COMP4920 Senior Design Project 2, Spring 2025
Advisor: Mehmet Ufuk Çağlayan

AccessFit: Enhancing Gym Management with QR Code-Based Access Control
High Level Design
Design Specifications Document

Revision 2.0
12.06.2025

By:
Aytun Yüksek, 20070001026
Emirhan Kurşun, 21070001038
Fehmi Mert Tezdoğan, 21070001021

Revision History

Revision	Date	Explanation
0.1	04.01.2025	Initial high level design
0.2	21.01.2025	Supervisor Feedback
0.3	27.01.2025	The **Introduction** section has been modified. The **AccessFit Architecture** section has been renamed and revised; its new title is **AccessFit Software Architecture**. Elements from **Section 2** have been moved to **Section 3** or revised based on feedback. User interface-related processes and visuals have been added. The **UML Package Diagram** has been relocated, and the **UML Class Diagram** has been updated. They have been aligned for consistency. A new table has been created to show which class belongs to which package and which layer those packages are part of. The **References** section has been restructured. Section 4 revised based on new Class Diagram.
1.0	30.01.2025	Final Draft
2.0	12.06.2025	Final Draft 2.0 revised

Table of Contents

1. Introduction

This Design Specification Document (DSD) is developed based on the AccessFit Requirements Specification Document (RSD), Revision 1.2, finalized on 03.01.2025. The document outlines the detailed design for AccessFit, a gym management application, and serves as a reference for the project's implementation phase. The comprehensive design will be completed during COMP4920.
AccessFit is a React Native-based mobile application designed to modernize gym access systems. The application utilizes QR code-based access control while integrating features such as personalized workout plans, progress tracking, and motivational tools. These functionalities aim to address the limitations of existing gym applications, enhancing user engagement and providing a seamless fitness experience.
The DSD emphasizes a structured approach to system architecture and component design, incorporating UML notations and industry standards. The design ensures compatibility and scalability across major platforms, including iOS and Android, adhering to React Native best practices. This document facilitates the successful realization of AccessFit, creating a dynamic tool for gym-goers to achieve their fitness goals.

2. AccessFit Software Architecture

The AccessFit system has **layered architecture**, ensuring **scalability, modularity, and maintainability**. Each layer has a distinct role in managing system functionalities and interactions.

The choice of a layered architecture was made to meet the system's non-functional requirements effectively. For instance, the system is required to maintain a response time of less than one second for up to one million simultaneous users. While a distributed system architecture could also fulfill this requirement, it would introduce unnecessary complexity given the project's scope and constraints. A layered approach, on the other hand, simplifies development by clearly separating concerns across presentation, business logic, and data layers. This design choice ensures that the system remains maintainable and scalable while achieving the required performance metrics. Furthermore, it aligns with the team's expertise and resources, providing a balance between performance and practicality for the AccessFit application.

3. AccessFit Software Structure

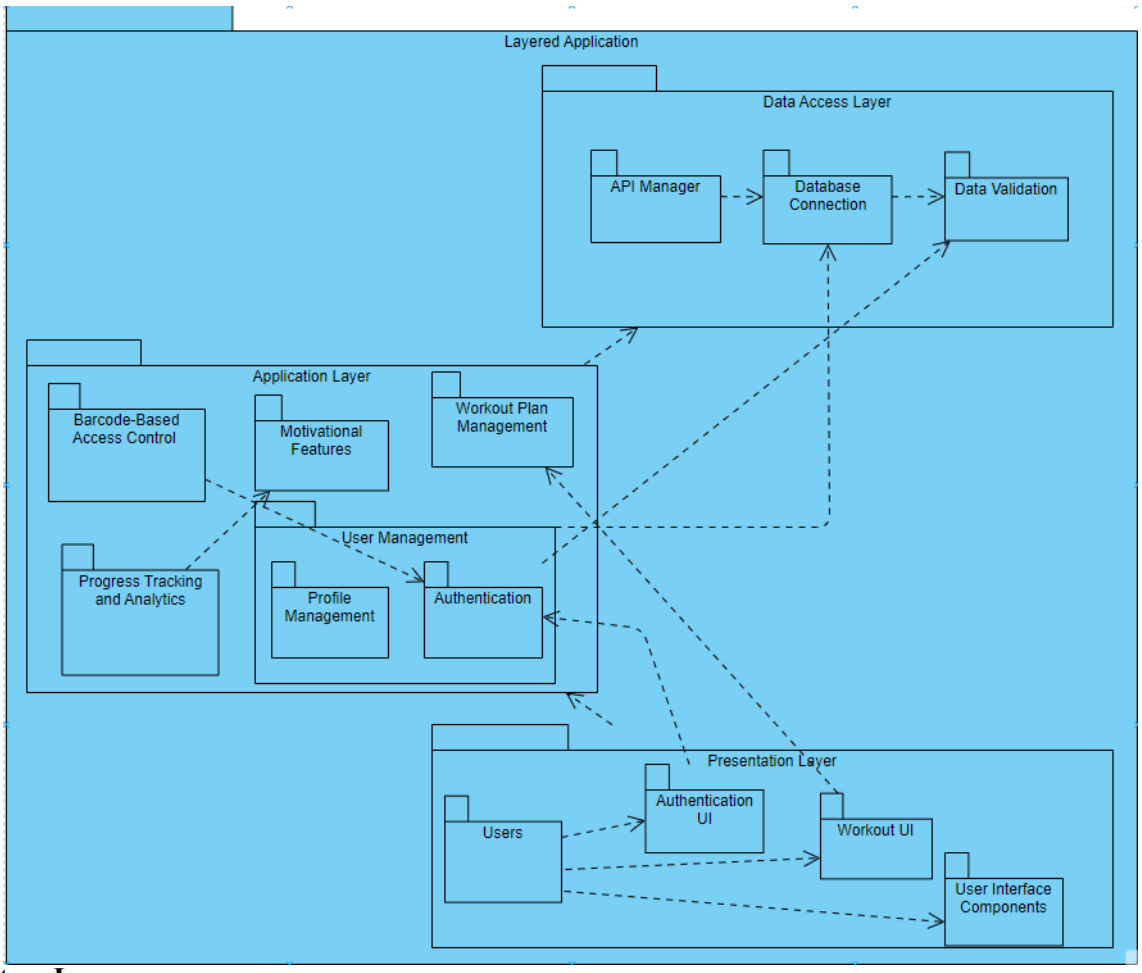
3.1 Overview of Software Structure

The AccessFit software system is structured has a three layer structure corresponding to layered architecture to ensure modularity, scalability, and maintainability. It consists of various subsystems that interact through well-defined interfaces. These subsystems are responsible for handling different functionalities such as user management, workout tracking, barcode-based access control, and analytics.

3.2 UML Package Diagram

A 3-layer UML package diagram should be placed here to visually represent the system structure. Ensure the diagram reflects the Presentation Layer, Application Layer, and Data Access Layer clearly.

3.3



System Layers

3.3.1 Presentation Layer

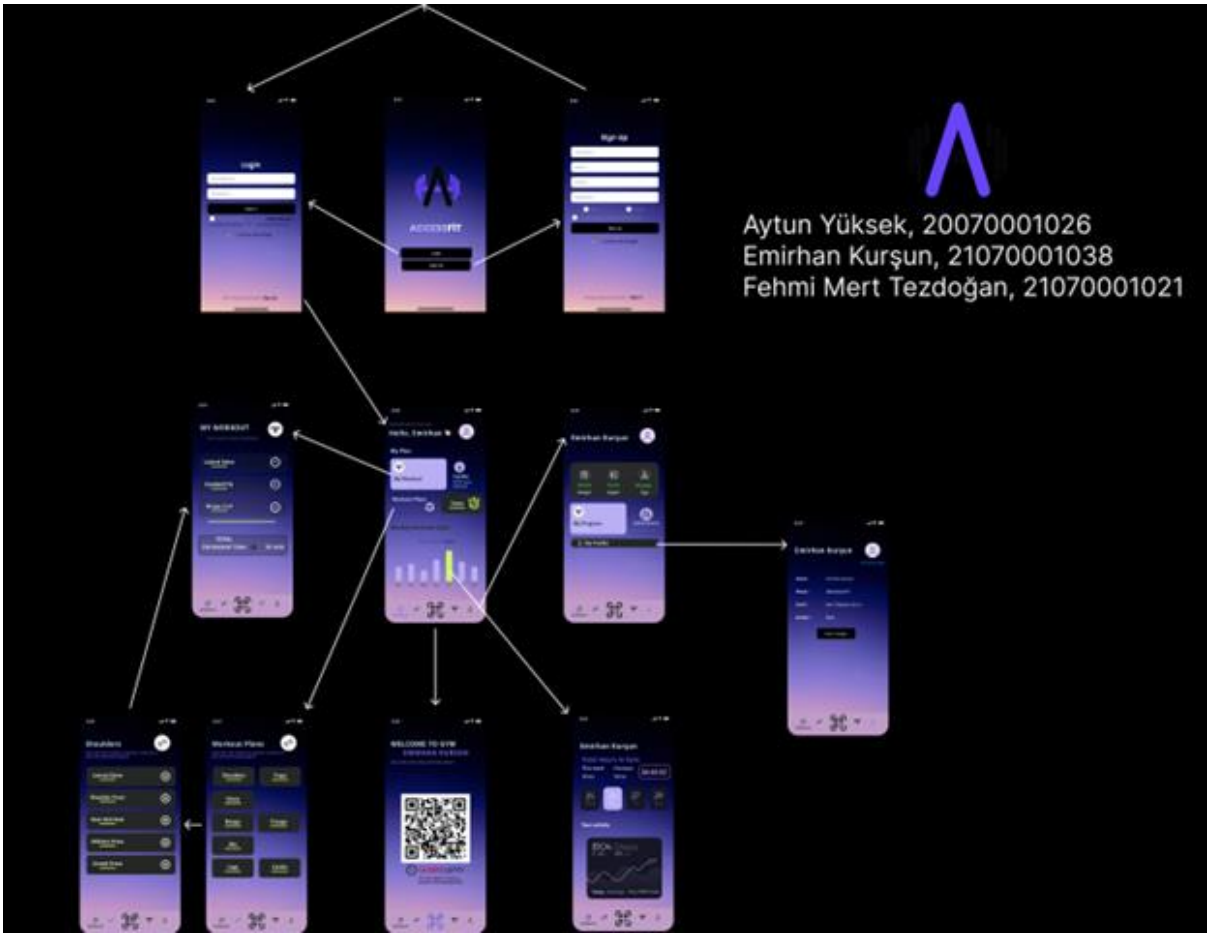
This layer is responsible for user interactions and the frontend interface. It is implemented using React Native, providing a cross-platform UI experience. The Presentation Layer does not contain business logic; instead, it communicates with the Application Layer via API requests.

- **Components**
    1. Authentication UI → Handles login, registration, and authentication interfaces.
    2. Workout UI → Provides interfaces for creating and tracking workouts.
    3. User Interface Components → Manages buttons, forms, and visual components.
- **Responsibilities**
    1. Sends user requests to the Application Layer.
    2. Displays processed data received from the Application Layer.
    3. Provides a responsive and intuitive UI using React Native.

User Interface Components

1. **Login Screen:** Allows users to enter their credentials to access their accounts. Includes a "Forgot Password" option for account recovery.

2. **Signup Screen:** Enables new users to register by filling in required fields, such as name, email, and password. Provides a seamless registration process for first-time users.
3. **Welcome Screen:** Displays a QR code for gym entry and important membership details, such as the number of days left on the subscription.
4. **Workout Dashboard:** Summarizes weekly workout statistics, including total time spent on workouts. Displays motivational badges earned by the user.
5. **User Profile Overview:** Shows personal details, including name, email, and membership information. Allows users to edit and update their profile details.
6. **My Workout Section:** Lists personalized workout plans with estimated times for each exercise. Offers a clear breakdown of exercises grouped by muscle groups (e.g., shoulders, legs, back).
7. **Workout Plans:** Provides tailored plans for different fitness goals. Enables users to select and follow structured routines.
8. **Analytics Screen:** Displays user progress through visual data, including workout history and step count. Helps users track long-term performance and set new goals.



3.3.2 Application Layer

This layer contains the core business logic and application workflows. It processes user requests from the Presentation Layer, applies business rules, and interacts with the Data Access Layer to fetch or update data.

- **Components**
    1. User Management
        - Profile Management → Handles user profiles and preferences.
        - Authentication → Manages user login, registration, and token validation.
    2. Workout Plan Management → Processes workout routines, stores progress, and applies validation rules.
    3. Barcode-Based Access Control → Generates and verifies QR codes for gym entry.
    4. Motivational Features → Sends reminders, manages achievement badges, and provides motivational feeds.
    5. Progress Tracking and Analytics → Logs user workout data, calculates insights, and provides performance summaries.

- **Responsibilities**
    1. Implements business rules and processes user actions.
    2. Handles authentication, validation, and session management.
    3. Manages workflow orchestration between different components.
    4. Calls Data Access Layer to retrieve or update database records.

3.3.3 Data Access Layer

The Data Access Layer (DAL) is responsible for managing communication with backend APIs and databases. It acts as a bridge between the Application Layer and the database, ensuring data integrity, validation, and security.

- **Components**
    1. API Manager → Handles API requests and responses between the frontend and backend.
    2. Database Connection → Manages database read/write operations.
    3. Data Validation → Ensures incoming data conforms to the expected formats and business constraints.
- **Responsibilities**
    1. Provides an abstraction layer for database operations.
    2. Ensures data consistency and integrity through validation mechanisms.
    3. Prevents direct database access from the Application Layer by exposing APIs.
    4. Uses secure authentication to manage API access.

3.4 High-Level System Flow

1. A user interacts with the UI (Presentation Layer).
2. The UI sends requests to the Application Layer via API calls.
3. The Application Layer processes the request, applies business logic, and forwards data requests to the Data Access Layer.
4. The Data Access Layer fetches data from the database, validates it, and returns it to the Application Layer.
5. The Application Layer processes the data (if needed) and sends the response back to the UI.
6. The UI updates and presents the data to the user.

3.5 Why This Architecture?

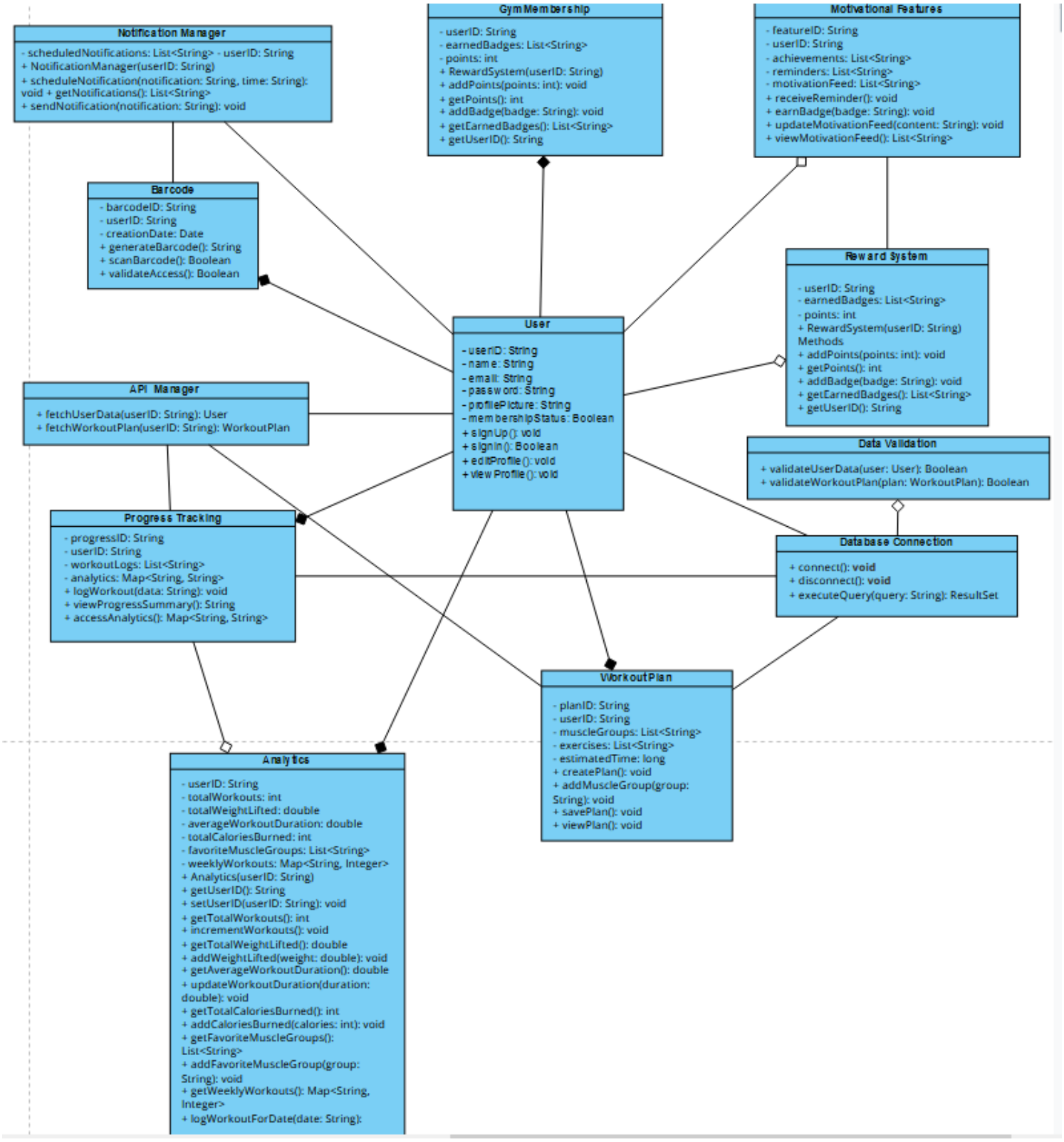
1. Separation of Concerns:
Each layer has a specific responsibility, preventing code duplication and increasing maintainability.

2. Scalability:
Since business logic is separated from the UI and data access, each layer can be modified independently.

3. Security:
Data access and authentication are isolated, reducing security risks.

4. Easier Debugging and Testing:
Issues can be pinpointed in a specific layer without affecting the rest of the system.

3.6 UML Class Diagram



3.7 Layered System Design

This

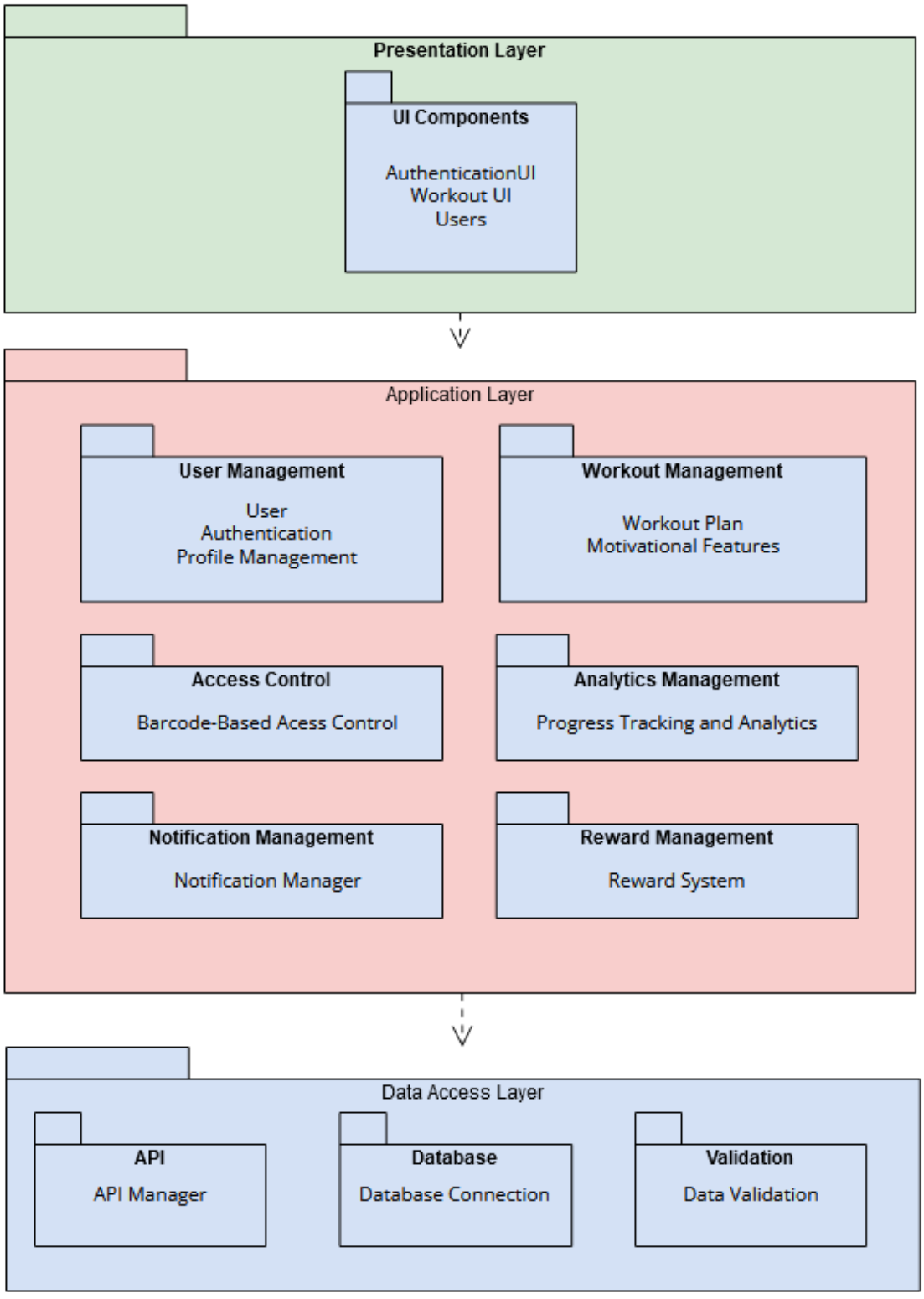


diagram illustrates the layered architecture of the AccessFit system, which is structured into three primary layers: Presentation Layer, Application Layer, and Data Access Layer. Each layer has distinct responsibilities, ensuring modularity, maintainability, and scalability. By logically organizing classes into appropriate packages within these layers, the system follows best software design practices, facilitating future updates and enhancements while maintaining a clear separation of concerns.

4. AccessFit Software System Detailed Design

4.1 AccessFit Main Module/Class

The AccessFitMainController class serves as the entry point for the application and manages the overall flow of all subsystems.

- **Main Class:**
  - Name: AccessFitMainController
  - Responsibilities:
    - Initialize the application.
    - Manage navigation between components (e.g., login, dashboard, barcode scanner).
    - Handle global error and exception management.
- **Design Details:**
  - The UML Class Diagram content is based on the diagram provided above.
  - Internal methods and logic will be designed to ensure seamless control over subsystems.

4.2 AccessFit Subsystem S1 Classes: User Management

Class: User

- **Responsibilities:**
  - Manages user registration and login operations.
  - Edits and displays user profiles.
- Methods:
  - signUp()
  - signIn()
  - editProfile()
  - viewProfile()

4.3 AccessFit Subsystem S2 Classes: Progress Tracking and Analytics

Class: ProgressTracking

- **Responsibilities:**
  - Records user workout progress.
  - Analyzes fitness data and provides summaries.
- **Methods:**
  - logWorkout(data: String)
  - viewProgressSummary()
  - accessAnalytics()

Class: Analytics

- **Responsibilities**:
  - Analyzes user fitness data (e.g., total workouts, calories burned, muscle groups).
  - Provides weekly analysis and progress reports.
- **Methods:**
  - addWeightLifted(weight: double)
  - getTotalWorkouts()
  - logWorkoutForDate(date: String)

4.4 AccessFit Subsystem S3 Classes: Motivational Features

Class: MotivationalFeatures

- **Responsibilities:**
  - Provides reminders and motivational content to the user.
  - Manages badges and achievement rewards.
- **Methods:**
  - receiveReminder()

    - earnBadge(badge: String)
    - viewMotivationFeed()

Class: RewardSystem

- **Responsibilities:**
    - Tracks user achievements and rewards.
    - Manages points and badge mechanisms.
- **Methods:**
    - addPoints(points: int)
    - addBadge(badge: String)
    - getEarnedBadges()

4.5 AccessFit Subsystem S4 Classes: Utility and Infrastructure

Class: NotificationManager

- **Responsibilities:**
    - Sends notifications using Firebase Cloud Messaging.
    - Schedules workout reminders and delivers them to the user.
- **Methods:**
    - sendNotification(notification: String)
    - scheduleNotification(notification: String, time: String)
    - getNotifications()

Class: Barcode

- **Responsibilities:**
    - Generates and validates barcodes.
- **Methods:**
    - generateBarcode()
    - scanBarcode()
    - validateAccess()

Class: APIManager

- **Responsibilities:**
    - Manages API requests and interactions.
- **Methods:**
    - fetchUserData()
    - fetchWorkoutPlan()

Class: DataValidation

- **Responsibilities:**
    - Validates user data and workout plans.
- **Methods:**
    - validateUserData()
    - validateWorkoutPlan()

Class: DatabaseConnection

- **Responsibilities:**
    - Manages database connection and queries.
- **Methods:**
    - connect()
    - disconnect()
    - executeQuery()

4.6 AccessFit Subsystem S5 Classes: User Gym Membership

Class: GymMembership

- **Responsibilities:**
    - Manages the user's gym membership.
    - Handles membership plan and renewal operations.

- **Methods:**
    - renewMembership()
    - isActive()
    - setMembershipType()

Class: WorkoutPlan

Responsibilities:

- Manages user workout plans.

- **Methods:**
    - createPlan()
    - addMuscleGroup()
    - SavePlan()
    - viewPlan()

5. Testing Design

In this course, the emphasis on testing design is minimal. Therefore, detailed test case designs for individual modules, methods, or classes, as well as integration test designs, are not included in this section. However, the following general remarks outline the approach and considerations for testing in the context of the AccessFit application.

5.1 General Testing Remarks

1. **Unit Testing**:
    - Each component (e.g., backend services, UI modules) will undergo unit testing to validate individual functionalities.
    - Automated testing tools such as JUnit (for Java) and Jest (for JavaScript/React Native) will be utilized.
2. **Integration Testing**:
    - The interaction between backend services and frontend modules will be tested.
    - Scenarios like barcode scanning, user authentication, and workout tracking workflows will be validated for seamless integration.
3. **System Testing**:
    - The entire application will be tested in an environment mimicking the production setup.
    - Key functionalities like user registration, barcode entry, and data persistence will be assessed.
4. **User Acceptance Testing (UAT)**:
    - Final testing will be conducted with end users to ensure the app meets their expectations and usability requirements.
5. **Performance and Scalability Testing**:
    - Tests will be conducted to ensure minimal latency during barcode scanning and stable performance under high user activity.
6. **Testing Tools**:
    - Postman for API testing.
    - Firebase Test Lab for Android/iOS compatibility and device-specific testing.

References

1. Requirements Specification Document (RSD) for AccessFit (Revision 1.2, finalized January 3, 2025).
2. Figma. (n.d.). *Figma: Design and prototyping tool for UI/UX development*. Retrieved January 25, 2025, from https://www.figma.com
3. Visual Paradigm. (n.d.). *Visual Paradigm: Software for UML diagrams and design models*. Retrieved January 25, 2025, from https://www.visual-paradigm.com
4. Visual Paradigm. (n.d.). *What is package diagram? UML unified modeling language guide*. Retrieved January 25, 2025, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/
5. Visual Paradigm. (n.d.). *What is class diagram? UML unified modeling language guide*. Retrieved January 25, 2025, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/