

## TABLE OF CONTENTS

<b>1. Introduction.....</b>	<b>1</b>
1.1 Purpose .....	1
1.2 Scope .....	1
1.3 Overview .....	2
1.4 Reference Material .....	2
1.5 Definitions and Acronyms.....	3
<b>2. System Overview .....</b>	<b>3</b>
<b>3. System Architecture.....</b>	<b>4</b>
3.1 Architectural Design.....	4
3.2 Decomposition Description .....	7
3.3 Design Rationale .....	10
<b>4. Data Design.....</b>	<b>12</b>
4.1 Data Description .....	12
4.2 Data Dictionary .....	12
<b>5. Component Design.....</b>	<b>13</b>
<b>6. Human Interface Design.....</b>	<b>17</b>
6.1 Overview of User Interface .....	19
6.2 Screen Images.....	19
6.3 Screen Objects and Actions.....	20
<b>7. Requirements Matrix.....</b>	<b>22</b>
<b>8. Appendices.....</b>	<b>23</b>

## **1. Introduction**

### **1.1 Purpose**

This software design document (SDD) describes the architecture and system design of a gym training site. The intended audience for this document may include software developers, system architects, project managers and quality assurance testers involved in the development of the gym training system. It can also be used as a reference for stakeholders interested in understanding the technical details of software design and architecture. SDD outlines high-level system design, including software components, data flow, user interface design and other technical details necessary for the implementation of the web-based gym training system.

### **1.2 Scope**

The gym training site is a web-based software system designed to provide users with a comprehensive platform for managing their fitness routine and achieving their health goals. The software enables users to create personalized workout plans, track their progress, and connect with trainers to get tips and advice.

The scope of the gym training site includes the following features:

- User account management: Users can create their profiles, set fitness goals, and track their progress over time.
- Workout planning and tracking: Users can create and customize workout plans based on their fitness goals and track their progress by logging their workouts.
- Community features: Users can connect with trainers, get workout tips.
- Content management: The site will provide users with access to fitness-related articles, videos, and other resources to help them achieve their goals.

The goal of the gym training site is to provide a comprehensive platform for individuals to manage their fitness routine and achieve their health goals. The objectives of the project include:

- Creating a user-friendly web interface that is easy to navigate and use.
- Developing a robust workout tracking system that can be customized to meet the needs of individual users.
- Providing users with access to high-quality content and resources to help them achieve their goals.

The benefits of the gym training site include:

- Increased motivation and accountability: By tracking their progress and connecting with trainers, individuals are more likely to stay motivated and accountable to their fitness goals.

- Customized workout plans: Users can create personalized workout plans based on their fitness goals and preferences.
- Access to resources: The site will provide users with access to a variety of fitness-related resources, including articles, videos, and other educational materials.

Overall, the gym training site aims to help individuals achieve their fitness goals in a fun, social, and engaging way.

### **1.3 Overview**

This document is a software design document (SDD) that outlines the architecture and system design of a gym training site. The purpose of this document is to provide a detailed description of the software design and architecture of the gym training site, as well as the goals, objectives and benefits of the project.

The document is organized into several chapters that provide a comprehensive overview of the design and functionality of the gym training site. Below is a brief overview of each section:

**Introduction:** This section provides an overview of the purpose, scope, and target audience of the document.

**Software Description:** This section describes the features and functionality of the gym training site, including user account management, workout scheduling and tracking, community features and content management.

**System Architecture:** This section provides an overview of the system architecture of the gym training site, including software components, data flow and other technical details.

**User Interface Design:** This section describes the user interface design of the gym training site, including layout, navigation, and visual design.

**Implementation Plan:** This section outlines the implementation plan for the gym training site, including the development process, project timeline, and resource requirements.

**Conclusion:** This section summarizes the key points of the document and provides recommendations for future improvements or enhancements to the gym training site.

Overall, this document is organized to provide a comprehensive understanding of the design and functionality of the gym training site as well as the steps required to implement the software and achieve the aims and objectives of the project.

## 1.4 Reference Material

Ref 1. IEEE Computer Society, "IEEE Standard for Information Technology—Systems Design— Software Design Descriptions" IEEE Std 1016-2009, March. 19, 2009.

## 1.5 Definitions and Acronyms

**Product:** The outcome of the project, the end product.

**User/Member:** Person who has subscribed to the gym membership.

**Trainer/Coach:** Person who is employed by the gym and works with users to create training plans.

**Payment:** The action taken by the user to obtain a membership.

## 2. System Overview

**Introduction:** The Software Design Document (SDD) provides a comprehensive overview of the functionality, content, and design of a gym site. This report outlines the architectural design, key components and implementation details of the project, which aims to create a robust and user-friendly gym management system.

**Project Background:** The gym site is designed to facilitate the management of a fitness center, offering features such as member registration, exercise customization, instructor management, attendance tracking and billing. The primary goal is to provide an efficient and intuitive platform for both gym managers and members to streamline their activities.

### Functionality:

The gym site covers the following core functionalities:

**a. Member Registration:** Allows individuals to sign up for gym memberships by providing their personal information, contact details and membership preferences.

**b. Exercise customization:** Allows gym users to create a personalized exercise program based on their desires and expectations.

**c. Trainer feedback:** It provides mutual communication between gym instructors and users.

**d. Attendance Tracking:** Helps managers monitor member attendance by providing a mechanism to track member participation in fitness classes.

**e. Billing and Payments:** Ensures accurate and timely billing for members by supporting the creation of invoices, tracking payments, and managing membership renewals.

**Context:** The gym site ensures code reusability, maintainability and extensibility. The site is designed to be accessible through a web-based interface that provides a seamless user experience across different devices and browsers.

**Design:** The design of the gym website includes the following components:

**a. User Interface (UI):** The UI is responsible for providing users with a visually appealing and intuitive interface, allowing them to interact with the system seamlessly. It includes features such as member registration, exercise customization forms, and forms for trainer feedback, as well as dashboards for administrators and members.

**b. Backend Logic:** The backend logic covers the business rules and processes of the gym site. It handles user requests, performs data validation, communicates with the database and organizes the flow of information between different components.

**c. Database:** The gym site uses a database to store and manage relevant information such as member profiles, class schedules, instructor details, attendance records, and billing information. It ensures data integrity and persistence across sessions.

**d. Security:** The gym site uses security measures, including user authentication and authorization mechanisms, to protect sensitive information.

**Outcome:** The gym website consists of a comprehensive gym management system covering features such as member registration, exercise customization, instructor feedback, attendance tracking and billing. By following sound software engineering principles and using appropriate design patterns, the project aims to provide a reliable, scalable, and user-friendly solution for gym managers and members.

### 3. System Architecture

#### 3.1 Architectural Design

Below is an overview of the modular program structure and the responsibilities assigned to each subsystem:

##### User Management Subsystem:

- Responsible for user registration and profile management.

- Manages user roles and permissions.
- Manages user data and access control.

#### Membership & Payment Subsystem:

- Handles membership plans and subscriptions.
- Handles membership registration, renewal, and cancellation.
- Manages membership-related data such as pricing, duration, and benefits.
- Manages the payment and billing process.
- Ensures secure and reliable financial transactions.

#### Training Subsystem

- Keeps track of the exercises the user has completed.
- Communicates with the Progress Tracking Subsystem to keep track of improvement.

#### Progress Tracking Subsystem:

- Monitors members' fitness progress and goals.
- Manages exercise routines, workout logs and achievements.

#### Exercise Library:

- Keeps data about each exercise, mainly categorized by their workout type.
- Provides links to the External Resource Subsystem where necessary.

#### External Resource Subsystem:

- Stores external resources such as instructional videos and workout guides.

#### Communication Subsystem:

- Facilitates communication between the gym employees and members through the Training Subsystem.
- Handles member inquiries, feedback and support requests.

These subsystems collaborate with each other to achieve the desired functionality of the system.:

-User Management Subsystem mainly interacts with the payment subsystem and the training subsystem as it manages user registration and permissions. It provides user data and access control to other subsystems based on their roles and permissions.

-The Membership and Payment Subsystem communicates with User Management during registration, renewal, or cancellation processes. It also provides membership-related data for charging and billing purposes in the process of making a payment. It also keeps track of the previous transaction records.

-The Training Subsystem with all other subsystems to bring the User Management Subsystem together with the Exercise Library. It is mainly used so that the user can view their training plan and mark the exercises they have completed as done. It communicates with the Progress Tracking Subsystem to keep track of the user's achievements and improvement.

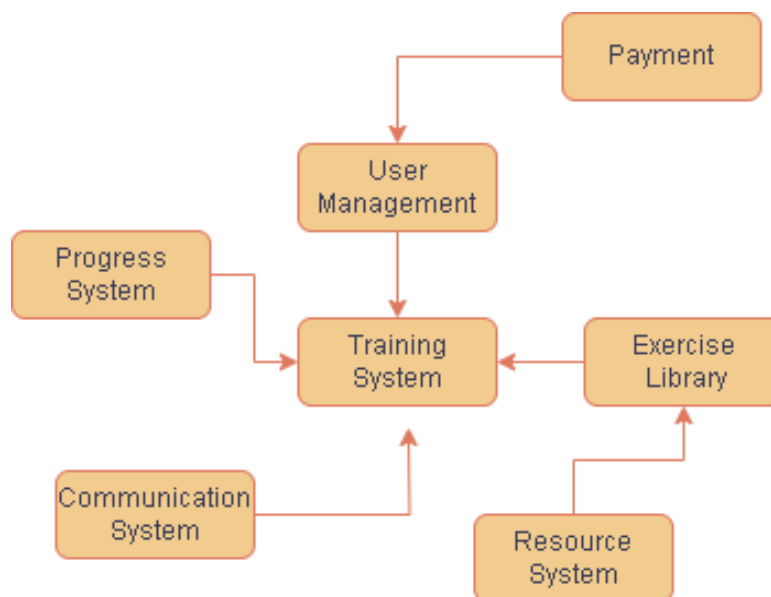
-The Progress Tracking Subsystem interacts with the Training Subsystem to access user profiles and retrieve relevant data to track fitness progress. It can also provide progress reports upon request.

-Exercise Library is used to show what exercises the user has in their training plan via the Training Subsystem. It also communicates with the External Resource Subsystem in order to provide more information about the exercises where necessary.

-External Resource Subsystem keeps external links to several workout and exercise related guides so that users can reach more detailed information through the Exercise Library.

-The Communication Subsystem interacts with User Management to access user contact information to send support responses. It may also receive user feedback and questions from User Management.

Here is a simplified diagram showing the major subsystems and their interconnections:



### 3.2 Decomposition Description

#### User Management Subsystem:

**User registration:** The subsystem allows new users to register by providing necessary information, such as username, email address, and password. It validates the input, checks for duplicate usernames or email addresses, and creates a new user account.

**Profile management:** Users can update their profile information through the subsystem. This includes managing personal details, contact information, and preferences. The subsystem ensures the accuracy and integrity of the user profile data.

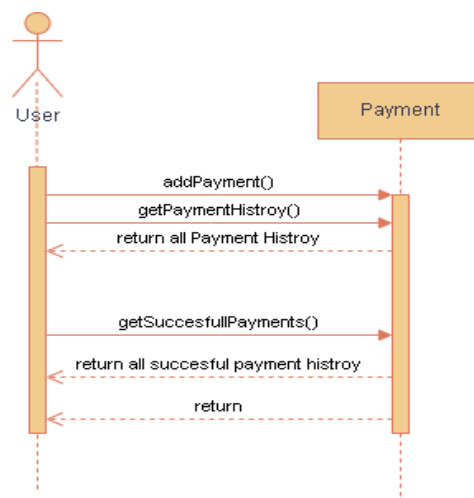
**User roles and permissions:** The subsystem manages different user roles and permissions within the system. It defines roles such as administrator, moderator, or regular user and assigns appropriate permissions to each role. User roles determine the level of access and functionality available to users within the system.

#### Membership & Payment Subsystem:

**Membership plans and subscriptions:** The subsystem manages different membership plans and subscription options available to users. It stores and maintains membership-related data, including pricing, duration, benefits, and any additional details associated with each plan or subscription option.

**Payment and billing process:** The subsystem facilitates the payment and billing process for membership fees. It integrates with payment gateways or third-party payment processors to securely handle financial transactions. Users can make payments for membership fees through various payment methods, and the subsystem ensures secure and reliable transactions.

**Financial transaction security:** The subsystem prioritizes the security of financial transactions. It implements encryption, secure communication protocols, and compliance with industry standards to protect sensitive financial information. It may also include mechanisms for fraud detection and prevention.





### Training Subsystem

**Exercise completion tracking:** The subsystem keeps a record of the exercises completed by users. It tracks the exercises attempted, completed, and the corresponding timestamps. The completion status of each exercise is recorded to monitor the user's progress.

**Progress monitoring and improvement tracking:** The subsystem communicates with the Progress Tracking subsystem to track the user's improvement over time. It shares exercise completion data with the Progress Tracking subsystem, allowing it to analyze the user's performance, calculate scores or metrics, and generate insights on the user's progress and improvement.

**Communication with Progress Tracking subsystem:** The Exercise Tracking subsystem establishes communication channels with the Progress Tracking subsystem to exchange relevant data. This communication enables the Progress Tracking subsystem to update the user's overall progress based on the exercises completed and provide personalized feedback or recommendations.

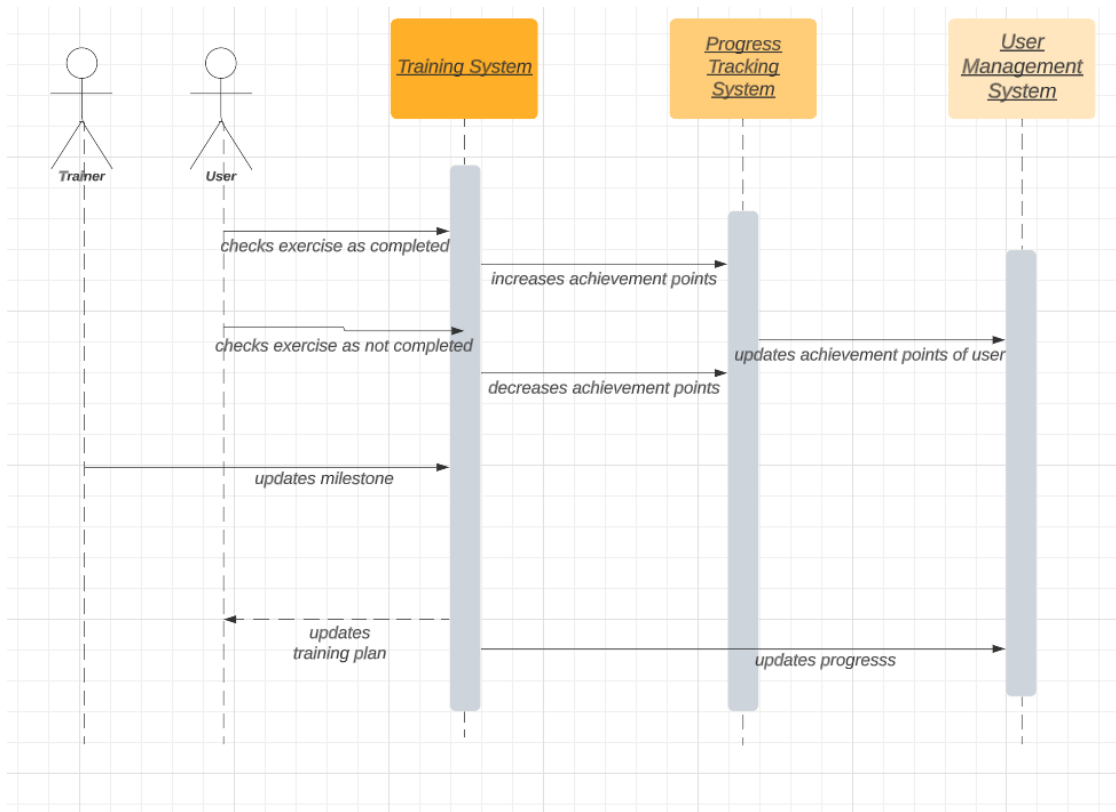
### Progress Tracking Subsystem:

**Goal management:** The subsystem allows members to set and manage their fitness goals. Members can define specific objectives, such as weight loss, muscle gain, or improved endurance. The subsystem tracks progress towards these goals and provides visualizations or indicators to help members stay motivated and focused.

**Fitness progress monitoring:** The subsystem continuously tracks and monitors members' fitness progress. It collects data from various sources, such as exercise completion records, performance metrics, and body measurements. It analyzes this data to assess members' progress towards their fitness goals.

**Achievement tracking:** The subsystem tracks members' achievements in their fitness journeys. This can include milestones reached, personal bests, or badges earned based on specific accomplishments. Achievements provide recognition and motivation to members as they progress towards their goals.

**Exercise routines:** The subsystem manages exercise routines for members. It provides a library of pre-defined routines or allows members to create personalized routines. The subsystem may include features such as exercise recommendations, progression tracking, or guidance on proper form and technique.



### Exercise Library:

**Exercise data management:** The subsystem stores and manages data about each exercise. This includes exercise names, descriptions, instructions, and any additional relevant information. The exercise data is structured in a way that facilitates categorization and retrieval.

**Categorization by workout type:** The subsystem categorizes exercises based on their workout types, such as strength training, cardio, flexibility, or specific sports-related exercises. This categorization allows users to browse and search for exercises based on their specific workout goals or preferences.

**External Resource Subsystem integration:** The Exercise Library subsystem provides links to the External Resource Subsystem when additional external resources are available for an exercise. These external resources may include video demonstrations, articles, or tutorials related to the exercise. Integration with the External Resource Subsystem enhances the user experience by providing comprehensive and supplementary information.

**Exercise variations and progressions:** The subsystem may include information about variations or progressions of exercises. For example, it may provide different levels of difficulty or modifications to suit users with varying fitness levels or specific needs. This allows users to progress and challenge themselves gradually.

### External Resource Subsystem:

**Storage and organization:** The subsystem provides storage capabilities for external resources. It stores files or references to external resources, such as video files, documents, or URLs. The resources are organized in a structured manner for easy retrieval and management.

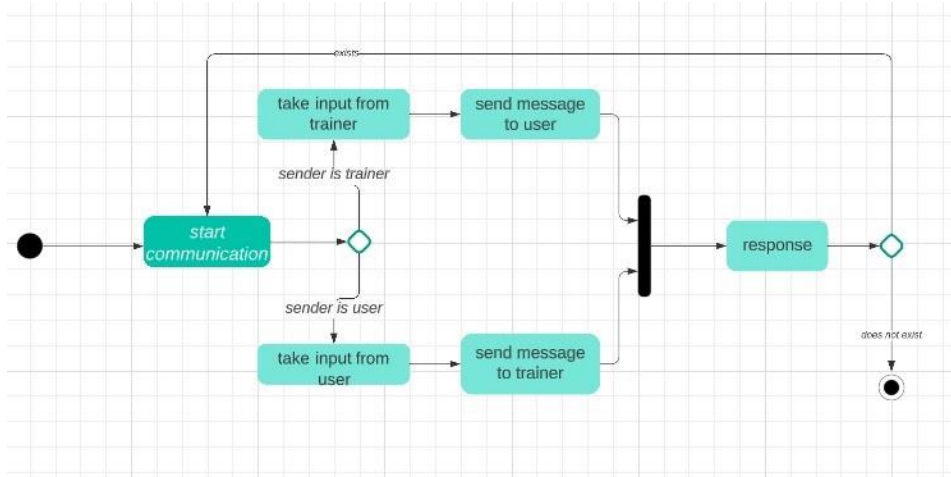
**Instructional videos:** The subsystem accommodates instructional videos related to exercises, workouts, or fitness routines. It stores video files or links to video hosting platforms. These videos provide visual demonstrations and guidance for users to learn proper exercise techniques or follow workout routines.

**Categorization and tagging:** The subsystem categorizes external resources and applies relevant tags or metadata. This categorization enables efficient searching, filtering, and retrieval of resources based on their content, topics, or intended use. Users can easily locate resources within the system based on their specific needs or preferences.

### Communication Subsystem:

**Communication channels:** The subsystem provides various channels for communication between gym employees and members. This may include email, chat/messaging systems, notification systems, or dedicated communication interfaces within the system. These channels enable seamless and convenient communication between the parties involved.

**Feedback management:** The subsystem manages member feedback. It provides mechanisms for members to submit feedback, suggestions, or complaints regarding their gym experience, training programs, facilities, or other aspects. It collects and organizes feedback data for further analysis and action.



### 3.3 Design Rationale

The rationale for selecting the architecture, with the identified subsystems, is to achieve a modular and scalable design that promotes separation of concerns and maintainability. The critical issues and trade-offs considered during the selection process are shown below, as well as alternative architectures that were evaluated:

Critical Issues Considered:

**Modularity and Separation of Concerns:** The chosen architecture decomposes the system into distinct subsystems, each responsible for specific functionalities. This promotes modularity, making it easier to understand, develop, test, and maintain individual components.

**Scalability and Flexibility:** By dividing the system into subsystems, the architecture allows for the independent scaling of different components based on demand. It provides flexibility to add or modify subsystems to accommodate future growth and evolving requirements.

**Collaboration and Integration:** The subsystems collaborate through well-defined interfaces and interactions, enabling seamless communication and data exchange between components.

Trade-offs and Considered Architectures:

**Monolithic Architecture:** An alternative considered architecture is a monolithic approach, where all functionalities are tightly integrated into a single codebase. While a monolithic architecture can be simpler to develop initially, it can become complex and challenging to maintain as the system grows. The lack of modularity makes it difficult to scale and can lead to dependencies between different functionalities.

**Microservices Architecture:** Another architecture considered is a microservices approach, where each functionality is developed as an independent service. While microservices offer benefits such as flexibility, scalability, and fault isolation, they introduce additional complexity in terms of service orchestration, communication, and deployment. Considering the scope and complexity of a gym site, a microservices architecture might be overly complex and resource intensive.

**Layered Architecture:** A layered architecture separates the system into horizontal layers, such as presentation, business logic, and data access. While this approach provides a good separation of concerns, it may not offer the same level of modularity and scalability as the chosen subsystem-based architecture. Layered architectures can also lead to tight coupling between layers, making maintenance and updates challenging.

In the context of a gym site, the selected architecture strikes a balance between simplicity, maintainability, and scalability. It allows for modular development, easy integration of new functionalities, and independent scaling of subsystems. The chosen architecture also aligns with industry best practices for web system development, facilitating future enhancements and modifications.

## **4. Data Design**

### **4.1 Data Description**

The system as a whole stores the data it obtains from its functional methods. These data are then transformed into data structures in a standard form and are classified and stored in a database management system that provides capabilities such as data retrieval, storage, and manipulation. The major data entities in this are as follows:

#### **User Profile**

The user profile stores data about the personal information of the user (such as name, age, gender etc.), their contact information (the phone number, the email address), and their subscription status. The data are transformed into a standard format and stored in a User Profile database.

#### **Exercise Library**

The exercise library stores exercises in a categorized way based on their exercise type, intensity level, a list of necessary equipment, the muscle group the exercise targets. It is kept in an Exercise Library database.

#### **Training Plans**

Customized training plans for each user are based on users' preferences, goals, and limitations that are created by trainers. These plans keep a user ID, list of exercises, the trainer's contact information, and a set milestone for progress comparison and are stored in the according Training Plan database.

#### **Progress Data**

The progress data consists of the user's data (most importantly their achievement points), and the training plan data that stores the milestone set by the trainer. This data is stored in a Progress database, which allows for easy comparison of users' progress and adjustment of their training programs accordingly.

#### **Resource Library**

The Resource Library database consists of a list of resources including instructional videos, workout guides, and nutritional advice. These resources are categorized by their type, workout type and the data stored includes the name of the exercise and the relevant muscle groups.

#### **Payment Records**

The payment records store information about the credit/debit card information of the user, the user information including their subscription, and previous transactions in the payment records database. In order to ensure the secure management of user payment information and to speed the handling of subscription-related procedures, the data is securely stored in a separate Payment Records database.

## 4.2 Data Dictionary

### Exercise

#### Attributes

**String** name: exercise name

**String** type: type of exercise (category it belongs to)

**String** intensity: level of intensity (easy/medium/hard)

**String []** necessary\_equipment: array of equipment the exercise requires

**String** muscle\_group: the muscle group the exercise targets

### Exercise Library

#### Attributes

**Exercise [] []** exercise\_lists: holds the array of exercises for each type of workout such as 'cardio\_list'

#### Methods

**addExercise**(String type, Exercise e): adds exercise *e* to the exercise array *type*

**removeExercise**(String type, Exercise e): removes exercise *e* from the exercise array *type*

**Exercise []** **getExercises** (String type): returns the array *type* from exercise\_lists

### Payment

#### Attributes

**User** user: the user who makes the payment

**String** name\_on\_card: the name on the card used to make the payment

**String** card\_no: the card number used to make the payment

**int** cvv\_no: the cvv security code of the card

**String** valid\_thru: the last date the card can be used

**int** price: the price paid

**boolean** successful: shows whether the payment was successful

#### Methods

**makePayment**(User u, String name\_on\_card, String card\_no, int cvv\_no, String valid\_thru, int price): connects to bank API and processes payment using the given *name\_on\_card*, *card\_no*, *cvv\_no*, and

*valid\_thru* in the amount of *price* and assigns the result to a boolean and calls ***changePlanStatus*** and ***changeEndDate*** accordingly

**changePlanStatus**(User *u* , Boolean *b*, int *price*): changes the purchased membership plan to the plan that has *price* of the user *u* if *b* is true

**changeEndDate**(User *u* , Boolean *b*): calculates and changes the end date of subscription of the user *u* if *b* is true

## Payment Records

### Attributes

**Payment []** *payment\_history*: an array that stores all payments

### Methods

**addPayment**(Payment *new\_payment*): add *new\_payment* to *payment\_history*

**Payment []** **getPaymentHistory**(): returns all payments

**Payment []** **getSuccessfulPayments** (): returns all successful payments

**Payment []** **getPayments** (User *u*): returns all payments of the user *u*

## Progress Data

### Attributes

**User** *user*: the user whose progress is tracked

**int []** *milestone*: the number of exercises user needs to complete before getting a new training plan, it is used to determine the progress of the user by comparing it with a user's achievement point

### Methods

**int** **checkProgress**(): returns how many more points are needed to reach the milestone

**updateAchievementPoints**(int *i*): updates the achievement points with *i*

**updateMilestone**(int *i*): updates the milestone with *i*

## Resource

### Attributes

**String** *name*: exercise name

**String** *resource\_type*: type of resource

**String** *workout\_type*: type of exercise

## Resource Library

### Attributes

**Resource []** resources: list of all resources in the system

### Methods

**addResource**(Resource resource): adds *resource* to the resource library

**removeExercise**(Resource resource): removes *resource* from the resource library

**Resource [] getResources ()**: returns the list of resources

## Training Plans

### Attributes

**User** user: the user the training plan belongs to

**Exercise []** training\_list: list of exercises planned for the user

**boolean []** exercises\_done: keeps track of the exercises the user has completed

**String** trainer\_email: the contact information (email) of the trainer

### Methods

**addExercise**(String type, Exercise e): adds exercise *e* to the exercise array *type*

**removeExercise**(Exercise e): removes exercise *e* from the exercise array *type*

**Exercise [] getExercise ()**: returns the array *type* from exercise\_lists

**markCompleted**(Exercise e): marks exercise as completed in the boolean array and increases achievement point by one.

**markNotCompleted**(Exercise e): marks exercise as not completed in the boolean array and decreases achievement point by one.



## **User**

### **Attributes**

**int** userID: unique identifier for each

**String** name: user's name

**int** age: user's age

**char** gender: user's gender

**String** email: user's contact information, email address

**String []** equipment: array of equipment user has

**String** workout\_type: type of selected workout

**String** workout\_intensity: preferred intensity of workout

**boolean** has\_injury: shows if the user has an injury to be mindful of

**String []** injury\_places: shows the places of injury

**String** goals: description of the user's goals

**int** achievement\_point: points the user has earned by completing exercises

**String** subscription\_type: type of subscription a user has

**Date** subscription\_end: end date of the subscription

## 5. Component Design

```
Exercise[] create_workoutlist(User u,Trainer t){
    String wt = u.workout_type();

    Exercise[] temp ;
    //temp = list of the selected workout type

    boolean b = false;
    String level = u.intensity_level();

    for(int i=0;i<temp.length;i++){
        String[] equip = u.hasEquipment();
        if(user doesnt have necessary equipments){
            //remove this exercise from temp;}
        if(temp[i].intensity != u.workout_intensity){
            //remove this exercise from temp;}
        if(user has injury that temp[i] affects){
            //remove this exercise from temp;}

    }

    return temp;
}
```

**Personalized Training Plan:** The Training Plan Creation Module creates a personalized training list for a user based on the intensity level of the workout they choose whether they have a certain discomfort, and whether they have equipment. It takes in a User object and a Trainer object as parameters and returns an array of Exercise objects. The function first assigns the user's preferred workout type to the variable wt and initializes the temp array with Exercise objects that match the user's preferred workout type. It then checks if the user has the necessary equipment, preferred intensity level, and any injuries that may be affected by each Exercise object in the temp array and removes any that don't meet the user's preferences. Finally, the function returns the temp array, now containing only the Exercise objects that meet the user's needs.

```
void payment(User u,String plan){
    int price;
    if(plan.equals("Classic")){
        price = 620;}
    else if(plan.equals("Gold")){
        price = 810;}
    else{
        //Incorrect input ERROR}
        //Scan credit card informations(credit card number,name on card,
        valid thru, cvv)

    make_payment(u, name, card_no, cvv, valid_thru, price);
}
```

```
void make_payment(User u, String name, String card_no, int cvv, String
valid_thru, int price) {

    //connect to bank API and process payment
    //assign if succesful to a boolean successful

    //change the membership plan and end date accordingly
    changePlanStatus(u, successful, price);
    changeEndDate(u, successful);

    //create new payment
    addPayment(payment); }
```

**Payment System:** The Payment Processing Module processes a payment for a user based on their selected membership plan. The payment function initializes the price variable based on the selected membership plan, prompts the user to enter their credit card information, and calls the make\_payment function. The make\_payment function connects to the bank API to process the payment, updates the user's membership plan status and end date based on the payment status, and creates a new payment record for the user.

```

void exercise(TrainingPlan tp){
    Exercise[] list = tp.getExercise();
    int done;

    while(done!=list.length()){
        //get input from the user about what exercises they
        mark

        if(input = exercise done) {
            markCompleted(exercise);
            done++;
        }

        else if(input = exercise not done) {
            markNotCompleted(exercise);
            done--;
        }
    }
}

```

```

void markCompleted(Exercise e){
    for(int i=0; i<list.length(); i++){
        if(list[i] == e)
            //mark exercise as True on the boolean array
    }

    User u = tp.user();
    u.achievement_point ++;
}

void markNotCompleted(Exercise e){
    for(int i=0; i<list.length(); i++){
        if(list[i] == e){
            //mark exercise as False on the boolean array
        }

        User u = tp.user();
        u.achievement_point --;
    }
}

```

**Progress Tracking:** The Progress Tracking Module tracks a user's progress in their customized training plan. The exercise function takes in a Training Plan object as a parameter, gets the list of exercises from the training plan, and prompts the user to input which exercises they have completed. The markCompleted and markNotCompleted functions are called to update the completion status of the exercises and adjust the user's achievement points accordingly. The mark Completed function marks the exercise as completed in the boolean array and increases the user's achievement points. The mark Not Completed function marks the exercise as not completed in the boolean array and decreases the user's achievement points.

**Communication System:** The Communication Module facilitates communication between a user and their trainer. The communicate function takes in a TrainingPlan object as a parameter, gets the user and trainer email addresses from the training plan, and prompts the user to input a message to send to their trainer. The function then determines the sender and receiver of the message based on whether the user or trainer initiated the communication. The message is then sent to the appropriate recipient.

```

void communicate(TrainingPlan tp){

    User u = tp.user;
    String user_email = u.email;
    String trainer_email = tp.trainer_email();

    String sender, receiver;
    if(user sends message) {
        sender = user_email;
        receiver = trainer_email;
    }
    else{
        sender = trainer_email;
        receiver = user_email;
    }

    //get text input from sender

    //send input to receiver
}

```

## **6. Human User Interface Design**

### **6.1 Overview of User Interface**

The system offers five key features designed to enhance user experience and provide convenient functionality.

**Customization Form:** Users are required to provide important personal information such as their age, weight, height, and fat percentage via a form. This information is used to generate a personalized fitness program tailored to the user's needs. Upon completing the form, the system automatically recommends the appropriate fitness program for the user.

**Exercise Tracking:** Users have access to a training program created by the coaches. They can track their progress by recording the amount of exercise they have completed. The system provides a user-friendly dashboard where users can visualize and monitor their progress. This feature serves as a source of motivation and allows users to track their improvement over time.

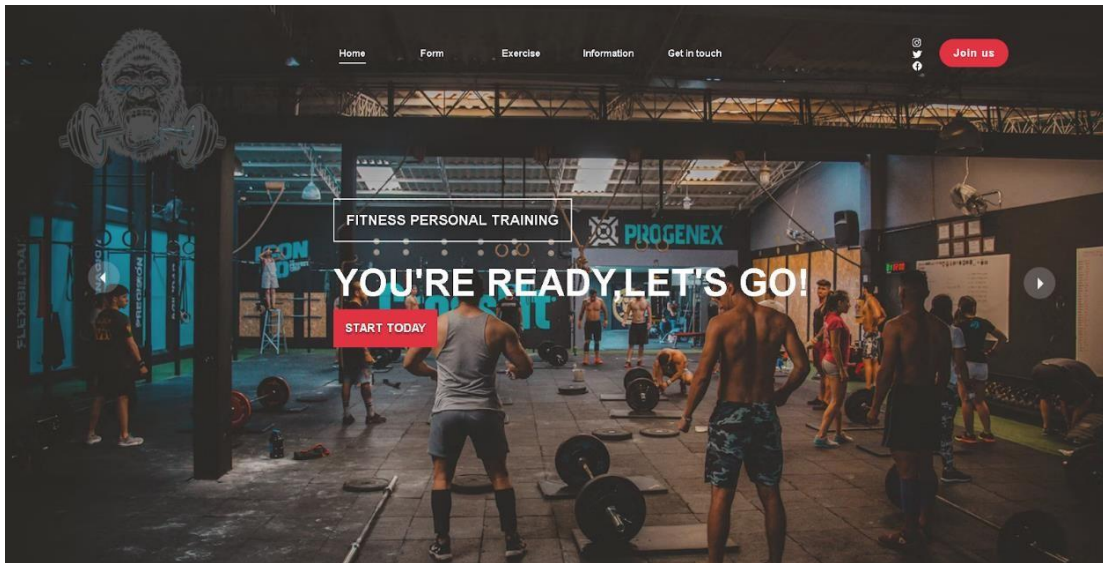
**External Resources:** The system provides a comprehensive information section where users can access useful information related to their fitness journey. Users can find detailed information about different exercises on a single page. Additionally, the system offers access to various instructional videos to further support users' fitness knowledge.

**Communication Platform:** Users can engage in communication with their trainers using the communication section. They can seek guidance, ask questions, and receive feedback on their workouts. The system ensures efficient communication by delivering notifications based on the user's inputs, ensuring prompt responses and assistance.

**Secure Payment System:** Users can purchase memberships by using the payment system. The system offers multiple payment options to cater to users' preferences. Once a transaction is completed, users receive a confirmation message to acknowledge the successful payment.

These features are designed to provide users with a seamless and user-friendly experience, ensuring their comfort and convenience throughout their fitness journey.

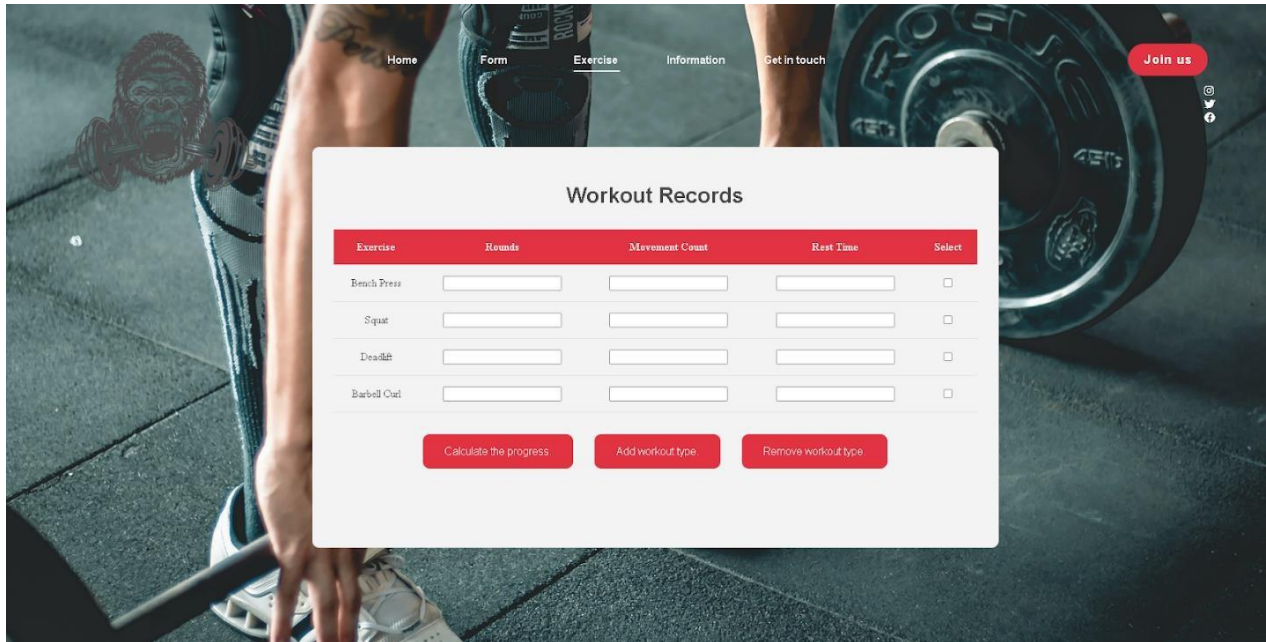
## 6.2 Screen Images



Home page.

A screenshot of the Form page of the Gym Management System. The background is a wooden surface with gym equipment, including a barbell and a resistance band. In the top left corner, there is a logo featuring a gorilla's head with a barbell. The top navigation bar includes links for Home, Form, Exercise, Information, and Get in touch. On the right side of the navigation bar, there are social media icons and a red 'Join us' button. The main content area is a white form with the following fields: Full Name, Age, Weight, Height, Fat percentage, and Which exercise do you prefer?. At the bottom of the form is a red 'Submit' button.

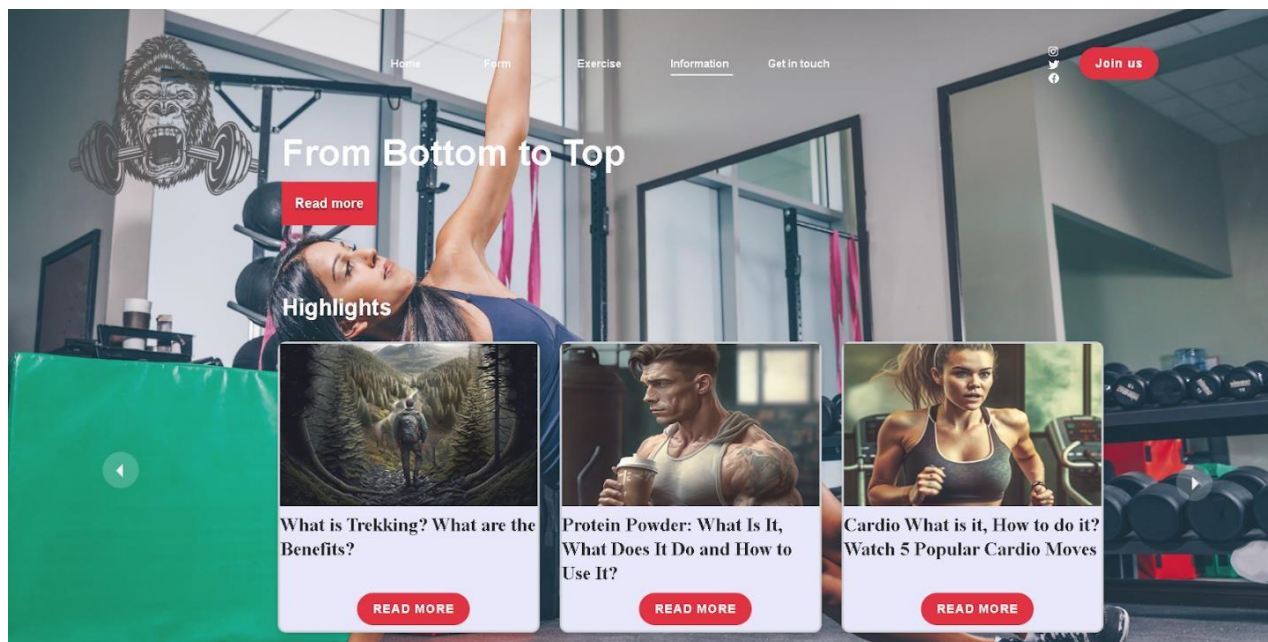
A form in which the training program will be prepared according to the data to be entered by the user.



Exercise	Rounds	Movement Count	Rest Time	Select
Bench Press	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
Squat	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
Deadlift	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
Barbell Curl	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>

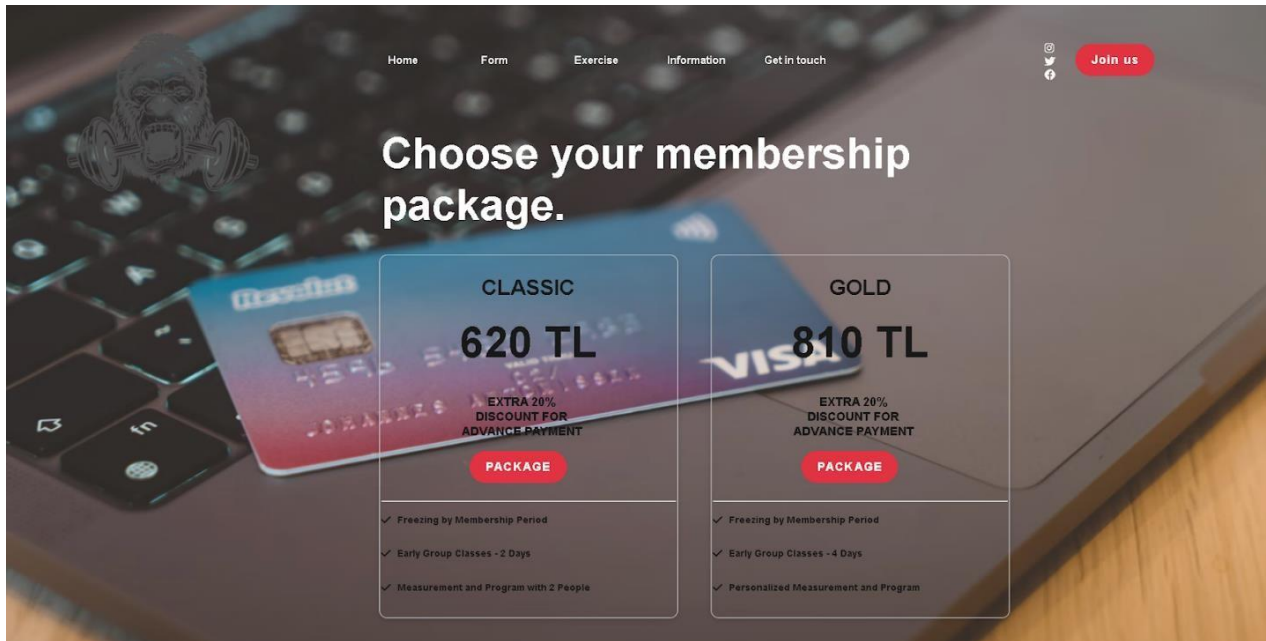
[Calculate the progress](#) [Add workout type](#) [Remove workout type](#)

Exercise page where the user will receive various rewards after entering data.



Library page where the user will find various exercise and sports related information and videos.





The page where the user can make payments and purchase subscriptions.

### 6.3 Screen Objects and Actions

On the main page, users can log in to other pages using the categories displayed. The "Join Us" option allows access to the payment system to become a member of the GYM. Social media buttons are available to log in to the GYM's social media accounts. The "Start Today" section serves the same purpose as the "Join Us" section. Images on the homepage can be changed using the arrow options on the side.

The form page shares similarities with the homepage, except for the form itself. After filling in the required information in the form section, users can submit the form to their trainers using the "Submit" button. Trainers then determine the user's future training program based on the provided information.

In the Exercise section, a reward system is implemented based on results such as "Rounds," "Movement Count," and "Rest Time." Users can calculate their progress by clicking on the "Calculate the progress" button. Relevant information is entered in the corresponding input sections. The "Add workout type" option allows users to include their own movements in the record, while the "Remove workout type" option enables the deletion of specific movements from the training program.

The Information section provides the ability to read various topics using the "Read More" options. Users can navigate between different reading sections using the side arrows.

In the "Get in Touch" section, users enter their name in the Name section, select the desired coach's email address in the To Instructor section, and write their message in the message section. The message can be sent to the coach by clicking the "Submit" button.

The Payment section allows users to choose their preferred membership type by selecting the appropriate package button.

## 7. Requirements Matrix

FUNCTIONAL REQUIREMENT	SYSTEM COMPONENT	DATA STRUCTURES
Customization Form	User Interface Module	User Profile Database
Personalized Training Plan	Training Plan Generation Module	User Profile Database Exercise Library Database Training Plans Database
Progress Tracking	Progress Tracking Module	User Profile Database Training Plans Database
Communication System	Communitacion Module	User Profile Database Training Plans Database
External Resource Library	Resource Management	Resource Library Database
Payment Management	Payment Processing Module	User Profile Database Payment Records Database

**Customization Form:** Customization Form functional requirement is satisfied within The User Interface Module, while the User Profile Database stores the data structures related to the User Profile. This ensures that the user's personal and contact information is stored in a standardized format and can be easily accessed by the system.

**Personalized Training Plan:** Training Plan Generation Module requirement satisfied the Personalized Training Plan functional requirement, while the User Profile Database, Exercise Library Database, and Training Plans Database store the necessary data structures related to the user profile, exercise library, and training plans. This ensures that customized training plans can be generated for each user based on their preferences, goals, and limitations, and that the necessary data is stored in a standardized format for easy access by the system.

**Progress Tracking:** Progress Tracking Module requirement satisfied the Progress Tracking functional requirement, while the User Profile Database and Training Plans Database store the necessary data structures related to the user profile and training plans. This ensures that progress can be tracked for each user based on their customized training plan, and that the necessary data is stored in a standardized format for easy access by the system.

**Communication System:** Communication Module requirement satisfied the Communication System functional requirement, while the User Profile Database and Training Plans Database store the necessary data structures related to the user profile and training plans. This ensures that the communication system can access the necessary data to facilitate communication between users and trainers, and that the necessary data is stored in a standardized format for easy access by the system.



**External Resource Library:** Resource Management system component requirement is satisfied by the External Resource Library functional requirement, while the Resource Library Database stores the necessary data structures related to the resource library. This ensures that the external resource library can be accessed by users and trainers, and that the necessary data is stored in a standardized format for easy access by the system.

**Payment Management:** Payment Processing Module requirement satisfied the Payment Management functional requirement, while the User Profile Database and Payment Records Database store the necessary data structures related to the user profile and payment records. This ensures that payment information can be securely managed and processed for each user, and that the necessary data is stored in a standardized format for easy access by the system.