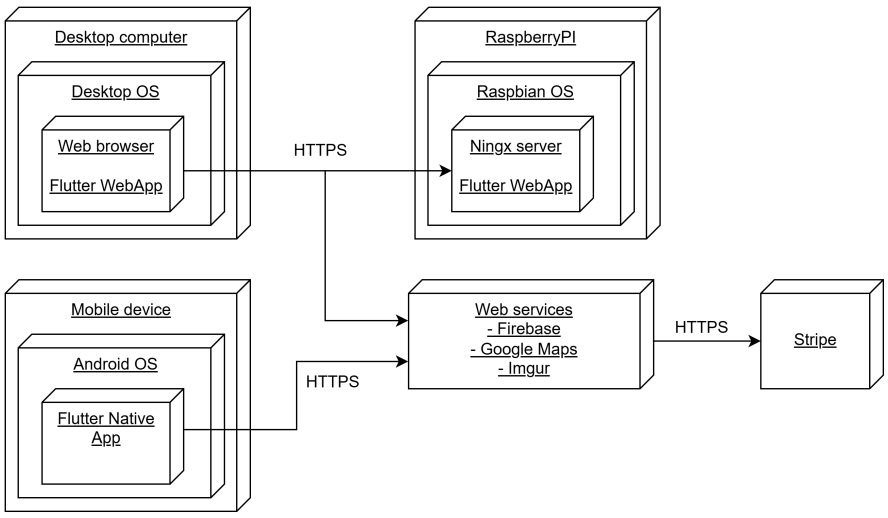


Figure 2.4: Deployment View

- *Mobile Application (Flutter)*
  - Runs on Android smartphones and tablets.
  - Communicates with Firebase Firestore for data storage and Firebase Authentication for user authentication.
  - Uses Google Maps API for location-based functionalities.
  - Optionally integrates with the device's calendar through an external API.
- *Web Application (Flutter Web)*
  - Runs on web browsers and is hosted on a Raspberry Pi for internal use.
  - Provides similar functionality to the mobile application.
  - Uses Firebase Firestore, Firebase Authentication, Google Maps API, and Calendar API.
- *Firebase Services*
  - Firestore: Cloud database storing gyms, users, bookings, and schedules.
  - Authentication: Handles user sign-in (email/password, Google Sign-In) and role management.
  - Cloud Messaging: Sends push notifications for booking reminders.
  - Hosting: Provides storage and content delivery for the web application if needed.
- *External APIs*
  - Google Maps Platform: Displays gym locations and provides location-based search.
  - Calendar API: Allows users to synchronize bookings with their personal calendars.

- *User Devices*
  - Smartphones (Android) for mobile app usage.
  - PCs and laptops running web browsers for web app access.

2.5 Runtime View

The runtime view describes how the system behaves during execution by illustrating key use cases and their interactions with different components. The following scenarios highlight essential functionalities of the application:

User Login

- The user opens the app and logs in using email/password or Google Sign-In.
- Firebase Authentication verifies credentials and retrieves the user role (customer or manager).
- The app loads user-specific data from Firestore and displays the appropriate interface.

Exploring Gyms

- The user views the list of gyms or explores them on the map using the Google Maps API.
- Users can search for specific gyms or filter the results.
- If location access is granted, the app displays the nearest gyms.

Booking a Course/Slot/Trainer

- The user selects a gym and views available courses, slots or trainers.
- Upon selecting a session, the user confirms the booking.
- The app updates Firestore and sends a confirmation notification.
- If calendar synchronization is enabled, an event is added to the user's calendar.

Managing Gym Data (for Managers)

- A manager logs in and navigates to the management section.
- The manager can create, edit, or delete gyms, courses, and schedules.
- Any modifications update Firestore in real time.

Receiving Notifications

- Firebase Cloud Messaging sends notifications to remind users of upcoming bookings.
- Notifications appear both as push notifications and in the in-app notification center.

Searching for Gyms/Courses

- The user enters a search query.
- The app queries Firestore to retrieve relevant results.
- If the search is location-based, Google Maps API helps locate relevant gyms.

Managing Favorite Gyms

- The user selects a gym and marks it as a favorite.
- The app stores this preference in Firestore and caches it locally for quick access.
- The user can view saved gyms in the dedicated "Favorite Gyms" section.
- If a gym is removed from favorites, the app updates Firestore and the cache.

2.6 Component Interfaces

This section provides a comprehensive description of the interfaces between Flutter components and external services, detailing how the application communicates with Firebase services, third-party APIs, and internal provider methods.

2.6.1 Firebase Service Interfaces

Firestore Database Interfaces

Firestore serves as the primary data storage solution for the application, handling all persistent data including gym information, user profiles, bookings, and activity schedules. In the following section we present some examples of interfaces that are implemented:

Gym Management Operations:

- `addGym(Gym gym)`: Creates a new gym document in the 'gyms' collection with complete gym information including name, address, facilities, and contact details
- `updateGym(Gym gym)`: Updates specific fields of an existing gym document using Firestore's merge functionality
- `deleteGym(String gymId)`: Performs a soft delete by updating the gym's status field rather than removing the document entirely
- `fetchGymList()`: Fetches all active gyms with real-time listeners for live updates

Activity and Schedule Management:

- `setActivity(Gym gym)`: Creates activity documents within a gym's collection, including scheduling information, capacity limits, and pricing
- `setActivity(Gym gym)`: Updates activity details while maintaining referential integrity with existing bookings
- `deleteActivity(Gym gym, Activity activity)`: Removes activities and handles cascade deletion of related bookings

2.6.2 Booking and Reservation Management:

- `addBooking(Booking booking, Slot slot)`: Creates booking documents with atomic transactions to prevent overbooking
- `confirmPayment(String bookingId)`: Updates booking payment status with timestamp tracking
- `fetchBookings()`: Retrieves all bookings for a specific user
- `deleteBooking(Booking booking)`: Handles booking cancellation

User Profile Management:

- `createUser(User user)`: Creates comprehensive user profiles with role-based permissions
- `updateUserProfile(User user)`: Updates user information while maintaining data consistency
- `fetchUser()`: Retrieves complete user profile information
- `updateUserFavourites(User user)`: Manages user's favorite gym list with real-time synchronization

Firebase Cloud Functions Interfaces

Cloud Functions serve as the secure backend layer for payment processing and sensitive operations that cannot be performed directly from the client application.

Payment Processing Functions:

- `goToPayment(Booking booking)`: Initiates Stripe payment intents with proper metadata for tracking and reconciliation
- `confirmPayment(String bookingId)`: Confirms successful payments and updates booking status atomically

Firebase Authentication Interfaces

The authentication system handles user management with role-based access control and seamless integration with Google OAuth.

Authentication Operations:

- `signInWithEmailAndPassword(String email, String password)`: Handles traditional email/password authentication with proper error handling
- `signInWithGoogle()`: Manages Google OAuth flow with automatic profile creation and role assignment
- `signOut()`: Properly clears authentication tokens and local cache
- `fetchUser()`: Retrieves current authenticated user with role information
- `resetPassword(String email)`: Initiates password reset flow with secure email verification

2.6.3 External API Interfaces

Stripe Payment Processing

Stripe integration handles all payment operations with PCI DSS compliance and comprehensive error handling.

Payment Intent Management:

The application creates payment intents through Cloud Functions to maintain security. The process involves:

1. Client requests payment intent creation with booking details
2. Cloud Function validates request and creates Stripe PaymentIntent
3. Client receives secret for secure payment confirmation
4. Payment confirmation triggers webhook events processed by Cloud Functions
5. Booking status updates automatically upon successful payment

Webhook Integration:

Cloud Functions handle all Stripe webhook events including `payment_intent.succeeded` and `payment_intent.payment_failed` to maintain data consistency and trigger appropriate business logic.

2.6.4 Imgur Image Management

Imgur serves as the content delivery network for all application images including gym photos and user profile pictures.

Image Upload Operation:

- `uploadImage(String base64Image)`: Uploads images to Imgur

Usage Implementation:

The application implements a comprehensive image management system:

1. Images are compressed locally before upload to reduce bandwidth usage
2. Upload progress is tracked and displayed to users

2.6.5 Google Maps Platform Integration

Google Maps provides location services, geocoding, and interactive map functionality for gym discovery and navigation.

Map Display and Interaction:

- `GoogleMap widget`: Displays interactive maps with custom markers for gym locations
- `MarkerManager`: Handles dynamic marker creation and clustering for performance optimization
- `LocationService`: Manages user location permissions and GPS coordinates

Geocoding and Search:

- `geocodeAddress(String address)`: Converts addresses to coordinates for gym location storage
- `searchPlaces(String query, LatLng center)`: Implements place search with proximity-based results

2.6.6 Google Calendar Integration

Calendar integration allows users to automatically sync their gym bookings with their personal calendars for better schedule management.

Calendar Operations:

- `addToCalendar(Booking booking)`: Creates calendar events with proper timezone handling and reminder settings

2.6.7 Internal Provider Methods

Authentication Provider

The AuthenticationProvider manages user authentication state and role-based access control throughout the application.

Core Methods:

- `signIn(String email, String password)`: Handles user authentication with proper error handling and state management
- `signInWithGoogle()`: Manages Google OAuth flow with automatic user profile creation
- `signOut()`: Clears authentication state and navigates to login screen
- `getCurrentUser()`: Returns current authenticated user with role information
- `checkAuthState()`: Monitors authentication state changes for automatic login/logout

State Management:

The provider uses ChangeNotifier to update UI components automatically when authentication state changes, ensuring consistent user experience across the application.

Gym Provider

The GymProvider handles all gym-related data operations and state management for efficient data access and caching.

Data Management Methods:

- `getGymList()`: Fetches all gyms with real-time listeners and local caching
- `addGym(Gym gym)`: Creates new gyms with image upload integration
- `updateGym(Gym gym)`: Updates gym information with proper validation
- `removeGym(Gym gym)`: Handles gym deletion with dependency checking

Search and Filtering:

- `searchGyms(String query)`: Implements text-based search across gym names and descriptions

Booking Provider

The BookingProvider manages all booking operations with integrated payment processing and calendar synchronization.

Booking Management:

- `createBooking(Gym gym, Activity activity, Slot slot)`: Creates bookings with payment intent generation and calendar event creation
- `removeBooking(Booking booking)`: Handles booking cancellation with refund processing and calendar cleanup
- `getUserBookings()`: Retrieves user bookings with pagination and filtering options

Payment Integration:

- `goToPayment(Booking booking)`: Integrates Stripe payment processing with booking creation
- `confirmPayment(String bookingId)`: Processes successful payments and updates booking status

2.7 Selected Architectural Styles and Patterns

2.7.1 MVVM (Model-View-ViewModel) Pattern

The application implements the MVVM architectural pattern to ensure separation of concerns, maintainability, and testability throughout the codebase. This pattern is particularly well-suited for Flutter applications and provides clear boundaries between different layers of the application.

Model Layer

The Model layer represents the data and business logic of the application. It includes:
Data Models:

- `Gym`: Represents gym entities with properties like name, address, facilities, contact information, and geographic coordinates

- `Activity`: Defines gym activities including schedules, capacity, pricing, and instructor information
- `Booking`: Handles reservation data with user information, payment status, and cancellation policies
- `User`: Manages user profiles with authentication details, preferences, and role assignments
- `Instructor`: Represents the instructor assigned to an Activity
- `Subscription`: Represents the subscription to a gym for an user
- `GymLocation`: Represents gym location data with coordinates

Data Sources:

- `FirestoreDataSource`: Handles all Firestore database operations with proper error handling and offline support
- `StripeDataSource`: Manages payment processing operations through secure Cloud Function calls
- `GoogleMapsDataSource`: Provides location services and geocoding functionality
- `CalendarDataSource`: Handles calendar integration and event synchronization

Repository Pattern Implementation: The application uses the Repository pattern within the Model layer to abstract data access:

- Gym repository: Aggregates data from Firestore and caches frequently accessed gym information
- Booking repository: Combines booking data with payment information for comprehensive reservation management
- User repository: Manages user profiles with authentication state synchronization

2.7.2 ViewModel Layer

The ViewModel layer acts as an intermediary between the View and Model layers, handling business logic and state management using the Provider pattern.
Provider-Based ViewModels:

- `UserProvider`: Manages user authentication state, login/logout operations, and role-based access control
- `GymProvider`: Handles gym data retrieval, search operations, and favorite gym management
- `BookingProvider`: Manages booking operations, payment processing, and calendar synchronization
- `NotificationProvider`: Controls notification delivery, user preferences, and message history

State Management Characteristics: Each ViewModel provider implements ChangeNotifier to automatically update UI components when data changes. This ensures reactive user interfaces that respond immediately to data modifications without manual refresh operations.
Business Logic Separation: ViewModels contain all business logic including:

- Data validation and formatting
- Complex calculations (pricing, availability)
- Cross-service coordination (booking + payment + calendar)
- Error handling and user feedback generation

2.7.3 View Layer

The View layer consists of Flutter widgets that provide the user interface and delegate user interactions to ViewModels.

Widget Organization:

- Screen Widgets: Top-level pages like HomePage, GymPage, BookingsPage
- Component Widgets: Reusable UI components like GymCard, BookingCard, ActivityList
- Custom Widgets: Specialized components like MapView, PaymentForm, CalendarWidget

2.8 Authentication Architecture

2.8.1 Firebase Authentication Integration

The authentication system leverages Firebase Authentication to provide secure, scalable user management with support for multiple authentication providers.

Authentication Flow

- User initiates login through email/password or Google Sign-In.
- Firebase Authentication validates credentials and returns authentication tokens.
- Custom claims are used to assign user roles (`customer`, `admin`).
- Authentication state is monitored through `StreamBuilder` for automatic UI updates.
- Secure tokens are automatically refreshed by the Firebase SDK.

Google OAuth Implementation

The Google Sign-In integration provides seamless authentication with the following features:

- Automatic account discovery and selection.
- Secure token exchange with Firebase Authentication.
- Profile information synchronization (name, email, profile picture).
- Automatic user profile creation in Firestore upon first login.

2.8.2 Security Implementation

- JWT tokens are automatically managed by the Firebase SDK.
- Authentication state persistence across app sessions.
- Secure logout with proper token invalidation.
- Protection against common authentication vulnerabilities (CSRF, session hijacking).

2.8.3 Session Management

The application implements stateless authentication using Firebase's built-in session management:

- Authentication tokens are automatically refreshed before expiration.
- Offline authentication support for temporary network interruptions.
- Multi-device synchronization of authentication state.
- Secure storage of authentication credentials using platform-specific secure storage.

2.9 Payment Processing Architecture

2.9.1 Stripe Integration Strategy

The payment processing architecture prioritizes security and PCI DSS compliance by routing all sensitive operations through Firebase Cloud Functions.

Secure Payment Flow

- Client initiates payment request with booking details.
- Cloud Function creates Stripe `PaymentIntent` with proper metadata.
- Client receives `client_secret` for secure payment confirmation.
- Stripe processes payment with 3D Secure authentication when required.
- Webhook notifications update booking status automatically.
- Receipt generation and email delivery through automated systems.

Error Handling and Retry Logic

- Network failure handling with automatic retry mechanisms.
- Payment failure scenarios with user-friendly error messages.
- Webhook event processing with idempotency keys to prevent duplicate processing.
- Refund processing with proper audit trails and notification systems.

2.10 Other Design Decisions

This section lists some design characteristics for which the system should be run correctly.

2.10.1 Availability

The availability of the system ensures that users can reliably access the application and its services with minimal downtime. Several design decisions contribute to maintaining high availability, such as using the Firebase Infrastructure itself as the backend services are hosted on Firebase, which provides built-in scalability, redundancy, and automatic failover to minimize service disruptions.
This also brings the Firestore Replication as Firestore operates in a multi-region setup, reducing latency and ensuring data availability even in case of regional failures.

Another aspect is that the app uses local caching for frequently accessed data (e.g., favorite gyms, recent searches) to provide limited functionality even when the device is offline.

The web version of the application can be accessed on different devices and benefits from Firebase Hosting, ensuring global availability.
Firebase also guarantees Load Balancing as it dynamically scales resources based on demand, distributing traffic efficiently to avoid bottlenecks.

Updates to the Flutter app are rolled out in a way that minimizes service interruptions, with over-the-air updates available for mobile users via Flutter's hot reload features, so to grant Minimal Downtime Deployment.

These measures collectively enhance the availability of the application, ensuring a reliable user experience.

2.10.2 Notification service

In-App Notifications:

- Users can access past and upcoming notifications within a dedicated section of the app.
- Notifications are stored locally and synchronized with Firestore to ensure availability across devices.

User Preferences:

- Integration with Calendar API allows users to receive additional event reminders in their preferred calendar application.

Chapter 3

User Interface Design

3.1 Design principles

This section outlines the foundational color palettes and visual elements used in designing both the light and dark modes of the application interface. These themes ensure visual consistency, accessibility, and a seamless user experience across all screens and user roles (User, Admin).

The color schemes are based on Material Design 3 principles, adapted and customized to suit our brand tone and UI needs. The system includes semantic color groupings (primary, secondary, tertiary, error, surface, etc.), each with corresponding "on-color" and container values to ensure clarity and visual hierarchy in different UI contexts.
Each palette supports:

- Theming flexibility between light and dark modes;
- Accessibility through adequate contrast;
- Component consistency through reusable color roles;
- Role-based variation, allowing visual cues for Admin and User contexts when appropriate.

3.1.1 Light theme

The light scheme emphasizes bright surfaces, clear shadows, and vibrant contrast. It ensures optimal readability in high-light environments. Refer to the palette below:

- Primary: Deep navy blue (text and interactive elements);
- Secondary: Mustard yellow (accents, highlights);
- Tertiary: Muted magenta (support visuals);
- Error: Bold red/orange for error messaging;
- Surface: Soft off-whites and greys for backgrounds and containers;
- Inverse colors: Used in tooltips, overlays, and reversed contrast elements.

3.1.2 Dark theme

The dark scheme prioritizes eye comfort, using darker surfaces with pops of light for emphasis and interaction.

- Primary: Lavender tones on dark backgrounds;
- Secondary: Bright yellow on dark containers for contrast
- Tertiary: Pink-magenta with clear contrast text
- Error: Orange/red tones remain bold and noticeable
- Surface: Deep greys and blacks for background and depth
- Inverse: Light tones used in dark popovers or contrasting elements

3.2 User screens mockup

3.2.1 Mobile

Login

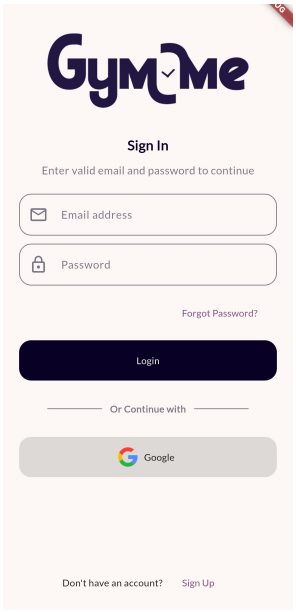


Figure 3.1: Login screen

This screen represents the Login interface of the GymMe application, adapted for mobile devices using a light theme. It retains all functional elements of the desktop version while optimizing layout and spacing for smaller screens.

Layout Overview

- **App Logo:**
  The GymMe logo is positioned at the top and scaled appropriately for mobile screens. It remains a central visual element to reinforce branding.

- **Login Prompt:**
  Below the logo, the user is presented with a simple title Sign In and a supporting message: *"Enter valid email and password to continue"*.

- **Form Fields:**

  – **Email address** – displayed with an email icon inside a rounded input box.
  – **Password** – shown with a lock icon, indicating secure input.

  Both fields use large touch-friendly inputs with clear borders.

- **Forgot Password:**
  A *"Forgot Password?"* link is placed directly beneath the password field, aligned to the right, and styled in a purple accent.

- **Primary Action: Login Button**
  A prominent, dark-colored **Login** button spans nearly the full width of the screen, with significant padding and rounded edges for touch accessibility.

- **Alternative Sign-In Divider:**
  A horizontal divider with the label *"Or Continue with"* separates the email/password login from social login options.

- **Google Login Button:**
  A large, rounded button with the Google logo and label **Google**. It features a light gray background and accommodates mobile interaction standards.

- **Bottom Prompt: Sign Up Link**
  At the very bottom, the user is prompted with: *"Don't have an account?"* followed by a clickable **Sign Up** link in purple.

Home

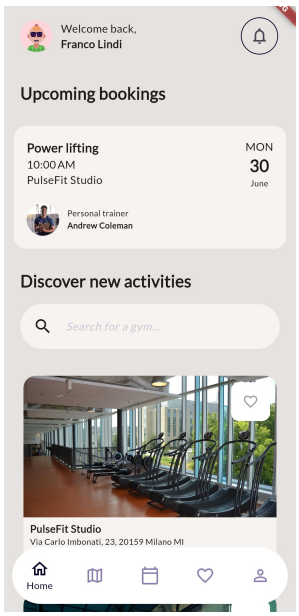


Figure 3.2: Home screen

This screen represents the Home Page of the mobile application. It is the main dashboard that users see upon logging in, offering quick access to current bookings and discovery of new gym activities.

Key Elements

- **Welcome Banner:** A greeting message at the top addresses the user by name, alongside a personalized avatar and a notification bell icon to alert the user of any updates or reminders.
- **Upcoming Bookings:** This section highlights the user's next scheduled activity—in this case, a *Power lifting* session at 10:00 AM at *PulseFit Studio*, scheduled for Monday, June 30. The booking card includes the gym name and an image of the personal trainer, *Andrew Coleman*, enhancing familiarity and trust.
- **Discover New Activities:** Below the bookings section is a search field allowing users to explore other gyms. It includes a magnifying glass icon and placeholder text (*"Search for a gym..."*) to guide user interaction.
- **Gym Discovery Card:** A featured card introduces *PulseFit Studio* with an image of the facility, its name, and address (*Via Carlo Imbonati, 23, 20159 Milano MI*). A heart icon in the top-right corner of the card allows users to save or favorite the gym for later access.
- **Navigation Bar:** At the bottom, a fixed navigation bar includes five icons for key sections of the app: *Home*, *Map*, *Calendar*, *Favorites*, and *Profile*. The *Home* icon is filled in, indicating the current active page.

Gym

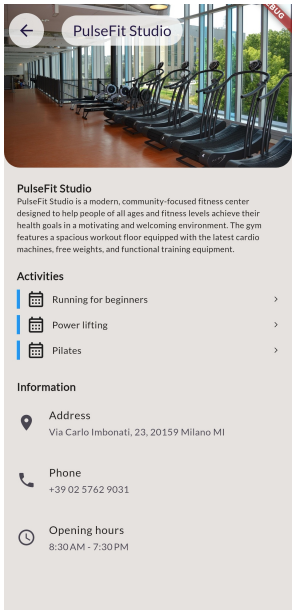


Figure 3.3: Gym screen

This screen represents the Gym Detail Page for *PulseFit Studio* in the mobile application. It is designed to provide users with essential information about the fitness center in a clean and user-friendly layout. The visual style follows the previously established light theme, utilizing soft background tones with dark, high-contrast typography for readability.

Key Elements

- **Header Section:** A full-width banner image of the gym provides visual context, overlaid with the gym name in a semi-transparent container to ensure legibility without obscuring the background. A back arrow in the top left corner supports easy navigation.
- **Gym Description:** Below the image, the gym name is repeated in bold for emphasis, followed by a brief but informative paragraph outlining the studio's philosophy, facilities, and focus on inclusivity.
- **Activities Section:** A vertical list of offered classes—*Running for beginners*, *Power lifting*, and *Pilates*—each accompanied by a calendar icon and chevron indicator, suggests additional details are available upon tapping. The list uses icons and blue accent bars for visual differentiation and interactivity.
- **Information Section:** Includes the gym's *Address, Phone Number, and Opening Hours*

Booking

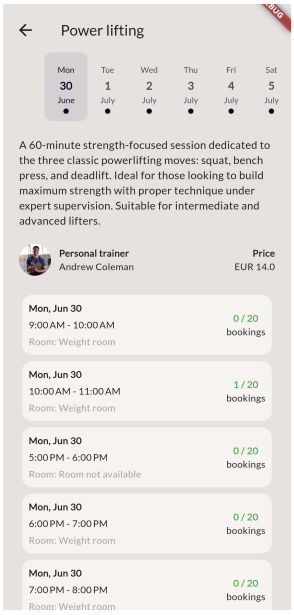


Figure 3.4: Booking an activity screen

This screen represents the detailed booking interface for the Power Lifting activity within the mobile application. It allows users to read about the session, check availability, and make bookings for specific time slots.

Key Elements

- **Header and Navigation:** At the top, a back arrow icon enables users to return to the previous screen. The title "Power lifting" clearly indicates the current activity being viewed.
- **Date Selector:** A horizontal date picker allows the user to browse available sessions for the week. The selected date (*Monday, June 30*) is highlighted with a filled background and bold text.

- **Activity Description:** A paragraph provides a concise overview of the session, stating it is a 60-minute strength-focused class covering squats, bench press, and deadlifts. It mentions the target audience (intermediate and advanced lifters) and emphasizes proper technique and supervision.
- **Trainer and Pricing Info:** Beneath the description is a compact row showing the trainer name and the price.
- **Session Time Slots:** A vertical list displays all available time slots for the selected day:
  - Each entry shows the time range, date, and room name (e.g., "Weight room" or "Room not available").
  - Booking availability is shown on the right as *0 / 20 bookings*, indicating the number of users currently signed up.

Profile

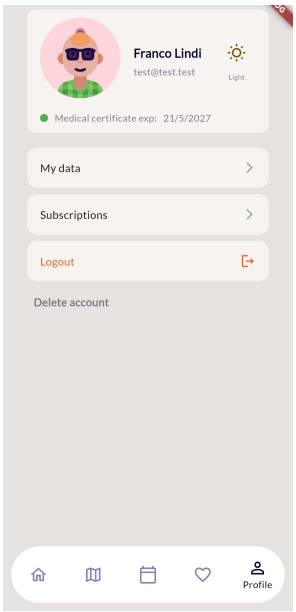


Figure 3.5: Profile screen

The Profile screen provides the user with a centralized area for managing their personal account details and settings. This mockup is rendered in the light theme, as indicated by the sun icon next to the user's name.

- **User Information:** At the top, a card component displays the user's profile picture, full name, email address, and the currently active theme. A green dot with accompanying text indicates the expiration date of the user's medical certificate.
- **Navigation Options:** Two navigable items are presented using standard list-style rows:
  - `My data`: Redirects to a screen where users can view and update personal information.
  - `Subscriptions`: Opens a section for managing active plans and billing details.

- **Session Controls:** A clearly distinguished `Logout` button is styled in orange, making it easy to spot, with an accompanying logout icon for clarity. Below that, a textual link allows the user to `Delete account`, styled in a more neutral color to reduce accidental clicks.

3.2.2 Desktop

Login

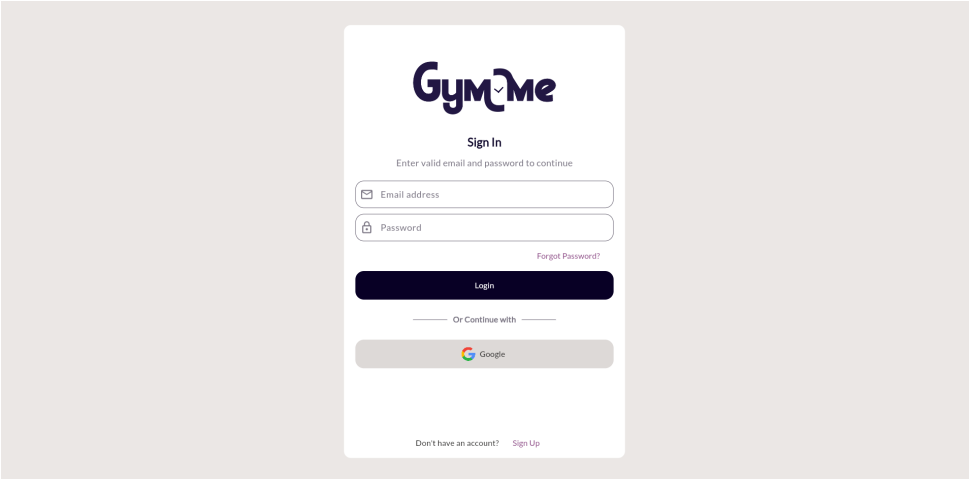


Figure 3.6: Profile screen

This screen represents the Login interface of the GymMe application, optimized for desktop display in light theme mode. It enables users to access their accounts through traditional email/password login or via Google authentication.

Layout Overview

- **Branding Header:**
  The GymMe logo is prominently displayed at the top center of the card, reinforcing brand identity and visual consistency throughout the app.
- **Login Form:**
  The central portion of the screen contains the login form with the following fields and elements:
  - *Email address input field*, labeled with an envelope icon for recognition.
  - *Password input field*, labeled with a lock icon to indicate security.
  - *Forgot Password?*, a link placed to the right of the password field, providing access to the recovery process.
- **Primary Action Button:**
  A large, dark-colored Login button is placed below the form fields. Its prominent size and contrasting color indicate it as the primary action on the screen.
- **Alternative Login Option:**
  Below the main button, a horizontal divider with the label *"Or Continue with"* separates the alternative sign-in method. A secondary button allows login via Google, visually distinguished with the Google logo and neutral background color.

- **Sign Up Prompt:**
  At the bottom of the card, users are prompted with *"Don't have an account?"*, followed by a *Sign Up* link for new user registration.

Home

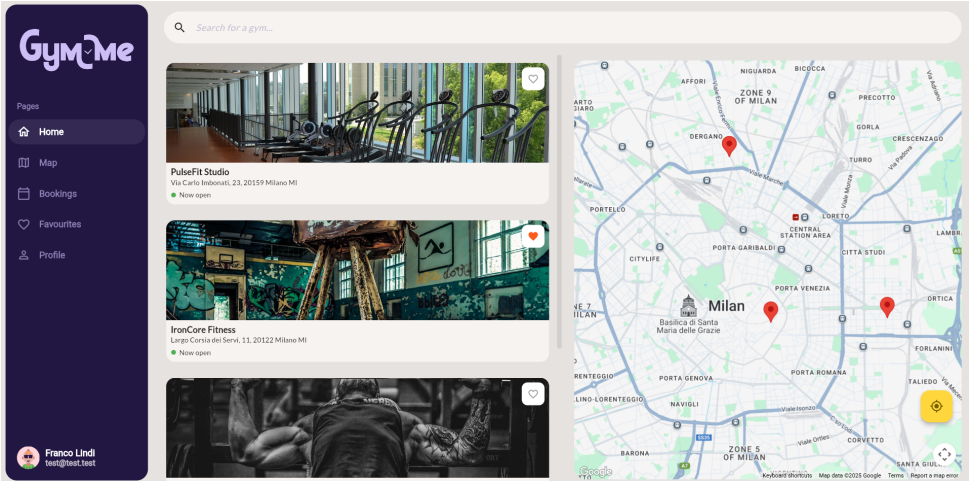


Figure 3.7: Home screen

The Home Screen serves as the primary entry point for users to discover gyms available in their vicinity.

Layout Breakdown:

- **Sidebar Navigation (Left Panel):**
  - The sidebar features a vertical menu with clearly labeled icons for quick access to key sections: *Home*, *Map*, *Bookings*, *Favourites*, and *Profile*.
  - At the bottom, a user avatar and profile information are persistently visible, enhancing personalization and accessibility.
- **Top Search Bar:**
  - Positioned centrally at the top of the screen, the search input allows users to find specific gyms by name.
- **Content Area (Center-Left):**
  - Each gym is presented as a card with a high-resolution image, name, address, and status indicator (e.g., "Now open").
  - A heart icon allows users to add or remove gyms from their favourites.
- **Map View (Right Panel):**
  - An interactive *Google Map* integration displays gym locations with pins, helping users visualize proximity and make informed choices.
  - The map corresponds to the listed gyms on the left.

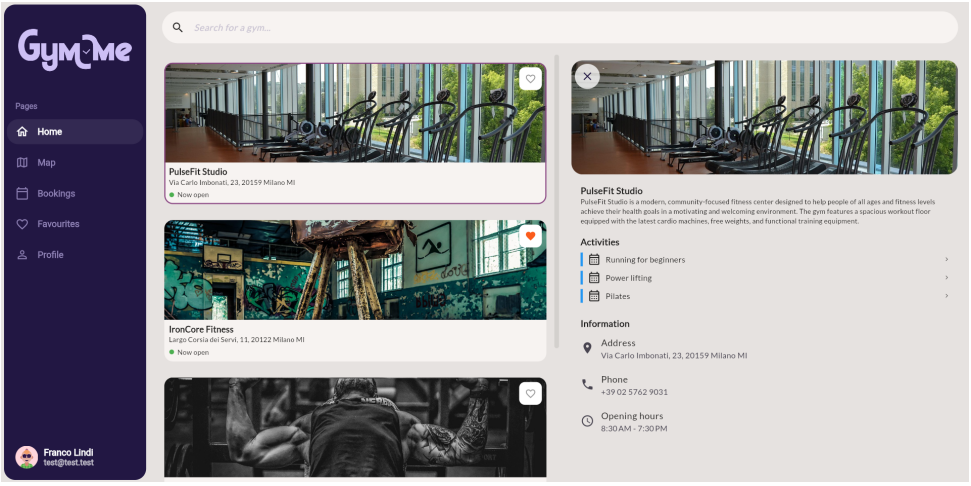


Figure 3.8: Gym screen

Gym

This screen illustrates the behavior of the application when a user selects a gym from the main list. The layout dynamically adjusts to display a two-column interface: the left side retains the gym list, while the right side reveals detailed information about the selected gym.
Upon selecting a gym (in this case, *PulseFit Studio*), the interface expands to present a details panel on the right. This design supports a seamless, context-preserving user experience by avoiding full screen transitions and maintaining the user's browsing state.

Left Column:

- The left column remains consistent with the home screen, displaying the gym cards.
- The selected gym is visually highlighted with a colored border, indicating it is currently active.

Right Column – Details Panel:

- **Header and Description:** A large banner image of the selected gym is displayed, followed by a short description that conveys the gym's focus and atmosphere.
- **Activities:** A section listing available activities.
- **General Information:** A structured section with clear iconography providing key information, such as *Address, Phone, and Opening hours*.
- **Close Button:** Located at the top left of the details panel, allowing the user to collapse the panel and return to the default list view.

Booking

This screen represents the interface displayed when a user selects a specific activity from a gym's detailed view. The layout shifts to reveal all relevant details of the selected class along with a calendar and booking options.

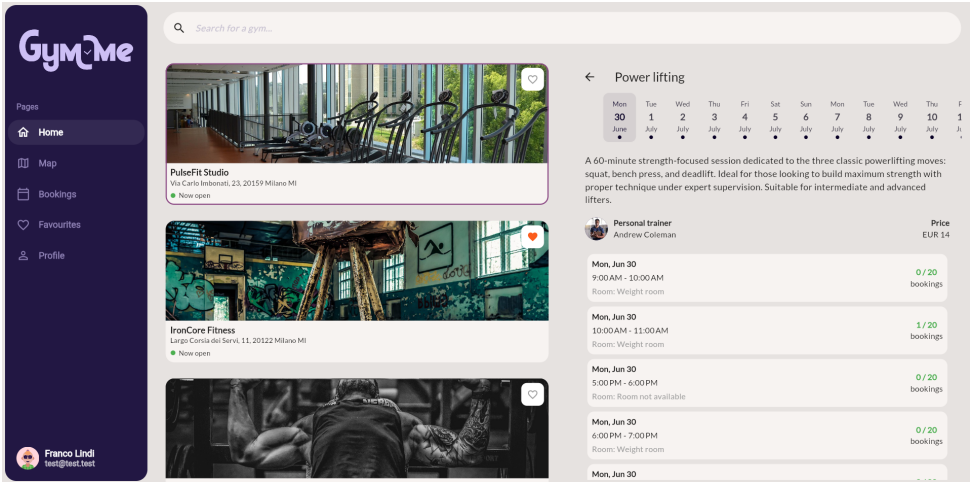


Figure 3.9: Booking an activity screen

Layout Overview: The interface maintains the two-column layout:

- **Left Column:** Remains unchanged, showing the main gym list, with the selected gym highlighted.
- **Right Column:** Displays the selected activity details, schedule, trainer, and available time slots for booking.

Activity Header:

- The title "*Power lifting*" is prominently displayed.
- A horizontal scrolling date selector allows the user to choose the desired day for booking.
- A detailed description outlines the class objectives, including exercises (squat, bench press, deadlift), and suitability (intermediate and advanced users).

Trainer and Pricing:

- Displays the assigned personal trainer (*Andrew Coleman*) with a small profile image.
- The price for the session is clearly stated.

Session Times:

- A vertical list of available time slots for the selected day is shown.
- Each card includes:
  - Time range;
  - Location;
  - Booking availability.

Profile

This screen represents the User Profile section of the GymMe application in **light theme mode**. It provides users with access to their personal data, subscription details, logout functionality, and the option to delete their account.

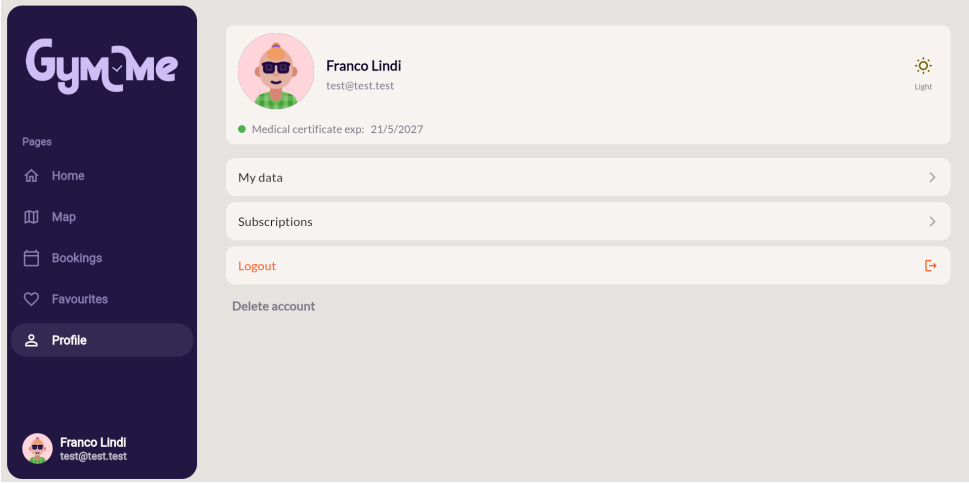


Figure 3.10: Profile screen

Main Content Area (Right Panel) This section is dedicated to profile-related actions and information. It includes:

- **User Info Card:** Displays the user's avatar, full name, email, and the expiration date of the medical certificate, with a green status indicator showing validity.
- **Theme Toggle:** A sun icon labeled *Light* on the top-right of the card allows users to switch between light, automatic, and dark themes.
- **Navigation Buttons:**
  - *My data*: opens a detailed view and the editor for personal information.
  - *Subscriptions*: displays the user's current and past gym subscription plans.
- **Logout Button:** Clearly highlighted in orange to ensure visibility and ease of session termination.
- **Delete Account Link:** Styled in grey and positioned at the bottom to prevent accidental interactions, initiating the account deletion process.

3.3 Admin screens mockup

3.3.1 Mobile

Home

This screen serves as the landing page for users, presenting a welcoming and engaging interface for discovering new gyms and activities. This particular mockup is rendered in the light theme and corresponds to the administrator view, as indicated by the presence of the "Add a new gym" button.

- **Header Section:** At the top, a personalized greeting is displayed with the user's profile picture and name , fostering a welcoming experience. To the right, a bell icon provides access to notifications.
- **Search Bar:** A prominent search bar is placed centrally, allowing users to quickly find gyms by name or keyword. It features a search icon and placeholder text: `Search for a gym...`.

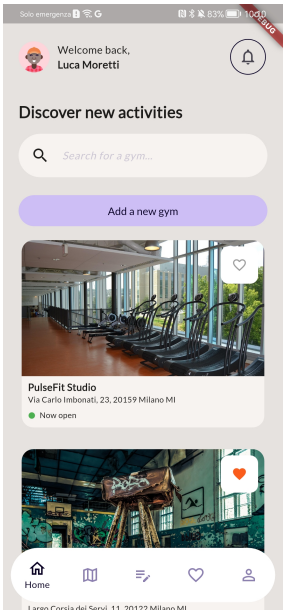


Figure 3.11: Home screen

- **Admin-Specific Action:** Below the search bar, a wide, lavender-colored button labeled `Add a new gym` enables administrators to submit new entries to the database. This feature is only visible in the admin role.
- **Gym Cards:** The main content area displays a vertically scrollable list of gym cards.
- **Bottom Navigation Bar:** The screen includes a persistent navigation bar with five icons: *Home*, *Map*, *Members*, *Favorites*, and *Profile*. The current screen (Home) is visually highlighted.

This layout balances discovery, quick access, and administrative functionality, while maintaining a friendly and spacious design.

Gym

This screen provides detailed information about a specific gym, in this case `PulseFit Studio`, and includes administrative controls. The clean and informative layout supports both general users and administrators, offering an overview of the gym's offerings and metadata.

- **Header:** The top navigation bar features a back arrow for returning to the previous screen and the name of the gym, clearly displayed.
- **Gym Description:** A brief paragraph introduces `PulseFit Studio`.
- **Activities Section:** A list of available gym activities is presented with calendar icons for visual clarity. An `Add activity` link is displayed below the list, available exclusively to admin users.
- **Information Section:** This section contains structured details about the gym, such as the address, the phone number, and the opening hours. Each piece of information is preceded by an appropriate icon for better readability and UX.
- **Admin Actions:** Positioned at the bottom, this section includes two key administrative controls:

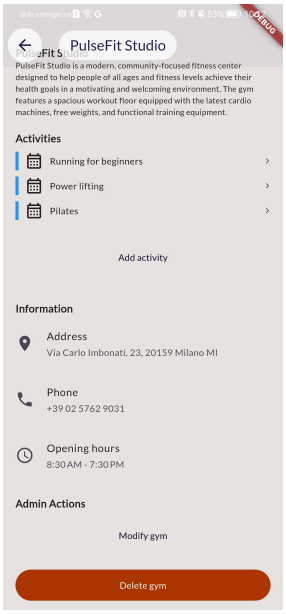


Figure 3.12: Gym screen

- `Modify gym` – A link allowing admins to edit the gym's details.
- `Delete gym` – A prominently styled red button enabling deletion of the gym record.

This screen supports clear information presentation while incorporating essential administrative tools in a secure and visually distinct manner.

Booking

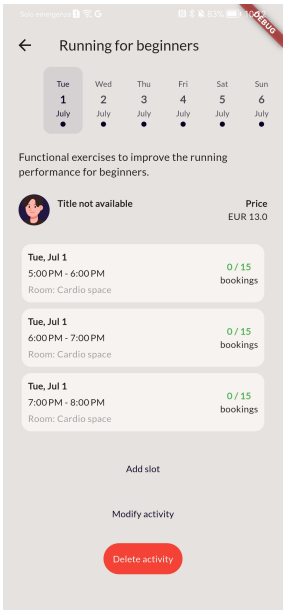


Figure 3.13: Booking an activity screen

This screen represents the administrative interface for managing a recurring beginner-level running class titled *"Running for beginners"*. It follows the light theme configuration, featuring a clean, minimalistic layout with clearly segmented components.

- **Date selector:** At the top, a weekly date selector allows the administrator to view and manage session slots for specific days. The currently selected date is highlighted to distinguish it from the rest of the week.
- **Activity description:** The central section provides a brief description of the activity, aimed at improving running performance through functional exercises. An avatar and a placeholder for the instructor's title are displayed, followed by the session pricing.
- **Slots information:** Each slot indicates its availability and the room where it is held, maintaining consistency in layout and typography. Slots are designed with subtle background coloring and bordered cards to enhance visual separation without overwhelming the interface.

At the bottom, three administrative actions are available:

- **Add slot** – Allows creation of additional time slots.
- **Modify activity** – Opens the interface to change the activity's details.
- **Delete activity** – A prominent red button for removing the activity, visually distinguished to emphasize its critical function.

The layout ensures usability, quick information scanning, and accessibility, aligning with the overall design principles discussed earlier.

Subscriptions

- **Subscription tab** Under Subscription Details, the admin can fill in:

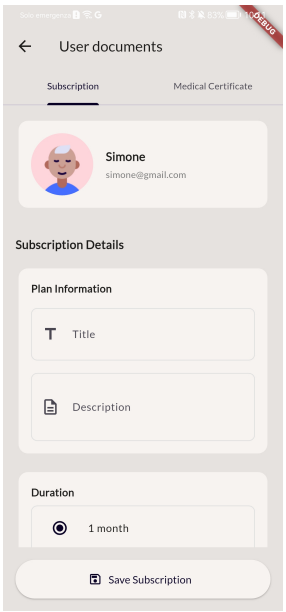


Figure 3.14: Subscription screen

  - **Plan Information** – title and description of the subscription plan.
  - **Duration** – selectable via radio buttons.
  - **Price** – price of the subscription.

The interface ends with a call-to-action button labeled Save Subscription, styled with an icon and rounded design to encourage completion of the form. The screen is part of a tabbed interface, with the "Subscription" tab selected.

- **Medical certificate tab** This screen handles the medical certificate management for a user.

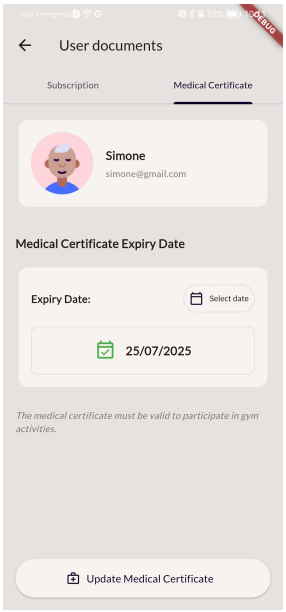


Figure 3.15: Medical certificate screen

It appears in the same tabbed interface as the Subscription tab, with "Medical Certificate" selected.

Key elements include:

- The user's avatar, name, and email.
- A section labeled Medical Certificate Expiry Date, where the expiry can be set via a date picker.
- The selected expiry date is shown with a confirmation icon (e.g., `25/07/2025`).

A short informational message reminds the user that a valid certificate is required to participate in gym activities.

At the bottom, a button labeled Update Medical Certificate allows changes to be submitted. The button is styled to be easily distinguishable, consistent with the overall UI.

3.3.2 Desktop

Home

The Home Page for administrators is designed to provide a clear overview of available gyms and facilitate quick access to essential management functionalities.

- On the left-hand side, a vertical navigation bar provides access to the main sections of the application: *Home*, *Map*, *Members*, *Favourites*, and *Profile*. The currently active page is highlighted to help users quickly orient themselves.
- At the top, a search bar allows filtering gyms by name, while a button labeled *"Add a new gym"* enables administrators to create a new gym entry.

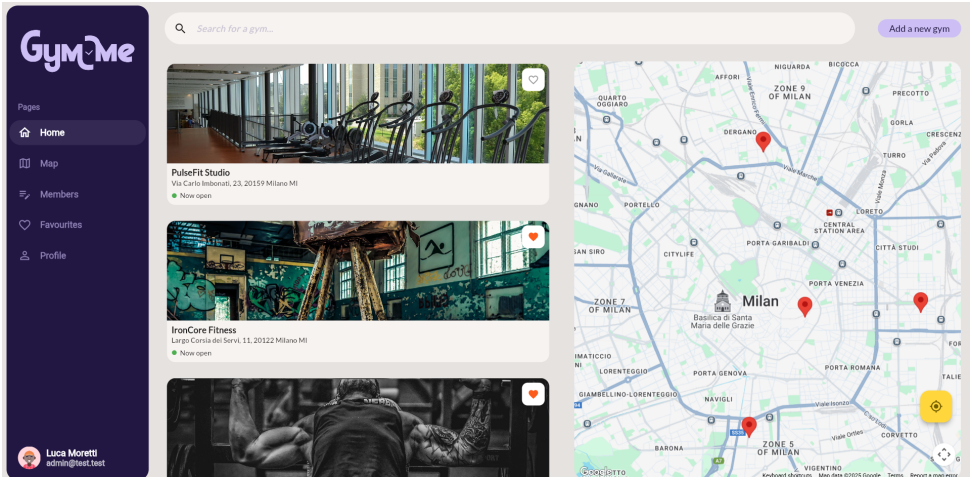


Figure 3.16: Home screen

- The main content area displays a list of gyms in card format, each showing a cover image, name, address, and real-time status (e.g., "Now open"). A heart icon allows marking gyms as favourites. On the right-hand side, an interactive map displays the locations of all listed gyms using red location markers, providing spatial context at a glance.

The user's profile information, including name, role, and email, is displayed in the lower-left corner of the navigation bar.

Gym

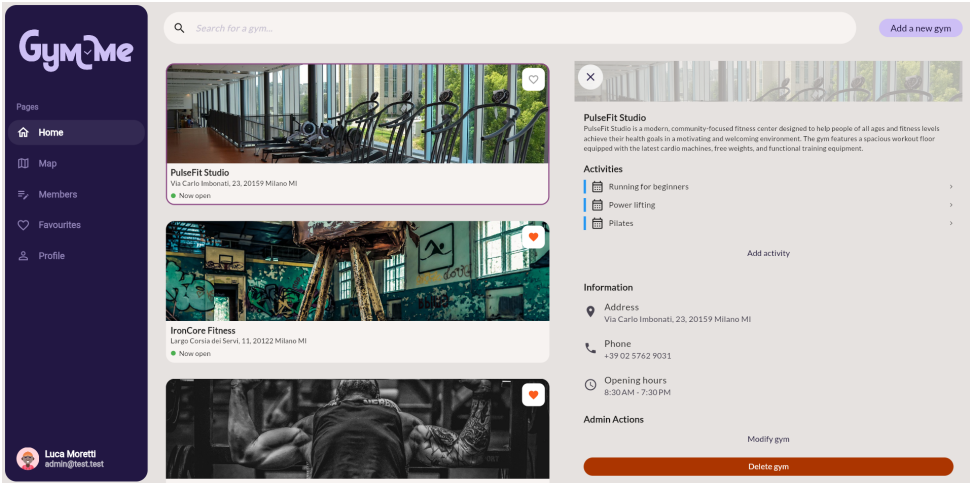


Figure 3.17: Gym screen

When a gym card is selected, detailed information about the gym is displayed in a panel on the right-hand side, as shown below.
The details panel includes:

- A larger header image of the gym.
- The gym's full name and description.
- A list of available activities, each accompanied by an icon.

- Key information such as address, phone number, and opening hours.
- Administrative actions, including:
  - **Add activity** — to associate new activities with the gym.
  - **Modify gym** — to edit gym details.
  - **Delete gym** — a clearly highlighted button for permanent removal of the gym.

The selected gym is visually distinguished in the list by a colored border, ensuring clarity of the current context. This layout allows administrators to efficiently manage gym information while maintaining visual consistency with the overall application design.

Booking

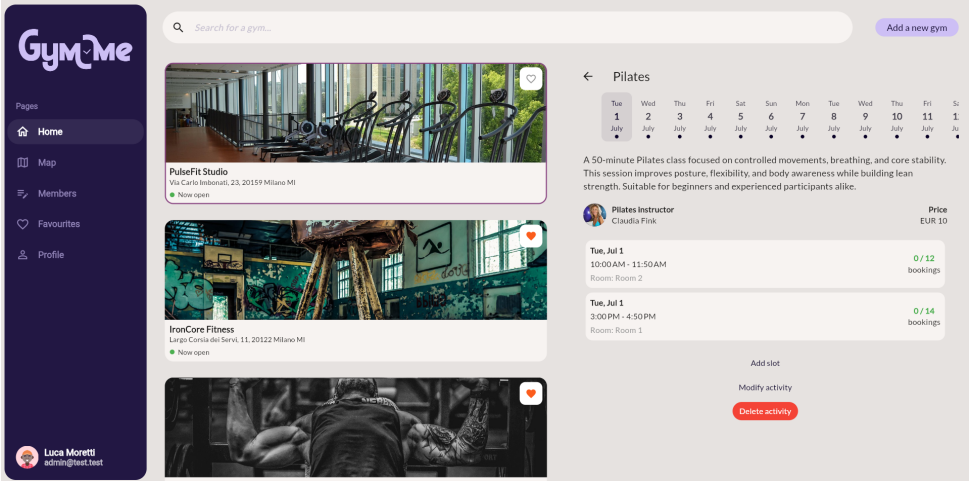


Figure 3.18: Booking an activity screen

The following screen displays the detailed view of an activity within a gym, along with administrative options for managing time slots and bookings.
When an activity of a gym is selected from the list, detailed information appears in a dedicated panel on the right-hand side. This panel includes:

- A horizontal date selector at the top for browsing upcoming sessions.
- The activity name and description, providing essential information about the session.
- The instructor's name and profile picture.
- The price of the activity.
- A list of scheduled time slots for the selected day, with details including:
  - Start and end times.
  - Assigned room.
  - Current booking status (e.g., "0 / 12 bookings") with real-time availability.
- Administrative actions at the bottom:
  - **Add slot** — to create a new time slot for the activity.

  - **Modify activity** — to edit activity details such as description or price.
  - **Delete activity** — a clearly marked, red-highlighted button to permanently remove the activity.

This interface enables administrators to efficiently manage activities, monitor bookings, and make adjustments to schedules with minimal navigation, ensuring streamlined operational control of the platform.

Subscriptions

The **User Documents** section is divided into two tabs: *Subscription* and *Medical Certificate*, both accessible within the same page. These tabs allow gym administrators to manage key user-related data conveniently and efficiently.

- **Subscription Tab:**

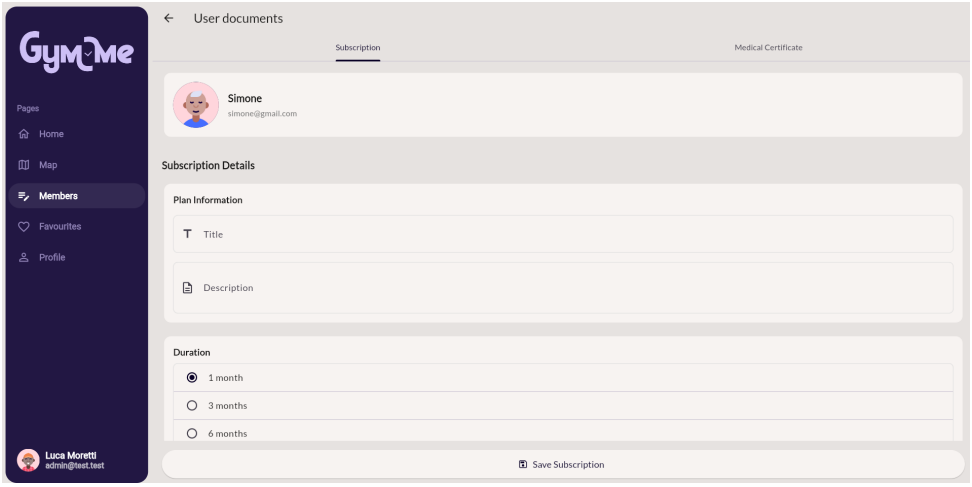


Figure 3.19: Subscription screen

  In the *Subscription* tab, the administrator can view and assign subscription details for the selected user. At the top of the interface, the user's avatar, name, and email address are prominently displayed to provide context and reduce errors in data entry.

  Below the user information, the Subscription Details area is divided into two main sections:

  - **Plan Information:** This includes editable fields for the subscription *Title* and a brief *Description*. These fields allow the admin to specify the type of membership or plan (e.g., "Standard", "Premium") and provide additional details if necessary.
  - **Duration Selection:** This section offers radio buttons for selecting the subscription period. Only one option can be selected at a time, ensuring clear and error-free input.

  A primary button labeled *Save Subscription* is located at the bottom, allowing the admin to save the selected subscription plan for the user.

- **Medical Certificate Tab:**

  The Medical Certificate tab is designed to store and manage the expiration date of the user's required medical certificate. The user's information is again shown at the top for consistency and quick reference.

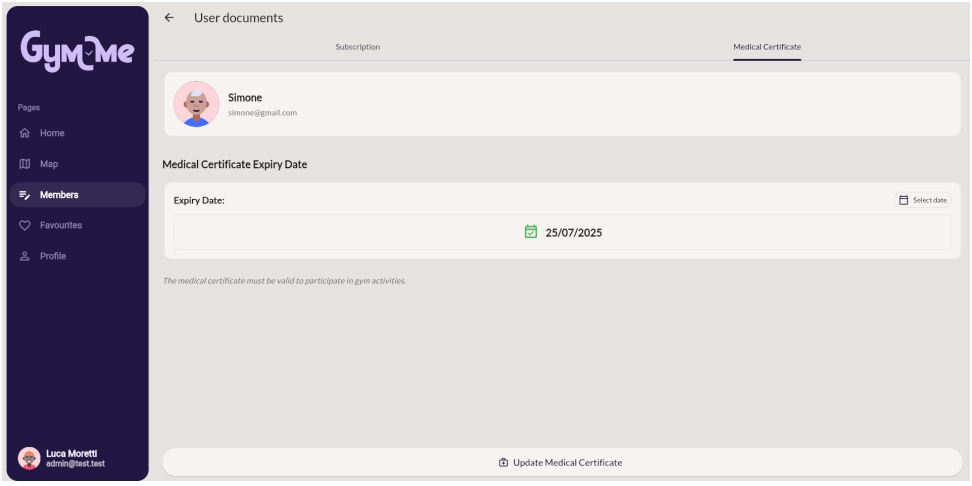


Figure 3.20: Medical certificate screen

The main component of this tab is a date input field labeled *Expiry Date*, which uses a calendar picker for ease of use and standardized formatting. A note below the field reminds the administrator that the certificate must be valid for gym participation.

At the bottom of the screen, the *Update Medical Certificate* button allows the admin to save changes to the expiry date.

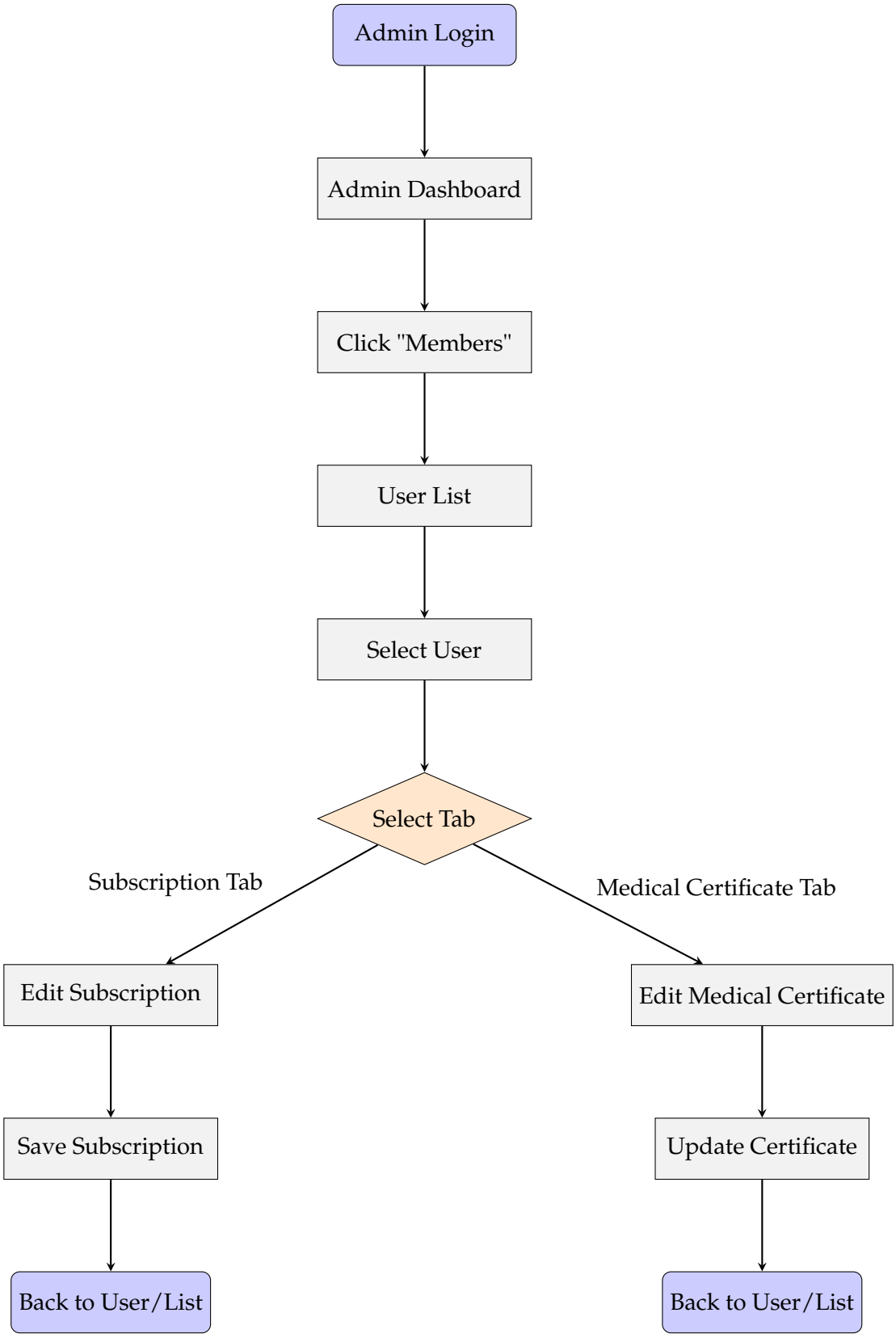
Both tabs are structured for clarity and usability, ensuring that all actions related to user documentation are straightforward and consistent with the application's design system.

3.4 User Flow

3.4.1 Admin Managing User Documents

The following flow represents the steps followed by an administrator to update a registered user's documents — specifically the subscription plan and the medical certificate expiration date.

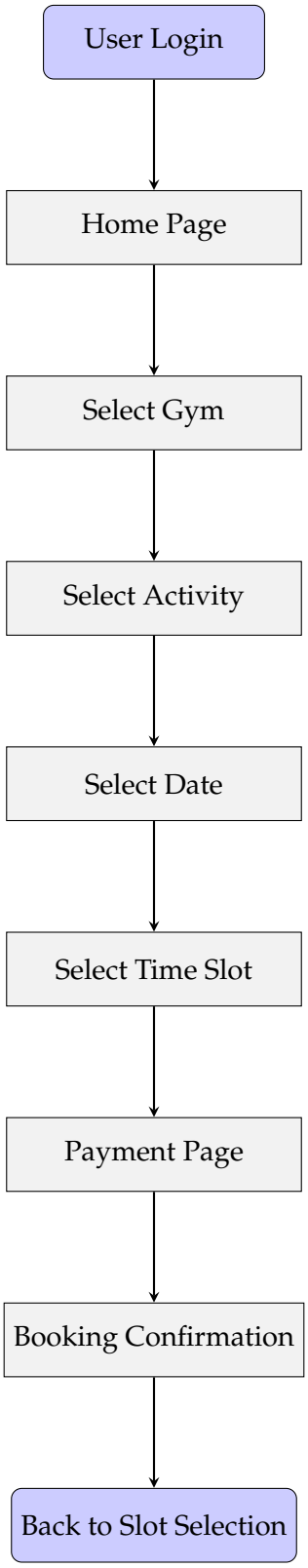
- The administrator logs into the system and lands on the main dashboard.
- From the navigation menu, the admin selects the "Members" section.
- A list of users is displayed, and the administrator selects a specific user.
- The system loads the **User Documents** page, which includes two separate tabs:
  - **Subscription Tab**, where the admin can update the subscription title, description, duration, and price.
  - **Medical Certificate Tab**, where the admin can set or modify the expiration date of the user's medical certificate.
- Each tab includes a corresponding action button to save the updated data:
  - *Save Subscription*
  - *Update Certificate*
- After saving, the administrator can return to the user profile or the member list.



3.4.2 User Booking an Activity

This flow outlines the steps a user follows to book a gym activity through the application, from login to payment and return to slot selection.

- The user logs into the application.
- From the home screen, the user navigates to the “Gyms” section.
- A list of available gyms is displayed. The user selects a specific gym to view its details.
- On the gym’s page, the user accesses the list of available activities.
- The user selects a specific activity (e.g., Yoga, CrossFit).
- A calendar view is displayed. The user selects a date.
- The system displays the available time slots for that activity on the selected date.
- The user selects a slot and proceeds to the payment page.
- After completing the payment, the system confirms the booking.
- The user is redirected back to the time slot selection page to optionally book additional sessions.

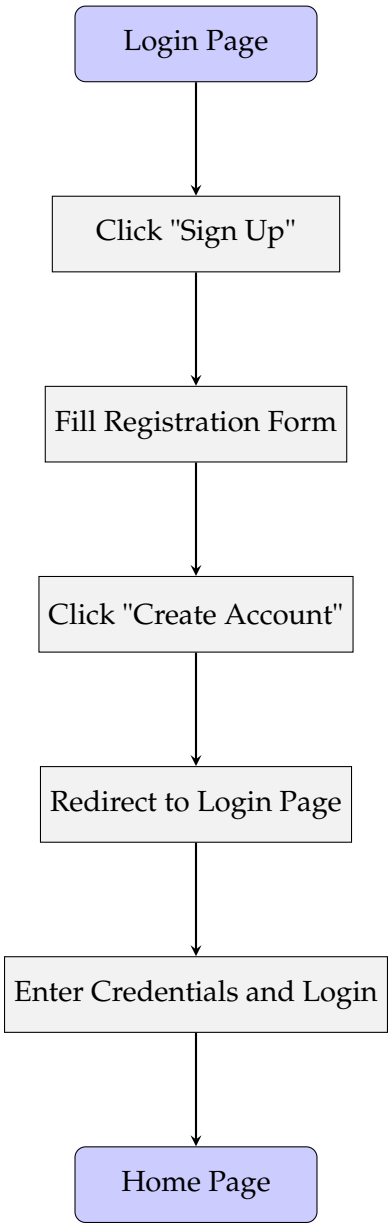


3.4.3 User Flow – Registration and Login

This user flow illustrates the process followed by a new user who registers an account and then logs in for the first time.

- The user lands on the Login page.

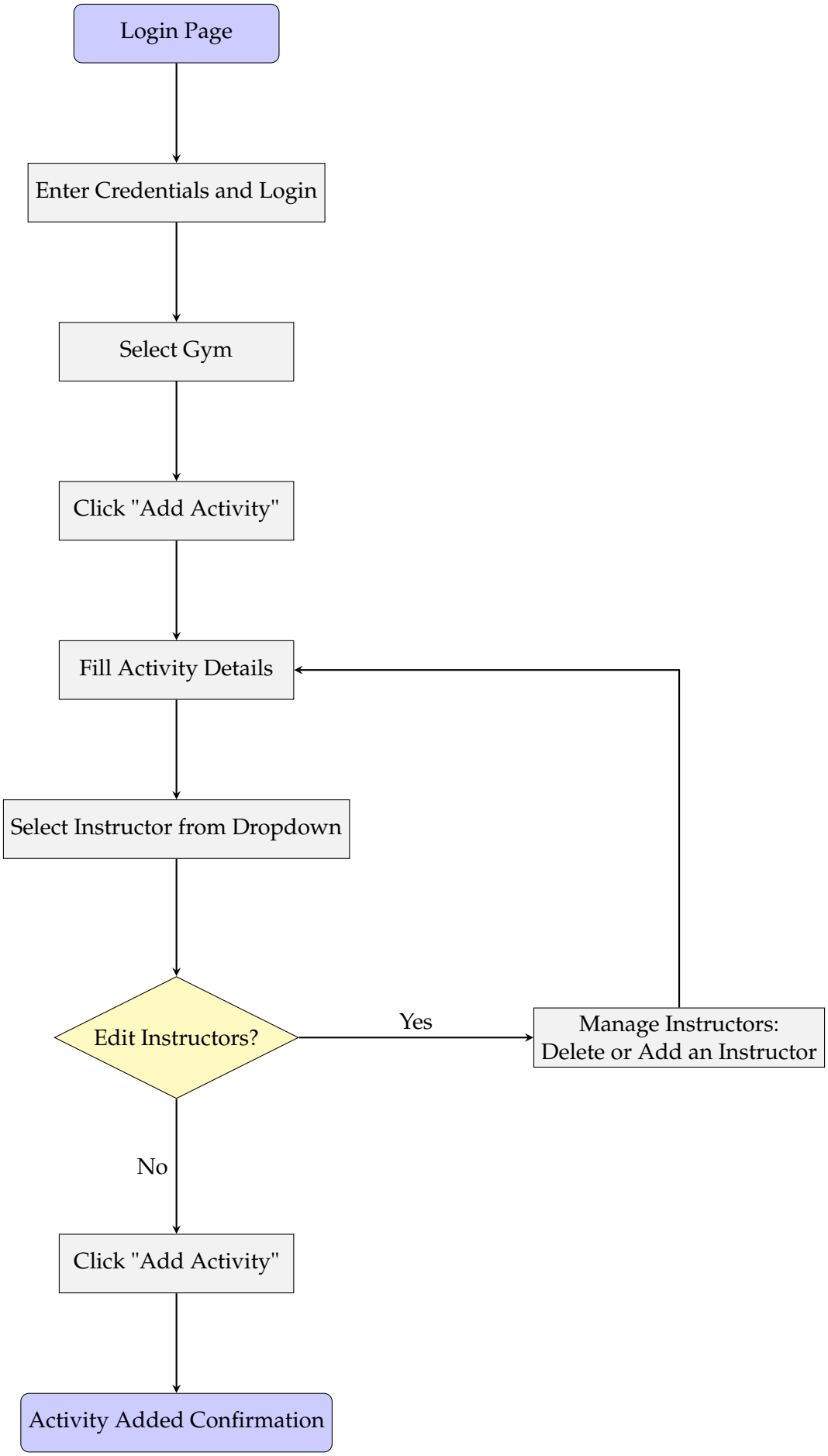
- Instead of logging in, the user taps the **"Sign Up"** button to start the registration process.
- A registration form appears. The user fills in their full name, address, and chooses a password.
- After completing the form, the user taps the **"Create Account"** button.
- The system confirms that the account has been successfully created and redirects the user to the Login page.
- The user enters their credentials and logs into the system.
- After login, the user is redirected to the Home page.



3.4.4 User Flow – Admin Managing Activities

This user flow illustrates the process followed by an admin who logs in and manages gym activities, including adding new activities and editing instructors.

- The admin lands on the Login page.
- The admin enters credentials and logs in.
- After login, the admin selects the gym from a list.
- The admin clicks **"Add Activity"**.
- The admin fills in the activity details: name, description, and price.
- The admin selects an instructor from a dropdown menu.
- Alternatively, the admin can click **"Edit Instructors"** to manage instructors:
  - The admin can delete an existing instructor.
  - The admin can create a new instructor by entering name, title, and profile image.
- After managing instructors, the admin returns to the activity form.
- The admin clicks **"Add Activity"** to save the new activity.
- The system confirms the activity has been added.



Chapter 4

Requirements Traceability

4.1 Overview

This chapter defines the requirements of the application and details the necessary functionalities, technical constraints, and performance expectations. Since the app serves two types of users, **User** and **Admin**, the requirements will be categorized accordingly to ensure clarity and specificity.
The requirements are divided into the following:

- Functional Requirements, which describe the features the application must provide.
- Non-Functional Requirements, which define aspects such as performance, security, and usability.
- System Requirements, which list the technologies and dependencies necessary for the proper functioning of the application.

This section will serve as a reference throughout the development process to ensure that the final product meets the expected needs.

4.2 Functional Requirements

4.2.1 User Functional Requirements

These requirements define the functionalities available to a regular User of the app.

Login and Authentication

FR-1 A User must be able to log in using email and password.

FR-2 A User must be able to sign up with an email and password.

FR-3 A User must be able to log in using Google Sign-In.

FR-4 A User must be able to sign up using Google Sign-In.

Home Screen

FR-5 A User must be able to search for a gym.

FR-6 A User must be able to view and open notifications.

FR-7 The User must see a list of nearby gyms.

FR-8 The User must be able to view and interact with the map displaying nearby gyms.

FR-9 The User must be able to view a list of their active bookings.

FR-10 The User must be able to access and manage their list of favorite gyms.

FR-11 The User must be able to view and update their personal profile page.

FR-12 The User must be able to add a gym to his favorite gyms.

FR-13 The User must be able to visualize the list of his favorite gyms.

Notifications Screen

FR-14 A User must be able to scroll through the list of notifications.

FR-15 The User must be able to clearly identify unread notifications, which should be visually highlighted.

Gym Map Screen

FR-16 If location access is granted, the map must center on the User's current location. Otherwise, it must default to Milan.

FR-17 The User must be able to re-center the map on their current location.

Active Bookings Screen

FR-18 The User must be able to see a list of their active bookings.

FR-19 The User must be able to add a booking to their Google Calendar.

FR-20 The ser must be able to pay for the activity he intend to partecipate to.

FR-21 The User must see booking details including date, time slot, gym name, and booked activity.

FR-22 The User must be able to view past bookings in the history tab.

Search Screen

FR-23 The User must be able to search for a gym by name.

FR-24 The User must be able to search for a gym by address.

Profile and User Area

FR-25 The User must be able to view their profile information, including name and basic details.

FR-26 The User must be able to access and modify personal information.

FR-27 The User must be able to view their active bookings.

FR-28 The User must be able to view their gym memberships.

FR-29 The User must be able to view if their medical certificate is still valid.

FR-30 The User must be able to switch between light and dark mode.

FR-31 The User must be able to select automatic theme selection based on the operating system's appearance setting.

FR-32 The User must be able to delete their account permanently.

FR-33 A User must be able to log out and be redirected to the log-in/sign-up screen.

Personal Data Page

FR-34 The User must be able to upload a profile picture.

FR-35 The User must be able to update their name, phone number, and address.

FR-36 The User must be able to update their date and place of birth.

FR-37 The User must be able to update their tax code.

FR-38 The User must be able to view their active memberships.

FR-39 The User must see membership details, including validity and price.

FR-40 The User must be able to view their medical certificate status.

FR-41 The User must see the expiration date of their medical certificate.

Gym Page

FR-42 The User must be able to view the gym's name and opening hours.

FR-43 The User must be able to view the gym's address.

FR-44 The user must be able to read the description of the gym.

FR-45 The User must be able to access the slot booking page for the Weight Room.

FR-46 The User must be able to access the slot booking page for Courses.

FR-47 The User must be able to access the slot booking page for Personal Training.

Slot Booking Pages

FR-48 The User must be able to browse available slots in a calendar view.

FR-49 The User must be able to see available slots with details such as time, activity name, coach name (if applicable), available spots, and price.

FR-50 The User must be able to book a slot.

FR-51 The User must be able to pay to book an activity.

4.2.2 Admin Functional Requirements

These requirements define the functionalities available to an Admin, who manages the app.

Authentication

FR-1 The Admin must be able to log in and sign up.

FR-2 The Admin must be able to sign up with an email and password.

FR-3 The Admin must be able to log in using Google Sign-In.

FR-4 The Admin must be able to sign up using Google Sign-In.

Home

FR-5 The Admin must be able to search for gyms.

FR-6 The Admin must be able to view a list of nearby gyms.

FR-7 The Admin must have access to the gym map.

FR-8 The Admin must be able to access a dedicated section to add new gyms.

FR-9 The Admin must be able to access a list of favorite gyms.

FR-10 The Admin must be able to access the user search section.

FR-11 The Admin must be able to add a gym to his favorite gyms.

FR-12 The Admin must be able to visualize the list of his favorite gyms.

Gym Management

FR-13 The Admin must be able to add a new gym by filling out a dedicated form.

FR-14 The Admin must be able to modify gym details.

FR-15 The Admin must be able to update gym informations.

FR-16 The Admin must be able to delete a gym.

User Management

FR-17 The Admin must be able to view and navigate the list of users.

FR-18 The Admin must be able to add or modify user subscriptions and membership cards.

FR-19 The Admin must be able to update and validate a user's medical certificate.

FR-20 The Admin must be able to view the expiration date of a user's medical certificate.

Notifications Screen

FR-21 The Admin must be able to scroll through the list of notifications.

FR-22 The Admin must be able to clearly identify unread notifications, which should be visually highlighted.

Slot and Booking Management

FR-23 The Admin must be able to view and manage slots for Weight Rooms.

FR-24 The Admin must be able to view and manage slots for Courses.

FR-25 The Admin must be able to view and manage slots for Personal Training.

FR-26 The Admin must be able to view details of each slot.

FR-27 The Admin must be able to view a list of users who booked a slot.

FR-28 The Admin must be able to modify slot details.

FR-29 The Admin must be able to delete slots.

FR-30 The Admin must be able to add new slots.

Maps and Location

FR-31 If location access is granted, the map must center on the Admin's current location. Otherwise, it must default to Milan.

FR-32 The Admin must be able to re-center the map on their current location.

FR-33 The Admin must be able see the gyms on the map.

FR-34 The Admin must be able to be redirected to the gym page when tapping on a gym on the map.

Profile and User Area

FR-35 The Admin must be able to view their name and email.

FR-36 The Admin must be able to switch between light and dark mode.

FR-37 The Admin must be able to select automatic theme selection based on the operating system's appearance setting.

FR-38 The Admin must be able to delete their account permanently.

FR-39 A Admin must be able to log out and be redirected to the log-in/sign-up screen.

4.3 Non-Functional Requirements

Firebase, as the chosen backend infrastructure for this application, inherently provides several non-functional requirements that contribute to security, scalability, reliability, performance, and cost management. These requirements are predefined by the framework and ensure a robust and efficient system architecture.

NFR-1 The application must support automatic scaling to handle a growing number of users and data, leveraging Firebase's cloud infrastructure.

NFR-2 Firebase ensures high availability and reliability with its globally distributed servers, guaranteeing minimal downtime for critical services.

NFR-3 Firebase Authentication provides secure user sign-in methods.

NFR-4 All data stored in Firebase Realtime Database and Firestore is encrypted at rest and in transit, ensuring compliance with security best practices.

NFR-5 Firebase Hosting delivers content via a global CDN, reducing latency and improving application performance for users worldwide.

NFR-6 Firebase Firestore and Realtime Database include built-in backup mechanisms to prevent data loss and ensure recovery options.

NFR-7 Firebase Realtime Database allows instant updates across connected clients, ensuring a seamless user experience.

NFR-8 Firebase provides built-in logging and monitoring tools, such as Firebase Crashlytics and Performance Monitoring, to track and analyze app performance and errors.

NFR-9 Firebase follows industry standards for data privacy, including GDPR and CCPA compliance, to protect user information.

4.4 System Requirements

4.4.1 Mobile Application (Android)

SR-1 The mobile application must be compatible with Android versions 8.0 (Oreo) and above

SR-2 The app must support both Wi-Fi and mobile data connections (3G, 4G, 5G) for real-time functionality, such as booking, notifications, and map display.

SR-3 The application must request permission to access the device's GPS for location-based services.

SR-4 The application must request permission to send notifications.

SR-5 The app must be integrated with Firebase services for authentication, real-time data synchronization, notifications, and analytics.

4.4.2 Web Application (Browser)

SR-6 The web application must be compatible with the latest versions of the following browsers: Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.

SR-7 The web application must be supported on Windows, macOS, and Linux operating systems.

SR-8 The web application must support modern internet connections, including both high-speed broadband and mobile networks.

SR-9 The web application must integrate with Firebase for user authentication, real-time updates, and data synchronization across devices.

Chapter 5

Implementation, Integration and Test Plan

5.1 Overview

This chapter illustrates the implementation, integration and test plan of the system. These last steps are crucial for a development project because they may be sources of errors and lead to the quality of the final deliverables. It is worth noting that testing allows developers to find the presence of bugs but never shows their absence. Therefore the scope of testing, composed by the verification and validation processes, is to discover the majority of the running platform's errors and to solve them within the time terms defined in the scheduling of the project management. Furthermore, the presence of bugs can directly lead to the plan of development in terms of costs, quality and time, and deviation may change the allocation of resources, such as working people, along with budget deviation. Finally, a rigorous test plan should be designed because it can be relevant and worthy for the sake of the system to be.

5.2 Implementation Plan

The development of the GymMe system was organized using Notion to manage the project planning, track the progress of each feature, and assign tasks among team members. Each feature, bug fix, or improvement was tracked as a separate item on a Kanban-style board, ensuring a clear view of the development status.

Version control was handled using Git and GitHub. Each feature or fix was developed on a dedicated branch derived from the `main` branch. The team adopted a Gitflow-inspired branching model to isolate development activities and avoid regressions on the production code.

The implementation order followed the requirements prioritization, starting from the core functionalities such as authentication and database structure, then proceeding to user and admin flows separately. For users, features such as browsing activities, subscribing, and payments were developed. For admins, features like creating and managing activities were implemented. UI development for both web and Android platforms occurred in parallel, taking advantage of Flutter's multiplatform capabilities.

Firebase was chosen as the backend service due to its tight integration with Flutter and its real-time database capabilities, authentication services, and built-in support for web and mobile platforms.

5.3 Component Integration and Testing

The components of GymMe were integrated progressively. The architecture is modular and consists of several layers including *models*, *providers*, *services*, and *widgets*. Each module is designed to be loosely coupled and testable.
Integration followed a top-down approach. For example, models were implemented and tested first, followed by the corresponding services interacting with Firebase. Then, providers were added to handle the app's state, and finally, widgets were connected to the providers and tested through the UI.
The integration process was managed through GitHub using pull requests (PRs). Developers opened a PR when a feature was completed and pushed to its branch. Each PR triggered an automated pipeline via GitHub Actions, which executed Flutter tests. The main branch was protected by branch rules requiring the successful completion of these tests and approval from at least one reviewer before merging.
The GitHub Actions pipeline also automated the post-merge deployment process, as detailed in the next section.

5.4 System Testing

The GymMe application adopts a comprehensive and layered testing strategy to ensure correctness, stability, and maintainability of the codebase. The testing is divided into three major types:

Unit Tests: Verify isolated pieces of logic, such as model serialization or input validation.

Widget Tests: Verify the correctness of individual Flutter widgets and user interactions.

Integration Tests: Simulate complete user workflows across multiple components.

Technologies Used

Flutter Test Framework: Native support for unit and widget tests using `flutter_test`.

Mockito: For mocking services and Firebase interactions.

Integration Test Package: For full app-level testing on device/emulator.

GitHub Actions: Automatically runs all tests on each pull request.

Testing Layers and Coverage

Models — All data classes are tested for:

- JSON (de)serialization
- Equality/hashCode correctness
- Null safety and constructor behavior

Example: testing the `ActivityModel.fromJson()` method with edge case values.

Providers — All state managers (based on `ChangeNotifier`) are tested for:

- Correct state transitions
- Error handling and loading states
- Notification of listeners

Example: checking that subscribing to an activity updates the internal state and UI accordingly.

Widgets — All views are tested for:

- Rendering based on state
- Navigation and routing
- Interaction responses (e.g., button taps, form submission)

Example: the subscription screen correctly disables the confirm button when no slot is selected.

Services — Firebase service abstractions are tested using mocks:

- Firestore queries and updates
- Authentication flow (login, registration)
- Payments or any backend integration logic

Example: verifying that calling `ActivityService.addActivity()` results in the correct Firestore write.

Test Coverage Overview

High test coverage is a key goal of the GymMe project, both for ensuring quality and enabling safe refactoring. The test coverage is continuously monitored through automated workflows, and pull requests are blocked if tests fail.
The table below summarizes the current line coverage across key project components:

Component	Lines Covered	Coverage %
All Files	3951 / 4297	**91.9%**
models	226 / 233	**97.0%**
providers	409 / 441	**92.7%**
content (widgets, views)	3096 / 3369	**91.9%**
services	220 / 254	**86.6%**

Table 5.1: Test coverage by component

The overall average coverage exceeds **91.9%**, which is considered excellent for a production-level mobile/web application. The goal is to maintain this level throughout the project's lifecycle.

Continuous Integration and Verification

Each pull request triggers a GitHub Actions workflow that runs all Flutter tests. The CI pipeline enforces the following rules:

A pull request cannot be merged unless all tests pass.

A reviewer must approve the PR.

The `main` branch is protected against direct pushes.

Once the pull request is merged, another GitHub Action is triggered to:

Build the release APK

Build and deploy the Flutter Web version to a Raspberry Pi via SCP

Create a GitHub Release with tagged version

Notify a Telegram chat about the deployment status

This infrastructure ensures a fully automated and reliable release cycle for GymMe.

