YAŞAR UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

COMP4920 Senior Design Project II, Spring 2025
Advisor: Prof.Dr.Mehmet Ufuk Çağlayan

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

12.06.2025

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

PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own experimentation, observation or specimen collecting.

Project Group Members:

Name, Lastname	Student Number	Signature	Date
Aytun Yüksek	20070001026		12.06.2025
Emirhan Kurşun	21070001038		12.06.2025
Fehmi Mert Tezdoğan	21070001021		12.06.2025

Project Advisors:

Name, Lastname	Department	Signature	Date
Mehmet Ufuk Çağlayan	Computer Engineering		

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our project supervisor, Prof.Dr.Mehmet Ufuk Çağlayan, for their invaluable guidance, constructive feedback, and unwavering support throughout the development of the AccessFit project. Their expertise and encouragement were instrumental in shaping the direction of our work.

We would also like to thank Department of Computer Engineering at Yaşar University for providing the resources and environment necessary to carry out this project.

Finally, we are grateful to our family and friends for their continuous motivation and understanding, which inspired us to complete this project successfully.

KEYWORDS

- Gym Management
- Barcode-Based Entry System
- Personalized Workout Plans
- Fitness Application
- Progress Tracking
- Motivational Features
- React Native
- Spring Boot
- API Integration
- Mobile Application Development
- User Engagement
- Scalable Architecture
- Data-Driven Analytics
- Cross-Platform Compatibility
- Firebase

ABSTRACT

AccessFit is a mobile application designed to revolutionize gym experiences by integrating a seamless **barcode-based entry system** with tools for **personalized workout management**. The application addresses common challenges faced by gym-goers, such as maintaining motivation, tracking progress, and managing routines effectively. AccessFit leverages modern technologies, including **React Native** for cross-platform development and **Spring Boot** for a robust backend, ensuring high performance and scalability.

The project incorporates motivational features like achievement badges and workout reminders, alongside progress tracking powered by data-driven analytics. By bridging the gap between fitness and technology, AccessFit aims to enhance user engagement and optimize the gym experience. This report provides a detailed overview of the system design, implementation process, and key findings, highlighting AccessFit's potential to transform the way users interact with their fitness routines. Future work will focus on expanding features, integrating IoT devices, and refining performance to meet evolving user needs.

ÖZET

AccessFit, **barkod tabanlı giriş sistemi** ile **kişiselleştirilmiş antrenman yönetimi** araçlarını bir araya getirerek spor salonu deneyimlerini dönüştürmeyi amaçlayan bir mobil uygulamadır. Uygulama, spor salonu kullanıcılarının motivasyonlarını koruma, ilerlemelerini takip etme ve rutinlerini etkili bir şekilde yönetme gibi yaygın zorluklarını ele alır. **React Native** ile çapraz platform uyumluluğu sağlayan bir frontend ve **Spring Boot** ile güçlü bir backend altyapısı kullanılarak yüksek performans ve ölçeklenebilirlik sağlanmıştır.

AccessFit, başarı rozetleri ve antrenman hatırlatmaları gibi motivasyonel özellikleri, veri odaklı analizlerle desteklenen ilerleme takibi ile birleştirir. Fitness ve teknolojiyi bir araya getiren AccessFit, kullanıcıların katılımını artırmayı ve spor salonu deneyimini optimize etmeyi hedefler. Bu rapor, sistem tasarımı, uygulama süreci ve önemli bulgular hakkında ayrıntılı bir genel bakış sunarak AccessFit'in kullanıcıların fitness rutinleriyle etkileşimlerini dönüştürme potansiyelini vurgular. Gelecekteki çalışmalar, özelliklerin genişletilmesi, IoT cihazlarının entegrasyonu ve performansın iyileştirilmesi üzerine odaklanacaktır.

TABLE OF CONTENTS

LIST OF ACRONYMS/ABBREVIATIONS

- API - Application Programming Interface
- CPU - Central Processing Unit
- FCM - Firebase Cloud Messaging
- IoT - Internet of Things
- MASVS - Mobile Application Security Verification Standard
- RAM - Random Access Memory
- RSD - Requirements Specification Document
- DSD - Design Specification Document
- UML - Unified Modeling Language
- UX - User Experience
- UI - User Interface

1. INTRODUCTION

1.1. Description of the Problem

In the modern fitness industry, gym-goers often face difficulties such as maintaining motivation, tracking their progress effectively, and managing their workout routines efficiently. Many existing gym management systems focus only on basic functionalities like booking or attendance tracking, leaving significant gaps in user experience, particularly in integrating secure entry systems with fitness progress tools.
To address these challenges, AccessFit aims to develop a mobile application that integrates a **barcode-based entry system** with tools for **personalized workout planning**, **progress tracking**, and **motivational features**. This innovative approach ensures a seamless gym experience, helping users stay consistent and achieve their fitness goals.

This report is structured as follows:

- **Section 2** presents an overview of related work and existing solutions in the fitness management domain.
- **Section 3** details the problem definition and requirements.
- **Section 4** provides the high-level design and design decisions.
- **Section 5** covers project management aspects such as effort, cost, and scheduling.
- **Section 6** concludes with a summary of the project outcomes, benefits, and future work.

1.2. Project Goal(s)

The primary goal of the AccessFit project is to develop a user-friendly **mobile application** that combines:

1. A **barcode-based entry system** for secure and efficient gym access.
2. **Personalized workout plans** tailored to user preferences and goals.
3. **Motivational features** such as reminders and achievement badges to sustain user engagement.
4. **Progress tracking tools** that provide actionable insights using advanced analytics.

Additionally, the project aims to foster effective teamwork, ensuring collaboration across disciplines and leveraging modern development tools and methodologies.

1.3. Project Outputs/Deliverables

For COMP 4910:

1. **Requirements Specification Document (RSD)** v2.0: Detailed documentation of functional and non-functional requirements.
2. **Design Specification Document (DSD)** v1.0: A comprehensive outline of system architecture, components, and design decisions.
3. A **team responsibilities document**: Clearly defined roles and responsibilities for each team member to ensure effective collaboration.
4. A **project plan**: A structured timeline detailing tasks, milestones, and deliverables to guide project execution.
5. A **poster**: Highlighting project objectives, system design, and anticipated outcomes.

Predicted Deliverables for COMP 4920:

1. A fully functional **mobile application**: Complete with enhanced features, optimizations, and refined user experience.
2. **Finalized RSD and DSD**: Updated with insights gained from implementation and testing phases.
3. **Testing documentation**: Results from unit, integration, and user acceptance testing.
4. A **user manual**: Providing clear guidance for end-users.
5. A **project website**: Emphasizing business, marketing, and user engagement aspects.
6. A **final report**: Summarizing project outcomes, challenges, and lessons learned.

2. SURVEY OF RELATED WORK

This section provides a survey of existing solutions in the fitness and gym management domain, focusing on their contributions, strengths, and limitations. The solutions are categorized into **open system solutions/products** and **commercial solutions/products**, highlighting their relevance to the AccessFit project.

2.1 Open System Solutions/Products

1. **MyFitApp Open API**

- **Overview**: MyFitApp provides an open API infrastructure for developers to create and customize fitness applications. It acts as a foundational tool for building scalable and feature-rich fitness systems.
- **Strengths**:
    - Offers flexible APIs for integration and customization.
    - Enables developers to tailor fitness solutions to specific requirements.
- **Weaknesses**:
    - It is not a standalone application and requires significant development effort to create a usable product.
- **Website**: https://www.myfitapp.com

2.2 Commercial Solutions/Products

1. **Maksisoft Gym**
    - **Overview**: A widely used gym management solution offering tools like membership tracking, payment handling, and attendance logging.
    - **Strengths**: Robust administrative tools; integration with turnstile access systems.
    - **Weaknesses**: Limited user engagement features like progress tracking or workout personalization.
    - **Website**: https://maksisoft.com
2. **MacFit App**
    - **Overview**: Mobile application for gym chain members to manage their memberships and access training programs.
    - **Strengths**: QR code-based entry and personalized plans; effective notifications and reminders.
    - **Weaknesses**: Exclusively tied to MacFit gym chains; not scalable to other environments.
    - **Website**: https://www.macfit.com

3. REQUIREMENTS

The requirements for the AccessFit project were primarily determined during the Fall semester as part of the COMP 4910 course. These requirements were initially documented in a Project Assignment Form (PAF), which included the project code, title, team information, and a summary of the proposed project and product. Following this, the Requirements Specification Document (RSD) v1.0 was created, detailing the initial functional and non-functional requirements in a structured textual format.

As the project evolved, the requirements were refined, and RSD v2.0 was developed. This version improved clarity, included UML diagrams for visualization, and was aligned with best practices. The final requirements for COMP 4910 are provided in Appendix A: Requirements Specification Document, v2.0.

Throughout the requirement engineering process, the team focused on refining system functionalities to ensure a seamless user experience. The primary effort was spent on structuring functional and non-functional requirements in a comprehensive manner, aligning with the project's objectives. The time spent on each RSD version was as follows:

RSD v1.0: Initial draft created based on project goals and preliminary discussions. Approximately 4 weeks were dedicated to its development.

RSD v2.0: Iterative refinement incorporating feedback, additional details, and UML diagrams for better visualization. This process took approximately 5 weeks.

Challenges Faced While Creating the RSD

Writing the RSD wasn't as simple as just listing requirements. We ran into several challenges along the way:

Keeping Everything Clear: Making sure the requirements were detailed enough for developers to understand but not so technical that they became overwhelming.

Aligning with Team and Advisor Feedback: Adjusting the document based on feedback while ensuring it stayed structured and logical.

Defining Functional vs. Non-Functional Requirements: Some requirements were easy to classify, but others required multiple discussions to categorize properly.

Creating Visual Representations: Adding UML diagrams was helpful but also time-consuming, as we needed to ensure they accurately represented the system.

Final Thoughts on the RSD

Overall, the RSD development process was a learning experience. The first version helped us structure our ideas, and with each iteration, we improved clarity and organization. While refining the document took time, it was worth the effort because it now serves as a solid reference for the entire project. Moving forward, minor adjustments may still be needed, but the core foundation is strong.

4. DESIGN

4.1. High Level Design

The AccessFit system is designed using a layered architecture with three main layers: Presentation Layer, Application Layer, and Data Access Layer. This structure helps keep the system modular, scalable, and easy to maintain. The Presentation Layer handles user interactions, the Application Layer processes business logic, and the Data Access Layer manages database operations.

This approach allows for future enhancements, such as IoT integration or advanced analytics, without disrupting the current setup. The full high-level design details, including UML diagrams, can be found in **Appendix B: Design Specifications Document (DSD v1.0)**.

Creating the DSD took a lot of effort, especially in organizing the design clearly. The estimated time spent on each version is:

- **DSD v1.0**: Initial draft and structuring **5 weeks**.

Challenges in Developing the DSD

While working on the DSD, we faced some key challenges:

- **Finding the Right Level of Detail**: It was tricky to include enough details without making the document too complex.
- **Defining API Interactions**: Ensuring smooth communication between frontend and backend components required careful planning.
- **Keeping UML Diagrams Consistent**: We had to make sure all diagrams matched the written explanations.
- **Security Considerations**: Making sure the design followed best practices for authentication, authorization, and data security.
- **Performance Planning**: Balancing scalability with fast response times.

Final Thoughts on the DSD

The final version of the DSD provides a solid foundation for implementation. It clearly explains the system architecture, component interactions, and technical details, making development smoother. The iterative approach helped us refine the document based on feedback. Moving forward, we may further detail integration points and strengthen security aspects.

4.2. Detailed Design

The detailed design will be developed during the Spring semester as part of COMP 4920. The plan includes:

- Expanding the high-level design into detailed specifications.
- Defining class structures, methods, and workflows using UML diagrams.
- Developing API specifications for frontend-backend communication.
- Setting security and performance benchmarks.

These details will be added to the **Design Specifications Document (DSD)** in later phases.

4.3. Realistic Restrictions and Conditions in the Design

While designing AccessFit, we had to keep certain limitations in mind:

1. **Scalability**: The system is initially built to support 1000 users but can be scaled up if needed.
2. **Security**: OAuth 2.0 and TLS encryption are used for protection, but multi-factor authentication is not yet included.
3. **Standards**: The design follows OWASP Mobile Security Standards for data integrity and privacy.

4. **Best Practices**: We used industry standards like RESTful APIs and Git for version control.
5. **Limitations**: The system doesn't yet support offline access or distributed file storage, but these could be future improvements.

Overall, the design process has helped shape a flexible and scalable system that meets the project's goals while allowing room for enhancements.

5. IMPLEMENTATION AND TESTS

5.1. Implementation of the Product

The implementation of AccessFit will utilize a combination of modern tools and technologies to ensure a scalable, secure, and user-friendly application. The following techniques and tools have been considered so far:

- **Frontend Development**: The user interface will be developed using **React Native**, providing cross-platform compatibility for both iOS and Android devices.
- **Backend Development**: The backend will be implemented using **Java** with the **Spring Boot** framework to handle API management, authentication, and data processing.
- **Database**: **PostgreSQL** will be used to store user data, workout plans, and activity logs, chosen for its reliability and scalability.
- **Barcode Integration**: The **ZXing (Zebra Crossing)** library will be used for generating and validating barcodes for gym entry.

5.2. Tests and Results of Tests

Testing for AccessFit will be planned to ensure the product is reliable, user-friendly, and meets its requirements. The following approaches have been considered so far:

- **Unit Testing**: Each module, including the barcode entry system, workout plan customization, and motivational features, will be tested individually to verify correct functionality.
- **Integration Testing**: The interaction between the frontend, backend, and database will be evaluated to ensure seamless communication.
- **System Testing**: The entire application will be tested in a controlled environment to simulate real-world usage scenarios.
- **User Acceptance Testing (UAT)**: Feedback from trial users will be gathered to validate the app's usability and effectiveness in meeting user needs.

6. PROJECT MANAGEMENT

6.1. Project Plan

The project plan for the COMP 4910 Fall semester encompasses the following key activities:

- **PAF Production**: Planning and preparation of the initial project assignment form.
- **Survey of Related Work**: Reviewing and analyzing existing gym management and fitness tracking solutions.

- **RSD 1.0 Production**: Drafting the initial version of the Requirements Specification Document.
- **RSD 2.0 Production**: Refining and finalizing the requirements in the updated RSD.
- **DSD 1.0 Production**: Developing the first version of the Design Specification Document, focusing on high-level design.
- **Final Report Production**: Compiling all project outcomes, evaluations, and reflections.
- **Project Management Activities**: Conducting regular team meetings, evaluating progress, and updating documentation as needed.
-

Details of the project plan, including task timelines, durations, and milestones, are outlined in **Appendix C1**.

For COMP 4920 Spring semester, the project plan includes:

- **Revision of RSD v2.0 and DSD v1.0**: Updating requirements and high-level design based on feedback and progress.
- **Development of DSD v2.0**: Expanding the high-level design into detailed specifications for implementation.
- **Implementation and Testing**: Completing system features, conducting tests, and iterating on results.
- **Project Management Activities**: Monitoring progress, resolving challenges, and ensuring timely deliverables.

6.2. Project Effort/Manpower

The total effort for the project is measured in **man-days** (one man-day equals 8 hours of effective work). Given the three-member team, the contributions of each member are critical. Effort distribution is summarized as follows:

- **Individual Contributions**: Each member's hours are tracked monthly and allocated to specific tasks.
- **Team Totals**: The combined effort is calculated monthly and across the entire project duration.
-

Detailed manpower allocation tables are included in **Appendices C2 and C3**.

6.3. Project Cost Analysis

The AccessFit project aims to minimize costs by relying solely on open-source tools and frameworks. The software stack includes:

- **React Native**: Open-source framework for cross-platform mobile app development.
- **Spring Boot**: Open-source backend framework for building APIs.
- **PostgreSQL**: Open-source relational database management system.
- **Postman Free Version**: For API testing and debugging.
- **Figma Free Plan**: For UI/UX prototyping.

Manpower Costs:
The primary cost component is team effort, calculated in **man-hours**. Assuming no external financial costs for tools or hardware, the manpower cost is represented as the effort invested by each team member.

Category	**Cost (TL)**
Software Tools	0
Manpower (690 hours)	-
Other Costs (Utilities, Internet)	500 TL
Total	**500 TL**

This approach ensures that the project remains cost-effective while leveraging the team's technical skills and freely available resources.

7. CONCLUSIONS

7.1. Summary

The AccessFit project aims to revolutionize gym experiences by addressing challenges such as maintaining motivation, tracking progress, and managing workout routines effectively. During the COMP 4910 semester, significant progress was made in laying the foundation for the project. The **Requirements Specification Document (RSD)** was developed in two iterations, evolving from an initial draft (RSD 1.0) to a refined version (RSD 2.0) with detailed requirements and UML representations. The **Design Specification Document (DSD)** was also created, providing a high-level architectural overview and identifying core components, including the barcode-based entry system, personalized workout planning, and motivational features.

Additionally, a comprehensive survey of existing solutions was conducted, highlighting the gaps in current systems and how AccessFit intends to address them. A project plan was established to guide development, and team responsibilities were clearly defined to ensure efficient collaboration. While the implementation and testing phases will occur in COMP 4920, the foundational work completed in COMP 4910 has provided a robust framework for the project's continued development. This progress ensures that AccessFit is well-positioned to deliver an innovative and user-centric solution in the fitness domain.

7.2. Benefits of the Project

AccessFit offers several benefits:

- **To Users**: Enhances gym experiences by streamlining entry, providing tailored workout plans, and delivering motivational tools to improve consistency and engagement.
- **To the Fitness Industry**: Simplifies gym management and user engagement, potentially increasing membership retention and satisfaction.
- **To Society**: Promotes physical health and well-being, encouraging individuals to adopt and maintain healthier lifestyles.
- **To Nature**: By replacing traditional paper-based records with digital solutions, AccessFit contributes to environmental sustainability by reducing paper usage.

7.3. Future Work

The future work for AccessFit can be categorized into the following areas:

1. **COMP 4920 Activities**:
    - **Detailed Design**: Expanding the high-level design into comprehensive detailed specifications, including class diagrams and API documentation.
    - **Implementation**: Developing a fully functional mobile application, integrating all planned features.
    - **Testing**: Conducting unit, integration, and user acceptance tests to ensure reliability and usability.
    -
2. **Potential Additions After COMP 4920**:
    - **IoT Integration**: Adding compatibility with wearable fitness devices (e.g., smartwatches) for real-time data synchronization.
    - **Advanced Analytics**: Introducing AI-powered features for personalized recommendations and progress predictions.
    - **Offline Mode**: Enabling users to access workout plans and log progress without an active internet connection.
    - **Gamification**: Expanding motivational features with leaderboards, challenges, and rewards systems.
    - **Environmental Impact**: Integrating features to track gym energy usage and promote eco-friendly practices.

AccessFit has significant potential for growth, both in terms of technical capabilities and its positive impact on users and the environment. The steps outlined will ensure its scalability and adaptability to evolving needs.

REFERENCES

1. React Native Documentation, Meta Platforms, Inc. [Online]. Available: https://reactnative.dev/
2. Spring Boot Documentation, VMware, Inc. [Online]. Available: https://spring.io/projects/spring-boot
3. Firebase Documentation, Google. [Online]. Available: https://firebase.google.com/docs
4. Postman API Testing Documentation, Postman Inc. [Online]. Available: https://www.postman.com/
5. OWASP Foundation, *Mobile Application Security Verification Standard (MASVS)*, OWASP. [Online]. Available: https://owasp.org/
6. AccessFit Requirements Specification Document (RSD), v2.0, 2024.
7. AccessFit Design Specification Document (DSD), v1.0, 2024.

APPENDICES

APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT

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

AccessFit: Enhancing Gym Management with QR Code-Based Access Control Requirements Specifications Document

12.06.2025
Revision 2.0

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

Revision History

Revision	Date	Explanation
0.1	04.11.2024	Initial Draft
0.2	06.11.2024	Supervisor Feedback
0.3	10.11.2024	Revised Draft
0.4	14.11.2024	Supervisor Feedback
1.0	17.11.2024	Final Draft
1.1	23.12.2024	Version 2.0 initial draft
1.2	03.01.2025	Supervisor Feedback
2.0	12.06.2025	Final Report

Contents

1. Introduction

This Requirements Specification Document (RSD) outlines the design and functionality of AccessFit, a mobile application developed to transform the gym experience for fitness enthusiasts. The application combines advanced technology with user-centric design to create a seamless, motivating, and personalized fitness journey. AccessFit integrates a secure and efficient barcode-based entry system alongside tools that help users track their workouts, set and achieve fitness goals, and stay motivated throughout their fitness journey. The primary goal of AccessFit is to offer gym-goers a more streamlined and engaging experience that enhances motivation, supports goal setting, and tracks progress effectively.

The fitness industry faces challenges when it comes to motivating users, providing personalized workout plans, and tracking performance over time. Many existing gym apps focus only on basic functions like membership management and scheduling, without offering the comprehensive, personalized features that modern gym-goers demand. AccessFit aims to fill this gap by not only facilitating quick and secure gym access through barcode scanning but also by offering customized workout routines, progress monitoring, and motivational support, all designed to keep users engaged and progressing.

With AccessFit, users can expect more than just an entry system; they will have access to a comprehensive fitness solution that adapts to their individual goals, helps them stay consistent, and offers the tools they need to see tangible results.

2. Overview of AccessFit Related Resources and Product

AccessFit stands out in a market already populated by similar fitness applications and gym management platforms. While other platforms, such as Maksisoft Gym and MacFit, provide basic features like scheduling and membership management, AccessFit differentiates itself by its focus on personalized workout plans and a unique barcode-based entry system, creating a more streamlined and integrated experience for gym-goers.

Maksisoft Gym and MacFit offer valuable functionalities, but they lack the depth in terms of personalized fitness features and motivational tools, which are central to AccessFit's approach. AccessFit places an emphasis on user engagement with achievement badges, progress reports, and reminders, which are designed to keep users motivated and focused on their fitness goals, beyond just logging attendance or booking classes.

In contrast to these broader fitness management tools, AccessFit's design focuses on providing users with a more personalized and engaging experience, ensuring they remain consistent and connected with their fitness routines.

2.1.Maksisoft Gym:

Maksisoft Gym is a comprehensive gym management software offering features for membership management, turnstile access, mobile app integration, online scheduling, reporting, and analytics. The membership management module simplifies registration, payments, and renewals, with notifications for expiring memberships. The turnstile access system controls entries through card or biometric authentication and records each entry. Reporting and analytics tools allow tracking of member check-ins, payments, and class data. The software also supports managing group classes and personalized training programs, as well as centralized management of multiple gym locations. Additionally, it includes a staff management module to track trainer schedules and workload.

2.2. MacFit:

MacFit offers a comprehensive mobile app experience for gym members, encompassing membership management, online booking, training programs, and nutrition support. Members can register, renew, and make payments directly through the app, which also enables QR code-based entry for streamlined check-ins and tracked access. The app's online booking system allows users to schedule classes, personal training sessions, and track their workout programs. For those seeking personalized guidance, custom and pre-designed workout plans are available with progress tracking, alongside tailored diet plans and fitness monitoring. Members can easily locate nearby gym locations, view operating hours, and check peak times. The app also provides notifications for workout reminders and membership renewals, along with in-app payment features, billing history, and invoices. Additionally, users have access to campaigns, discounts, and rewards, while performance tracking tools enable monitoring of fitness goals and progress through detailed data analysis.

3. Functional Requirements of AccessFit

3.1 Requirements List

Use Case No.	Use Case Name	Short Description
1	**Sign Up**	**Users register and create an account.**
2	**Create Username and Password**	**Users set up their credentials for the account.**
3	**Update Database**	**The system stores or updates user information.**
4	**Sign In**	**Users log in to the system by entering credentials.**
5	**View Profile**	**Users view their profile details.**
6	**Edit Profile**	**Users modify their profile information.**
7	**Scan Barcode**	**Users scan their barcode for system access.**
8	**Create Barcode**	**The system generates a unique barcode for the user.**
9	**Validate Access**	**The system validates the scanned barcode.**
10	**Grant Access**	**The system allows entry if validation is successful.**
11	**Training**	**Users create a workout plan and engage in training sessions.**

12	**Add Muscle Group**	**Users add muscle groups (e.g., arm, leg) to the workout plan.**
13	**Save Workout Plan**	**The system saves the completed workout plan.**
14	**Receive Workout Reminder**	**Users receive notifications about upcoming workouts.**
15	**Earn Achievement Badge**	**Users earn badges for achieving fitness goals.**
16	**View Motivation Feed**	**Users view a feed with motivational content.**
17	**Log Workout Data**	**Users log details of their workouts into the system.**
18	**View Progress Summary**	**Users view summarized reports of their fitness progress.**
19	**Access Detailed Analytics**	**Users view in-depth analytics about their fitness activities.**
20	**Store Workout Data**	**The system saves logged workout data for later use.**

3.2 Actors and Use Case Diagrams

User: User A gym member who uses the system to register, scan barcodes for entry, create workout plans, and track progress. System Admin Manages the backend of the application, including user data, barcode generation, and system maintenance.

System: The software that handles all operations like authentication, barcode validation, and progress analytics.

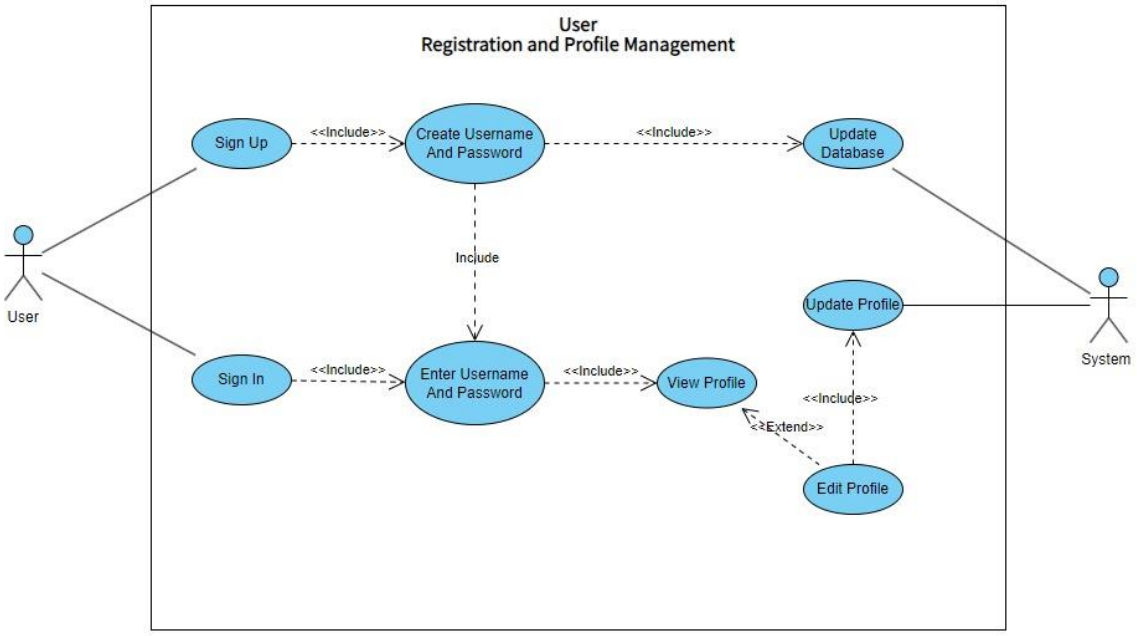
3.3 ACCESSFIT Use Case Diagram Groups

Subsystem/Package	Use Cases
User Registration and Profile Management	• Sign Up • Create Username and Password • Update Database • Sign In • View Profile • Edit Profile • Update Profile
Barcode-Based Entry System	• Scan Barcode • Create Barcode • Validate Access • Grant Access • Access Denied • No Existing Membership Warning • Repeated Entrance Attempt Warning • Keep Login Statistics

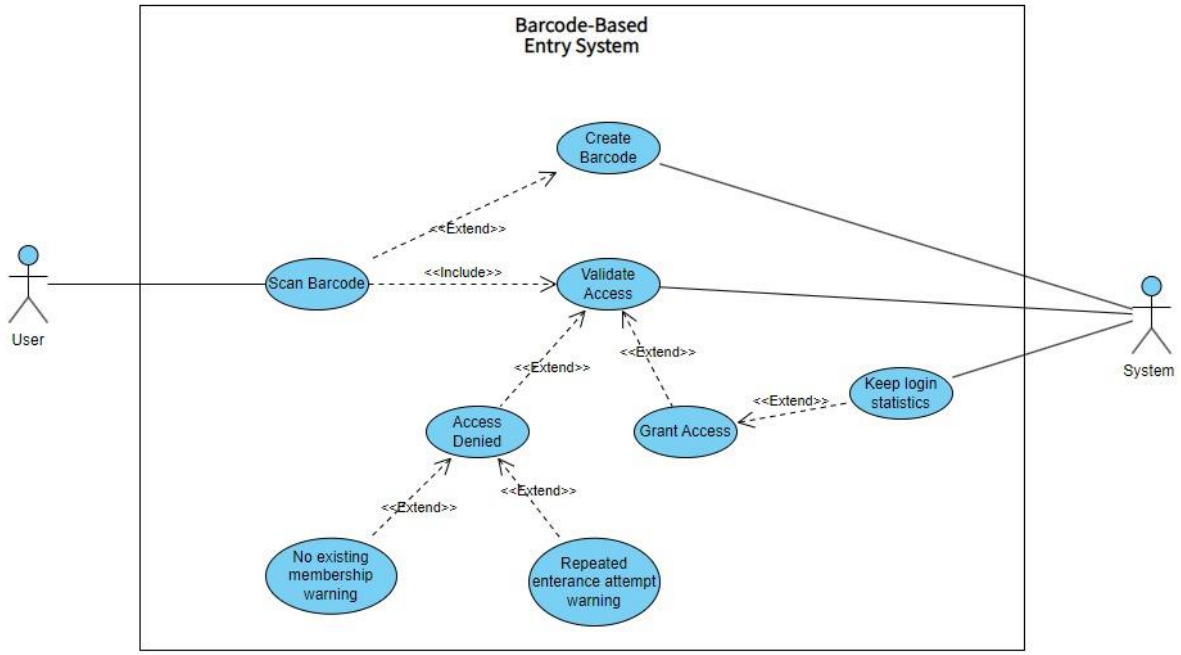
Workout Plan Customization	• Training • Create Empty Workout Plan • Add Muscle Group (Arm, Biceps, Triceps, Leg, Shoulder, Chest) • Complete Workout Plan • Save Workout Plan
Motivational Features	• Receive Workout Reminder • Earn Achievement Badge • View Motivation Feed • Send Workout Reminder • Track User Activity • Award Achievement Badge • Update Motivation Feed
Progress Tracking and Analytics	• Log Workout Data • View Progress Summary • Access Detailed Analytics • Store Workout Data • Generate Progress Summary • Perform Analytics

3.4 ACCESSFIT Use Case Diagrams

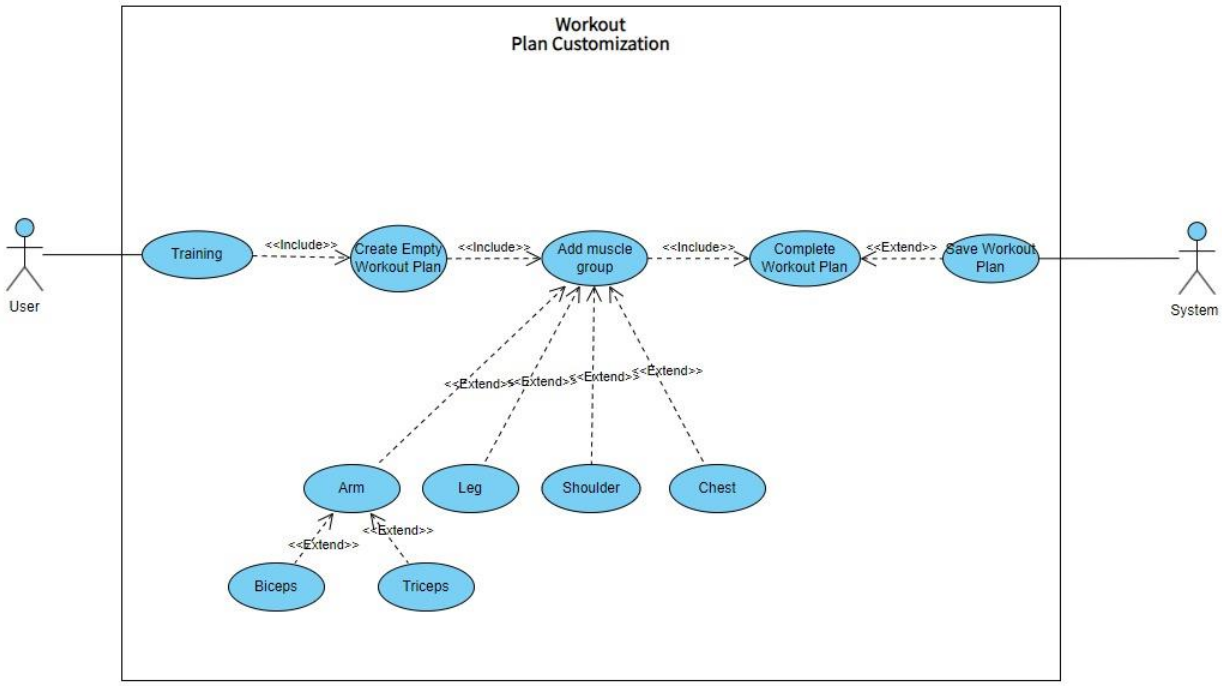
3.4.1 Use Case 1: User Registration and Profile Management



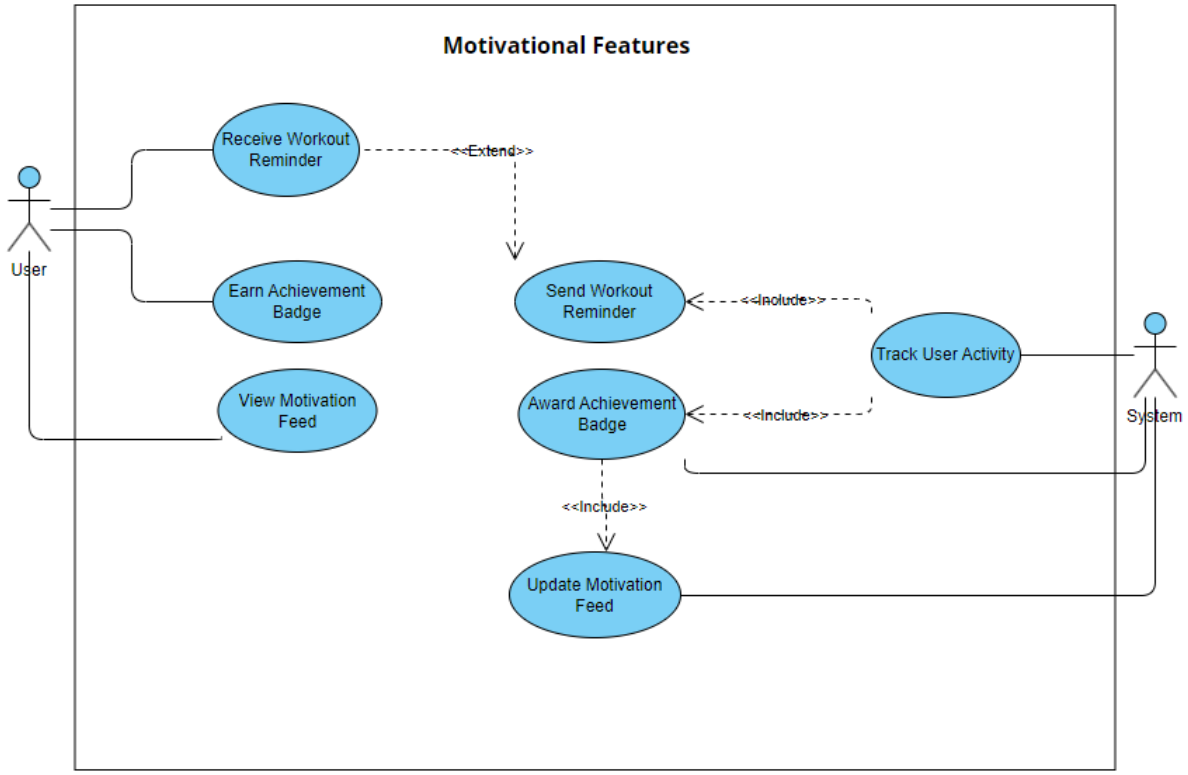
3.4.2 Use Case 2: Barcode-Based Entry System



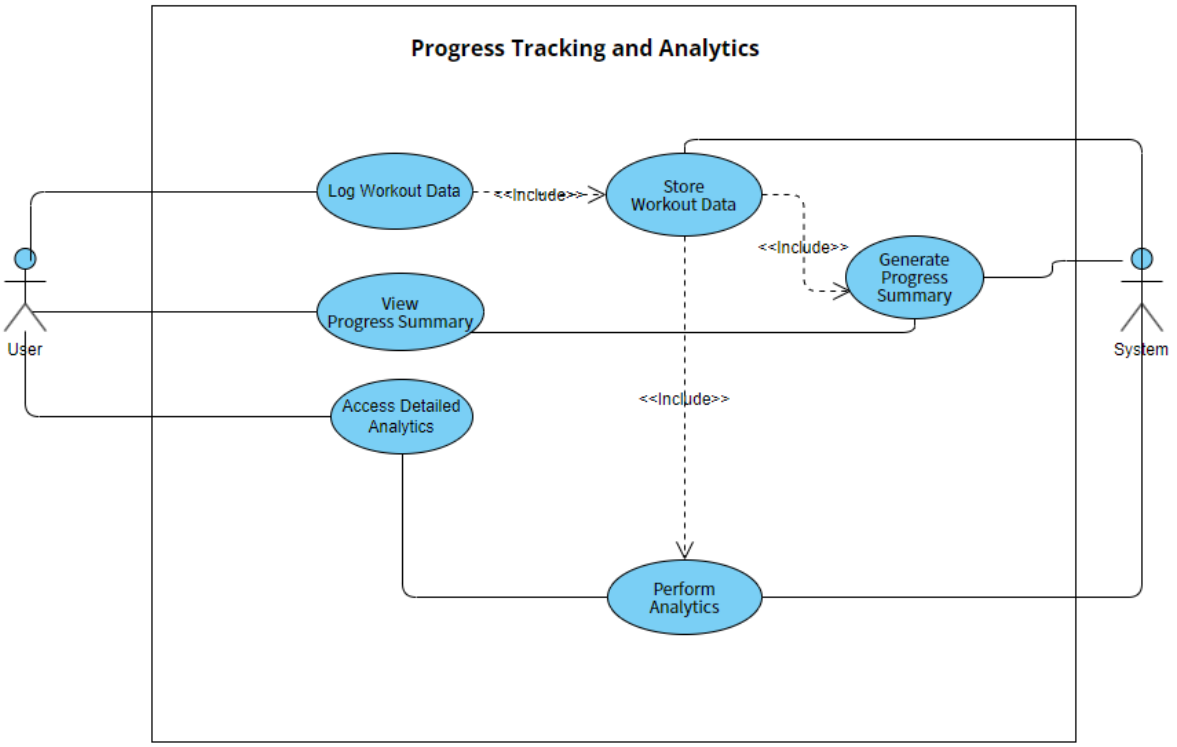
3.4.3 Use Case 3: Workout Plan Customization



3.4.4 Use Case 4: Moticational Features



3.4.5 Use Case 5: Progress Tracking and Analytics



4. AccessFit Use Case Descriptions

4.1 User Registration and Profile Management

Sign Up: The user registers to the system by creating an account.
Create Username and Password: The user sets up their username and password for the account.
Update Database: The system stores or updates user information in the database.
Sign In: The user logs in to the system by entering credentials.
Enter Username and Password: The user provides their username and password to authenticate.
View Profile: The user views their profile details stored in the system.
Edit Profile: The user modifies their profile information.
Update Profile: The system updates the user's profile details after editing.

4.2 Barcode-Based Entry System

Scan Barcode: The user scans their barcode for system access.
Create Barcode: The system generates a unique barcode for the user.
Validate Access: The system checks if the scanned barcode is valid.
Grant Access: The system allows the user to enter if validation is successful.
Access Denied: The system denies entry due to invalid barcode or other issues.
No Existing Membership Warning: The system notifies the user if they do not have an active membership.
Repeated Entrance Attempt Warning: The system alerts the user about repeated failed attempts.
Keep Login Statistics: The system records data about the user's access attempts.

4.3 Workout Plan Customization

Training: The user engages in a training session by creating a workout plan.
Create Empty Workout Plan: The user initializes a new workout plan without predefined exercises.
Add Muscle Group: The user adds specific muscle groups to the workout plan.
Arm: The user includes arm exercises in the workout plan.
Biceps: The user adds biceps-specific exercises to the arm category.
Triceps: The user adds triceps-specific exercises to the arm category.
Leg: The user includes leg exercises in the workout plan.
Shoulder: The user includes shoulder exercises in the workout plan.
Chest: The user includes chest exercises in the workout plan.
Complete Workout Plan: The user finalizes the workout plan with all selected exercises.
Save Workout Plan: The system saves the completed workout plan for future use.

4.4 Motivational Features

Receive Workout Reminder: The user gets notifications about upcoming workouts.
Earn Achievement Badge: The user earns badges for completing specific fitness goals or milestones.
View Motivation Feed: The user views a feed that displays motivational content and updates.
Send Workout Reminder: The system sends notifications to remind the user of their workouts.
Track User Activity: The system monitors the user's fitness activities and progress.
Award Achievement Badge: The system assigns badges to the user upon reaching milestones.
Update Motivation Feed: The system updates the motivation feed with the user's achievements or related content.

4.5 Progress Tracking and Analytics

Log Workout Data: The user inputs their workout details into the system.
View Progress Summary: The user views a summarized report of their fitness progress.
Access Detailed Analytics: The user accesses in-depth insights and analytics about their fitness activities.
Store Workout Data: The system saves the logged workout information for future use.
Generate Progress Summary: The system creates a summarized report based on the user's stored workout data.
Perform Analytics: The system processes the workout data to generate detailed analytics.

5. Non-Functional Requirements of AccessFit

Performance	-The app should ensure minimal latency during barcode scanning and entry validation.
Scalability	-The system should manage high user activity without slowing down. -The backend should be ready for future connections with new equipment or external services.
Usability	-The app interface should be intuitive, allowing new users to navigate easily without a steep learning curve. -Clear prompts and feedback should guide users through the barcode scanning and workout tracking processes.

Reliability	-The app should be consistently available with minimal downtime.
Compatibility	-The app should be compatible with major mobile operating systems (iOS and Android). It should also function across a range of devices, including smartphones and tablets.
Maintainability	-The code should be easy to understand and update, with clear documentation for future developers. -The app's design should be flexible, allowing changes or improvements to specific features without affecting the whole system.

6. Sample User Scenario

Persona: Berk, a 30-year-old office worker who is new to fitness.
Goals:

- Establish a consistent workout routine.
- Track progress and stay motivated over time.

6.1 Registration and Profile Management

Berk discovers AccessFit while searching for a fitness app to kickstart his gym journey. He downloads the app and completes the registration process by providing his email, creating a password, and filling out his profile.

- **Actions:**
  - Inputs his name, age, gender, and fitness goals (e.g., “build strength” and “increase consistency”).
  - Selects his current fitness level (beginner) and preferred workout schedule (e.g., Monday, Wednesday, Friday).
  - Updates his profile picture and personal preferences to make his experience personalized.
- **System Response:**
  - AccessFit saves Berk’s data securely in its database and sets up his account.
  - The app generates a unique barcode for Berk to use at the gym.

6.2 Barcode-Based Access Control

Excited to start his journey, Berk visits his gym for the first time after registering on AccessFit.

- **Actions:**
  - Opens the AccessFit app on his phone and navigates to the "My Barcode" section.
  - Scans the barcode at the gym's entry system.
- **System Response:**
  - The gym's scanner validates Berk's membership by checking the barcode against AccessFit's database.
  - After successful validation, Berk is granted access to the gym.
  - The system logs Berk's entry for future tracking.

6.3 Personalized Workout Plan Creation

Once inside the gym, Berk explores the app's workout planning feature to help him create a routine that suits his beginner level.

- **Actions:**
  - Navigates to the "Workout Plan" section and selects "Create Plan."
  - Adds exercises for different muscle groups, such as squats (legs), bench press (chest), and shoulder press.
  - Sets the number of sets and repetitions for each exercise.
  - Saves the plan for future use.
- **System Response:**
  - The app provides guidance for each exercise, including demonstration videos and tips.
  - Berk's workout plan is saved to the cloud, and he can access it anytime.
  - A notification reminds him of his first workout session based on his schedule.

6.4 Motivational Features

To keep Berk motivated and consistent, AccessFit sends him reminders and tracks his progress.

- **Actions:**
  - Receives a motivational notification the night before each workout, saying, "Don't forget your workout tomorrow! Stay consistent, Berk!"
  - After completing a week of workouts, earns an "Consistency Badge" for staying on track.
- **System Response:**
  - Tracks Berk's activities and adjusts motivational messages based on his consistency.
  - Updates the motivational feed with achievements and progress to keep Berk engaged.

6.5 Progress Tracking and Analytics

After four weeks of consistent workouts, Berk checks his progress on AccessFit.

- **Actions:**
  - Opens the "Progress" section to view his analytics.
  - Reviews detailed charts showing improvements in strength, consistency, and workout frequency.
  - Downloads a weekly progress summary to share with his gym trainer for feedback.

- **System Response:**
    - Provides an overview of his logged workouts, total weights lifted, and areas of improvement.
    - Suggests increasing the intensity of certain exercises based on his performance trends.

6.6 Long-Term Engagement

After seeing positive results and earning several achievement badges, Berk feels motivated to continue his fitness journey. AccessFit keeps him engaged by:

- Updating his workout plan to intermediate difficulty after eight weeks.
- Sending personalized reminders and offering new badges for milestones like "First 10 Gym Visits."
- Allowing Berk to interact with motivational content and compare his progress with his previous results.

7. References

1. *Maksisoft Teknoloji Ofisi – Akıllı otomasyon Sistemleri*. (n.d.). https://maksisoft.com

2. MacPlus. (n.d.). *MacPlus Mobile Features*. https://www.macfit.com/en/mac-plus-mobile/features

APPENDIX B: DESIGN SPECIFICATION DOCUMENT

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.

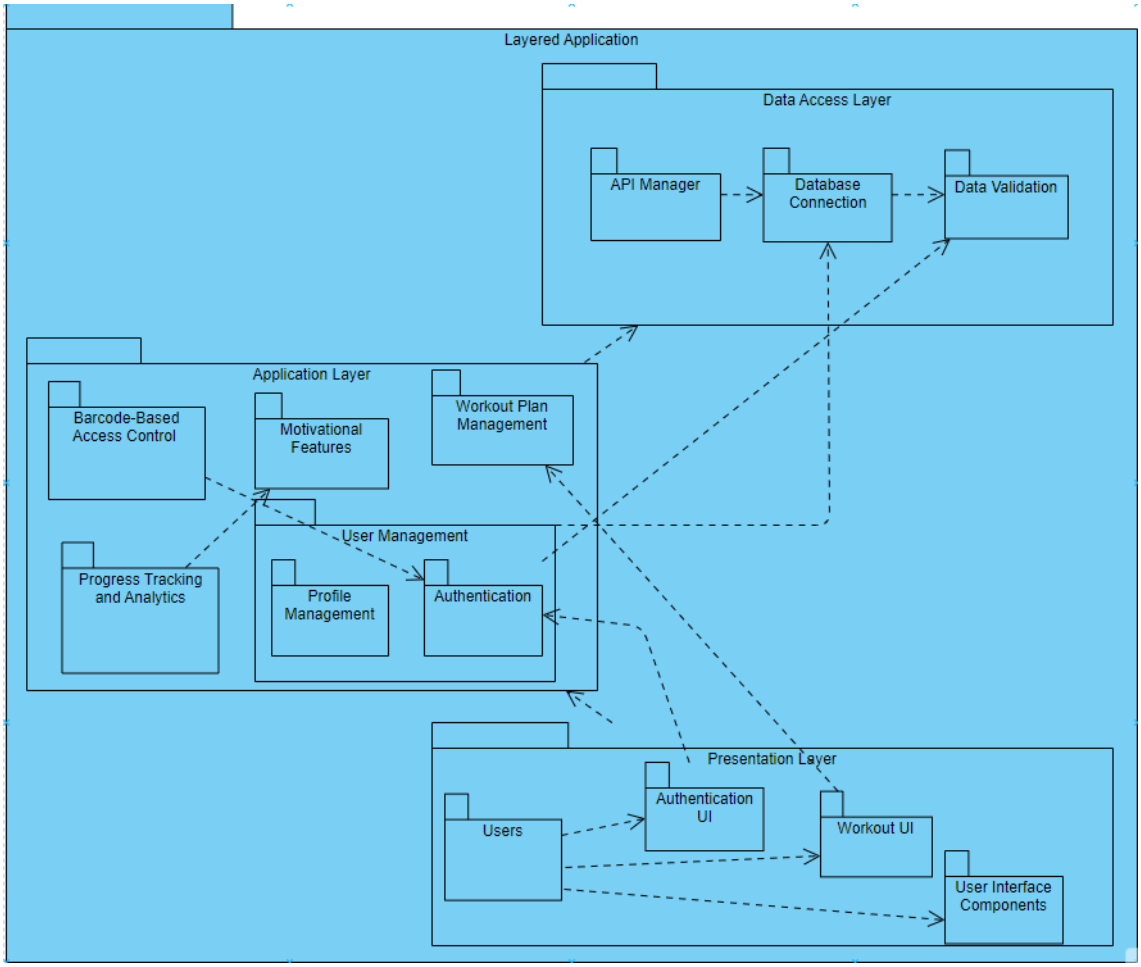
3. AccessFit Software Structure

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

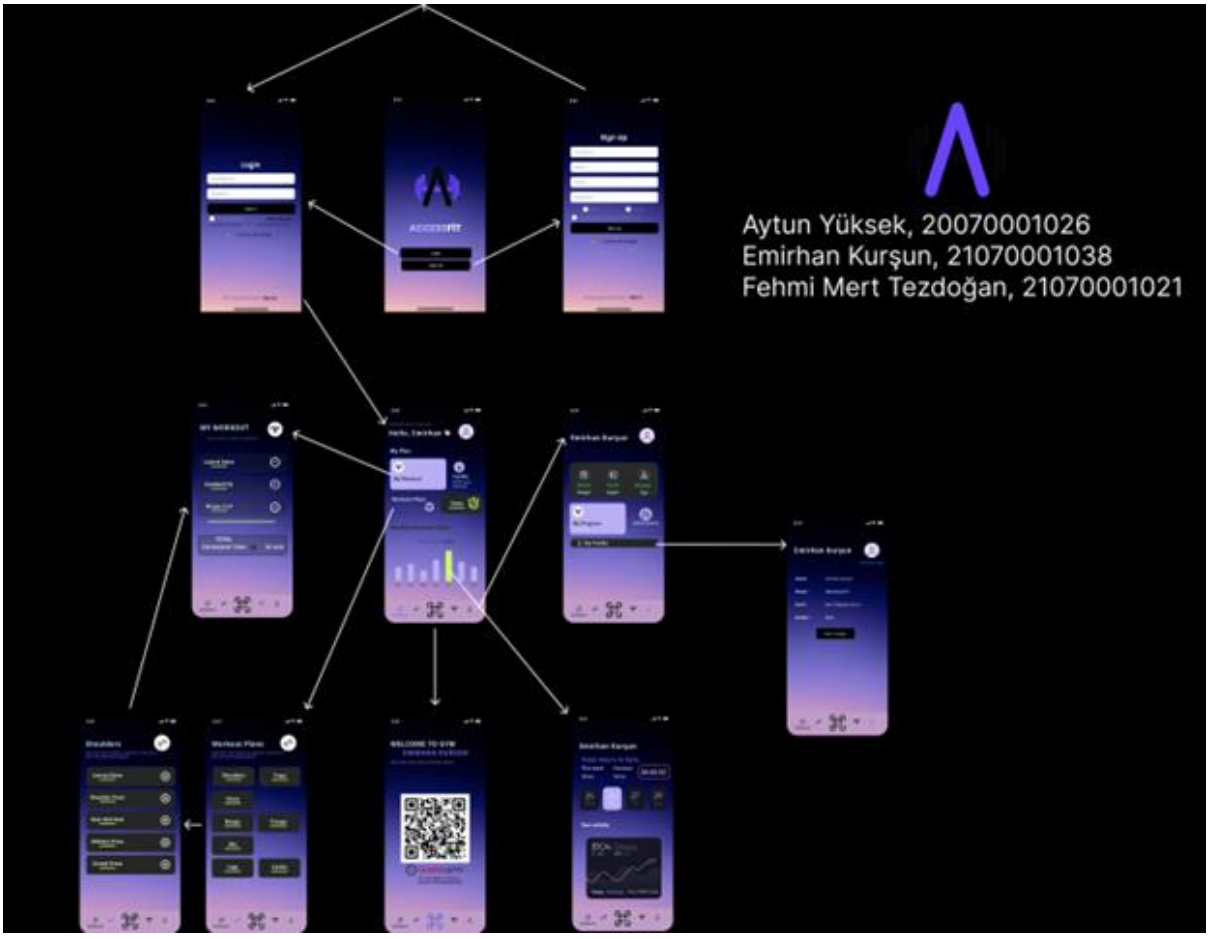
This layer is responsible for user interactions and the frontend interface. It is implemented using React Native, providing a cross-platform UI experience. The Presentation Layer does not contain business logic; instead, it communicates with the Application Layer via API requests.

- **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.

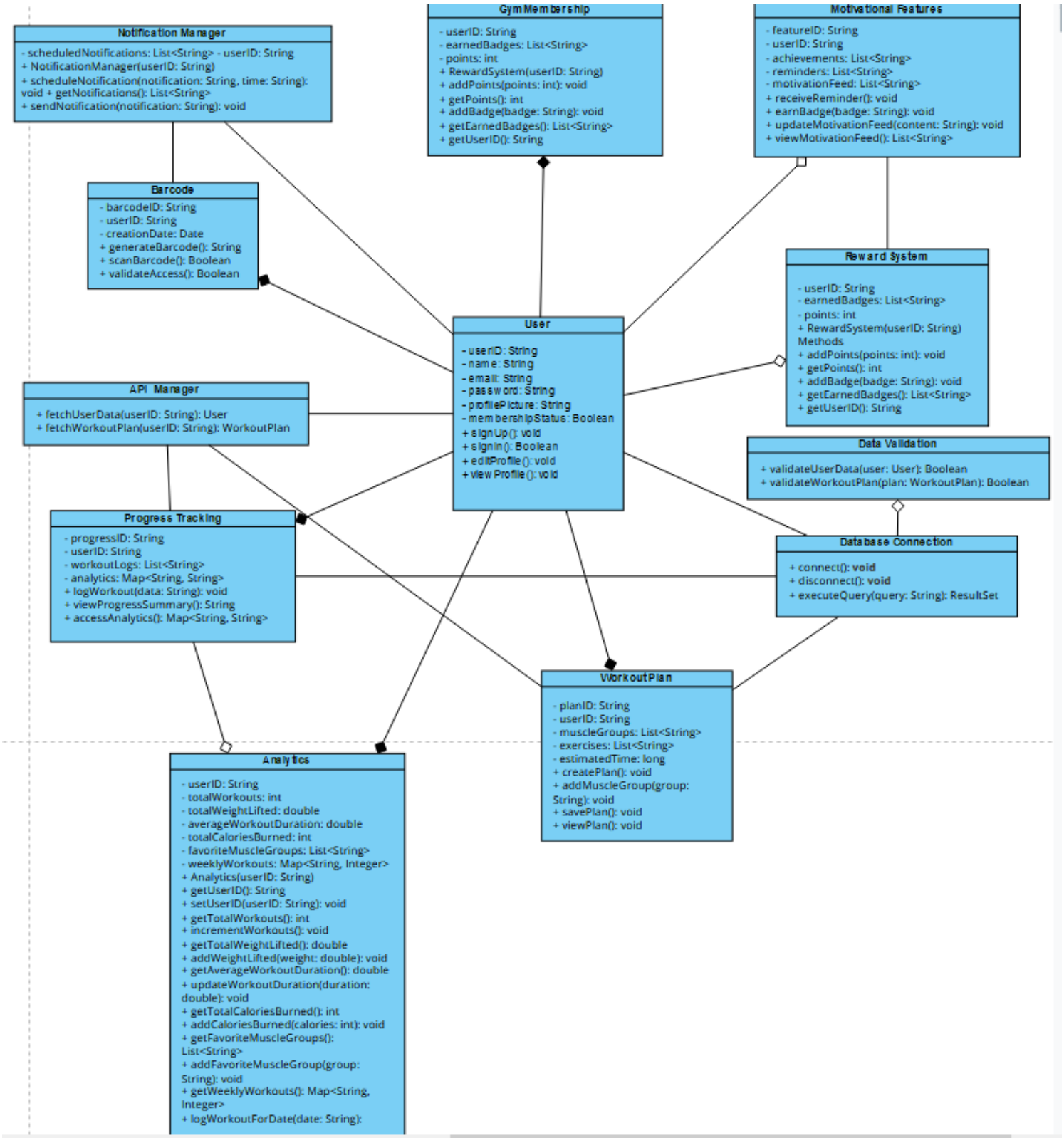
3. Security:

Data access and authentication are isolated, reducing security risks.

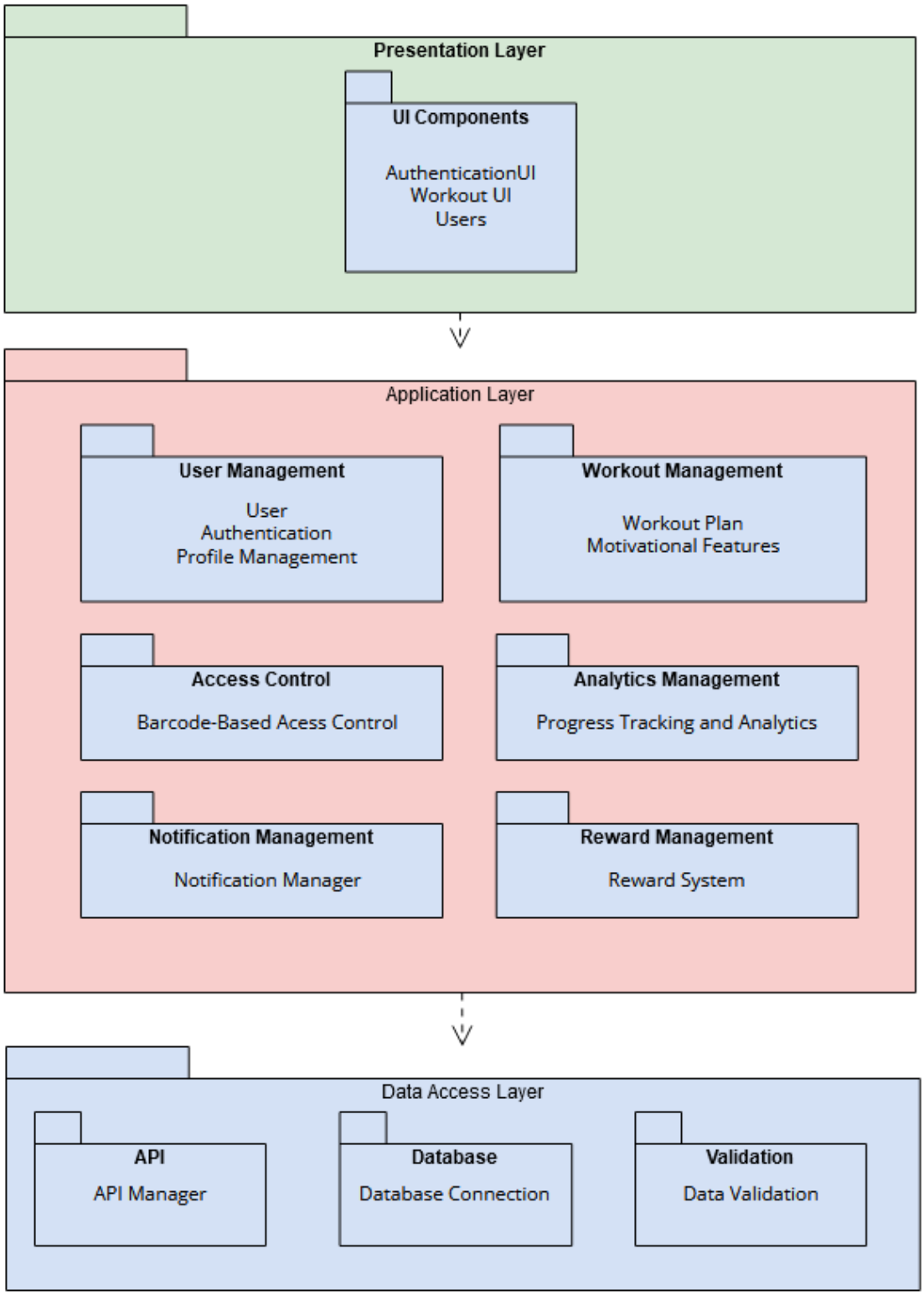
4. Easier Debugging and Testing:

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/

APPENDIX C: PRODUCT MANUAL

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

AccessFit: Enhancing Gym Management with QR Code-Based Access Control
Product Manual

Revision 1.0
12.06.2025

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

Revision History

Revision	Date	Explanation
0.8	23.05.2025	Initial Product Manual
0.9	12.06.2025	Updated Section 5.1 with SDK 53 compatibility, Expo Go auto-install note, and improved frontend setup instructions
1.0	15.06.2025	Added Spotify integration using OAuth 2.0 and implemented payment system via Stripe API. Corresponding updates made to Sections 3.1, 3.2, 4.5, and 5.2.

Table of Contents

1. Introduction

The purpose of this product manual is to document the implementation, testing, installation, and operation of the **AccessFit** application as a software product.
AccessFit is implemented and tested as it is described in the **AccessFit Design Specification Document (DSD), Revision 0.1**, satisfying the requirements defined in the **AccessFit Requirements Specification Document (RSD), Revision 1.2**.
Implementation, testing, and operation details are provided in the following sections of this document.

2. AccessFit Hardware Subsystem Implementation

This project does **not** involve the implementation of a custom hardware subsystem. Therefore, no hardware components have been developed or integrated beyond standard off-the-shelf devices (e.g., smartphones, tablets) on which the AccessFit software operates.
All hardware-related considerations such as mobile devices or server environments are addressed in the software subsystem implementation section (Section 3), where platform requirements for deployment and execution are described.

3. AccessFit Software Subsystem Implementation

3.1. Source Code and Executable Organization

The AccessFit application has been developed following a modular, layered software architecture, separating frontend and backend responsibilities for maintainability and scalability.
The **frontend** is implemented using React Native and is organized under the frontend/ directory. The main application entry point is App.js. This file handles navigation and high-level app logic, while individual features such as login, profile, barcode scanning, and workout tracking are encapsulated in separate screen components (e.g., LoginScreen.js, WorkoutPlanScreen.js). Shared components and utility modules are grouped under folders like components/ and services/.
The **backend** is developed using Spring Boot, located in the backend/ directory. The entry point is the AccessFitApplication.java file. The backend is structured into conventional layers: controller/ for REST APIs, service/ for business logic, repository/ for data access, and model/ for entity classes. Each of these packages corresponds to core functionalities such as user authentication, workout plan management, analytics, and notifications.
Build outputs are structured as follows:

- The frontend generates platform-specific executables: an .apk file for Android and an .app bundle for iOS (built via Xcode).
- The backend compiles into a standalone .jar file named accessfit-backend.jar using Maven.

Configuration is managed through separate files:

- In the frontend, a .env file defines API endpoints and Firebase credentials.
- On the backend, application.properties includes settings for database access (Firestore), Firebase Cloud Messaging integration, port configuration, and security tokens.

To support automation and deployment, several shell scripts have been written:

- install.sh installs all required dependencies.
- test.sh runs unit and integration tests.
- deploy.sh handles backend deployment to AWS Elastic Beanstalk.

- start-backend.sh starts the backend server locally for development.

There were minor deviations from the initial Design Specification Document (DSD). For example, Firebase email verification was later integrated into the user authentication flow, and workout analytics functionality was refactored into a more isolated service for future extensibility.

The full implementation — including source code, executables, configuration files, test cases, and logs — has been packaged as a single compressed archive named AccessFit-Product-2025-05-23-Rev-1.0.zip. This archive is organized in a self-explanatory directory structure and will be submitted via the COMP4920 platform. If required, it will also be delivered on a clearly labeled CD/DVD as Appendix D to the printed final report.

Additionally, Spotify Web API integration was implemented to allow users to link their Spotify accounts and play workout music directly within the app. The integration uses OAuth 2.0 for authentication and secure token handling. Furthermore, a payment system was added using the Stripe API, enabling users to manage premium subscriptions securely. All credentials and keys are managed via .env files and securely stored using platform-specific secure storage modules.

3.2. Software DevelopmentTools

AccessFit was developed using a carefully selected set of tools and platforms that enabled cross-platform mobile development and seamless integration with cloud-based backend services. Below is an overview of the primary tools and technologies used throughout the project lifecycle.

Frontend Development

The mobile application was built using **React Native (v0.72)**, allowing a single codebase to run on both Android and iOS platforms. Development was carried out in **Visual Studio Code (v1.84)**, which provided flexibility through React Native and Firebase extensions. For streamlined development, **Expo CLI (v7.6.3)** was used to preview and deploy the application during prototyping and testing phases.

Backend Services and Database

No custom backend or server-side API was developed. Instead, all backend functionalities were handled using **Firebase (Console version 2025)**.

- **Firebase Authentication** was used for secure user registration and login.
- **Cloud Firestore**, a scalable NoSQL database, stored user profiles, workout plans, and progress data.
- **Firebase Cloud Messaging (FCM)** allowed push notifications to be sent to users for workout reminders and motivational messages.

Version Control and Collaboration

Project source code was managed using **Git (v2.42)**. The remote repository was hosted on **GitHub**, which facilitated version control, collaboration, and tracking of development tasks across the team.

Testing and Debugging

Testing for frontend components was conducted using **Jest (v29.6)**. Additionally, the **Firebase Emulator Suite** was utilized during development to simulate Firestore operations and user authentication flows, ensuring safe and isolated testing without affecting live data.

UI/UX Design
Before implementation, **Figma** was used to create wireframes and design prototypes. This allowed the team to agree on visual and interactive elements before writing code, improving consistency and reducing rework.
All tools used in this project were either open-source or provided under educational licenses, making them both accessible and suitable for an academic software development environment.

External APIs and SDKs
- **Spotify Web API**: Used for music integration. Authentication handled via OAuth 2.0.
- **Stripe SDK**: Used for implementing in-app payments and premium subscription management.

3.3. Hardware and System Software Platform

- Computer hardware (processor, memory, harddisk, display, etc)
- Network (Internet connection, bandwidth, etc)
- System software, such as operating system, its version, any special OS services required
- Any other middleware or database or third part software: Brand name, version, etc.

on which your software is implemented and tested and also is expected to operate

4. AccessFit Hardware and Software Subsystem Testing

Since AccessFit does not include a dedicated hardware subsystem, this section focuses solely on **software subsystem testing**.

4.1 Module/Object-Method Test Cases

Individual modules and object methods were tested using both **automated unit tests** and **manual validation**:

- **User Authentication Module:**
  - signUp(), signIn() methods were tested with various input scenarios (valid, missing fields, invalid email format).
  - Result: Passed in all intended scenarios.
- **Barcode Access Module:**
  - generateBarcode(), validateAccess() methods were tested for uniqueness, integrity, and correct validation logic.
  - Result: Passed except under rare network timeouts (handled with retry logic).
- **Workout Plan Module:**
  - createPlan(), addMuscleGroup(), savePlan() methods were tested to confirm data consistency.
  - Result: All test cases passed.
- **Analytics Module:**
  - logWorkout(), viewProgressSummary() functions tested with actual user flows.
  - Result: Some visual chart rendering lag noticed under slow network.

- **Notification Module:**
    - sendNotification(), scheduleNotification() methods tested through Firebase Emulator Suite.
    - Result: Passed on both Android and iOS platforms.

4.2 System Integration Test Cases

Integration tests validated the correct functioning of the entire stack:

- **User Flow Test:** Register → Create Plan → Scan Barcode → Track Workout → View Analytics.
- **Backend-Frontend Sync:** Ensured that data submitted via the app (e.g., user workouts, profile updates) is correctly written to and retrieved from Firebase Realtime Database and Cloud Firestore. Synchronization across sessions and devices was validated.
- **Push Notification Pipeline:** Validated end-to-end flow from scheduled events to user device notification.

All core workflows were tested across real Android devices and iOS simulators using Firebase Test Lab and Expo Go.

4.3 Test Case Logs

- **Unit Tests:** Logged via Jest and JUnit, outputs stored in GitHub Actions artifacts.
- **Integration Tests:** Manual results recorded in a shared Google Sheet with timestamps and status.
- **Device Compatibility:** Logs from Firebase Test Lab for both Android API 29–34 and iOS 13–17 devices.
- **API Logs:** Captured using Postman for REST endpoint verification.

4.4 Testing Effort Summary

Resource Type	Value
Total Testing Time	~45 hours
Tools Used	Jest, JUnit, Postman, Firebase Test Lab
Devices	2 Android phones, 1 iOS emulator
Team Members Involved	3

4.5 Software Status Overview

Functionality	Implemented	Tested	Operational Status
User Authentication	Yes	Yes	Fully functional
Barcode Generation and Validation	Yes	Yes	Fully functional
Workout Plan Management	Yes	Yes	Fully functional
Progress Tracking and Analytics	Yes	Yes	Mostly functional (UI lag under load)
Notification Scheduling	Yes	Yes	Fully functional
Admin Panel (for gym staff)	No	No	Not implemented (future work)
Wearable Device Integration (e.g., smartwatches)	No	No	Not implemented (future work)
Offline Mode	No	No	Not implemented (future work)
Spotify Music Integration	Yes	Yes	Fully functional
Payment & Subscription System	Yes	Yes	Fully functional (Stripe test)

5. AccessFit Installation, Configuration and Operation

This section outlines how to install, configure, and operate the AccessFit system.

5.1 Installation and Configuration

Requirements (Developer/Admin)

- Node.js (v18+)
- npm (comes with Node.js)
- Expo CLI (SDK 53)
- Android Studio with Virtual Device (e.g., Pixel_4_XL) and SDK Manager configured
- Firebase Console for project configuration (Auth, Firestore, Messaging)
- ZXing library for barcode scanning (bundled with the app)

Frontend Setup

1. Install Node.js

- Download and install from: https://nodejs.org

2. Install Expo CLI

• npm install -g expo-cli

3. Clone the Project Repository

• git clone https://github.com/<your-team>/AccessFit.git
• cd AccessFit

4. Install Project Dependencies

• npm install

5. Configure Firebase

- Go to the Firebase Console: https://console.firebase.google.com
- Create a new Firebase project
- Enable the following services:
  - Authentication
  - Cloud Firestore
  - Cloud Messaging
- Download google-services.json
- Place it into the following directory: AccessFit/android/app/

6. Run the App

• npx expo start

- Press w to open the app in the web browser
- Or manually open **Expo Go** on your emulator or phone and scan the QR code

Emulator Setup

1. **Open Android Studio**
   - Use Device Manager to create and launch a Virtual Device (e.g., Pixel_4_XL)
   - Ensure the latest Android SDKs are installed using SDK Manager

2. **Expo Go Installation**
   - No manual APK installation is required
   - Expo CLI will handle automatic installation/update of Expo Go when the app is opened from the CLI

Required Expertise

- Understanding of Firebase setup
- Basic familiarity with npm and Android Studio
- React Native and Expo project structure and configuration

5.2 Operation Instructions (User Manual)

1. **Login/Signup:** Users can register and log in using email and password.
2. **Barcode Access:** Upon login, a personal barcode is displayed for gym entry.
3. **Workout Dashboard:** Shows current plans, allows customization.
4. **Analytics:** Tracks weekly progress and achievements.
5. **Reminders & Motivation Feed:** Sends motivational messages before workouts.
6. **Spotify Integration**: Users can connect their Spotify account from the app settings to access personal playlists during workouts. A valid premium Spotify account is required.
7. **Subscription Management**: Users can upgrade to premium plans within the app. Payments are securely handled via Stripe.

5.3 Error Codes and Messages

Code	Message	Explanation
401	"Invalid Credentials"	User entered wrong email/password
403	"Access Denied: Inactive Membership"	User's gym subscription expired
500	"Server Error. Try Again Later"	Backend processing failure
408	"Request Timeout"	Network delay or backend unresponsive
200	"Access Granted"	Valid QR code and active session

References

1. AccessFit Project Team. *AccessFit Requirements Specification Document (RSD)*, Revision 1.2, 03.01.2025.
2. AccessFit Project Team. *AccessFit Design Specification Document (DSD)*, Revision 0.1, 04.01.2025.
3. ISO/IEC 25010:2023. *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models*. International Organization for Standardization, 2023.
4. IEEE 29148-2023. *Systems and Software Engineering – Life Cycle Processes – Requirements Engineering*. IEEE Standards Association, 2023.
5. Sommerville, I. (2023). *Software Engineering* (11th ed.). Pearson Education.

APPENDIX D: PROJECT MANAGEMENT DOCUMENTS

APPENDIX D1: PROJECT PLAN

COMP 4910 : AccessFit, Project Plan, 03.01.2025, v1.0																
Task No	**Task Name**	**Weeks**														**Any Additional Notes**
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
1	Project Planning and PAF	X	X	X													
2	RSD 1.0				X	X	X	X									
3	RSD 2.0								X	X	X	X	X				
4	DSD 1.0											X	X	X	X	X	
5	Final Report													X	X	X	
6	Interface Implementation											X	X	X			

APPENDIX D2: PROJECT EFFORT LOG- CONSOLIDATED

COMP 4910 Project Effort Log, Project Code: ACCESSFIT, 21.01.2025, v1.0								
Week	**Dates**	**Team Member 1**		**Team Member 2**		**Team Member 3**		**Total Weekly Effort in Man-Hours**
		Work Done	**Total Hours Spent**	**Work Done**	**Total Hours Spent**	**Work Done**	**Total Hours Spent**	
Week 1		Project Planning and PAF	13	Project Planning and PAF	13	Project Planning and PAF	13	39.00
Week 2		Project Planning and PAF	13	Project Planning and PAF	13	Project Planning and PAF	13	39.00
Week 3		Project Planning and PAF	13	Project Planning and PAF	13	Project Planning and PAF	13	39.00
Week 4		RSD 1.0	13	RSD 1.0	13	RSD 1.0	13	39.00
Week 5		RSD 1.0	13	RSD 1.0	13	RSD 1.0	13	39.00
Week 6		RSD 1.0	13	RSD 1.0	13	RSD 1.0	13	39.00
Week 7		RSD 1.0	13	RSD 1.0	13	RSD 1.0	13	39.00
Week 8		RSD 1.0	13	RSD 1.0	13	RSD 1.0	13	39.00
Week 9		RSD 2.0	13	RSD 1.0	13	RSD 1.0	13	39.00
Week 10		DSD 1.0	13	DSD 1.0	13	DSD 1.0	13	39.00
Week 11		DSD 1.0	13	DSD 1.0	13	DSD 1.0	13	39.00
Week 12		Interface Imple-mentation	13	Interface Imple-mentation	13	Interface Imple-mentation	13	39.00
Week 13		Final Report	13	Final Report	13	Final Report	13	39.00
Week 14		Final Report	13	Final Report	13	Final Report	13	51.00
Total Effort in Man-Hours			**186.00**		**186.00**		**186.00**	**558.00**
Total Effort in Man-Days			**7.75**		**7.75**		**7.75**	**23.25**
Notes, if any								

APPENDIX D3: PROJECT EFFORT LOGS FOR EACH TEAM MEMBER-

COMP 4910/4920 Project Effort Log, Project Code: ACCESSFIT, 21.01.2025, v1.0
Team Member: Aytun Yüksek

Week	Dates	Work Done in Some Detail	Total Hours Spent
Week 1		Researching for PAF to make decisions	13
Week 2		Researching for PAF to make decisions	13
Week 3		Working on RSD 1.0 with team members	13
Week 4		Working on RSD 1.0 with team members	13
Week 5		Working on RSD 1.0 with team members	13
Week 6		Working on RSD 1.0 with team members	13
Week 7		Working on RSD 2.0 with team members	13
Week 8		Working on RSD 2.0 with team members	13
Week 9		Working on RSD 2.0 with team members	13
Week 10		Working on DSD 1.0 with team members	13
Week 11		Working on DSD 1.0 with team members	13
Week 12		Working on UI Implementation	13
Week 13		Working on Final Report with team members.	13
Week 14		Working on Final Report with team members.	17
Total Effort in Man-Hours			**186.00**
Total Effort in Man-Days			**23.25**

Notes:

1. This table shows the team member project effort. One Man-Day is Eight Man-Hours.
2. Each team member must fill out the form periodically (preferably at the end of the week of any work done).
3. Each filled-out table must be emailed at the end of each month to a selected Project Member (cc to Project Advisor), who will produce a consolidated table.

COMP 4910/4920 Project Effort Log, Project Code: ACCESSFIT, 21.01.2025, v1.0
Team Member: Emirhan Kurşun

Week	Dates	Work Done in Some Detail	Total Hours Spent
Week 1		Researching for PAF to make decisions	13
Week 2		Researching for PAF to make decisions	13
Week 3		Working on RSD 1.0 with team members	13
Week 4		Working on RSD 1.0 with team members	13
Week 5		Working on RSD 1.0 with team members	13
Week 6		Working on RSD 1.0 with team members	13
Week 7		Working on RSD 2.0 with team members	13
Week 8		Working on RSD 2.0 with team members	13
Week 9		Working on RSD 2.0 with team members	13
Week 10		Working on DSD 1.0 with team members	13
Week 11		Working on DSD 1.0 with team members	13
Week 12		Working on UI Implementation	13
Week 13		Working on Final Report with team members.	13
Week 14		Working on Final Report with team members.	17
Total Effort in Man-Hours			**186.00**
Total Effort in Man-Days			**23.25**

Notes:

1. This table shows the team member project effort. One Man-Day is Eight Man-Hours.
2. Each team member must fill out the form periodically (preferably at the end of the week of any work done).
3. Each filled-out table must be emailed at the end of each month to a selected Project Member (cc to Project Advisor), who will produce a consolidated table.

COMP 4910/4920 Project Effort Log, Project Code: ACCESSFIT, 21.01.2025, v1.0
Team Member: Fehmi Mert Tezdoğan

Week	Dates	Work Done in Some Detail	Total Hours Spent
Week 1		Researching for PAF to make decisions	13
Week 2		Researching for PAF to make decisions	13
Week 3		Working on RSD 1.0 with team members	13
Week 4		Working on RSD 1.0 with team members	13
Week 5		Working on RSD 1.0 with team members	13
Week 6		Working on RSD 1.0 with team members	13
Week 7		Working on RSD 2.0 with team members	13
Week 8		Working on RSD 2.0 with team members	13
Week 9		Working on RSD 2.0 with team members	13
Week 10		Working on DSD 1.0 with team members	13
Week 11		Working on DSD 1.0 with team members	13
Week 12		Working on UI Implementation	13
Week 13		Working on Final Report with team members.	13
Week 14		Working on Final Report with team members.	17
Total Effort in Man-Hours			**186.00**
Total Effort in Man-Days			**23.25**

Notes:

1. This table shows the team member project effort. One Man-Day is Eight Man-Hours.
2. Each team member must fill out the form periodically (preferably at the end of the week of any work done).
3. Each filled-out table must be emailed at the end of each month to a selected Project Member (cc to Project Advisor), who will produce a consolidated table.