Virtual Fitness Trainer

Architecture Design w/Data Management

Version 4.0

04/10/2025

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>
<03/20/25>	<Version 3.0>	Group-13	<SDS>
<04/10/25>	<Version 4.0>	Group-13	<DM>

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

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

1.5 Overview

This document details the Virtual Fitness Trainer software's functional and non-functional requirements, providing design constrains, 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, adaptative 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

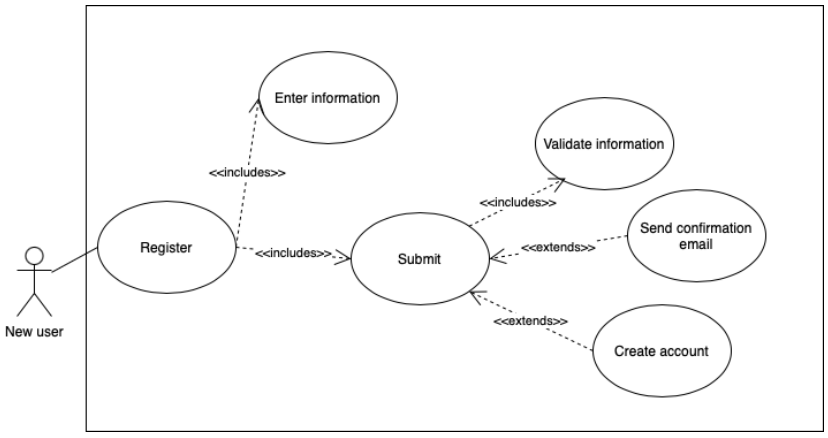
- The VFT must implement strong security measures to protects user data, including encryption and secure authentication methods
- Users should have control over their privacy setting, allowing them to manage data sharing and the visibility of their profiles

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:

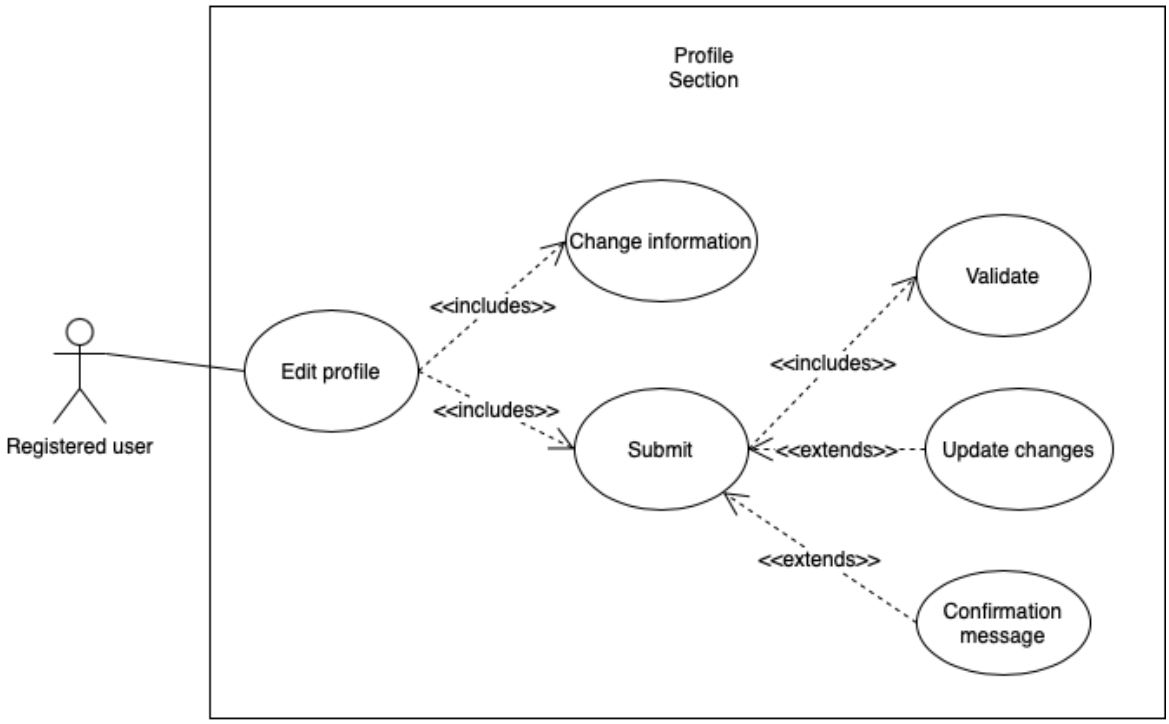
1. The user navigates to the registration page
2. Enters required information
3. Submit the registration form
4. The system validates the information
5. If valid, the system creates a new user account and send confirmation email
6. The user receives confirmation email and can log in



3.3.2 Update profile information

Actor: Registered user
Description: a registered user changes any information of his profile
Main flow:

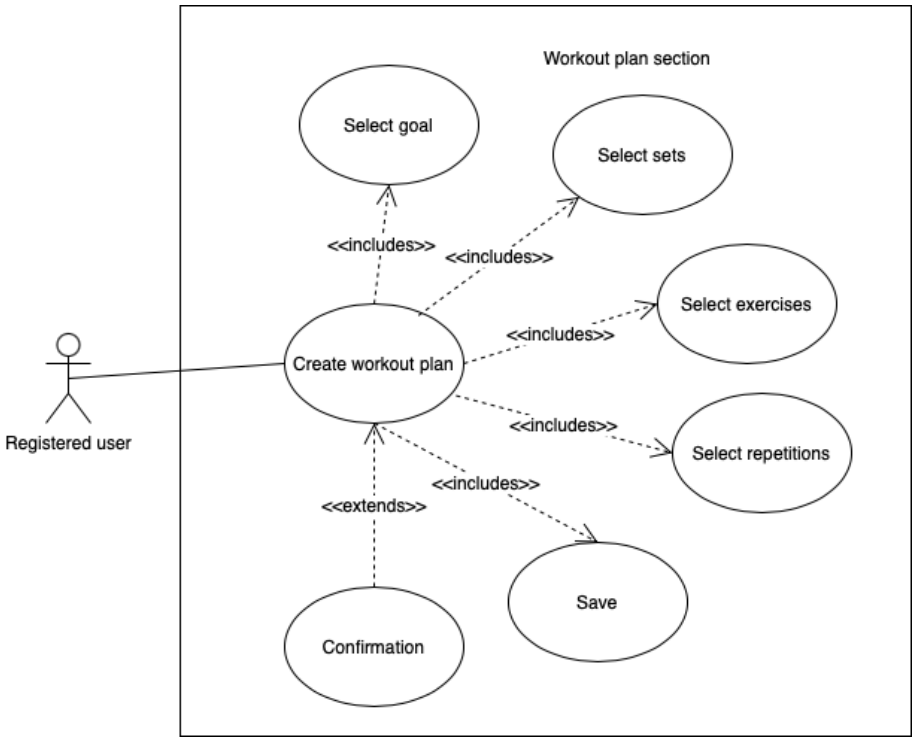
1. The user navigates to the “Profile” section
2. The user updates personal information
3. The user submits the update information
4. The system validates the changes
5. If valid, the system updates the user profile and displays a confirmation message



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.

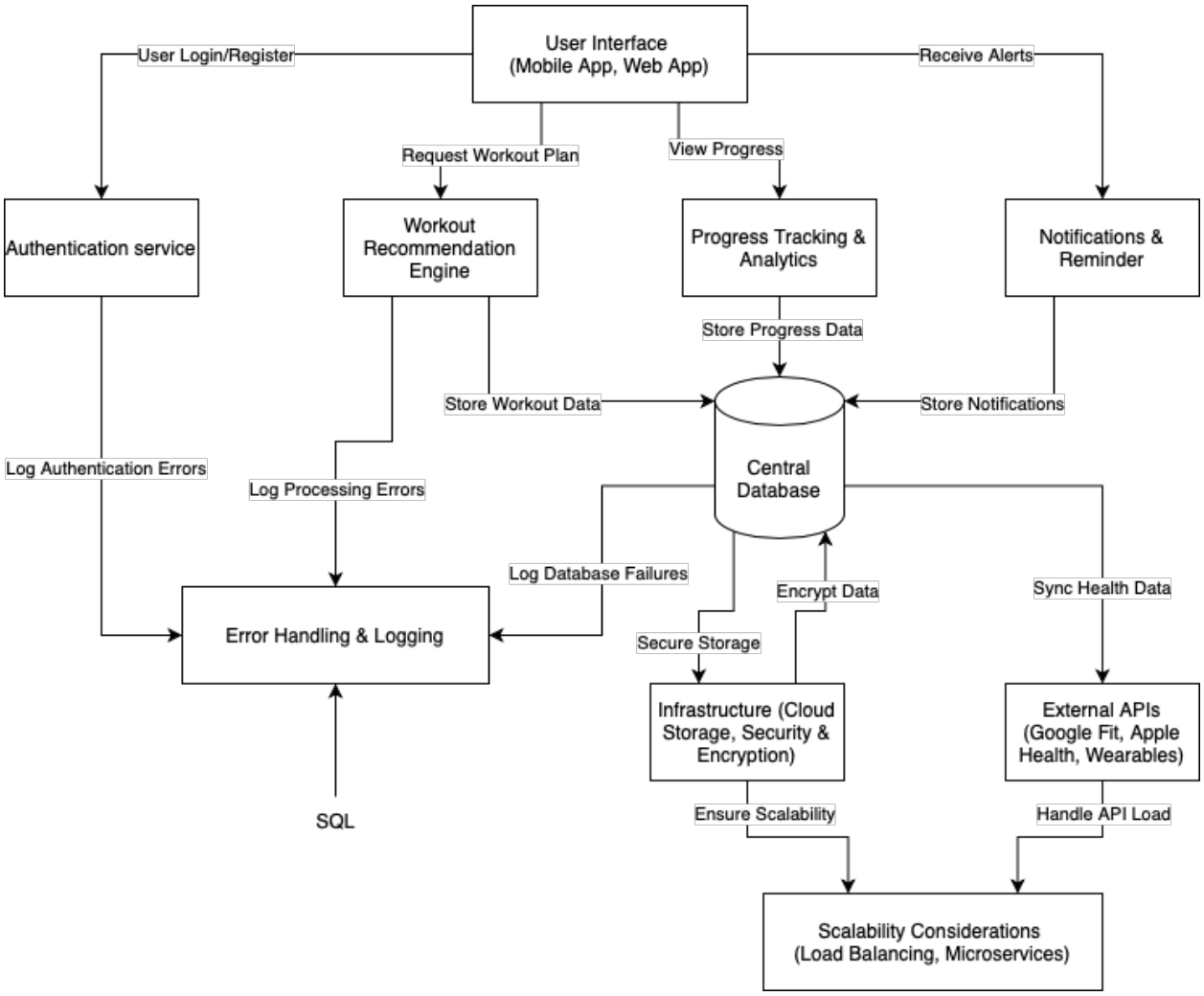
4.2 Data Processing Flow

To provide a clear understanding of how data moves within the Virtual Fitness Trainer (VFT), the following data flow process is defined:

1. User Input: Users enter fitness details, select workout preferences, or log progress.
2. Data Processing: The AI-based Workout Recommendation Engine processes input data and updates workout plans accordingly.
3. Data Storage: User information, progress metrics, and workout history are stored in a secure cloud database.
4. External Device Integration: Data from fitness wearables is processed via API calls.
5. Feedback & Reports: The system generates real-time workout feedback and analytics.
6. User Notification: Users receive personalized reminders and alerts.

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.

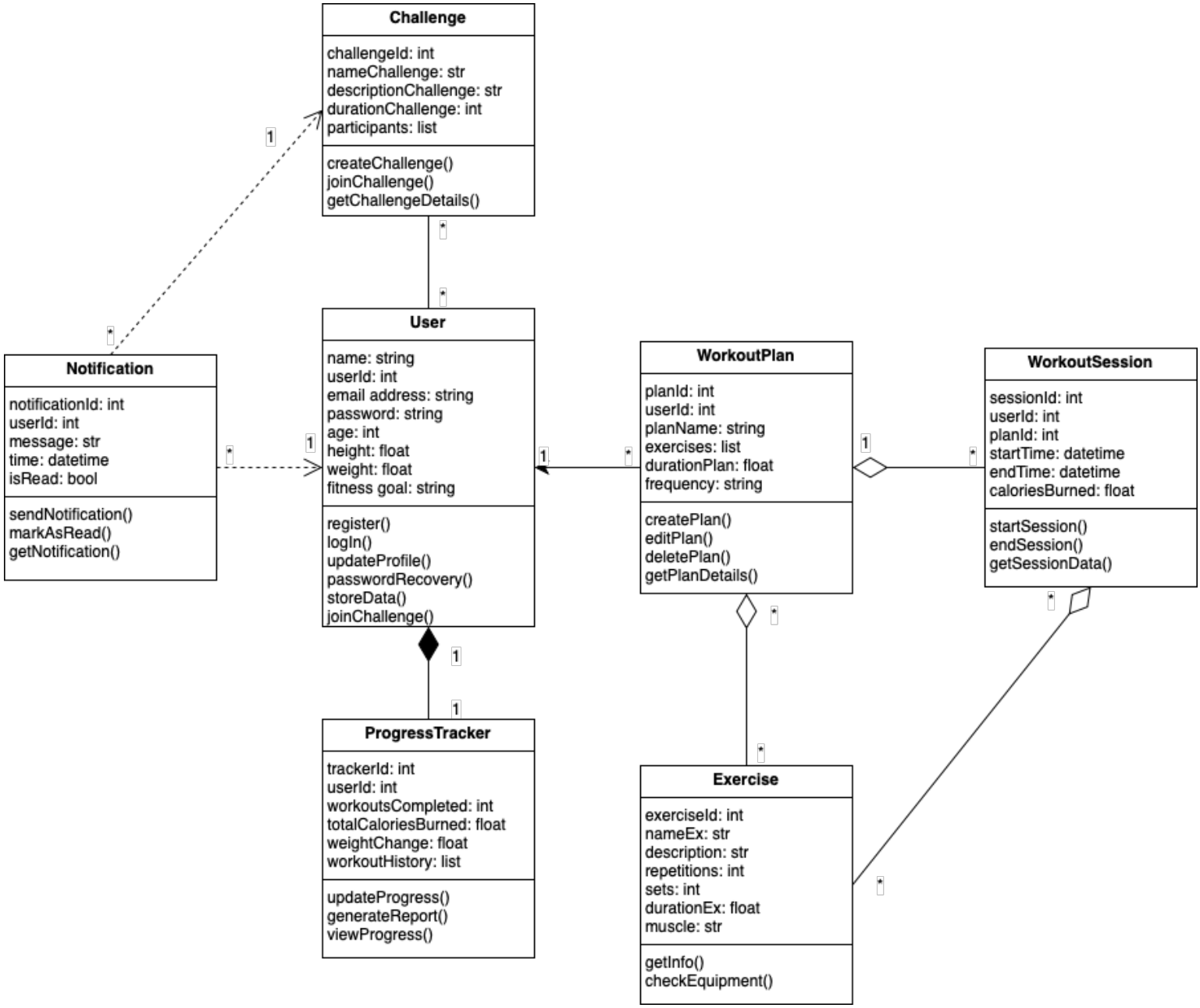
Virtual Fitness Trainer software is executed using scalable and modular design to achieve hassle-free user interaction, personalized exercise recommendation, and instant tracking of progress. There are four layers that form part of the system, i.e., User Interface, Application, Data, and Infrastructure, and all these are a vital contributor towards an impeccable fitness experience.
The User Interface Layer consists of mobile and web applications to allow the user to register, log in, view AI-based workout schedules, track progress, and receive notifications. It communicates with backend services through API calls, facilitating real-time synchronization of data. The

Application Layer manages critical features such as authentication, workout suggestion, tracking of progress, and notifications. Authentication Service secures user access, while the Workout Recommendation Engine employs AI to design personalized exercise regimens. The Progress Tracking module tracks user behavior, and the Notification System nudges users with reminders and motivational alerts.

The Data Layer contains a single database that securely stores user profiles, exercise histories, and analytics. It is also synchronized with external APIs such as Google Fit, Apple Health, and wearable sensors for correct synchronization of body metrics. The Infrastructure Layer offers support in the form of cloud storage, encryption, and scalability aspects such as load balancing and microservices, ensuring reliability with more users.

To enhance system resilience, Error Handling & Logging mechanisms track failures in authentication, workout generation, and database operations to improve debugging and overall system dependability. The Scalability module optimizes the system by processing increasing workloads through optimized API processing and distributed traffic management. With this architecture, the Virtual Fitness Trainer provides a secure, flexible, and data-driven fitness solution for users across different platforms.

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
- maskAsRead(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.

7. Test Plan

7.1 Overview

This section lays out test plans for verification and validation. It includes detailed test cases covering various system components, ensuring the software meets design and performance expectations.

7.2 Testing Approach

- Functional testing: this will ensure that all features and functionalities of the VFT work as expected, including exercises, workout tracking, and user interaction
- Usability testing: this will assess the ease of use of the system from the perspective of users, ensuring that navigation, workout setup, and progress tracking are intuitive and user-friendly
- Performance testing: this will evaluate how the system performs under various conditions, e.g., the number of users accessing the system at the same time and how well the system handles video streaming and workout data processing without delay
- Security testing: this will detect and fix any security weaknesses, making sure that personal data, such as health information and payment information, are stored and transmitted securely
- Compatibility testing: this will verify that the system is functioning as expected across different devices (smartphones, tablets, laptops) and operating systems, including Android and iOS

7.3 Test Case Categories

7.3.1 User Registration and Authentication

- Successful user registration and login
- Invalid login attempts
- Password reset functionality
- Multi-factor authentication (MFA) verification

7.3.2 Workout Setup and Customization

- Selecting workout plans based on user preferences
- Customizing workout routines (e.g., intensity, duration)
- Saving and editing personalized workout plans
- Scheduling workouts and notifications

7.3.3 Progress Tracking and Analytics

- Logging workout data (e.g., sets, reps, calories burned)
- Displaying progress over time (graphs, achievements)
- Verifying correct calculations of calories, distance, etc.
- Syncing with wearable devices or fitness trackers

7.3.4 Exercise and Video Streaming

- Playing exercise videos without buffering or interruptions
- Pausing, skipping, or rewinding workout videos
- Verifying video quality and resolution on different devices
- Correct display of exercise instructions and tips

7.3.5 User Profile and Account Management

- Updating user information (e.g., name, age, fitness goals)
- Managing connected devices (e.g., fitness trackers, smartwatches)
- Viewing workout history and progress reports
- Adjusting notification preferences (e.g., reminders, newsletters)

7.3.6 Admin Functionality

- Adding, editing, and removing workout programs
- Managing user accounts and subscriptions
- Generating reports on user activity and workout engagement
- Handling customer support requests and feedback

7.3.7 Error Handling and Error Edges

- Handling failed video streaming or playback issues
- Graceful degradation when the internet connection is lost
- Testing for security vulnerabilities in payment and personal data handling
- Verifying data integrity when syncing across devices or systems

7.4 Test Case Samples

7.4.1 Test#1 – User registration (Unit)

TestCaseId	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test Executed By
User_Registration_Test	User Management	P3-High	Verify that a new user can successfully register with valid credentials.	User is on the registration page.	1. Enter valid details. 2. Submit form. 3. Confirm email.	User account is created, and confirmation email is sent.	User account created, confirmation email sent.	Pass	Silvia Perez

The User Registration Test checked if new users could make an account with the right details. This test required going to the registration page, filling in details like email, password, age, height, and weight, and sending the form. We expected a new user account to be made and a confirmation

email to be sent. The actual outcome showed that the system worked well, as it made accounts and sent confirmation emails correctly, so the test was passed.

7.4.2 Test#2 – Login authentification (Unit)

Login_Authentication_Test	User Authentification	P3-High	Validate that a user cannot log in with incorrect credentials.	User has an existing account.	1. Enter invalid credentials. 2. Attempt login.	System displays an error message and denies access.	Error message displayed, access denied.	Pass	Silvia Perez

The User Registration Test checked if new users could make an account with the right details. This test required going to the registration page, filling in details like email, password, age, height, and weight, and sending the form. We expected a new user account to be made and a confirmation email to be sent. The actual outcome showed that the system worked well, as it made accounts and sent confirmation emails correctly, so the test was passed.

7.4.3 Test#3 – Workout Plan Creation (Functional)

Workout_Plan_Creation_Test	Workout Module	P1-Medium	Ensure that a user can create a workout plan with selected exercises.	User is logged in.	1. Select exercises 2. Define repetitions 3. Save plan	Workout plan is saved and displayed in the user profile.	Workout plan saved and displayed.	Pass	Silvia Perez

The Workout Plan Creation Test checked if a registered user could make their own workout plan. This functional test looked at choosing exercises, deciding how many times to do them, how often to work out, and saving the plan. It was expected that the workout plan would be saved and shown in the user's profile. The test was successful, which means users can create and save workout plans with no issues.

7.4.4 Test#4 – Progress Tracking (Functional)

Progress_Tracking_Test	Tracking System	P1-Medium	Verify that the system updates the user's progress after workout completion.	User has completed a workout.	1. Complete workout 2. View progress dashboard	Updated progress metrics appear in the dashboard.	Progress metrics updated successfully.	Pass	Silvia Perez

The Progress Tracking Test made sure the system correctly recorded when a workout was done and updated fitness stats. This test had a user complete a workout and checked if the system updated things like calories burned and exercises done. The expected result was the system showing the user's new fitness stats on the progress dashboard. The test was successful, showing the tracking feature works well.

7.4.5 Test#5 – Notification System (Functional)

Notification_System_Test	Notification System	P0-Low	Ensure that scheduled workout reminders are sent to the user.	User has scheduled a workout reminder.	1. Set a workout reminder. 2. Wait for scheduled time.	User receives a notification at the scheduled time.	Notification received at scheduled time.	Pass	Silvia Perez

The Notification System Test was done to make sure that users got reminders for planned workouts. This test included setting up a workout reminder and watching if the system sent a notification at the right time. The expected result was that workout reminders were sent on time. The test passed, showing that the notification system worked well.

7.4.6 Test#6 – Weareble Devide Integration (System)

Wearable_Device_Integration_Test	Integration Module	P1-Medium	Check that the app correctly syncs workout data with external devices.	User has a connected wearable device.	1. Connet wearable 2. Perform a workout. 3. Sync data.	Data from wearables is reflected in the user's profile.	Wearable data synced successfully.	Pass	Ines Ling Delgado

The Wearable Device Integration Test tested the accuracy of synchronization between the Virtual Fitness Trainer and real-time data from fitness wearables like the Apple Watch or Fitbit. The user had to connect a wearable device, finish an exercise, and confirm that the app accurately displayed the data that was gathered as part of this system test. The smooth synchronization and display of exercise metrics was the anticipated result. The test's success demonstrated that wearable integration functioned as planned.

7.4.7 Test#7 – API connectivity (System)

API_Connectivity_Test	API Module	P3-High	Ensure that the app correctly fetches and updates data	Third-party APIs are	1. Fetch data from API. 2. Validate response.	Data synchronization occurs	API data synchronization successful.	Pass	Ines Ling Delgado

			via third-party APIs.	accessible.		without errors.			

The application's ability to accurately retrieve and update data from external fitness tracking platforms like Google Fit and Apple Health was confirmed by the API Connectivity Test. In this system test, requests were sent to external APIs, and the responses were verified. Error-free, seamless data synchronization was the anticipated result. The test showed dependable connectivity and validated that the external API integration worked as intended.

7.4.8 Test#8 – Cocurrent User Load (Performance)

Concurrent_User_Load_Test	Performance	P3-High	Measure system response when 100 users access the app simultaneously.	100 users attempt to log in simultaneously.	1. Set log in simultaneously. 2. Wait for response time.	System remains responsive with minimal lag.	System handled 100 users without lag.	Pass	Ines Ling Delgado

The purpose of the Concurrent User Load Test was to assess how well the system performed when several users accessed it at once. Response times and system stability were assessed during this performance test, which mimicked 100 users logging in simultaneously. The system was supposed to remain responsive with little to no lag. The system's ability to manage heavy traffic without experiencing performance deterioration was demonstrated by the successful test.

7.4.9 Test#9 – Data Encryption (Security)

Data_Encryption_Test	Security	P3-High	Validate that user passwords are stored securely using encryption.	Passwords are stored in database.	1. Store password. 2. Attempt to retrieve it from the database.	Passwords are hashed and not stored in plain text.	Passwords stored securely with encryption.	Pass	Ines Ling Delgado

The Data Encryption Test checked to see if user passwords were safely kept in the database. In order to determine whether stored passwords were encrypted, this security test tried to retrieve them. Passwords were supposed to be hashed and hidden in plain text. Data security was ensured by the test, which verified that encryption protocols were applied correctly.

7.4.10 Test#10 – Unauthorized Access Prevention (Security)

Unauthorized_Access_Prevention_Test	Security	P3-High	Ensure that a user is locked out after multiple failed login attempts.	User has failed login attempts.	1. Attempt login with incorrect credentials. 2. Reapeat five times	Account is temporarily locked after five failed attempts.	Account locked after five failed attempts.	Pass	Ines Ling Delgado

The system's ability to appropriately lock an account following several unsuccessful login attempts was evaluated by the Unauthorized Access Prevention Test. In this security test, an account lockout mechanism was triggered by five consecutive incorrect credential entries. It was anticipated that the system would momentarily prevent additional login attempts following the fifth unsuccessful attempt. The successful completion of the test demonstrated that the system successfully thwarted brute-force attacks.

8. Data Management Strategy

8.1 General Description

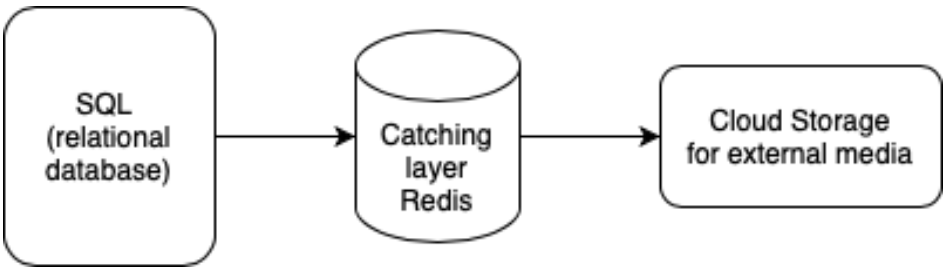
The data management strategy for the Virtual Fitness Trainer (VFT) prioritizes security, scalability, and real-time accessibility. The system uses a combination of database technologies to efficiently manage a wide range of data types, including personal fitness metrics, progress tracking, real-time workout data, and video content.

- Primary Database (Relational Database): A relational database (PostgreSQL) will serve as the primary data store for structured data such as:
    - User profiles (credentials, physical attributes, fitness goals)
    - Workout plans (exercise routines, sets, repetitions)
    - Progress tracking (workout history, calories burned, weight changes)
    - Workout sessions (session data, timestamps, performance)
    - Notifications and reminders
    - Challenges and leaderboards
- Caching Layer (Redis): Redis, a high-performance in-memory data store, will act as a caching layer to improve system performance. Redis will be used to store:
    - Real-time session data (e.g., active workouts)
    - Temporary synchronization from wearable devices
    - Frequently accessed AI-generated plan previews
- Cloud Storage (AWS S3 or Firebase Cloud Storage): Used for storing large media files, including:
    - Video tutorials and audio coaching
    - User-uploaded media (optional)
    - Auto-generated progress reports and charts

Key Considerations:

- Data Security: The system employs robust security measures including encryption at rest and in transit, secure authentication, and strict access controls.
- Scalability: The architecture allows to add user data and high concurrent usage at peak times.
- Real-Time Updates: This system also offers real-time syncing with your wearables and live workout progress tracking.
- Data Consistency: Referential integrity is ensured through structured relationships in the relational database.
- Backup and Recovery: Back-ups and disaster recovery protocols are applied to avoid loss of data.

8.2 Diagram



8.3 Relational Database (SQL)

Use of a structured SQL database ensures well-defined, enforced data relationships, secure user authentication, trackable workout history, and built-in data analytics. Robust searching, filtering, and reporting are possible due to structured query language (SQL). PostgreSQL was selected for its balance between performance, scalability, and modernity in the features it provides, such as JSON support and advanced indexing.

8.4 Single Database Design

For this system implementation, a single relational database instance contains all base tables and relationships. This design simplifies maintenance, backup processes, and access control and reduces system complexity. One database also makes cross-functional query operations easier, such as linking progress logs to workout plan changes or comparing metrics across challenges.

Although separating the database into separate services for users, workouts, and tracking could offer greater scalability, a design with such segregation would introduce significant coordination

overhead, especially at this development stage. The current centralized approach offers ease of development, reliability, and integrity to related data.

8.5 Table (Database) Structure

Below is a simplified representation of key tables that support core application functionality:

8.5.1 Users Table

- user_id (Primary Key)
- username
- email
- password
- age
- height
- weight
- fitness_goal

8.5.2 WorkoutPlan Table

- plan_id (Primary Key)
- user_id (Foreign Key referencing Users)
- plan_name
- duration
- frequency

8.5.3 Exercise Table

- exercise_id (Primary Key)
- name
- description
- sets
- repetitions
- muscle_group

8.5.4 ProgressTracker Table

- tracker_id (Primary Key)
- user_id (Foreign Key referencing Users)
- calories_burned
- weight_changed
- workouts_completed

8.5.5 WorkoutSession Table

- session_id (Primary Key)
- user_id (Foreign Key)
- plan_id (Foreign Key)
- start_time
- end_time
- calories_burned

8.6 Foreign Key Relationships

Foreign keys enforce consistency between entities:

- The WorkoutPlan.user_id field references Users.user_id, establishing ownership of workout plans.
- The WorkoutSession.user_id and plan_id reference Users and WorkoutPlan respectively.
- The ProgressTracker.user_id links user performance metrics to their account.
- These relationships enable robust data validation, reporting, and traceability throughout the system.

This strategy allows the Virtual Fitness Trainer to manage and secure user data effectively while enabling modular expansion in future iterations.

8.7 Alternatives Considered

There were a few technologies and design options under consideration during planning. A NoSQL solution such as MongoDB was looked at based on flexibility and scalability but was dropped due to high relational modeling and consistency needs, especially in watching performance over a period of time. Firebase/Firestore was also looked at as a mobile-focused platform with real-time changes. While tempting for rapid development, its weak querying and indexing capabilities made it less suitable for our feature set. Additionally, a multi-SQL database design was also considered, allowing modular isolation of features like user management and progress tracking. This would, however, complicate query logic and backups without offering significant advantages at our system size.

8.8 Trade Off Discussion

When designing the VFT's data management strategy, several tradeoffs were considered:

8.8.1 SQL vs. NoSQL

We selected PostgreSQL, a relational SQL database, due to its strong data consistency, integrity, and support for complex queries. While NoSQL solutions such as MongoDB offer schema flexibility and faster development in some scenarios, they lack the robust relationships and transactional support required by a fitness platform dealing with sensitive health data and multiple interrelated entities.

8.8.2 Single vs. Multiple Databases

A single database structure simplifies the system’s deployment, backup routines, and enforcement of referential integrity. However, multiple databases could improve modular scalability (e.g., separate services for progress tracking and community features). For this project’s scope, a unified schema reduces complexity and increases maintainability.

8.8.3 Caching with Redis

Redis dramatically improves the responsiveness of real-time features such as live workout tracking and AI recommendations. However, it introduces a need for proper cache invalidation and consistency management. Redis is used selectively where speed is prioritized over strict persistence.

8.8.4 Cloud Storage for Media

Separating large media assets from transactional records (via AWS S3 or Firebase) improves database performance. The tradeoff lies in managing secure access to media, ensuring correct file permissions, and handling latency during content delivery.

Overall, the chosen strategy balances performance, scalability, and data integrity, while maintaining a flexible and secure foundation for future expansion.

GitHub link

https://github.com/idelgadod/CS250-Spring-2025