

# Virtual Fitness Trainer

## Software Design Specification

Version 2.0

03/06/2025

Group 13

Ines Ling Delgado Dominguez  
Silvia Perez Servando

Prepared for  
CS 250- Introduction to Software Systems  
Instructor: Gus Hanna, Ph.D.  
Spring 2025

## Revision History

Date	Description	Author	Comments
<02/20/25>	<Version 1.0>	Group-13	<SRS>
<03/06/25>	<Version 2.0>	Group-13	<SDD>

## Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Group 13	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

# Table of Contents

<b>REVISION HISTORY .....</b>	<b>II</b>
<b>DOCUMENT APPROVAL .....</b>	<b>II</b>
<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 PURPOSE .....	1
1.2 SCOPE .....	1
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS .....	1
1.4 REFERENCES .....	1
1.5 OVERVIEW .....	2
<b>2. GENERAL DESCRIPTION .....</b>	<b>2</b>
2.1 PRODUCT PERSPECTIVE .....	2
2.2 PRODUCT FUNCTIONS .....	2
2.3 USER CHARACTERISTICS .....	3
2.4 GENERAL CONSTRAINTS .....	3
2.5 ASSUMPTIONS AND DEPENDENCIES .....	4
<b>3. SPECIFIC REQUIREMENTS .....</b>	<b>4</b>
3.1 EXTERNAL INTERFACE REQUIREMENTS .....	4
3.1.1 User Interfaces .....	4
3.1.2 Hardware Interfaces .....	4
3.1.3 Software Interfaces .....	4
3.1.4 Communications Interfaces .....	5
3.2 FUNCTIONAL REQUIREMENTS .....	5
3.2.1 User registration and profile management .....	5
3.2.2 Workout plan generation .....	5
3.2.3 Exercise library .....	5
3.2.4 Real time tracking and feedback .....	5
3.2.5 Progress tracking and analytics .....	5
3.2.6 Social features .....	5
3.2.7 Notifications and reminder .....	6
3.2.8 Integration with external devices .....	6
3.2.9 Security and privacy .....	6
3.3 USE CASES .....	6
3.3.1 User registration .....	6
3.3.2 Update profile information .....	7
3.3.3 Create workout plan .....	8
3.4 CLASSES / OBJECTS .....	9
3.5 NON-FUNCTIONAL REQUIREMENTS .....	9
3.5.1 Performance .....	9
3.5.2 Reliability .....	9
3.5.3 Availability .....	9
3.5.4 Security .....	9
3.5.5 Maintainability .....	9
3.5.6 Portability .....	9
3.6 INVERSE REQUIREMENTS .....	10
<b>4. SYSTEM DESCRIPTION .....</b>	<b>11</b>
4.1 OVERVIEW .....	11
<b>5. SOFTWARE ARCHITECTURE .....</b>	<b>11</b>
5.1 ARCHITECTURAL DIAGRAM .....	11

5.2 UML CLASS DIAGRAM .....	13
5.3 CLASSES, ATTRIBUTES AND FUNCTIONS .....	14
5.3.1 User .....	14
5.3.2 WorkoutPlan .....	14
5.3.3 Exercise .....	14
5.3.4 ProgressTracker .....	15
5.3.5 Notification .....	15
5.3.6 Challenge .....	16
5.3.7 WorkoutSession .....	16
<b>6. DEVELOPMENT PLAN AND TIMELINE .....</b>	<b>16</b>
6.1 PARTITIONING OF TASKS .....	16
6.1.1 Phase 1: Requirement Gathering and Analysis (2 weeks) .....	17
6.1.2 Phase 2: System Design (4 weeks) .....	17
6.1.3 Phase 3: Implementation (6 weeks) .....	17
6.1.4 Phase 4: Testing (3 weeks) .....	17
6.1.5 Phase 5: Deployment and Maintenance (2 weeks) .....	18
6.2 TEAM MEMBER RESPONSIBILITIES .....	18
6.2.1 Project Manager (PM) .....	18
6.2.2 Business Analyst (BA) .....	18
6.2.3 System Architect (SA) .....	18
6.2.4 Frontend Developer (FD) .....	18
6.2.5 Backend Developer (BD) .....	19
6.2.6 Database Administrator (DBA) .....	19
6.2.7 AI Developer (AID) .....	19
6.2.8 UI/UX Designer (UXD) .....	19
6.2.9 Data Scientist (DS) .....	19
6.2.10 QA Engineer (QA) .....	19
6.2.11 DevOps Engineer (DO) .....	19

## 1. Introduction

The Software Requirements Specification is critical introductory documentation that gives a good overview of the complete SRS having purpose, scope, definitions, acronyms, abbreviations, references, and the overview of SRS. It documents what is to be captured and analyzed deeply to provide insight on the total ongoing software work of Virtual Fitness Trainer, including intensive problem definition. The document, on the other hand, informs about the capabilities needed by the stakeholders and the need for these capabilities while stating high-level features of the product. This document has contained well-detailed requirements for Virtual Fitness Trainer software.

### 1.1 Purpose

The aim of this Software Requirements Specification document is to define the detailed functional and non-functional requirements of the Virtual Fitness Trainer software. This software is designed to provide users with personalized workout plans, progress tracking, and virtual coaching.

### 1.2 Scope

The Virtual Fitness Trainer will be a mobile and web application that provides users with AI-driven workout recommendations, video demonstrations, progress monitoring, and motivational feedback. It is intended for people who look for personalized training or rehabilitation. It supports features such as:

- Set fitness goals and receive customized workout plans
- Track workout progress and log exercises
- Access virtual coaching and real time feedback
- Sync with wearable fitness devices
- Participate in community challenges

The system aims to enhance users' fitness experiences by delivering a structured, engaging, and data-driven approach to personal training.

### 1.3 Definitions, Acronyms, and Abbreviations

VFT	Virtual Fitness Trainer
User	Individual who interacts with the software
API	Application Programming Interface
Biometric Data	Physiological and health related data
Workout Plan	Set of exercises

### 1.4 References

The references are:

- Course Materials – CS 250: “Lecture 3-ReqGathering”
- IEEE Guide for Software Requirements Specifications
- Your 2025 Guide to Writing a Software Requirements Specification (Andrew Burak)

- [https://allassignmentexperts.com/sample\\_assignments/programming/Software%20Requirements%20Specification%20-%20Gym%20App.pdf](https://allassignmentexperts.com/sample_assignments/programming/Software%20Requirements%20Specification%20-%20Gym%20App.pdf)

## 1.5 Overview

This document details the Virtual Fitness Trainer software's functional and non-functional requirements, providing design constraints, interface requirements, and use case scenarios. Section 2 is an overall description of the software including the product perspective and functions, and the user characteristics. In section 3, a full list of requirements is explained.

## 2. General Description

This section outlines the broader factors influencing the system and its requirements without specifying exact functional or technical details.

### 2.1 Product Perspective

The Virtual Fitness Trainer is an independent application. Rather similar to the currently available fitness-based apps like Nike Training Club or MyFitnessPal, it comes unique with its AI-driven adaptive workouts that continuously refine training plans through consistent improvements guided by user progress and real-time biometric feedback.

Although it can be used by itself, it can also work as a part of a broader digital health and wellness ecosystem by integrating with various external platforms. It can be connected to wearable fitness devices such as the Apple Watch or Fitbit to collect data on heart rate, step count, and calorie expenditure, as well as, it can synchronize with Apple Health or Google Fit. It also relies on cloud-based storage to securely saving user progress and analytics.

To ensure seamless integration, the VFT will interact with external systems through well-defined interfaces:

- API interfaces: RESTful APIs for connecting with other devices and platforms
- Database Interfaces: secure storage for user profiles, workout history, and AI-driven insights
- User Interface: web and mobile interfaces designed for intuitive navigations and interactive workout sessions

While some offline functionality will be supported, and internet connection is required for cloud synchronization and real-time tracking.

### 2.2 Product Functions

It provides a range of functions designed so that the users can engage in structured, adaptive workout routines that cater to their individual fitness levels and goals.

- User profile management: users can create a personal profile that store their fitness goals, activity history, and preferences.

- AI generated workout plans: the software offers personalized workout routines based on user fitness level, goals, and preferences. Algorithms analyze past performance and adjust future workout to optimize training effectiveness and user experience
- Real-time workout guidance: it provides step-by-step instructions and visual demonstrations to ensure users can follow the exercises correctly
- Exercise and progress tracking: the VFT provides users with metrics such as calories burned, workout duration, and repetitions completed
- Wearable device and API integration: the software connects with fitness tracking devices and platforms to collect real time data
- Social and community features: users can connect with friends, participate in challenges and share workout achievements
- Motivational features and notification: to keep users engaged, it includes motivational messages, goal reminders and achievement badges.
- Video and audio workout support: it includes video demonstrations and audio coaching for exercises
- Data analytics and reports: users receive insights based on their fitness journey
- Security and privacy management: the VFT ensures that all data users is securely stores and protected

### 2.3 User Characteristics

The Virtual Fitness Trainer is designed to accommodate a diverse range of users with varying levels of experiences, technical expertise, and fitness goals.

It is designed to be intuitive and accessible to users with a wide range of educational backgrounds. In most cases, users will understand at least the basic concepts of fitness concepts, but the system will be structured in a way that does not assume advanced knowledge of health or exercise science. At the same time, the system will offer more advanced content and explanations for users seeking detailed information about exercise techniques, health data, and workout theory.

The users would range from beginners to experts in their fitness abilities; therefore, the system must be adaptable and flexible, so it will provide clear instructions, visual demonstrations, and motivational prompts for beginners.

The interface will be simple, intuitive and easy to navigate, ensuring that users can easily interact with the system regardless of their technical background.

### 2.4 General Constraints

The Virtual Fitness Trainer (VFT) has several important limits that affect how it will be developed:

- It must follow privacy laws to protect user health data
- The system needs to work well on different devices, taking into account things like battery life and screen size
- It should connect smoothly with other apps and devices using standard formats
- The system should keep track of user activity and data access for security reasons
- Users should easily control key features, like adjusting workout settings or notifications

- The system will use modern programming languages that are efficient and commonly used
- The VFT must be reliable, with minimal downtime and quick responses to users
- The system needs to work correctly to keep users safe and motivated
- The VFT must protect user data with strong encryption and secure login methods

## **2.5 Assumptions and Dependencies**

The Virtual Fitness Trainer (VFT) is based on several assumptions and dependencies that could impact the system's requirements:

- Assumes users have smartphones, tablets, or compatible fitness wearables.
- Requires a stable internet connection for cloud syncing and data integration.
- Assumes users have devices running compatible operating systems (IOS or Android).
- Assumes users provide accurate information for personalized workouts.
- Depends on cloud storage and services for data management and syncing.
- Requires users to grant necessary permissions for data sharing with wearable devices and health apps.
- Assumes active participation from users to improve workout recommendations.

## **3. Specific Requirements**

### **3.1 External Interface Requirements**

#### **3.1.1 User Interfaces**

It should be intuitive, responsive, and accessible across devices

- Easy to use menu and layout for accessing workouts, progress, and settings
- Touchscreen or click-based inputs for workout selections, adjustments, and tracking
- Clear displays for exercise instructions, real time statistics and performance progress

#### **3.1.2 Hardware Interfaces**

The VFT must interface with various hardware devices, including:

- Compatible with iOS and Android operating systems. The VFT should function smoothly on devices with different screen sizes, processors, and memory.
- Integration with devices such as Fitbit or Apple Watch, for real-time fitness tracking. The VFT will need to communicate with the wearables via Bluetooth or compatible syncing technologies.
- Potential integration with other fitness equipment (gym machines) if supported by Bluetooth or Wi-Fi connectivity.

#### **3.1.3 Software Interfaces**

The VFT must integrate with other software systems and platforms, including:

- Health and fitness apps: like Google Fit and Apple Health for data synchronization, such as activity logs, sleep data, and health metrics.



- Cloud services: cloud storage for user data management and synchronization across multiple devices.
- Data sharing: use of APIS to exchange data between the VFT and third-party services

### **3.1.4 Communications Interfaces**

The VFT requires communication capabilities for:

- Real-time syncing of fitness data between the app and external devices or cloud services.
- Send reminders, progress updates, or motivational messages through push notifications or email.
- Support communication back to the user through alerts, recommendations, and achievements, as well as user input for adjusting fitness goals or preferences.

## **3.2 Functional Requirements**

### **3.2.1 User registration and profile management**

- Users must be able to create a personal account by providing their email address, creating a password and entering personal information (name, age, weight, height, fitness goal)
- Users should be able to update their profiles
- The system must provide a mechanism for users to recover their passwords in case they forget them

### **3.2.2 Workout plan generation**

- The VFT must generate customized workout plans based on user inputs
- It should adapt workout plans over time based on user progress

### **3.2.3 Exercise library**

- The system must include a comprehensive library of exercises, categorized by type and difficulty level
- Each exercise must include clear instructions and visual demonstrations to ensure proper form and safety

### **3.2.4 Real time tracking and feedback**

- The VFT should track user performance in real time during workouts, including metrics like durations, repetitions, sets and calories burned
- The software must provide real-time feedback during workouts, offering tips to form correction and motivation to encourage users

### **3.2.5 Progress tracking and analytics**

- The Virtual Fitness Trainer should display key performance metrics over time, such as workout frequency, total calories burned, and improvements
- Users must be able to set fitness goals and track their progress towards achieving them

### **3.2.6 Social features**

- Users should be able to connect with friends or other users within the app
- The systems should include options for users to participate in challenges and view leaderboards

### 3.2.7 Notifications and reminder

- Users should be able to set reminders for workouts, progress check-ins, and goals achievements
- The systems must send motivational messages based on user behavior and progress to keep users engaged

### 3.2.8 Integration with external devices

- The VFT must connect to with various wearable fitness devices for seamless data collection
- The software should connect with health and fitness applications for comprehensive tracking and data synchronization

### 3.2.9 Security and privacy

- The VFT must implement strong security measures to protects user data, including encryption and secure authentication methods
- Users should have control over their privacy setting, allowing them to manage data sharing and the visibility of their profiles

## 3.3 Use Cases

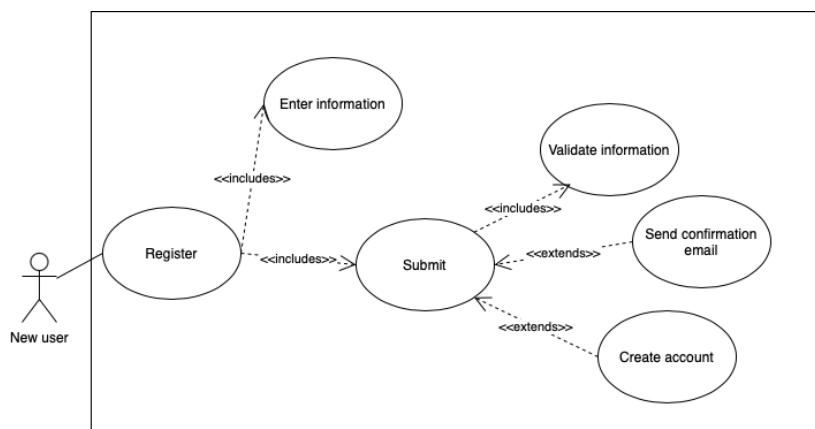
### 3.3.1 User registration

Actor: New user

Description: a new user creates a new account to the system

Main flow:

1. The user navigates to the registration page
2. Enters required information
3. Submit the registration form
4. The system validates the information
5. If valid, the system creates a new user account and send confirmation email
6. The user receives confirmation email and can log in



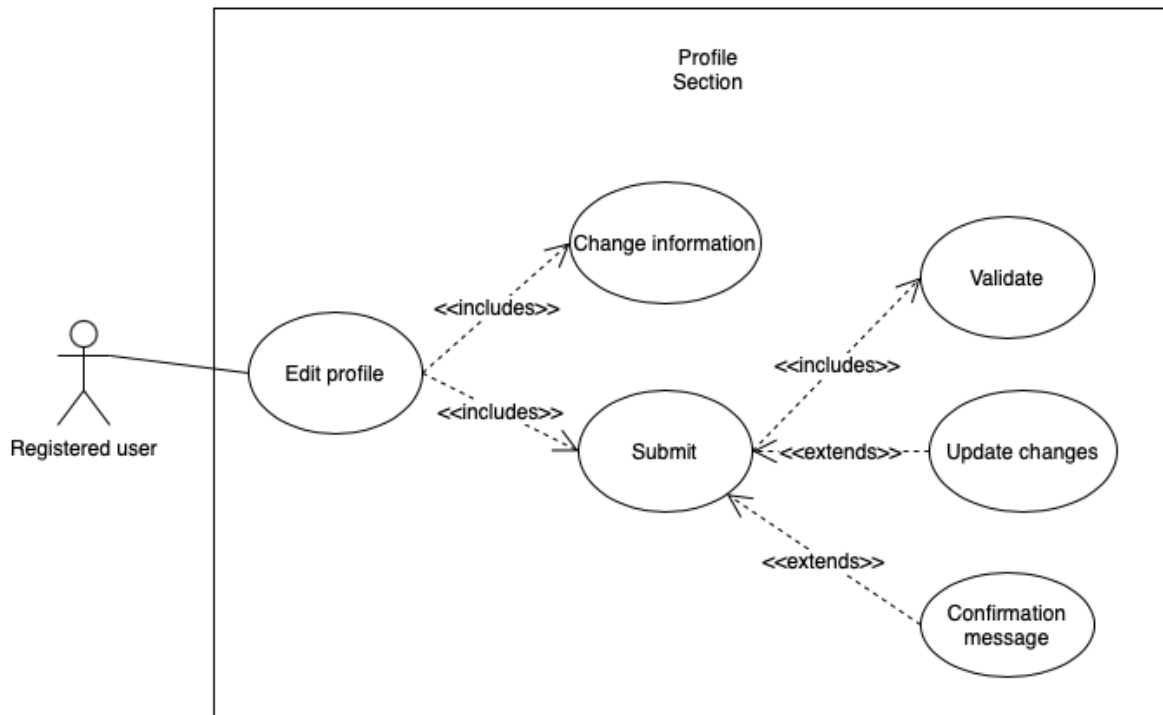
### 3.3.2 Update profile information

Actor: Registered user

Description: a registered user changes any information of his profile

Main flow:

1. The user navigates to the “Profile” section
2. The user updates personal information
3. The user submits the update information
4. The system validates the changes
5. If valid, the system updates the user profile and displays a confirmation message



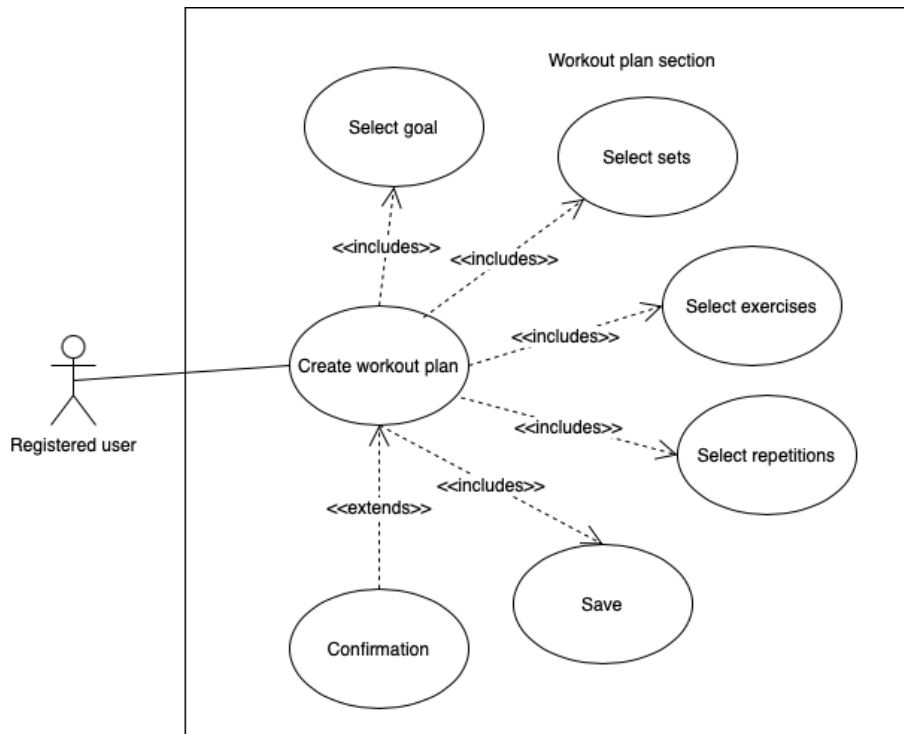
### 3.3.3 Create workout plan

Actor: Registered user

Description: a registered user creates a new set of exercises

Main flow:

1. The user navigates to the “Create Workout Plan” section
2. The user selects workout goals
3. The user chooses exercises, sets, and repetitions
4. The user saves the workout plan
5. The system confirms the successful creation of the workout plan



### **3.4 Classes / Objects**

The classes will be User, WorkoutPlan, WorkoutSession, Exercise, ProgressTracker, Challenge and Notification. More information about specific classes, methods, and attributes can be found in Section 5.

### **3.5 Non-Functional Requirements**

#### **3.5.1 Performance**

- The system should respond to user inputs within 2 seconds
- The system should handle at least 100 concurrent users without significant degradation in performance
- The system should be able to process user progress data and generate reports within 5 seconds for a dataset of up to 1,000 records

#### **3.5.2 Reliability**

- The system should maintain an uptime of 99.9%, always ensuring availability for users
- The system should have an error rate of less than 1% for all operations
- In the event of a failure, the system should have mechanisms in place to recover within 5 minutes, minimizing disruption to users

#### **3.5.3 Availability**

- Users should be able to access the system anytime
- The system should support users across various locations and time zones

#### **3.5.4 Security**

- All user data should be encrypted both at rest and in transit using industry-standard encryption protocols
- The system should implement strong user authentication methods to enhance account security
- Role-based access control should be implemented to restrict access to sensitive data bases on user roles and permissions

#### **3.5.5 Maintainability**

- The system should be designed using a modular architecture, allowing for easy updates and integration of new features without disrupting existing functionality
- Comprehensive documentation should be provided for both users and developers
- The system should include automated testing processes to ensure that updates do not introduce new defects

#### **3.5.6 Portability**

- The application should be compatible with major operating systems, including Windows, macOS, and Linux, as well as the mobile platforms IOS and Android
- The system should be accessible through a web browser without requiring installation
- The user interface should be designed responsively, ensuring optimal usability and experience on devices of different screen sizes

### **3.6 Inverse Requirements**

- The system shall not share user data with third parties without consent
- The system must not exceed a response time of 3 seconds for any user action
- The system should not be platform-specific or require proprietary software for access
- The system must not allow users to log in after 5 unsuccessful attempts to prevent attacks

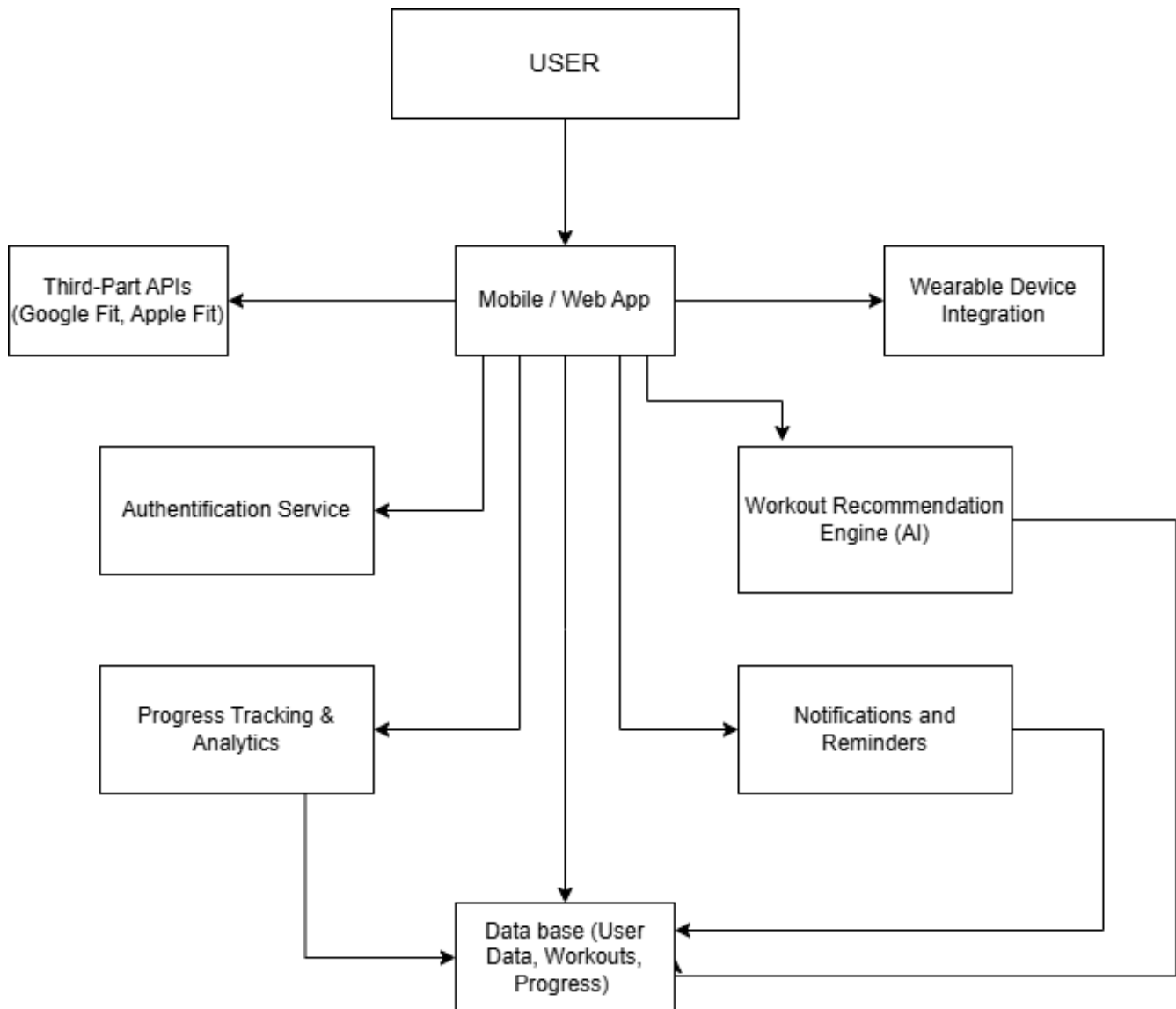
## 4. System Description

### 4.1 Overview

This Software Design Specification section continues with a detailed architecture, components, and design decisions made in building the system based on the SRS (Software Requirements Specification). The VFT can be accessed in both web and mobile platforms in order to create a scalable user experience allowing users a handy way to achieve their fitness goals. How these system components can collaborate to realize the target goals of the project will be explained in this document.

## 5. Software Architecture

### 5.1 Architectural Diagram

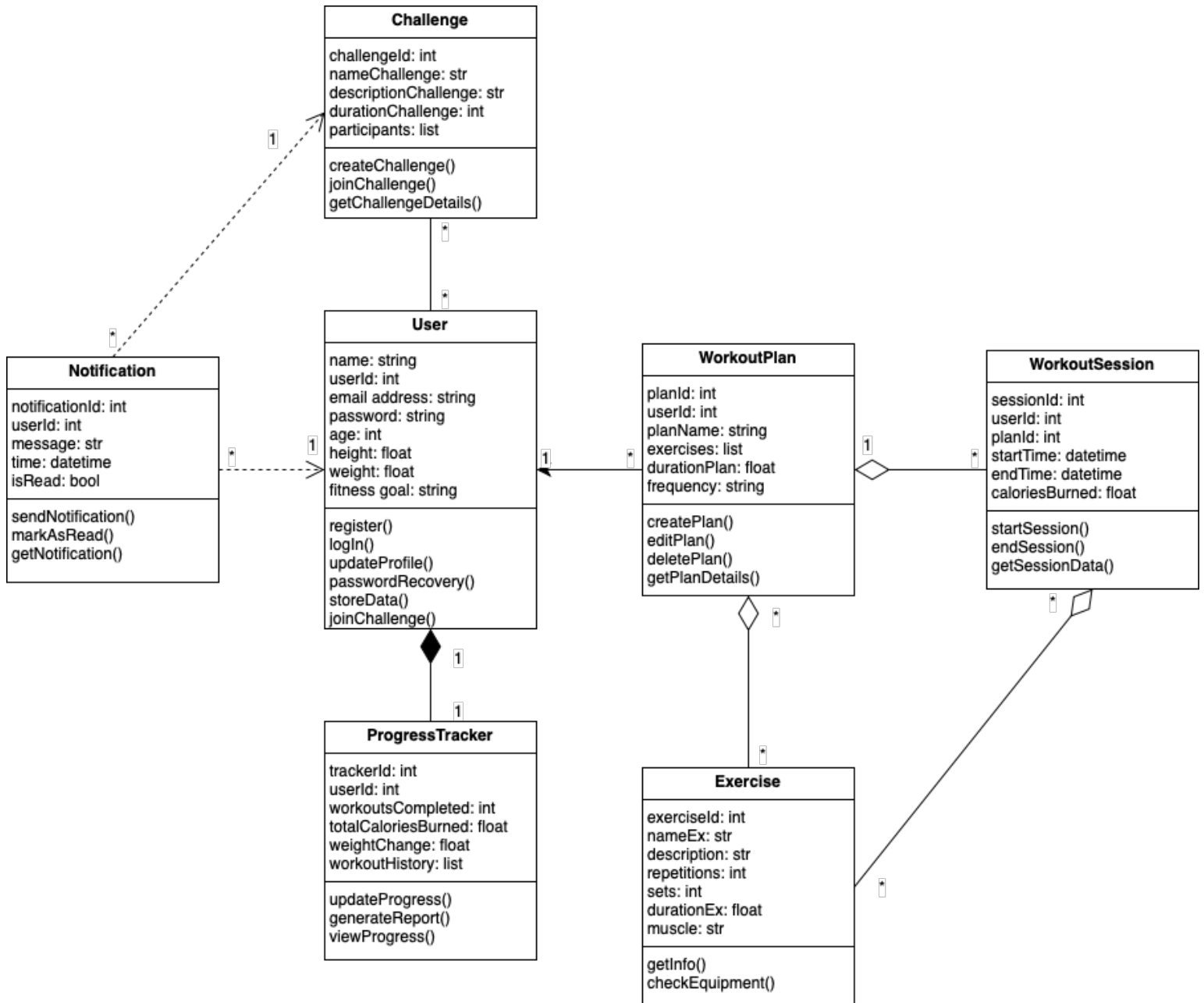


The Virtual Fitness Trainer is an application designed for mobile and web platforms to cater AI-driven personalized workout plans, real-time feedback, and tracking of progress. This platform provides a user-friendly interface, allowing users to register, set fitness goals, and track their fitness journey. The key features of the system consist of an Authentication Service that provides secure user access, an AI-based Workout Recommendation Engine that modifies workout plans based on user progress over time, and a Progress Tracking & Analytics module that tracks exercises and generates insights. A Notifications and Reminders subsystem interacts with users through motivational messages, workout reminders, and performance updates.

To enhance the user experience, the ecosystem is integrated with wearable fitness devices (Apple Watch, Fitbit) and third-party health platforms (Google Fit, Apple Health) to synchronize data in an automated manner. User profiles, workout histories, and AI-driven insights are housed in a central database with tight security provisions to facilitate decision-making. The architecture is designed for scalability, security, and real-time processing, which allows accommodating more users while protecting sensitive health data. This architecture ensures high availability, interoperability, and a riveting fitness experience, firmly entrenching the Virtual Fitness Trainer as a complete digital fitness solution.



## 5.2 UML Class Diagram



The system models fitness tracking with challenges, workout plans, and progress tracking. The User class interacts with multiple entities (WorkoutPlan, ProgressTracker, Notification). There's an emphasis on session tracking, which suggests a structured workout monitoring approach.

## 5.3 Classes, Attributes and Functions

### 5.3.1 User

Represents an individual user of the Virtual Fitness Trainer

#### 5.3.1.1 Attributes

- name: string -> name of the user
- usedId: integer -> unique identifier for the user
- emailAddress: string -> user's email address
- password: string -> encrypted user password
- age: integer -> user's age
- height: float -> user's height
- weight: float -> user's weight
- fitnessGoal: string -> user's fitness goal (weight loss, muscle gain...)

#### 5.3.1.2 Functions

- register() -> create new account
- login() -> authenticates user credentials
- logout() -> logs out the user
- updateProfile() -> updates personal details
- passwordRecovery() -> allows the user to recover passwords if forgotten
- storeData() -> saves or updates user information such as name, email, height, weight and fitness goal
- joinChallenge(challengeId) -> subscribe in challenge

### 5.3.2 WorkoutPlan

Represents a personalized work out plan for a user

#### 5.3.2.1 Attributes

- planName: string -> name of the workout plan
- planId: integer -> unique identifier
- userId: integer -> reference to the associated user
- exercises: list -> list of exercise objects included in the plan
- durationPlan: float -> total duration of the workout plan
- frequency: string -> frequency of workouts (daily, weekly...)

#### 5.3.2.2 Functions

- createPlan(userId) -> create a personalized workout plan based on user input
- editPlan(planId) -> modify workout plan based on user feedback or progress
- deletePlan(planId) -> removes a workout plan
- getPlanDetails(planId) -> retrieves details of a plan

### 5.3.3 Exercise

Represents an individual exercise in the workout library

### 5.3.3.1 Attributes

- exerciseId: integer -> unique identifier for the exercise
- nameEx: string -> name of the exercise.
- description: string -> explanation of the exercise
- repetitions: integer -> number of repetitions
- sets: integer -> number of sets
- durationEx: float
- muscle: string -> part of the body used in the exercise

### 5.3.3.2 Functions

- getInfo(exerciseId) -> give details and instructions for the exercise
- checkEquipment(exerciseId) -> check if specific equipment is required

## 5.3.4 ProgressTracker

Handle the tracking and analysis of user performance metrics

### 5.3.4.1 Attributes

- trackerId: integer -> unique identifier
- userId: integer -> reference to the associated user
- workoutsCompleted: list -> collection of completed workouts and associated metrics
- totalCaloriesBurned: float -> total calories burned by the user
- weightChanged: float -> difference in weight over time
- workoutHistory: list -> list of completed workouts

### 5.3.4.2 Functions

- updateProgress(userId, sessionId) -> updates the user's fitness progress based on completed workouts
- getProgress(userId) -> retrieves an overview of a user's fitness improvements, including weight change and calories burned
- viewProgress(userId, dateRange) -> displays fitness progress over specific period

## 5.3.5 Notification

Manage notification and reminder for the user

### 5.3.5.1 Attributes

- notificationId: integer -> unique identifier
- userId: integer -> reference to associated user
- message: string -> content
- time: datetime -> time when it is displayed
- isRead: boolean -> status indicating whether the notification has been read or not

### 5.3.5.2 Functions

- sendNotification(userId, message) -> sends a notification to user
- markAsRead(notificationId) -> mark the notification as read
- getNotification(userId) -> retrieve all unread notification for the user

### 5.3.6 Challenge

Represents a fitness challenge that users can participate in

#### 5.3.6.1 Attributes

- challengeId: integer -> unique identifier for the challenge
- nameChallenge: string -> name of the challenge
- descriptionChallenge: string -> description of the challenge
- durationChallenge: integer -> duration of the challenge (days, weeks...)
- participants: list -> users that are participating in the challenge

#### 5.3.6.2 Functions

- createChallenge() -> create new fitness challenge
- joinChallenge(userId) -> allow user to join
- getChallengeDetails(challengeId) -> retrieves details of the challenge
- getParticipants() -> returns a list of participants

### 5.3.7 WorkoutSession

#### 5.3.7.1 Attributes

- sessionId: integer -> unique identifier
- userId: integer -> reference to associated user
- planId: integer -> reference to workout plan
- startTime: datetime -> time when the session started
- endTime: datetime -> time when the session finished
- caloriesBurned: float -> number of calories that were burned during the session

#### 5.3.7.2 Functions

- startSession(sessionId) -> begins a new workout session, recording the start time and selected exercises
- endSession(sessionId) -> ends the workout session, logging data like duration and calories burned
- getSessionData(sessionId) -> retrieves specific details of a past workout session

## 6. Development Plan and Timeline

The objective is to develop a virtual fitness trainer system that enables users to create and follow workout plans, track progress, join fitness challenges, and receive notifications.

### 6.1 Partitioning of tasks

Phase	Duration	Start Date	End Date
Requirements Gathering	2 weeks	Week 1	Week 2
System Design	4 weeks	Week 3	Week 6
Implementation	6 weeks	Week 7	Week 12
Testing	3 weeks	Week 13	Week 15

Deployment and Maintenance	2 weeks	Week 16	Week 17
----------------------------	---------	---------	---------

### 6.1.1 Phase 1: Requirement Gathering and Analysis (2 weeks)

- Task 1: Conduct meetings with stakeholders to gather functional and non-functional requirements.
  - Responsible: Project Manager (PM), Business Analyst (BA)
- Task 2: Define use cases and user stories based on the requirements.
  - Responsible: BA, UI/UX Designer (UXD)
- Task 3: Identify constraints such as budget, timeline, and hardware/software limitations.
  - Responsible: PM, Technical Lead (TL)

### 6.1.2 Phase 2: System Design (4 weeks)

- Task 1: High-Level Architecture Design (1 week)
  - Define the overall system architecture, including the separation of frontend, backend, and database components.
  - Responsible: TL, System Architect (SA)
- Task 2: Database Design (1 week)
  - Create Entity-Relationship (ER) diagrams, database schema, and data flow diagrams.
  - Responsible: Backend Developer (BD), Database Administrator (DBA)
- Task 3: UI/UX Design (2 weeks)
  - Design wireframes, prototypes, and user flow diagrams. Review usability with stakeholders.
  - Responsible: UXD, UI Designer (UID)

### 6.1.3 Phase 3: Implementation (6 weeks)

- Task 1: Frontend Development (3 weeks)
  - Implement user-facing components based on the designs. Includes UI elements for user profiles, workout schedules, etc.
  - Responsible: Frontend Developer (FD)
- Task 2: Backend Development (3 weeks)
  - Implement logic for workout recommendation algorithms, user profile management, and database integration.
  - Responsible: BD, AI Developer (AID)

### 6.1.4 Phase 4: Testing (3 weeks)

- Task 1: Unit Testing (1 week)
  - Ensure individual components (e.g., AI algorithm, database, frontend components) are functioning correctly.
  - Responsible: QA Engineer (QA)
- Task 2: Integration Testing (1 week)
  - Test the interaction between components such as frontend, backend, and database.
  - Responsible: QA, BD, FD

- Task 3: User Acceptance Testing (1 week)
  - Stakeholder testing to validate that the system meets the business requirements.
  - Responsible: PM, QA

### **6.1.5 Phase 5: Deployment and Maintenance (2 weeks)**

- Task 1: Deployment (1 week)
  - Deploy the application to the live environment. Ensure proper scaling and security configurations are in place.
  - Responsible: DevOps Engineer (DO), TL
- Task 2: Post-Deployment Support & Maintenance (1 week)
  - Monitor the system, fix any bugs, and ensure that users can successfully interact with the system.
  - Responsible: PM, BD, DO

## **6.2 Team member responsibilities**

The success of the Virtual Fitness Trainer project relies on the collaboration of a multidisciplinary team. Each member plays a critical role in ensuring the system is designed, developed, and deployed efficiently. Below is a summary of the key responsibilities for each team member, outlining their contributions throughout the project lifecycle.

### **6.2.1 Project Manager (PM)**

- Ensure progress, manage scope, and stay within budget.
- Act as the main point of contact for stakeholders and clients.
- Identify and mitigate risks to project success.
- Monitor milestones and adjust plans as needed.

### **6.2.2 Business Analyst (BA)**

- Collect functional and non-functional requirements from stakeholders.
- Define clear use cases and user stories.
- Make sure the project aligns with business goals.
- Assist in user acceptance testing (UAT).

### **6.2.3 System Architect (SA)**

- Define system structure and choose technology stack.
- Design the system for future growth.
- Plan the integration of different system components

### **6.2.4 Frontend Developer (FD)**

- Implement user-facing components based on designs.
- Make the app responsive across devices.
- Ensure smooth app performance on all platforms.

### **6.2.5 Backend Developer (BD)**

- Develop backend logic and APIs.
- Work with DBA to integrate and manage the database.
- Implement secure user authentication and data handling.

### **6.2.6 Database Administrator (DBA)**

- Design the database structure for efficiency and integrity.
- Ensure that database queries run efficiently.
- Manage database security, backups, and recovery.

### **6.2.7 AI Developer (AID)**

- Implement personalized workout recommendations.
- Work with the backend team to integrate AI features.
- Continuously improve the AI models based on data.

### **6.2.8 UI/UX Designer (UXD)**

- Create wireframes, prototypes, and mockups.
- Gather feedback and iterate on design.
- Ensure the app is usable by all, including users with disabilities.

### **6.2.9 Data Scientist (DS)**

- Refine machine learning models based on user data.
- Extract insights from data to improve recommendations.
- Create new features from data to improve predictions.

### **6.2.10 QA Engineer (QA)**

- Conduct unit, integration, and system tests.
- Set up automated testing where possible.
- Conduct performance and security testing.

### **6.2.11 DevOps Engineer (DO)**

- Set up and manage CI/CD pipelines.
- Ensure the system is scalable and secure.
- Implement and monitor security measures across the infrastructure.