GYM MANAGEMENT SYSTEM

Mini -Project report submitted to
Kristu Jyoti College of Management and Technology,
Changanacherry

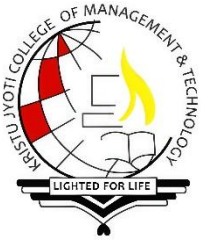
In partial fulfillment of the requirements for the award of the

BACHELOR OF COMPUTER APPLICATION

BY
ANAND S 230021080179

JOBAN SEBASTIAN 230021080221

Under the guidance of
MS. ABY ROSE VARGHESE



DEPARTMENT OF COMPUTER APPLICATIONS

KRISTU JYOTI COLLEGE OF MANAGEMENT
AND TECHNOLOGY, CHANGANACHERRY
NOVEMBER, 2025

Kristu Jyoti College of Management and Technology, Changanacherry
DEPARTMENT OF COMPUTER APPLICATION

CERTIFICATE

Certify that the report entitled *"GYM MANAGEMENT SYSTEM"* is a bonafide record of the mini-project work done by ANAND S (230021080179) and JOBAN SEBASTIAN (230021080221) under our guidance and supervision is submitted in partial fulfillment of the Bachelor Degree in Computer Applications, awarded by Mahatma Gandhi University Kerala and that no part of this work has been submitted earlier for the award of any other degree.

Ms. Aby Rose Varghese
Project guide

Mr.Roji Thomas
HOD

External Examiner

Internal Examiner

DECLARATION

I hereby declare that the project work entitled **"GYM MANAGEMENT SYSTEM"** submitted to Mahatma Gandhi University in partial fulfillment of requirement for the award of degree of Bachelor of Computer Application from Kristu Jyoti College of Management and Technology, Changanacherry is a record of bonafide work done under the guidance of **Ms.Aby Rose Varghese**, project guide, Kristu Jyoti College of Management and Technology ,Changanacherry.This project has not been submitted in partial or fulfillment of any other degree/diploma/fellowship or similar of this university or any other university.

Place : Changanacherry

Date : NOVEMBER 2025

ANAND S 230021080179

JOBAN SEBASTIAN 230021080221

ACKNOWLEDGEMENT

I would like to express our thankfulness towards a number of people who have been instrument in making of this project. First of all, I thank the college management who has been with me and provide most helpful facilities throughout the completion of the project. I am greatly indebted to Rev.Fr. Joshy George CMI, Principal Kristu Jyoti College, who whole heartedly gave me permission and whose advice was a real encouragement. I am deeply indebted to Ms.Aby Rose Varghese, Project Guide, for the valuable discussion and helpful counseling. I sincerely indebted to Asst. Prof. Mr. Roji Thomas, HOD, Department of Computer Applications, for the valuable guidance. Last but not least, I wish to express my gratitude and heartfelt thanks to my friends, family and whom with their valuable advice, encouragement and support, contributed to the successful completion of the project. Above all, I thank God Almighty.

ANAND S

JOBAN SEBASTIAN

INDEX

INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The *CrossFit Gym Management System* is a web-based project designed to make gym operations easier and more organized. It helps manage daily activities such as maintaining member records, assigning trainers, and keeping track of fitness packages in a simple digital way. This system also reduces paperwork, saves time, and improves communication between the admin, trainers, and members, making the entire gym management process smoother and more efficient.

In addition to managing gym data, the system allows trainers to interact with members and share fitness updates through an online platform. It also enables members to register, choose workout packages, and book trainers from anywhere at any time. Overall, this project helps create a modern, user-friendly, and transparent environment for both staff and clients in the gym.

1.1 PROBLEM STATEMENT

In traditional gym environments, administrators rely heavily on manual registers or spreadsheets to store member and trainer details. This approach often results in redundant data, slow updates, and limited access to critical information. Communication between trainers and members is also unstructured, leading to confusion and inconsistent results. Moreover, managing supplement stock and package subscriptions manually increases the risk of mismanagement and financial discrepancies.

The absence of an integrated system results in delays, mismanagement, and poor customer experience. Moreover, manual systems do not provide data security, real-time updates, or systematic tracking of activities. This creates a need for an automated platform that not only simplifies operations but also ensures reliability, accessibility, and effective communication between all stakeholders.

These challenges highlight the need for a centralized system that can automate all the key processes of a gym. The *CrossFit Gym Management System* aims to resolve these problems by providing an integrated digital solution that ensures data accuracy, real-time access, and efficient communication among all use

1.3 SCOPE AND RELEVANCE OF THE PROJECT

The *CrossFit Gym Management System* is developed to handle and organize all the major activities that take place in a fitness center. The system focuses on three main users — **Admin, Trainer, and Member** — each having specific roles and responsibilities. The Admin can manage trainers, members, packages, and supplement stocks; Trainers can view their clients, provide feedback, and manage their schedules; and Members can book packages, request trainers, and buy supplements online. This ensures that all gym operations are carried out in a smooth, efficient, and well-coordinated manner.

The project's relevance lies in its ability to bring automation and accuracy to the fitness industry. In today's world, where most services are becoming digital, this system helps gyms reduce manual errors, save time, and maintain proper records. It also improves communication between trainers and members through an online platform, making the gym experience more convenient and transparent. Overall, this project supports the modernization of fitness centers by combining technology with effective management.

1.4 OBJECTIVES

The main objective of the *CrossFit Gym Management System* is to create a user-friendly web-based platform that helps in the smooth and efficient management of gym activities. The project aims to reduce manual work and make all operations digital, accurate, and time-saving. It focuses on bringing all users — the admin, trainers, and members — under one system where each can perform their specific roles effectively. The system allows the admin to manage trainers, members, packages, and supplements with ease, while trainers can monitor clients and share feedback. Members can register, book packages, and buy supplements online, making the entire process more convenient. Overall, the project's objective is to ensure proper coordination, improve communication, and provide a transparent and efficient way to handle the daily tasks of a modern gym.

SYSTEM ANALYSIS

CHAPTER – 2

SYSTEM ANALYSIS

2.1 INTRODUCTION

The first stage of any software project, often known as the preliminary assessment, involves a detailed study of the system under consideration. System analysis helps in understanding the current working process, identifying its limitations, and designing solutions that improve performance. It is a problem-solving process that consists of several key phases—each requiring careful planning, investigation, and evaluation.

The main purpose of this stage is to determine the feasibility of the proposed project and understand the exact requirements of users. It begins with a feasibility study to check if the project can be implemented with the available time, cost, and resources. After feasibility evaluation, the next step is to study the existing system in order to identify inefficiencies, user needs, and areas that require improvement.

In this project, the system analysis aims to understand how gyms currently manage their operations such as member registration, trainer assignment, package booking, and supplement tracking. The findings from this phase help define a clear set of functional requirements and lead to the design of an effective digital solution—*The CrossFit Gym Management System*—that automates and simplifies these tasks.

2.2 EXISTI1NG SYSTEM

In many fitness centers today, gym operations are still managed manually or through basic digital tools like spreadsheets. Member details, trainer records, and package information are often stored in physical files or entered manually, which makes the process slow and prone to errors. Communication between trainers and members is usually done in person or through informal means like calls or messages, leading to inefficiency and lack of proper documentation,supplement stock difficult. As a result, both staff and members face confusion and delays in performing their daily tasks.

2.2.1 LIMITATIONS OF EXSTING SYSTEM

❖ **Lack of Centralized Data Management:**Member, trainer, and supplement records are scattered or manually maintained, making it hard to track and update information efficiently.

❖ **Communication Gaps:**There is no dedicated system for trainers and members to communicate effectively, resulting in poor coordination and inconsistent updates.

❖ **Error-Prone Manual Work:**Manual handling of registrations, bookings, and stock details often leads to duplication and data loss.

❖ **No Real-Time Updates:**Members and trainers cannot access updated package details or supplement availability in real time.

❖ **Limited Accessibility:**All tasks require physical presence or manual intervention, reducing convenience for both users and management.

These limitations highlight the need for an automated gym management system that simplifies daily operations, reduces paperwork, and ensures better coordination among all users.

2.3 PROPOSED SYSTEM

The *CrossFit Gym Management System* is designed to overcome the limitations of the existing manual process by providing an integrated, web-based platform that connects admins, trainers, and members. It automates core activities such as member registration, trainer assignment, supplement management, and package booking. The system also enables smooth communication between trainers and members through built-in messaging and feedback features.

This platform allows the admin to control and monitor all activities from a single dashboard, including adding trainers, updating supplement stock, and managing package details. Trainers can view their assigned clients, provide fitness updates, and manage their schedules. Members can log in, book packages,request trainers and buy supplements-all through an easy to use web interface Members can log in, book packages,request trainers and buy packages, request trainers, and buy supplements—all through an easy-to-use web interface.

Features of the Proposed System:

1. **Automated Member Management:** The system maintains detailed records of members, their chosen packages, and trainer allocations.
2. **Trainer and Package Management:** Trainers can manage assigned clients and members can choose from available packages.
3. **Supplement Inventory Control:** Admin can add, update, or remove supplement details and track stock levels efficiently
4. **Online Communication:** Built-in messaging allows trainers and members to communicate directly through the system.
5. **User Authentication:** Secure login ensures role-based access for Admin, Trainer, and Member users.
6. **Digital Record Keeping:** Eliminates paperwork by storing all data safely in a centralized database.
7. **Reports and Monitoring:** Provides an overview of membership trends, bookings, and supplement usage for better decision-making.
8. **BMI Calculator:**Provides a bmi calculator for memebrs to test their bmi level

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

❖ **Improved Efficiency:**

All major operations are automated, saving time and effort for both staff and members.

❖ **Enhanced Accuracy:**

Digital data storage minimizes human error and ensures updated, reliable records.

❖ **Better Communication:**

Trainers and members can interact through the system for guidance, progress tracking, and feedback.

❖ **Transparency:**

Members can view available packages, supplements, and trainer information anytime.

❖**Scalability:**

The system can easily be expanded to include additional modules like online payment or mobile app integration in the future. By addressing the gaps in existing solutions, the proposed system ensures a more reliable, efficient, and user-friendly experience for both member and trainer

2.4 FEASIBILITY STUDY

A feasibility study is conducted to determine whether developing the proposed system is practical and achievable within the available resources. It evaluates three main aspects—**technical**, **operational**, and **economic feasibility**—to ensure that the system can be successfully developed and implemented..

2.4.1 TECHNICAL FEASIBILITY

The *CrossFit Gym Management System* is technically feasible as it uses reliable and widely available technologies such as PHP for backend development, MySQL for database management, and HTML/CSS for the frontend interface. These tools are easy to maintain and cost-effective. The hardware requirements are minimal and can be supported by any modern computer with an internet connection. Hence, the technical feasibility of the project is very high.

2.4.2 OPERATIONAL FEASIBILITY

The system is operationally feasible because it simplifies daily gym tasks for all types of users. The Admin can efficiently monitor operations, trainers can manage clients easily, and members can book services without difficulty. Since the system is designed with a simple and user-friendly interface, it minimizes the learning curve for new users. Management support and user acceptance are expected to be strong because the system directly improves productivity and convenience.

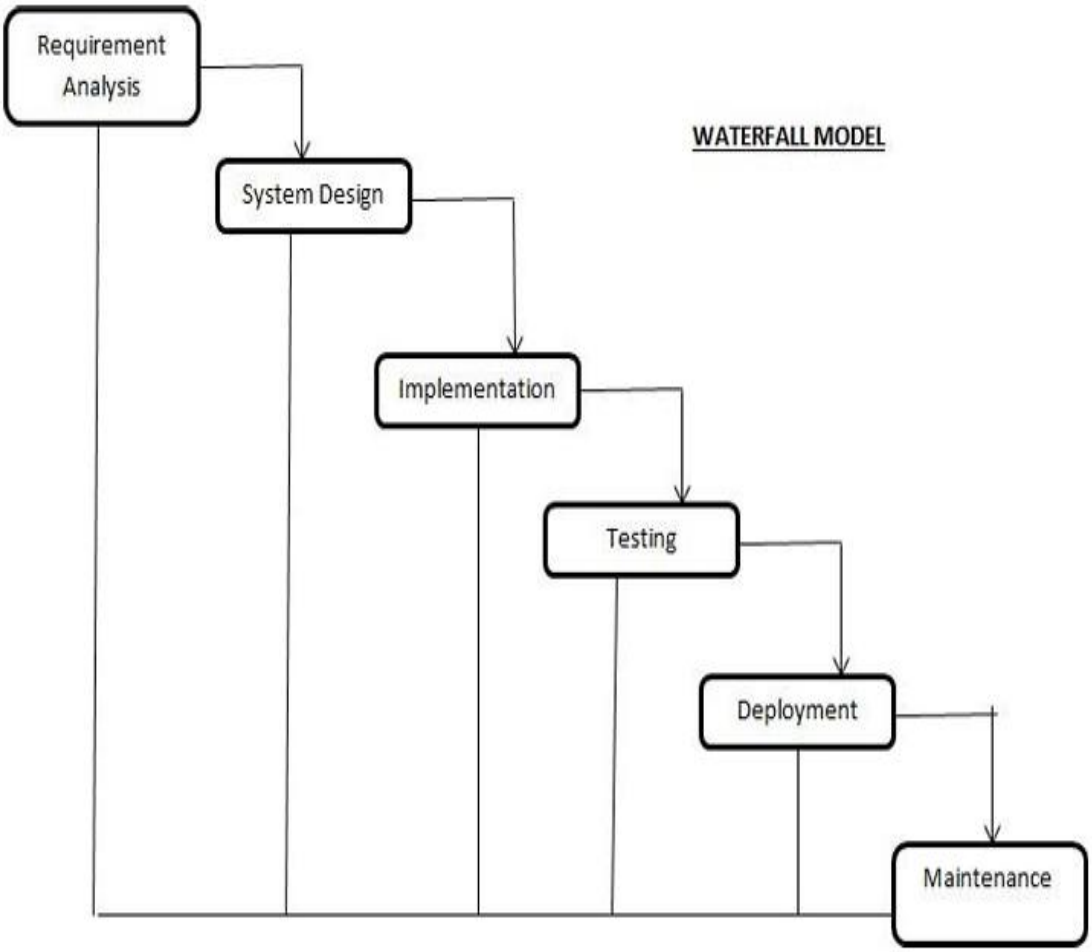
2.4.3 ECONOMIC FEASIBILITY

Economic feasibility examines whether the system is cost-effective in the long run. The *CrossFit Gym Management System* is economically feasible as it is developed using opensource technologies that do not require expensive licenses. It also reduces the cost of paperwork and manual labor, providing financial savings over time. The maintenance cost is minimal, making it a sustainable solution for the organization

2.5 SOFTWARE ENGINEERING PARADIGM APPLIED

The software process refers to the methods used to develop software. In addition to hiring skilled engineers and purchasing the latest development tools, an effective software development process is crucial for engineers to systematically apply the best technical and managerial practices to complete projects successfully.For this project, the Waterfall Model is followed.

The Waterfall approach was the first SDLC model to be widely used in software engineering to ensure the success of the project. In the Waterfall approach, the entire software development process is divided into separate phases. In this model, typically, the outcome of one phase acts as the input for the next phase sequentially.


Requirement Analysis
WATERFALL MODEL
System Design
Implementation
Testing
Deployment
Maintenance

SYSTEM DESIGN

CHAPTER 3

SYSTEM DESIGN

3.1 INTRODUCTION

The *CrossFit Gym Management System* is a web-based application developed to simplify and automate the daily operations of a gym. It helps manage members, trainers, fitness packages, and supplement stock in a systematic and digital way. The system aims to create a centralized platform where the **Admin**, **Trainer**, and **Member** can interact and carry out their respective functions efficiently.

This application provides key features such as user authentication, package booking, trainer assignment, supplement management, and communication modules. It ensures that every user can easily access information relevant to their role. The Admin manages the trainers, members, and inventory; the Trainer monitors clients and provides updates; and the Member can book packages and purchase supplements.

The system's design focuses on usability, data accuracy, and scalability. It follows a modular architecture that separates the front-end, back-end, and database layers for better maintenance and performance. By combining automation with a user-friendly interface, the *CrossFit Gym Management System* enhances the gym's productivity, improves communication, and ensures smooth management of all fitness-related activities.

3.2 DATABASE DESIGN

The database for the *CrossFit Gym Management System* is designed to handle all essential information related to members, trainers, packages, supplements, and transactions efficiently. It ensures secure, consistent, and organized data storage to support all operations within the system.

The main database tables include:

- **Admin Table:** Stores admin login credentials and profile details.
- **Trainer Table:** Contains information about each trainer, including ID, name, specialization, contact, and availability.
- **Member Table:** Stores member details like ID, name, contact, selected package, and assigned trainer.

- **Package Table:** Maintains data about various gym packages, including package ID, name, duration, and price.
- **Supplement Table:** Tracks supplement details such as stock quantity, type, and price.
- **Booking Table:** Records all package and trainer bookings made by members.
- **Feedback Table:** Stores messages or feedback between trainers and members for better communication.

Relationships between these entities are carefully structured to avoid redundancy and ensure data consistency. The use of primary and foreign keys maintains referential integrity, while indexing improves query performance, allowing fast data retrieval.

NORMALIZATION

Normalization is applied to organize the database into well-structured tables to reduce redundancy and ensure data integrity. The design process follows the standard normalization rules to achieve efficient data organization.

First Normal Form (1NF):

In 1NF, all the table columns contain atomic values — each field holds a single value, and there are no repeating groups. This ensures a structured and clear layout of the data.

Second Normal Form (2NF):

The database is in 2NF when it is already in 1NF and all non-key attributes are fully dependent on the primary key. This removes partial dependencies and ensures each attribute relates only to the entity it describes.

Third Normal Form (3NF):

A table is in 3NF if it is in 2NF and all non-key attributes depend only on the primary key, not on other non-key attributes. This eliminates transitive dependencies and provides an optimized database structure that supports consistency and accuracy.

3.2.1 ENTITY RELATIONSHIP MODEL

The **Entity-Relationship (ER) Model** is used to visually represent the logical structure of the database. It shows how different entities such as members, trainers, and admins are related to each other. This helps ensure a well-organized database design and efficient system development.

ENTITIES:

- **Admin:** Manages trainers, members, packages, and supplements
- **Trainer:** Provides training sessions and manages assigned members
- **Member:** Registers to the system, books packages, and communicates with trainers.
- **Package:** Defines gym offers with details like duration and cost.
- **Supplement:** Contains details of supplements available in stock.
- **Booking:** Records package or trainer bookings.
- **Feedback:** Stores communication between trainer and member.

Attributes:

Each entity includes specific attributes that describe its properties.

- **Admin:** Admin_ID (PK), Name, Email, Password.
- **Trainer:** Trainer_ID (PK), Name, Contact, Experience, Specialization.
- **Member:** Member_ID (PK), Name, Gender, Contact, Trainer_ID (FK), Package_ID (FK).
- **Package:** Package_ID (PK), Package_Name, Duration, Price.
- **Supplement:** Supplement_ID (PK), Supplement_Name, Stock, Price.
- **Booking:** Booking_ID (PK), Member_ID (FK), Package_ID (FK), Date, Status.
- **Feedback:** Feedback_ID (PK), Member_ID (FK), Trainer_ID (FK), Message, Date.

Relationships:

- A **Member** can book multiple **Packages** (1:N).
- A **Trainer** can train multiple **Members** (1:N).
- A **Member** can send multiple **Feedbacks** to a **Trainer** (1:N).
- **Admin** manages multiple **Trainers** and **Members** (1:N).

Components of the Entity-Relationship Diagram (ERD)

1. **Entities:** Represent real-world objects like Admin, Trainer, Member, Package, and Supplement.
    - Example: *Trainer*, *Member*, *Package*.
2. **Attributes:** Define details about each entity.
    - Example: *Member Name, Trainer Experience, Package Duration*.
3. **Primary Key (PK):** Uniquely identifies each record within a table.
    - Example: *Member_ID, Trainer_ID, Package_ID.*
4. **Foreign Key (FK):** Creates a relationship between two entities by referencing the primary key of another table.
    - Example: *Trainer_ID* in the Member table links to the Trainer entity.
5. **Relationships:** Show associations between entities.

- Example: *Assigned To* (between Trainer and Member), *Booked By* (between Member and Package).

6. **Cardinality:** Represents how many instances of one entity relate to another.
   - Example: A Trainer can train many Members (1:N).
7. **Connectivity:** Indicates whether a relationship is mandatory or optional.
   - Example: A Member must be linked to at least one Package.
8. **Generalization/Specialization (if applicable):** Defines hierarchical relationships, e.g., Admin being a higher-level entity managing Trainers and Members.

ER DIAGRAM

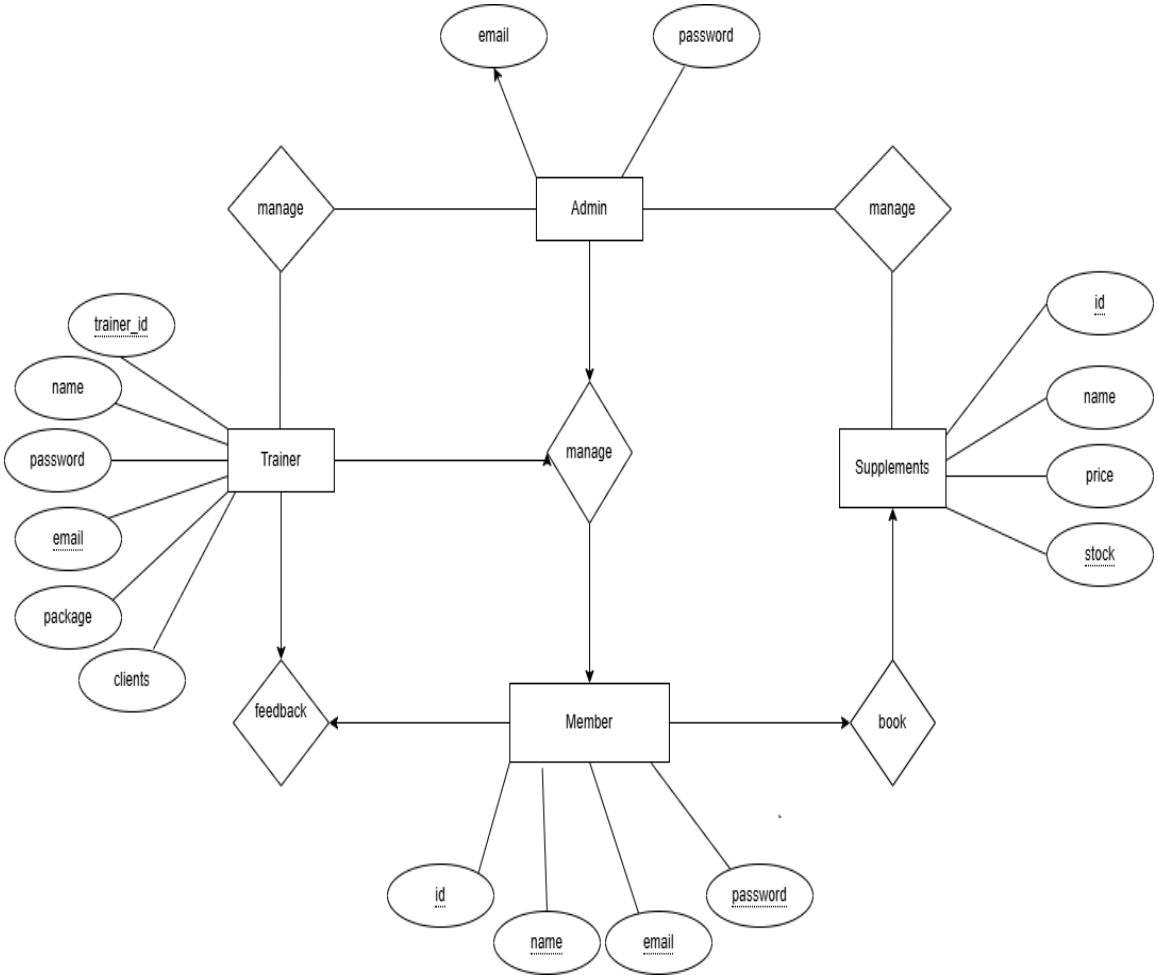


TABLE DESIGN:

DATABASE:crossfit_gym

Admin Table

Field Name	Data Type	Size	Description	Constraints
admin_id	INT	11	Unique identifier for admin	Primary Key, Auto Increment
name	VARCHAR	50	Admin's full name	Not Null
email	VARCHAR	100	Admin’s email address	Not Null, Unique
password	VARCHAR	255	Admin’s encrypted password	Not Null
created_at	TIMESTAMP	-	Record creation timestamp	Default: Current Timestamp

Trainer Table

Field Name	Data Type	Size	Description	Constraints
trainer_id	INT	11	Unique identifier for trainer	Primary Key, Auto Increment
name	VARCHAR	50	Trainer’s full name	Not Null
email	VARCHAR	100	Trainer’s email address	Not Null, Unique
password	VARCHAR	255	Trainer’s encrypted password	Not Null
specialization	VARCHAR	100	Area of expertise (e.g., weight training, cardio)	Not Null
contact_no	VARCHAR	15	Trainer’s phone number	Not Null
created_at	TIMESTAMP	-	Record creation timestamp	Default: Current Timestamp

Members Table

Field Name	Data Type	Size	Description	Constraints
member_id	INT	11	Unique identifier for member	Primary Key, Auto Increment
name	VARCHAR	50	Member's full name	Not Null
email	VARCHAR	100	Member's email address	Not Null, Unique
password	VARCHAR	255	Member's encrypted password	Not Null
age	INT	3	Member's age	Not Null
gender	VARCHAR	10	Gender of the member	-
contact_no	VARCHAR	15	Member's phone number	Not Null
joined_date	DATE	-	Date of joining the gym	Default: Current Date

Packages Table

Field Name	Data Type	Size	Description	Constraints
booking_id	INT	11	Unique identifier for booking	Primary Key, Auto Increment
member_id	INT	11	Reference to member who booked	Foreign Key (members.member_id)
trainer_id	INT	11	Reference to trainer booked	Foreign Key (trainers.trainer_id)
package_id	INT	11	Reference to package booked	Foreign Key (packages.package_id)
booking_date	DATE	-	Date of booking	Default: Current Date
status	VARCHAR	20	Booking status (Pending/Confirmed/Cancelled)	Default: 'Pending'

Bookings Table

Field Name	Data Type	Size	Description	Constraints
booking_id	INT	11	Unique identifier for booking	Primary Key, Auto Increment
member_id	INT	11	Reference to member who booked	Foreign Key (members.member_id)
trainer_id	INT	11	Reference to trainer booked	Foreign Key (trainers.trainer_id)
package_id	INT	11	Reference to package booked	Foreign Key (packages.package_id)
booking_date	DATE	-	Date of booking	Default: Current Date
status	VARCHAR	20	Booking status (Pending/Confirmed/Cancelled)	Default: 'Pending'

Supplements Table

Field Name	Data Type	Size	Description	Constraints
supplement_id	INT	11	Unique identifier for supplement	Primary Key, Auto Increment
name	VARCHAR	100	Name of supplement	Not Null
type	VARCHAR	50	Type (e.g., protein, vitamins)	Not Null
stock_quantity	INT	11	Available stock	Not Null
price	DECIMAL	10,2	Price of supplement	Not Null
updated_at	TIMESTAMP	-	Last updated timestamp	Default: Current Timestamp

Message Table

Field Name	Data Type	Size	Description	Constraints
message_id	INT	11	Unique identifier for message	Primary Key, Auto Increment
sender_id	INT	11	ID of sender (Member/Trainer)	Not Null
receiver_id	INT	11	ID of receiver (Member/Trainer)	Not Null
message_text	TEXT	-	Message content	Not Null
sent_at	TIMESTAMP	-	Time message was sent	Default: Current Timestamp

Supplements Order Table

Field Name	Data Type	Size	Description	Constraints
order_id	INT	11	Unique identifier for each supplement booking	Primary Key, Auto Increment
member_id	INT	11	Reference to the member booking the supplement	Foreign Key (members.member_id)
supplement_id	INT	11	Reference to the supplement being booked	Foreign Key (supplements.supplement_id)
quantity	INT	5	Quantity of supplement booked	Not Null
order_date	DATE	-	Date on which the supplement was booked	Default: Current Date
status	VARCHAR	20	Current status of the order (Booked/Collected/Cancelled)	Default: ‘Booked’

User Table

Field Name	Data Type	Size	Description	Constraints
member_id	INT	11	Unique identifier for each member	Primary Key, Auto Increment
name	VARCHAR	100	Member's full name	Not Null
gender	VARCHAR	10	Gender of the member	Not Null
age	INT	3	Age of the member	Not Null
email	VARCHAR	100	Member's email address	Not Null, Unique
phone	VARCHAR	15	Member's contact number	Not Null
password	VARCHAR	255	Member's login password (encrypted)	Not Null
address	VARCHAR	255	Residential address of the member	Optional
join_date	DATE	-	Date when the member joined the gym	Default: Current Date
package_id	INT	11	Reference to the selected gym package	Foreign Key (packages.package_id)
trainer_id	INT	11	Reference to the booked trainer (if any)	Foreign Key (trainers.trainer_id)
status	VARCHAR	20	Current status (Active / Inactive)	Default: 'Active'

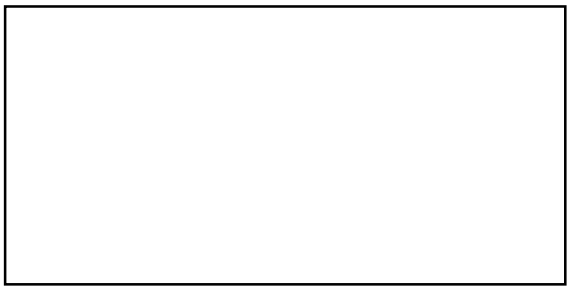
3.3 PROCESS DESIGN-DATAFLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing. A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

Data Flows Data flows show the passage of data in the system are represented by the lines joining System components. An arrow indicates the direction of flow and the line is labelled by name of data flow.

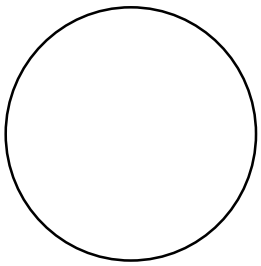
Basic data flow diagram symbols are: -

• A rectangle defines a source or destination of the data.



• An arrow identifies data flow, data in motion. It is a pipeline through which information flows

• A circle or bubble represents a process at transforms incoming data flows into outgoing data flows.



• An open rectangle is a data store

One of the tools of structured analysis is the diagram. A data flow diagram is a graphical representation of the system. The analyst can use data flow diagram to explain this understanding about the system

. • Data flows are an initiative way of showing how data is processed by a system

• Data flow models are used to show how data flows through a sequence of processing steps.

Four steps are commonly used to construct a DFD:

1. Process should be named and numbered for easy references.

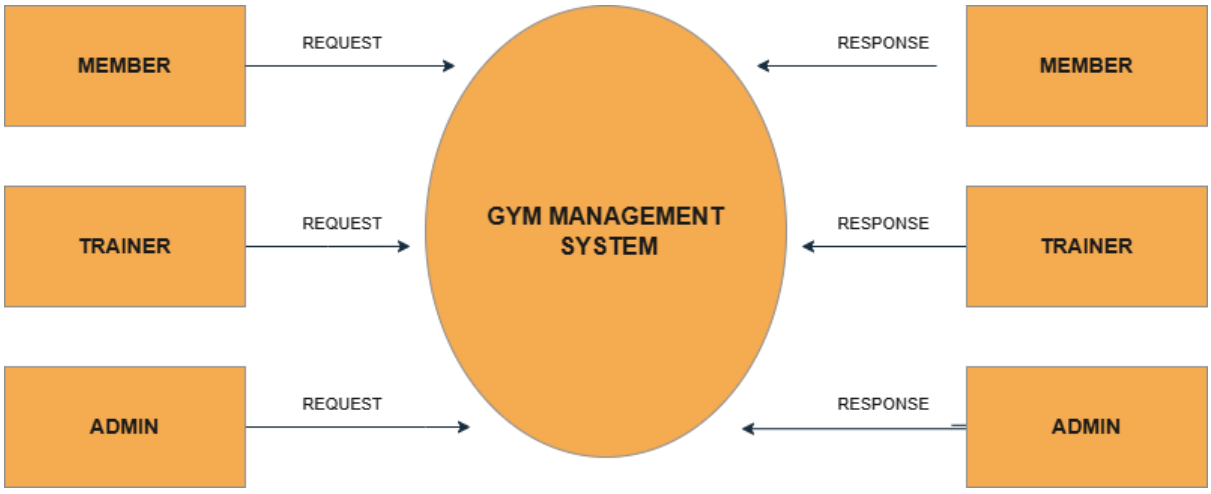
2. The direction of flow is from top to bottom and from left to right.

3. When a process is exploded into lower level details, they are numbered.

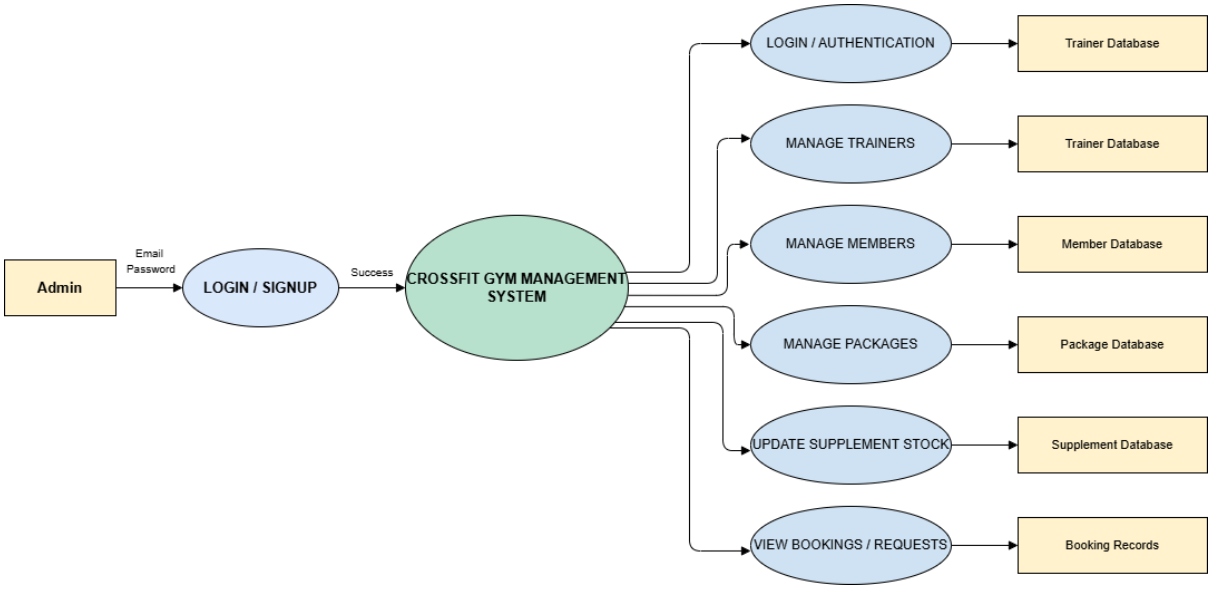
4. The name of data stores, sources and destinations are written in capital letters.

LEVELS

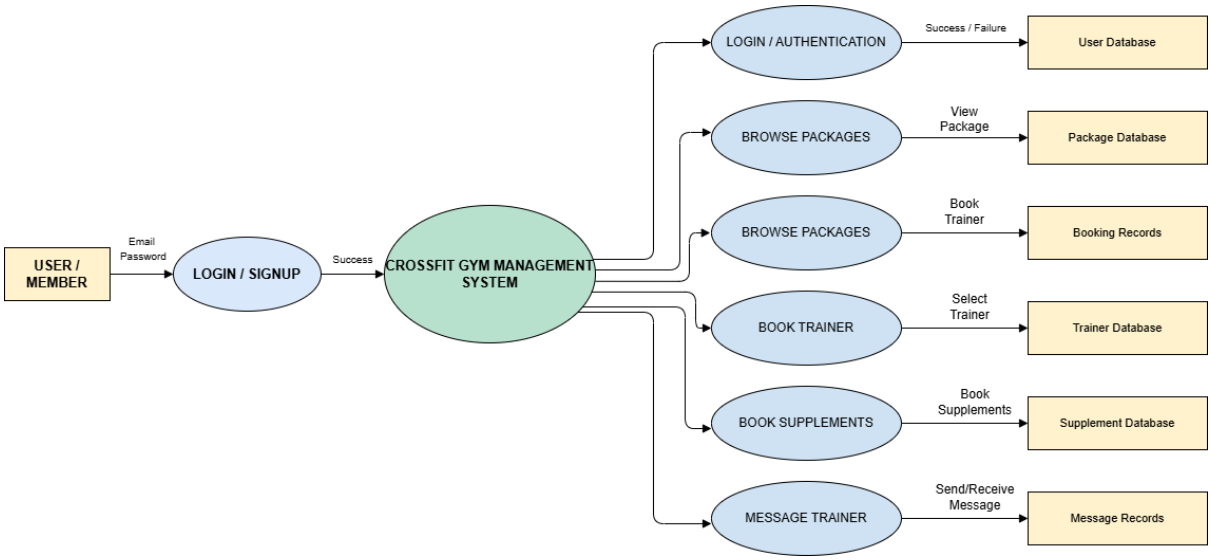
Level 0



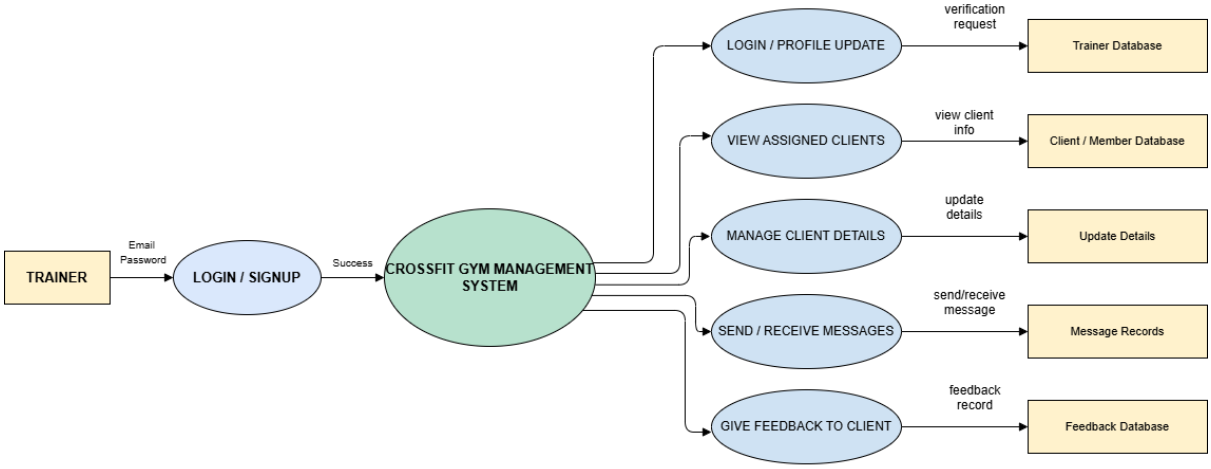
Admin Level 1



User Level 1



Trainer level 1



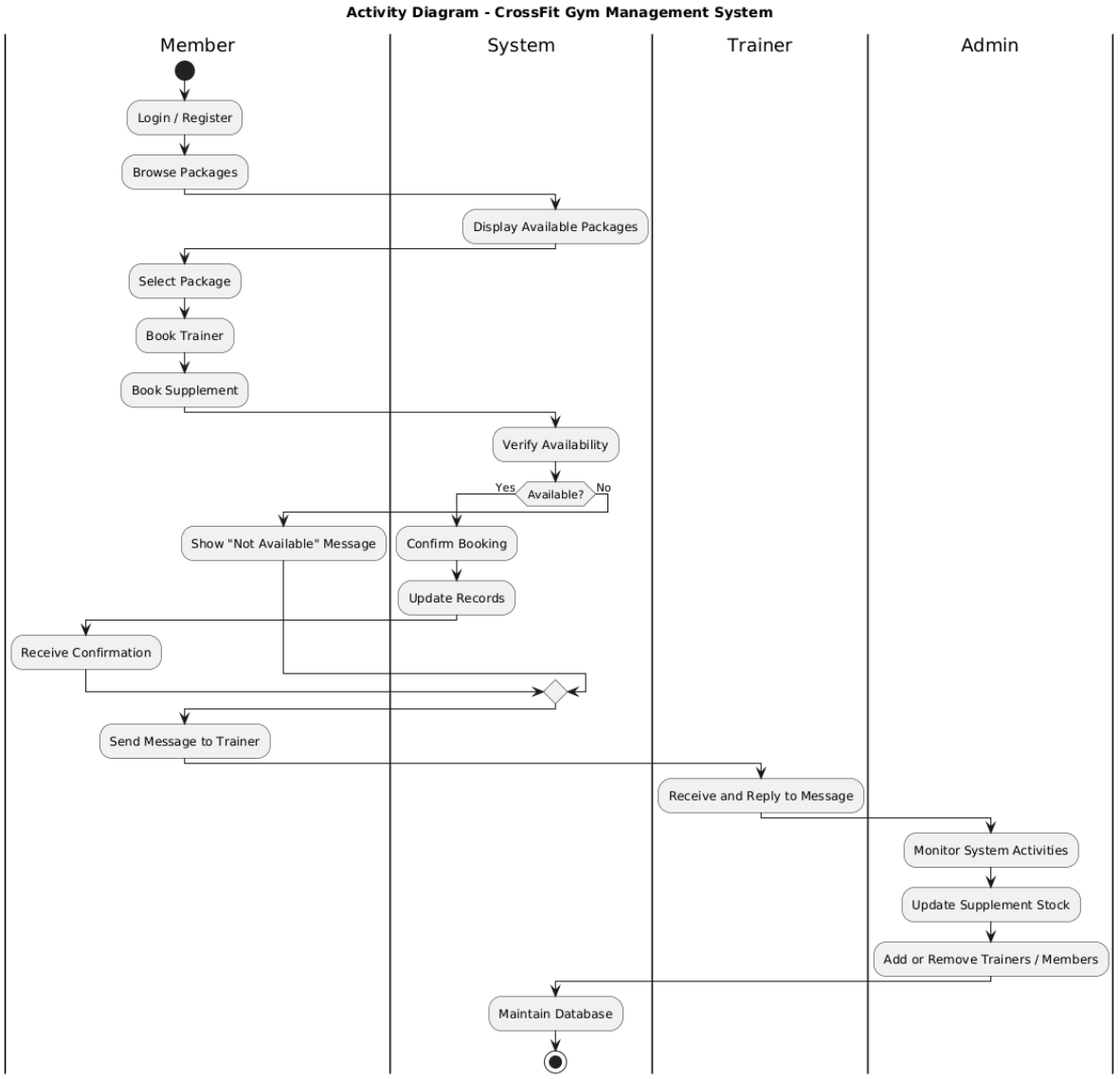
3.4 OBJECT ORIENTED DESIGN-UML DIAGRAMS

Object-oriented design (OOD) is a programming approach that structures a system around objects rather than actions. It emphasizes encapsulating data and behavior within objects and defining relationships between them. Unified Modeling Language (UML) diagrams are key tools in OOD, used to visually represent the architecture, components, and interactions of a system. UML diagrams are broadly categorized into structural and behavioral diagrams. Structural diagrams, like Class Diagrams, show the static relationships and structure of the system, including attributes, methods, and inheritance. Behavioral diagrams, such as Sequence and Use Case Diagrams, depict dynamic interactions and workflows. For instance, in the **CrossFit Gym Management System**, a **Class Diagram** represents key entities such as *Admin*, *Trainer*, *Member*, *Package*, and *Supplement*, along with their relationships and operations. A **Sequence Diagram** illustrates the step-by-step flow of interactions when a member performs activities such as *booking a trainer*, *purchasing supplements*, or *sending a message*. These UML diagrams together enhance the clarity of design, improve communication among stakeholders, and provide a clear blueprint for efficient system implementation and maintenance.

3.4.1 ACTIVITY DIAGRAM

An Activity Diagram is a type of behavioral UML diagram that visually represents the workflow or process in a system. It is particularly useful for modeling the dynamic aspects of a system by illustrating sequences of activities, their interdependencies, and the flow of control or data. Activity diagrams are often used in the analysis and design stages to capture business processes, system functionalities, or user interactions. The diagram consists of key components such as activities (rounded rectangles), transitions (arrows), decision nodes (diamonds for conditional branching), and start/end points (solid circle for start, double circle for end). Swim

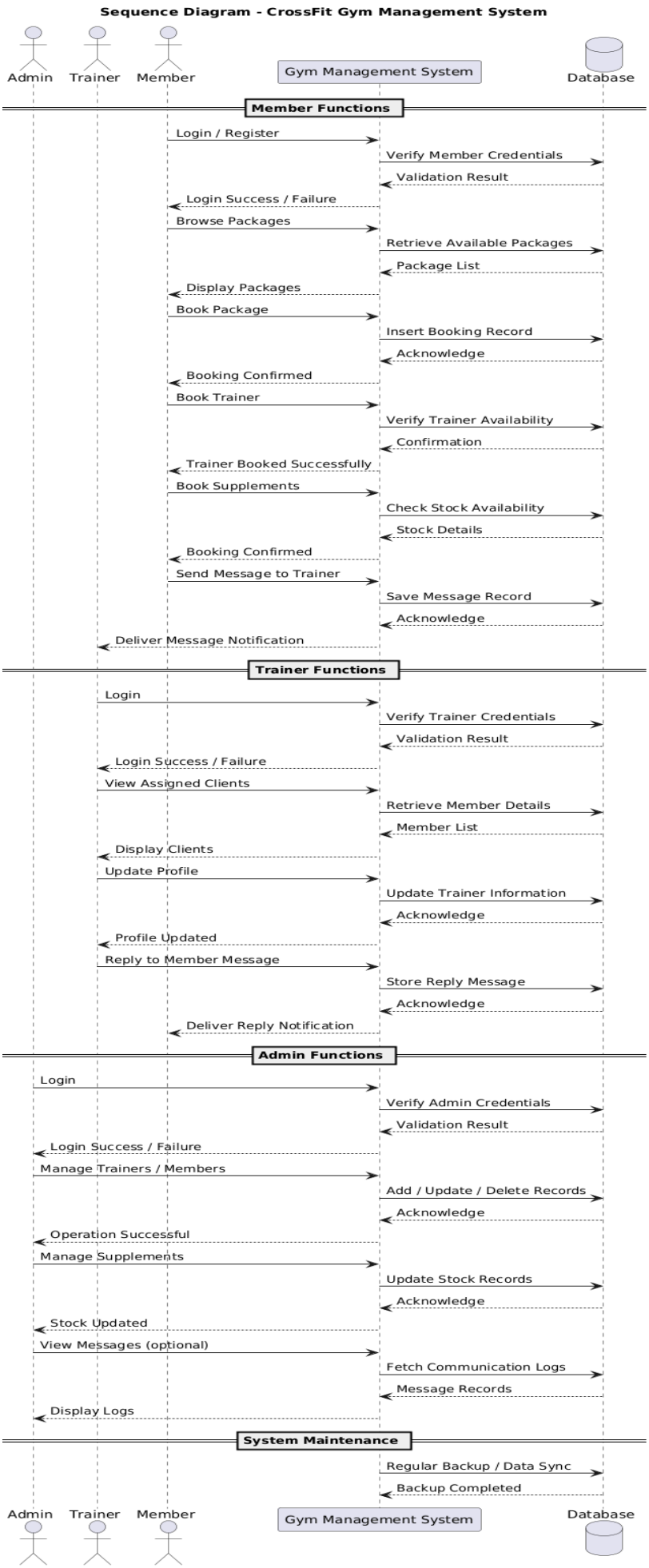
lanes are used to assign activities to specific actors or system components. For example, in the **CrossFit Gym Management System**, an activity diagram can represent the process of **member interaction within the system** — starting from the member logging in, browsing available fitness packages, booking a trainer or supplement, and proceeding to payment confirmation. It can also illustrate the trainer's workflow, where the trainer logs in, views assigned members, updates fitness plans, and communicates with members. Similarly, the admin's activities, such as managing trainers, updating supplement stock, or monitoring system operations, can be mapped within the diagram.



3.4.2 SEQUENCE DIAGRAM

A Sequence Diagram is a type of UML diagram used to model the flow of interactions between objects or components in a system over time. It captures the dynamic behavior of a system by showing how messages are exchanged in a specific sequence to accomplish a particular task. Sequence diagrams are particularly useful for visualizing use cases, clarifying system functionalities, and identifying object roles and communication. The diagram consists of **lifelines** (representing objects or actors), **activation bars** (indicating when an object is active), and **arrows** (representing the flow of messages). The horizontal arrangement shows the participating entities, while the vertical dimension represents the progression of time, illustrating the order of interactions.

For instance, in the **CrossFit Gym Management System**, a sequence diagram can illustrate the process of a **member booking a trainer or fitness package**. The interaction begins when the member logs into the system and browses available packages or trainers. The system retrieves relevant data from the database and displays it to the member. Upon booking, the system verifies availability, updates the booking records, and notifies both the member and the assigned trainer. Similarly, the sequence diagram can represent other workflows, such as the admin updating supplement stock or a trainer communicating with members.


Sequence Diagram - CrossFit Gym Management System
Admin
Trainer
Member
Gym Management System
Database
Member Functions
Login / Register
Verify Member Credentials
Validation Result
Login Success / Failure
Browse Packages
Retrieve Available Packages
Package List
Display Packages
Book Package
Insert Booking Record
Acknowledge
Booking Confirmed
Book Trainer
Verify Trainer Availability
Confirmation
Trainer Booked Successfully
Book Supplements
Check Stock Availability
Stock Details
Booking Confirmed
Send Message to Trainer
Save Message Record
Acknowledge
Deliver Message Notification
Trainer Functions
Login
Verify Trainer Credentials
Validation Result
Login Success / Failure
View Assigned Clients
Retrieve Member Details
Member List
Display Clients
Update Profile
Update Trainer Information
Acknowledge
Profile Updated
Reply to Member Message
Store Reply Message
Acknowledge
Deliver Reply Notification
Admin Functions
Login
Verify Admin Credentials
Validation Result
Login Success / Failure
Manage Trainers / Members
Add / Update / Delete Records
Acknowledge
Operation Successful
Manage Supplements
Update Stock Records
Acknowledge
Stock Updated
View Messages (optional)
Fetch Communication Logs
Message Records
Display Logs
System Maintenance
Regular Backup / Data Sync
Backup Completed
Admin
Trainer
Member
Gym Management System
Database

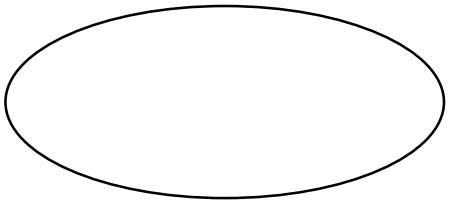
3.4.3 USECASE DIAGRAM

A Use Case Diagram is a UML diagram that visually represents the interactions between users (actors) and a system to capture its functionalities. It focuses on what the system does from a user's perspective rather than how it operates internally. The diagram consists of actors (stick figures), use cases (ovals), and associations (lines connecting actors to use cases). The main actors in this system are:

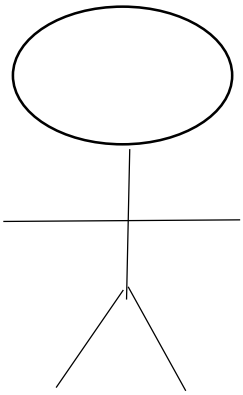
- **Admin:** Manages trainers, members, supplements, and oversees the overall system.
- **Trainer:** Views assigned members, provides fitness feedback, updates their profile, and communicates with members.
- **Member:** Registers and logs in to book packages, trainers, and supplements, and sends messages to trainers.

The system provides functionalities such as user authentication, booking, communication, and inventory.management.
This use case diagram establishes system boundaries, role responsibilities, and the main services offered by the gym management platform.

Represent the use case

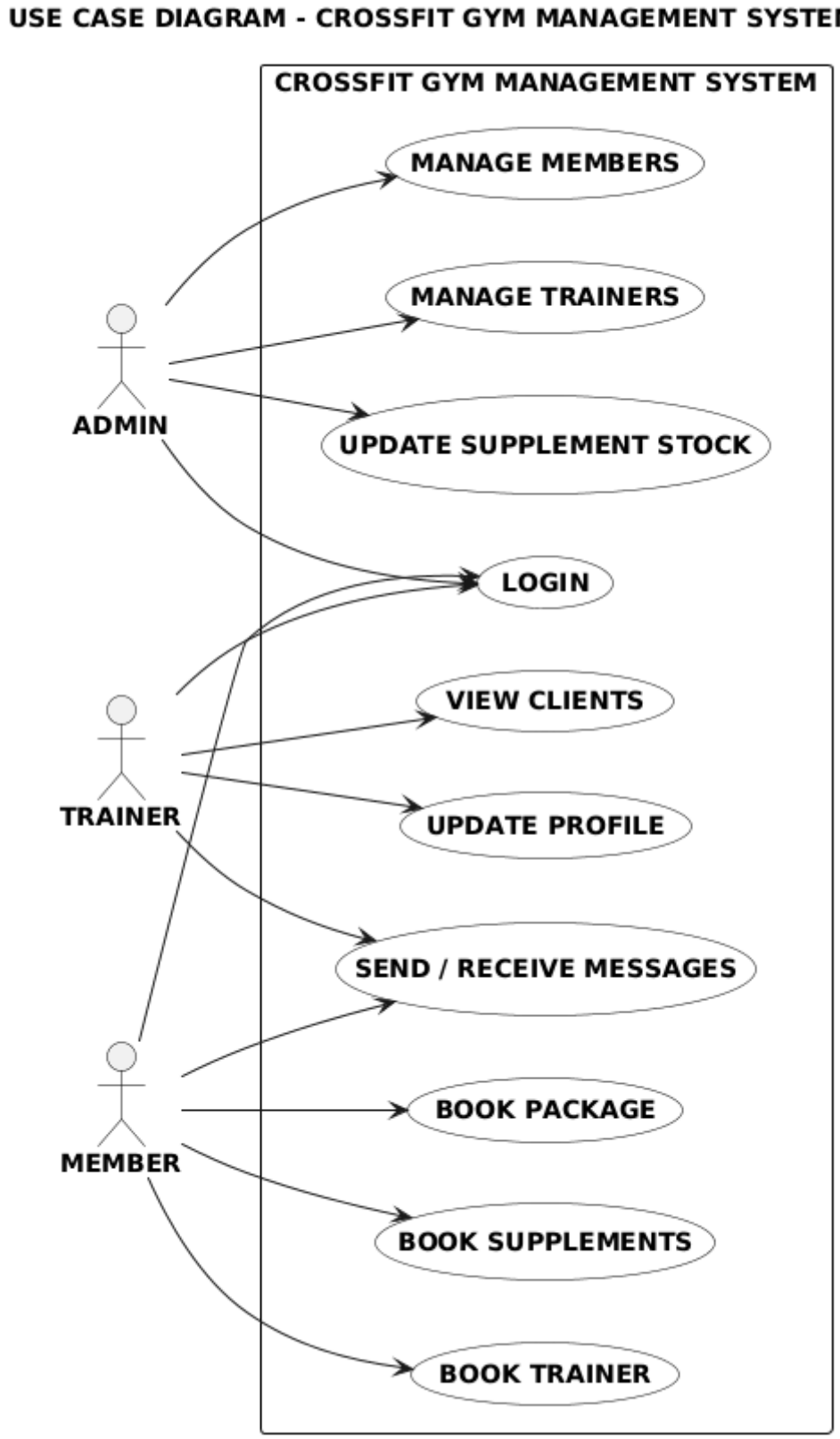


Actors are the entities which interact with the system



Association

Use Case Diagram



INPUT DESIGN

Input design is an essential phase in system development that focuses on how data is captured, validated, and entered into the system. It ensures that user interactions are simple, accurate, and efficient, minimizing data entry errors and maintaining the integrity of information across all modules. In the context of the **Gym Management System**, input design is structured to accommodate the needs of three primary users — **Admin, Trainer, and Member** — to support activities such as registration, package booking, trainer assignment, supplement management, and internal communication.

User Inputs

1. Admin Inputs

- **Login:** Requires a username and password for authentication. The credentials are validated using encrypted passwords to ensure secure access.
- **Add Trainer:** Includes fields for trainer name, email, contact number, specialization, and profile details.
- **Add Member:** Captures member details such as name, email, contact number, age, gender, and selected package.
- **Update Supplement Stock:** Allows the admin to input supplement details like supplement name, type, quantity, price, and expiry date.
- **Manage Packages:** Admin can enter package name, duration, price, and description for each membership type.

2. Trainer Inputs

- **Login:** Trainers enter their email and password for authentication.
- **Profile Management:** Trainers can update personal details such as contact information, specialization area, and working hours.
- **Client Feedback or Message:** Trainers can input messages or training feedback for members assigned to them.
- **Member Progress Tracking:** Inputs include fields for entering a member's progress, such as weight, BMI, workout schedule, or remarks.

3. Member Inputs

- **Registration:** New members fill in details like name, email, password, contact number, gender, and age to create an account.
- **Login:** Members input email and password for system access.
- **Book Package:** Members select package name, duration, and payment mode.
- **Book Trainer:** Allows members to choose a trainer from the available list based on specialization or experience.

- **Supplement Booking:** Members can select supplement name, quantity, and confirm order.
- **Messages:** Members can send messages or queries directly to trainers through the internal messaging system.

Design Features

- **Validation Controls:**Each form validates essential fields to ensure accuracy. Examples include email format checking, numeric field validation for phone numbers, and password length enforcement.
- **Error Handling:**Clear error prompts are displayed for missing or invalid data entries, guiding users to correct mistakes easily.
- **User-Friendly Interface:**Forms are designed with dropdown lists, checkboxes, and auto-fill suggestions for fields like package names and trainer selection.
- **Security Measures:**Passwords are encrypted before database storage, and session management prevents unauthorized access.
- **Consistency:**All input forms maintain consistent color themes and layout for easy navigation and visual uniformity.

3.6 OUTPUT DESIGN

Output design is a vital phase of system development that focuses on presenting processed data in a clear, accurate, and user-friendly manner. It ensures that the system's outputs are easily understandable, meaningful, and actionable for the intended users. Effective output design enhances decision-making, improves usability, and ensures that every user receives relevant information according to their role within the system.In the **Gym Management System**, output design ensures that the **Admin**, **Trainer**, and **Member** receive information that supports efficient gym operations, such as member management, training updates, and supplement tracking. The design also emphasizes clarity, responsiveness, and accessibility across devices.

Objectives of Output Design

- **Clarity:** Present system outputs in a structured, visually clear, and easily interpretable format.
- **Relevance:** Customize outputs according to each user's role and specific needs.
- **Timeliness:** Ensure that updates on bookings, messages, and stock levels are displayed in real-time.
- **Accessibility:** Provide responsive layouts compatible with multiple devices and browsers for ease of access.
- **Accuracy:** Display only validated and authorized data to maintain integrity and confidentiality.

Outputs for Members

- **Dashboard Overview:** Displays a summary of active gym packages, trainer details, supplement purchases, and membership status.
- **Trainer Booking Confirmation:** Shows booking details, including trainer name, session time, and schedule.
- **Package Details:** Displays information on available packages, their duration, cost, and benefits.
- **Messages:** Presents real-time communication between members and trainers for progress discussions or queries.
- **Supplement Orders:** Shows supplement names, quantity, price, and current stock status.
- **Notifications:** Provides timely updates for package expirations, trainer messages, and new offers.

Outputs for Trainers

- **Trainer Dashboard:** Displays assigned members, session schedules, and performance tracking data.
- **Member Progress Reports:** Shows member details such as attendance, weight changes, and training history.
- **Messages Panel:** Enables trainers to view and reply to member messages instantly for effective communication.
- **Profile Updates:** Reflects recent changes in trainer information such as specialization, working hours, and availability.
- **Feedback Section:** Displays feedback or ratings given by members regarding the trainer's performance.

Outputs for Admin

- **Admin Dashboard:** Provides an overview of the entire system, including total trainers, members, active packages, and supplement stock.
- **User Management Reports:** Displays lists of registered trainers and members with options to edit, suspend, or delete accounts.
- **Trainer & Member Activity Logs:** Shows login records, bookings, and communication history for better monitoring.
- **Supplement Inventory Report:** Displays supplement stock levels, expiry alerts, and total sales for efficient restocking.
- **Package Management:** Presents all available gym packages with the ability to view usage statistics and update or remove inactive ones.
- **System Alerts:** Notifies the admin about low stock, membership renewals, or inactive users.

Output Delivery Formats

- **Interactive Dashboards:** Real-time graphical dashboards for admins and trainers showing system statistics and member performance metrics.
- **Reports and Tables:** Structured tabular outputs for user details, bookings, and supplement data for easy review.
- **Notifications and Alerts:** Pop-up and email notifications for trainers, members, and admins to ensure timely updates.
- **Graphical Reports:** Visual charts displaying member progress, supplement sales, and attendance analytics.
- **Responsive Web Interface:** Ensures that outputs are accessible on desktops, tablets, and mobile devices with consistent quality.

SYSTEM ENVIRONMENT

CHAPTER – 4

SYSTEM ENVIRONMENT

An important aspect of building an application is selecting appropriate hardware and software components. The hardware supports and drives the software to deliver effective solutions. Factors such as cost, performance, reliability, and scalability are carefully considered when choosing the system configuration for any computerized project.

4.1 INTRODUCTION

The *CrossFit Gym Management System* is a web-based application developed using PHP and MySQL, aimed at simplifying and automating gym operations such as member registration, trainer management, package booking, and supplement tracking. The system runs on a standard computer configuration and requires minimal hardware resources to function efficiently.

This section describes the hardware and software requirements essential for the smooth installation and execution of the project. These requirements are determined based on the components of the operating environment that form the front-end and back-end systems.

The configuration can be divided into two phases:

- **Software Requirements**
- **Hardware Requirements**

SOFTWARE REQUIREMENTS

- **Operating System:** Windows 10 or higher
- **Front End:** HTML, CSS, JavaScript, PHP
- **Back End:** MySQL Server
- **Web Browser:** Google Chrome / Microsoft Edge
- **Platform:** Visual Studio Code

The software components work together to provide a stable and responsive environment. HTML, CSS, and JavaScript handle the interface design and client-side interactions, while PHP manages server-side scripting. MySQL serves as the database engine to store and retrieve gym-related data efficiently.

HARDWARE REQUIREMENTS

- **Processor:** Intel Core i3 or above
- **RAM:** 4 GB minimum
- **Processor:** Intel Core i3 or above
- **RAM:** 4 GB minimum
- **Hard Disk:** 100 GB or more
- **Monitor:** Standard Color Monitor
- **Clock Speed:** 2.4 GHz or above

The hardware setup is simple and cost-effective. The system can run on any modern desktop or laptop without requiring high-end configurations. This ensures easy deployment in gyms of different scales.

4.2 TOOLS AND PLATFORM

PHP

PHP (Hypertext Preprocessor) is a widely used open-source scripting language primarily designed for web development. It operates on the server side and is used to create dynamic and interactive web applications. In this project, PHP is responsible for handling backend logic, connecting to the MySQL database, and performing CRUD (Create, Read, Update, Delete) operations.

It is platform-independent and compatible with most web servers, offering simplicity, flexibility, and strong community support. Its integration with HTML enables seamless data processing and dynamic web page generation.

HTML

HTML (HyperText Markup Language) is used to design the structure and layout of web pages. It defines elements such as headings, tables, forms, and buttons, forming the skeleton of the system'sinterface.

Every HTML document consists of tags enclosed within <html>, <head>, and <body> elements, which define the overall page structure. In this project, HTML is used to build the core interface for the admin, trainer, and member dashboards.

CSS

CSS (Cascading Style Sheets) enhances the presentation of HTML elements. It controls colors, layouts, font styles, and overall aesthetics, ensuring a professional and user-friendly interface.By separating design from content, CSS makes the system visually appealing and

easy to maintain. It allows uniform styling across all pages, providing a consistent look and feel throughout the application.

JAVASCRIPT

JavaScript is a client-side scripting language that brings interactivity and dynamic behavior to web pages. It validates user input, manages client-side events, and enables real-time feedback to users.In this project, JavaScript is used to handle form validations (like checking empty fields), enable dynamic data loading, and improve the overall user experience. It allows trainers and members to interact with the system smoothly.

MYSQL SERVER

MySQL is an open-source relational database management system used to store all application data securely. It maintains records of members, trainers, packages, supplements, and bookings. MySQL ensures data integrity, consistency, and easy retrieval through structured queries. It supports multiple users, allows concurrent access, and provides efficient indexing and relationships among tables. In the *CrossFit Gym Management System*, MySQL is the backbone of data management, ensuring accurate tracking of gym operations.

Key Features:

- High performance and reliability
- Support for complex queries
- Secure data handling
- Compatibility with PHP
- Scalable for future enhancements

VISUAL STUDIO CODE

Visual Studio Code is a lightweight and powerful source-code editor used for developing and debugging PHP, HTML, CSS, and JavaScript. It provides built-in support for syntax highlighting, extensions, and integrated Git control, which help in efficient project development. The editor ensures easy project organization and faster code execution.

4.3 OPERATING SYSTEM

The *CrossFit Gym Management System* is developed and tested on **Windows 10**, which provides a stable and user-friendly platform for both development and deployment. Windows

10 supports the technologies used in the project and offers reliable compatibility with PHP, MySQL, and web browsers.

Main Features of Windows 10:

- Easy to use and manage
- High compatibility with software and web technologies
- Enhanced security and stability
- Support for multiple programming environments

SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 INTRODUCTION

Implementation is the process of transforming a newly designed system into an operational one. This stage demands close collaboration between the developers and the users to ensure the new system effectively meets organizational needs. Proper implementation is essential to demonstrate the system's value while minimizing resistance to change. Whether the system is entirely new or a modification of an existing one, careful execution is critical for success. The implementation phase includes key tasks such as:

• Comprehensive planning.

• Examination of system constraints.

• Designing changeover strategies.

• Evaluating the effectiveness of the implementation method.

The implementation method and timeline are initially determined, followed by rigorous testing and user training on the new procedures. For hardware implementation, this involves installing the system in its actual working environment to meet user satisfaction and ensure smooth operations. Often, those responsible for commissioning the system may not operate it themselves, leading to initial skepticism about its value. To address this, it is essential to ensure:

• Users are fully aware of the system's benefits.

• Confidence in the system is established.

• Adequate training and support are provided to make users comfortable with the application.

Implementation marks the transition from theoretical design to a functional system. This phase brings significant challenges and changes, requiring meticulous planning and management to prevent disruption. A successful implementation involves all the activities needed to replace an old system with a new one. This process includes thorough testing to confirm the system operates as specified.

Proper implementation guarantees the system's reliability and the organization’s smooth transition. It ensures users gain confidence in the new system's efficiency and capability. The complexity of the system determines the level of effort needed for analysis and design during implementation.

5.2 CODING

5.2.1 SAMPLE CODES

User Login Form

```
<?php
ob_start();
session_start();
require_once 'config/database.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
$email = $_POST['email'];
$password = $_POST['password'];

$stmt = $conn->prepare("SELECT * FROM users WHERE email = ?");
$stmt->execute([$email]);
$user = $stmt->fetch();

if ($user && password_verify($password, $user['password'])) {
 $_SESSION['user_id'] = $user['user_id'];
 $_SESSION['role'] = $user['role'];

  switch($user['role']) {
  case 'admin':
  header("Location: admin/dashboard.php");
  break;
  case 'trainer':
  // Check if trainer is verified
  $stmt = $conn->prepare("SELECT is_verified FROM trainers WHERE user_id = ?");
  $stmt->execute([$user['user_id']]);
```

```
$trainer = $stmt->fetch();


if ($trainer && $trainer['is_verified']) {
header("Location: trainer/dashboard.php");
} else {
// Trainer not verified yet
$_SESSION['info'] = "Your trainer account is pending verification. You'll have limited access until your certification is reviewed.";
header("Location: trainer/dashboard.php");
}
break;
case 'member':
header("Location: member/dashboard.php");
break;
default:
header("Location: index.php");
}
exit();
} else {
$_SESSION['error'] = "Invalid email or password";
header("Location: login.php");
exit();
}
}


$pageTitle = "Login";
$error = $_SESSION['error'] ?? null;
unset($_SESSION['error']);
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title><?= $pageTitle ?></title>
<!-- [Keep all your existing CSS styles] -->
<style>
/* Your existing CSS styles remain unchanged */
</style>
</head>
<body>
<div class="login-container">
<div class="login-card">
<div class="login-header">
<h2>Login</h2>
<p>Access your CrossFit account</p>
</div>

<?php if($error): ?>
<div class="alert"><?= htmlspecialchars($error) ?></div>
<?php endif; ?>

<form method="POST">
<div class="form-group">
<input type="email" class="form-control" id="email" name="email" placeholder="Email" required>
</div>
```

```
<div class="form-group">
<input type="password" class="form-control" id="password" name="password"
placeholder="Password" required>
</div>
<button type="submit" class="btn">Login</button>
</form>

<div class="login-links">
<p>Don't have an account? <a href="join.php">Join us</a></p>
</div>
</div>
</div>
</body>
</html>
<?php ob_end_flush(); ?>

$pageTitle = "Login";
$error = $_SESSION['error'] ?? null;
unset($_SESSION['error']);
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title><?= $pageTitle ?></title>
<style>
:root {
```

```
--primary: #FF5A1F;
--primary-dark: #E04A14;
--dark: #121212;
--darker: #0A0A0A;
--light: #F8F9FA;
--text-dark: #E0E0E0;
--text-light: #FFFFFF;
}

body {
font-family: 'Montserrat', sans-serif;
background-color: var(--dark);
color: var(--text-light);
line-height: 1.6;
margin: 0;
padding: 0;
}

h1, h2, h3, h4, h5, h6 {
font-family: 'Oswald', sans-serif;
font-weight: 700;
letter-spacing: 1px;
color: var(--text-light);
margin-top: 0;
}

.login-container {
display: flex;
justify-content: center;
```

```
align-items: center;
min-height: 100vh;
background-color: var(--darker);
padding: 20px;
}

.login-card {
background-color: var(--dark);
border-radius: 10px;
padding: 40px;
box-shadow: 0 5px 15px rgba(0,0,0,0.3);
border: 1px solid rgba(255,255,255,0.1);
width: 100%;
max-width: 450px;
transition: all 0.3s ease;
}

.login-card:hover {
transform: translateY(-5px);
box-shadow: 0 15px 30px rgba(255, 90, 31, 0.2);
}

.login-header {
text-align: center;
margin-bottom: 30px;
}

.login-header h2 {
font-size: 2.2rem;
```

```
text-transform: uppercase;
position: relative;
display: inline-block;
margin-bottom: 15px;
}

.login-header h2:after {
content: '';
position: absolute;
left: 50%;
bottom: -10px;
width: 60px;
height: 4px;
background: var(--primary);
transform: translateX(-50%);
}

.login-header p {
margin-bottom: 0;
color: var(--text-dark);
}

.form-group {
margin-bottom: 20px;
}

.form-control {
background-color: var(--darker);
border: 1px solid rgba(255,255,255,0.1);
```

```
color: var(--text-light);
padding: 12px 15px;
width: 100%;
border-radius: 5px;
font-size: 16px;
}

.form-control:focus {
background-color: var(--darker);
color: var(--text-light);
border-color: var(--primary);
box-shadow: 0 0 0 0.25rem rgba(255, 90, 31, 0.25);
outline: none;
}

.btn {
display: block;
width: 100%;
padding: 14px;
background-color: var(--primary);
color: white;
border: none;
border-radius: 50px;
font-weight: 700;
text-transform: uppercase;
letter-spacing: 1px;
cursor: pointer;
transition: all 0.3s ease;
font-size: 16px;
```

```
margin-top: 10px;
}

.btn:hover {
background-color: var(--primary-dark);
transform: translateY(-3px);
box-shadow: 0 8px 20px rgba(255, 90, 31, 0.4);
}

.alert {
padding: 15px;
margin-bottom: 25px;
border-radius: 5px;
background-color: rgba(220, 53, 69, 0.2);
border: 1px solid rgba(220, 53, 69, 0.3);
color: #dc3545;
}

.login-links {
text-align: center;
margin-top: 25px;
font-size: 14px;
}

.login-links a {
color: var(--primary);
text-decoration: none;
transition: all 0.3s ease;
}
```

```
.login-links a:hover {
color: var(--primary-dark);
text-decoration: underline;
}

@media (max-width: 576px) {
.login-card {
padding: 30px 20px;
}

.login-header h2 {
font-size: 1.8rem;
}
}
</style>
</head>
<body>
<div class="login-container">
<div class="login-card">
<div class="login-header">
<h2>Login</h2>
<p>Access your CrossFit account</p>
</div>

<?php if($error): ?>
<div class="alert"><?= htmlspecialchars($error) ?></div>
<?php endif; ?>
```

```
<form method="POST">
<div class="form-group">
<input type="email" class="form-control" id="email" name="email" placeholder="Email" required>
</div>
<div class="form-group">
<input type="password" class="form-control" id="password" name="password" placeholder="Password" required>
</div>
<button type="submit" class="btn">Login</button>
</form>

<div class="login-links">
<p>Don't have an account? <a href="join.php">Join us</a></p>

</div>
</div>
</div>
</body>
</html>
<?php ob_end_flush(); ?>
```

User Registration Form (Signup form)

```
<?php
ob_start();
session_start();
require_once 'config/database.php';

$error = null;
```

```
$success = null;

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
$full_name = trim($_POST['full_name']);
$email = trim($_POST['email']);
$password = $_POST['password'];
$role = 'member'; // Set default role to 'member'

try {
// Validate inputs
if (empty($full_name)) throw new Exception("Full name is required");
if (empty($email)) throw new Exception("Email is required");
if (empty($password)) throw new Exception("Password is required");

if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
throw new Exception("Invalid email format");
}

if (strlen($password) < 8) {
throw new Exception("Password must be at least 8 characters");
}

// Check for existing email
$stmt = $conn->prepare("SELECT user_id FROM users WHERE email = ?");
$stmt->execute([$email]);

if ($stmt->rowCount() > 0) {
// Email exists - provide helpful message with options
```

```
$error = "This email is already registered. Please <a href='login.php' style='color: #FF5A1F; font-weight: 600;'>login here</a> or <a href='forgot_password.php' style='color: #FF5A1F; font-weight: 600;'>reset your password</a> if you've forgotten it.";
} else {
// Hash password
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

// Begin transaction
$conn->beginTransaction();

// Insert user with default role
$stmt = $conn->prepare("INSERT INTO users (email, password, role) VALUES (?, ?, ?)");
$stmt->execute([$email, $hashed_password, $role]);
$user_id = $conn->lastInsertId();

// Insert member without join_date
$stmt = $conn->prepare("INSERT INTO members (user_id, full_name) VALUES (?, ?)");
$stmt->execute([$user_id, $full_name]);

$conn->commit();

$success = "Registration successful! You can now <a href='login.php' style='color: #28a745; font-weight: 600;'>login here</a>.";
}

} catch (PDOException $e) {
if ($conn->inTransaction()) {
$conn->rollBack();
}
```

```
// Check for duplicate entry error (MySQL error code 1062)
if ($e->errorInfo[1] == 1062) {
$error = "This email is already registered. Please <a href='login.php' style='color: #FF5A1F; font-weight: 600;'>login here</a> or <a href='forgot_password.php' style='color: #FF5A1F; font-weight: 600;'>reset your password</a> if you've forgotten it.";
} else {
$error = "Registration error. Please try again later.";
// Log the actual error for debugging: error_log($e->getMessage());
}
} catch (Exception $e) {
if ($conn->inTransaction()) {
$conn->rollBack();
}
$error = $e->getMessage();
}
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Join CrossFit Revolution</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.8.0/font/bootstrap-icons.css" rel="stylesheet">
<style>
:root {
--primary: #FF5A1F;
--primary-dark: #E04A14;
```

```
--dark: #121212;
--darker: #0A0A0A;
--light: #F8F9FA;
--text-dark: #E0E0E0;
--text-light: #FFFFFF;
--success: #28a745;
--info: #17a2b8;
--warning: #ffc107;
--danger: #dc3545;
}

* {
margin: 0;
padding: 0;
box-sizing: border-box;
}

body {
font-family: 'Montserrat', sans-serif;
background-color: var(--darker);
color: var(--text-light);
display: flex;
justify-content: center;
align-items: center;
min-height: 100vh;
padding: 20px;
line-height: 1.6;
}
```

```
.join-container {
width: 100%;
max-width: 500px;
}

.join-card {
background-color: var(--dark);
border-radius: 10px;
padding: 30px;
box-shadow: 0 10px 25px rgba(0,0,0,0.3);
border: 1px solid rgba(255,255,255,0.1);
}

.logo {
text-align: center;
margin-bottom: 25px;
}

.logo h1 {
color: var(--primary);
font-size: 2.2rem;
margin-bottom: 5px;
}

.logo p {
color: var(--text-dark);
font-size: 1rem;
}
```

```
.alert {
padding: 15px;
margin-bottom: 20px;
border-radius: 5px;
font-size: 0.95rem;
}

.alert-danger {
background-color: rgba(220,53,69,0.2);
border: 1px solid rgba(220,53,69,0.3);
color: var(--danger);
}

.alert-success {
background-color: rgba(40,167,69,0.2);
border: 1px solid rgba(40,167,69,0.3);
color: var(--success);
}

.alert a {
color: inherit;
font-weight: 600;
text-decoration: underline;
}

.form-group {
margin-bottom: 20px;
}
```

```
.form-group label {
display: block;
margin-bottom: 8px;
color: var(--text-light);
font-weight: 600;
}

.form-control {
background-color: var(--darker);
border: 1px solid rgba(255,255,255,0.1);
color: var(--text-light);
padding: 12px 15px;
width: 100%;
border-radius: 5px;
font-size: 1rem;
transition: all 0.3s ease;
}

.form-control:focus {
outline: none;
border-color: var(--primary);
box-shadow: 0 0 0 2px rgba(255,90,31,0.3);
}

select.form-control {
appearance: none;
-webkit-appearance: none;
-moz-appearance: none;
```

```
background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='16' height='16' fill='%23E0E0E0' viewBox='0 0 16 16'%3E%3Cpath d='M8 12L2 6h12L8 12z'/%3E%3C/svg%3E");
background-repeat: no-repeat;
background-position: right 15px center;
background-size: 16px;
padding-right: 40px;
}

.btn {
background-color: var(--primary);
color: white;
border: none;
padding: 14px;
width: 100%;
border-radius: 5px;
cursor: pointer;
font-weight: bold;
font-size: 1rem;
text-transform: uppercase;
letter-spacing: 1px;
transition: all 0.3s ease;
}

.btn:hover {
background-color: var(--primary-dark);
transform: translateY(-2px);
box-shadow: 0 5px 15px rgba(255,90,31,0.4);
}
```

```
.login-link {
text-align: center;
margin-top: 25px;
color: var(--text-dark);
}

.login-link a {
color: var(--primary);
text-decoration: none;
font-weight: 600;
}

.login-link a:hover {
text-decoration: underline;
}

.password-requirements {
font-size: 0.8rem;
color: var(--text-dark);
margin-top: 5px;
}
</style>
</head>
<body>
<div class="join-container">
<div class="join-card">
<div class="logo">
<h1><i class="bi bi-lightning-charge"></i> CrossFit Revolution</h1>
<p>Create your account</p>
```

```
</div>

<?php if ($error): ?>
<div class="alert alert-danger"><?= $error ?></div>
<?php elseif ($success): ?>
<div class="alert alert-success"><?= $success ?></div>
<?php endif; ?>

<form method="POST" id="registrationForm">
<div class="form-group">
<label for="full_name">Full Name</label>
<input type="text" id="full_name" class="form-control" name="full_name"
value="<?= isset($_POST['full_name']) ? htmlspecialchars($_POST['full_name']) : '' ?>"
required autofocus>
</div>

<div class="form-group">
<label for="email">Email Address</label>
<input type="email" id="email" class="form-control" name="email"
value="<?= isset($_POST['email']) ? htmlspecialchars($_POST['email']) : '' ?>"
required>
</div>

<div class="form-group">
<label for="password">Password</label>
<input type="password" id="password" class="form-control" name="password" required
minlength="8">
<div class="password-requirements">Must be at least 8 characters long</div>
</div>
```

```
<button type="submit" class="btn">Create Account</button>
</form>

<div class="login-link">
Already have an account? <a href="login.php">Login here</a>
</div>
</div>
</div>

<script>
// Simple form validation
document.getElementById('registrationForm').addEventListener('submit', function(e) {
const password = document.getElementById('password').value;

if (password.length < 8) {
e.preventDefault();
alert('Password must be at least 8 characters long');
return false;
}
});
</script>
</body>
</html>
<?php ob_end_flush(); ?>
```

Book Package [Member]

```
<?php
ob_start();
session_start();
require_once '../config/database.php';

// Verify member access
if (!isset($_SESSION['user_id']) || $_SESSION['role'] !== 'member') {
header("Location: ../index.php");
exit();
}

// Get member info
$stmt = $conn->prepare("SELECT * FROM members WHERE user_id = ?");
$stmt->execute([$_SESSION['user_id']]);
$member = $stmt->fetch();

// Check if member has active booking
$activeBooking = $conn->prepare("
SELECT b.*, p.name as package_name, p.duration_months, p.price, p.description, p.features
FROM bookings b
JOIN packages p ON b.package_id = p.package_id
WHERE b.member_id = ? AND b.status = 'active'
ORDER BY b.end_date DESC
LIMIT 1
");
$activeBooking->execute([$member['member_id']]);
$currentPackage = $activeBooking->fetch();
```

```
// Handle package booking only if no active booking exists
if ($_SERVER['REQUEST_METHOD'] == 'POST' && !$currentPackage) {
$package_id = $_POST['package_id'];
$start_date = $_POST['start_date'];

try {
// Get package details for success message
$stmt = $conn->prepare("SELECT name, duration_months, price FROM packages WHERE package_id = ?");
$stmt->execute([$package_id]);
$package = $stmt->fetch();

$end_date = date('Y-m-d', strtotime($start_date . " + " . $package['duration_months'] . " months"));

// Create booking
$stmt = $conn->prepare("INSERT INTO bookings (member_id, package_id, start_date, end_date, status) VALUES (?, ?, ?, ?, 'active')");
$stmt->execute([$member['member_id'], $package_id, $start_date, $end_date]);

$_SESSION['success'] = "Package booked successfully! You have subscribed to " . $package['name'] . " for " . $package['duration_months'] . " months.";
header("Location: book_package.php");
exit();
} catch(PDOException $e) {
$error = "Booking failed: " . $e->getMessage();
}
}

// Get the specific package if package_id is set and no active booking
```

```
$package = null;
if (isset($_GET['package_id']) && !$currentPackage) {
$package_id = $_GET['package_id'];
$stmt = $conn->prepare("SELECT * FROM packages WHERE package_id = ?");
$stmt->execute([$package_id]);
$package = $stmt->fetch();
}

// Get all packages if no specific package is selected and no active booking
if (!$package && !$currentPackage) {
$packages = $conn->query("SELECT * FROM packages")->fetchAll();
}

$pageTitle = "Book a Package";
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title><?= htmlspecialchars($pageTitle) ?></title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">
<style>
:root {
--primary: #FF5A1F;
--primary-dark: #E04A14;
--dark: #121212;
```

```
--darker: #0A0A0A;
--light: #F8F9FA;
--text-dark: #E0E0E0;
--text-light: #FFFFFF;
--success: #28a745;
--danger: #dc3545;
--info: #17a2b8;
}

* {
margin: 0;
padding: 0;
box-sizing: border-box;
}

body {
font-family: 'Montserrat', sans-serif;
background-color: var(--darker);
color: var(--text-light);
line-height: 1.6;
min-height: 100vh;
padding: 20px;
}

h1, h2, h3, h4, h5, h6 {
font-family: 'Oswald', sans-serif;
font-weight: 700;
letter-spacing: 1px;
color: var(--text-light);
```

```
margin-top: 0;
}

.booking-container {
max-width: 1200px;
margin: 0 auto;
}

.page-header {
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 2rem;
padding-bottom: 1rem;
border-bottom: 1px solid rgba(255,255,255,0.1);
}

.btn {
display: inline-flex;
align-items: center;
gap: 8px;
padding: 10px 20px;
background-color: var(--primary);
color: white;
text-decoration: none;
border-radius: 25px;
font-weight: 600;
transition: all 0.3s ease;
border: none;
```

```
cursor: pointer;
font-size: 14px;
}

.btn:hover {
background-color: var(--primary-dark);
transform: translateY(-2px);
box-shadow: 0 5px 15px rgba(255, 90, 31, 0.3);
}

.btn-outline {
background-color: transparent;
border: 2px solid var(--primary);
color: var(--primary);
}

.btn-outline:hover {
background-color: var(--primary);
color: white;
}

.btn-success {
background-color: var(--success);
}

.btn-success:hover {
background-color: #218838;
}
```

```
.card {
background-color: var(--dark);
border-radius: 15px;
padding: 2rem;
margin-bottom: 2rem;
border: 1px solid rgba(255,255,255,0.1);
box-shadow: 0 5px 15px rgba(0,0,0,0.2);
transition: all 0.3s ease;
}

.card:hover {
transform: translateY(-5px);
box-shadow: 0 10px 25px rgba(255, 90, 31, 0.2);
border-color: var(--primary);
}

.current-package-card {
border-left: 4px solid var(--success);
}

.form-group {
margin-bottom: 1.5rem;
}

.form-label {
display: block;
margin-bottom: 0.8rem;
font-weight: 600;
color: var(--text-light);
```

```
}

.form-control {
width: 100%;
padding: 14px 16px;
background-color: rgba(255,255,255,0.05);
border: 1px solid rgba(255,255,255,0.1);
border-radius: 8px;
color: var(--text-light);
font-family: 'Montserrat', sans-serif;
transition: all 0.3s ease;
}

.form-control:focus {
outline: none;
border-color: var(--primary);
box-shadow: 0 0 0 3px rgba(255, 90, 31, 0.2);
background-color: rgba(255,255,255,0.08);
}

.alert {
padding: 1.5rem;
border-radius: 10px;
margin-bottom: 2rem;
display: flex;
align-items: center;
gap: 15px;
border: 2px solid transparent;
animation: slideIn 0.5s ease-out;
```

```
}

@keyframes slideIn {
from {
opacity: 0;
transform: translateY(-20px);
}
to {
opacity: 1;
transform: translateY(0);
}
}

.alert-danger {
background-color: rgba(220, 53, 69, 0.15);
border-color: rgba(220, 53, 69, 0.3);
color: #f8d7da;
}

.alert-success {
background-color: rgba(40, 167, 69, 0.15);
border-color: rgba(40, 167, 69, 0.3);
color: #d4edda;
}

.alert-info {
background-color: rgba(23, 162, 184, 0.15);
border-color: rgba(23, 162, 184, 0.3);
color: #d1ecf1;
```

```
}

.alert-icon {
font-size: 1.5rem;
flex-shrink: 0;
}

.list-group {
list-style: none;
margin: 1.5rem 0;
}

.list-group-item {
background-color: var(--darker);
color: var(--text-light);
padding: 1rem;
border: 1px solid rgba(255,255,255,0.1);
border-radius: 5px;
margin-bottom: 0.5rem;
display: flex;
justify-content: space-between;
align-items: center;
}

.list-group-item:last-child {
margin-bottom: 0;
}

.package-grid {
```

```
display: grid;
grid-template-columns: repeat(auto-fit, minmax(350px, 1fr));
gap: 1.5rem;
margin-top: 1.5rem;
}

.package-card {
background-color: var(--dark);
border-radius: 15px;
padding: 2rem;
border: 1px solid rgba(255,255,255,0.1);
box-shadow: 0 5px 15px rgba(0,0,0,0.2);
transition: all 0.3s ease;
display: flex;
flex-direction: column;
height: 100%;
}

.package-card:hover {
transform: translateY(-5px);
box-shadow: 0 10px 25px rgba(255, 90, 31, 0.2);
border-color: var(--primary);
}

.package-header {
margin-bottom: 1.5rem;
}

.package-price {
```

```
font-size: 1.8rem;
font-weight: bold;
color: var(--primary);
margin: 1rem 0;
}

.package-features {
flex-grow: 1;
margin-bottom: 1.5rem;
}

.package-footer {
margin-top: auto;
text-align: center;
}

.feature-list {
list-style: none;
margin: 1rem 0;
}

.feature-list li {
padding: 0.5rem 0;
border-bottom: 1px solid rgba(255,255,255,0.05);
display: flex;
align-items: center;
gap: 10px;
}
```

```
.feature-list li:last-child {
border-bottom: none;
}

.feature-list li i {
color: var(--primary);
}

.form-actions {
display: grid;
gap: 1rem;
margin-top: 2rem;
}

.status-badge {
background-color: var(--success);
color: white;
padding: 6px 12px;
border-radius: 20px;
font-size: 0.8rem;
font-weight: 600;
display: inline-block;
margin-bottom: 1rem;
}

.package-details {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
gap: 1rem;
```

```
margin-top: 1rem;
}

.package-detail {
display: flex;
flex-direction: column;
}

.detail-label {
font-size: 0.8rem;
color: var(--text-dark);
margin-bottom: 0.2rem;
}

.detail-value {
color: var(--text-light);
font-weight: 600;
}

@media (max-width: 768px) {
.page-header {
flex-direction: column;
align-items: flex-start;
gap: 1rem;
}

.package-grid {
grid-template-columns: 1fr;
}
```

```
.card {
padding: 1.5rem;
}

.package-card {
padding: 1.5rem;
}

.alert {
padding: 1.2rem;
flex-direction: column;
text-align: center;
}
}

@media (max-width: 576px) {
body {
padding: 15px;
}

.booking-container {
padding: 0;
}

.btn {
width: 100%;
justify-content: center;
}
```

```
}

.text-muted {
color: var(--text-dark) !important;
}
</style>
</head>
<body>
<div class="booking-container">
<!-- Page Header -->
<div class="page-header">
<h1><i class="bi bi-box-seam"></i> Book a Package</h1>
<a href="dashboard.php" class="btn btn-outline">
<i class="bi bi-arrow-left"></i> Back to Dashboard
</a>
</div>

<!-- Alerts -->
<?php if(isset($error)): ?>
<div class="alert alert-danger">
<div class="alert-icon">⚠</div>
<div>
<strong>Booking Failed</strong><br>
<?= htmlspecialchars($error) ?>
</div>
</div>
<?php endif; ?>

<?php if(isset($_SESSION['success'])): ?>
```

```
<div class="alert alert-success">
<div class="alert-icon">✓</div>
<div>
<strong>Success!</strong><br>
<?= htmlspecialchars($_SESSION['success']) ?>
</div>
</div>
<?php unset($_SESSION['success']); ?>
<?php endif; ?>

<!-- Current Active Package -->
<?php if($currentPackage): ?>
<div class="card current-package-card">
<span class="status-badge">
<i class="bi bi-check-circle"></i> Active Membership
</span>
<div class="package-header">
<h2><?= htmlspecialchars($currentPackage['package_name']) ?></h2>
<p class="text-muted"><?= htmlspecialchars($currentPackage['description']) ?></p>
</div>

<div class="package-details">
<div class="package-detail">
<span class="detail-label">Duration</span>
<span class="detail-value"><?= $currentPackage['duration_months'] ?> month<?= $currentPackage['duration_months'] > 1 ? 's' : '' ?></span>
</div>
<div class="package-detail">
<span class="detail-label">Price</span>
```

₹<?= number_format($currentPackage['price'], 2) ?>

Start Date

<?= date('M j, Y', strtotime($currentPackage['start_date'])) ?>

End Date

<?= date('M j, Y', strtotime($currentPackage['end_date'])) ?>

<h4>Package Features</h4>

<?= htmlspecialchars($currentPackage['features']) ?>

i

Active Membership

You currently have an active package. You can book a new package only after your current membership expires on <?= date('F j, Y', strtotime($currentPackage['end_date'])) ?>.

<?php else: ?>

<!-- Package Booking Form or Package List -->

```
        <?php if($package): ?>
<!-- Single Package Booking Form -->
<form method="POST">
<input type="hidden" name="package_id" value="<?= $package['package_id'] ?>">

<div class="card">
<div class="package-header">
<h2><?= htmlspecialchars($package['name']) ?></h2>
<p class="text-muted"><?= htmlspecialchars($package['description']) ?></p>
</div>

<div class="package-features">
<ul class="list-group">
<li class="list-group-item">
<span><i class="bi bi-calendar"></i> Duration</span>
<strong><?= $package['duration_months'] ?> month<?= $package['duration_months'] > 1 ? 's' : '' ?></strong>
</li>
<li class="list-group-item">
<span><i class="bi bi-currency-rupee"></i> Price</span>
<strong>₹<?= number_format($package['price'], 2) ?></strong>
</li>
<li class="list-group-item">
<span><i class="bi bi-star"></i> Features</span>
<span><?= htmlspecialchars($package['features']) ?></span>
</li>
</ul>
</div>
 <div class="form-group">
```

```
<label for="start_date" class="form-label">
<i class="bi bi-calendar-check"></i> Start Date
</label>
<input type="date" class="form-control" id="start_date" name="start_date"
min="<?= date('Y-m-d') ?>" required>
</div>

<div class="form-actions">
<button type="submit" class="btn">
<i class="bi bi-check-circle"></i> Confirm Booking
</button>
<a href="book_package.php" class="btn btn-outline">
<i class="bi bi-arrow-left"></i> Back to Packages
</a>
</div>
</div>
</form>
<?php else: ?>
<!-- Package List -->
<div class="package-grid">
<?php foreach($packages as $p): ?>
<div class="package-card">
<div class="package-header">
<h3><?= htmlspecialchars($p['name']) ?></h3>
<p class="text-muted"><?= htmlspecialchars($p['description']) ?></p>
<div class="package-price">₹<?= number_format($p['price'], 2) ?></div>
</div>

<div class="package-features">
```

```
<ul class="feature-list">
<li><i class="bi bi-check-circle"></i> Duration: <?= $p['duration_months'] ?> month<?= $p['duration_months'] > 1 ? 's' : '' ?></li>
<li><i class="bi bi-check-circle"></i> <?= htmlspecialchars($p['features']) ?></li>
</ul>
</div>

<div class="package-footer">
<a href="book_package.php?package_id=<?= $p['package_id'] ?>" class="btn">
<i class="bi bi-bookmark-check"></i> Book Now
</a>
</div>
</div>
<?php endforeach; ?>
</div>
<?php endif; ?>
<?php endif; ?>
</div>

<script>
// Add hover effects to cards
document.querySelectorAll('.card, .package-card').forEach(card => {
card.addEventListener('mouseenter', function() {
this.style.transform = 'translateY(-5px)';
this.style.boxShadow = '0 10px 25px rgba(255, 90, 31, 0.2)';
});

card.addEventListener('mouseleave', function() {
this.style.transform = 'translateY(0)';
```

```
this.style.boxShadow = '0 5px 15px rgba(0,0,0,0.2)';
});
});

// Set minimum date for date input
document.addEventListener('DOMContentLoaded', function() {
const startDateInput = document.getElementById('start_date');
if (startDateInput) {
startDateInput.min = new Date().toISOString().split('T')[0];
}
});
</script>
</body>
</html>
<?php ob_end_flush(); ?>
```

Book Trainer[Member]

```
<?php
require_once '../includes/auth.php';

if ($_SESSION['role'] !== 'member') {
header("Location: ../index.php");
exit();
}

// Get member info
$stmt = $conn->prepare("SELECT * FROM members WHERE user_id = ?");
$stmt->execute([$_SESSION['user_id']]);
```

```
$member = $stmt->fetch();

// Get member's active package bookings
$memberPackages = $conn->query("
SELECT p.*, b.start_date, b.end_date
FROM packages p
JOIN bookings b ON p.package_id = b.package_id
WHERE b.member_id = {$member['member_id']}
AND b.status = 'active'
AND b.trainer_id IS NULL
ORDER BY b.start_date DESC
")->fetchAll();

// Handle trainer booking
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
$trainer_id = $_POST['trainer_id'];
$package_id = $_POST['package_id'];
$start_date = $_POST['start_date'];
$notes = $_POST['notes'] ?? '';

try {
// Verify the package belongs to the member
$validPackage = false;
foreach($memberPackages as $package) {
if ($package['package_id'] == $package_id) {
$validPackage = true;
break;
}
}
```

```
if (!$validPackage) {
throw new Exception("Invalid package selection");
}

// Get package duration
$stmt = $conn->prepare("SELECT duration_months FROM packages WHERE package_id = ?");
$stmt->execute([$package_id]);
$package = $stmt->fetch();

$end_date = date('Y-m-d', strtotime($start_date . " + " . $package['duration_months'] . " months"));

// Get trainer name for success message
$stmt = $conn->prepare("SELECT full_name FROM trainers WHERE trainer_id = ?");
$stmt->execute([$trainer_id]);
$trainer = $stmt->fetch();

// Create booking
$stmt = $conn->prepare("INSERT INTO bookings (member_id, trainer_id, package_id, start_date, end_date, notes, status) VALUES (?, ?, ?, ?, ?, ?, 'active')");
$stmt->execute([$member['member_id'], $trainer_id, $package_id, $start_date, $end_date, $notes]);

$_SESSION['success'] = "Trainer booked successfully! You are now working with " . $trainer['full_name'] . ".";
header("Location: book_trainer.php");
exit();
} catch(PDOException $e) {
$error = "Booking failed: " . $e->getMessage();
```

```
} catch (Exception $e) {
$error = $e->getMessage();
}
}

// Get all trainers
$trainers = $conn->query("SELECT * FROM trainers")->fetchAll();

// Get successfully booked packages with trainers (for display)
$successfulBookings = $conn->query("
SELECT b.*, p.name as package_name, p.duration_months, p.price,
t.full_name as trainer_name, t.specialization
FROM bookings b
JOIN packages p ON b.package_id = p.package_id
JOIN trainers t ON b.trainer_id = t.trainer_id
WHERE b.member_id = {$member['member_id']}
AND b.status = 'active'
AND b.trainer_id IS NOT NULL
ORDER BY b.start_date DESC
")->fetchAll();
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Book a Trainer - CrossFit Revolution</title>
<style>
```

```
:root {
--primary: #FF5A1F;
--primary-dark: #E04A14;
--dark: #121212;
--darker: #0A0A0A;
--light: #F8F9FA;
--text-dark: #E0E0E0;
--text-light: #FFFFFF;
}

* {
margin: 0;
padding: 0;
box-sizing: border-box;
}

body {
font-family: 'Montserrat', sans-serif;
background-color: var(--dark);
color: var(--text-dark);
line-height: 1.6;
min-height: 100vh;
padding: 20px;
}

h1, h2, h3, h4, h5, h6 {
font-family: 'Oswald', sans-serif;
font-weight: 700;
letter-spacing: 1px;
```

```
color: var(--text-light);
}

.container {
max-width: 1200px;
margin: 0 auto;
}

.page-header {
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 2rem;
padding-bottom: 1rem;
border-bottom: 1px solid rgba(255,255,255,0.1);
}

.btn {
display: inline-block;
padding: 12px 30px;
background-color: var(--primary);
color: white;
text-decoration: none;
border-radius: 50px;
font-weight: 700;
text-transform: uppercase;
letter-spacing: 1px;
transition: all 0.3s ease;
border: none;
```

```
cursor: pointer;
font-size: 0.9rem;
box-shadow: 0 5px 15px rgba(255, 90, 31, 0.4);
}

.btn:hover {
background-color: var(--primary-dark);
transform: translateY(-3px);
box-shadow: 0 8px 20px rgba(255, 90, 31, 0.6);
}

.btn-outline {
background-color: transparent;
border: 2px solid var(--primary);
color: var(--primary);
}

.btn-outline:hover {
background-color: var(--primary);
color: white;
}

.booking-form {
background-color: var(--darker);
padding: 2.5rem;
border-radius: 10px;
box-shadow: 0 10px 30px rgba(0,0,0,0.3);
margin-bottom: 2rem;
}
```

```
.form-label {
display: block;
margin-bottom: 0.5rem;
font-weight: 600;
color: var(--text-light);
}

.form-control, .form-select {
width: 100%;
padding: 12px 15px;
background-color: rgba(255,255,255,0.05);
border: 1px solid rgba(255,255,255,0.1);
border-radius: 5px;
color: var(--text-dark);
font-family: 'Montserrat', sans-serif;
transition: all 0.3s ease;
appearance: none;
-webkit-appearance: none;
-moz-appearance: none;
}

.form-select {
background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 24 24' fill='%23E0E0E0'%3E%3Cpath d='M7 10l5 5 5-5z'/%3E%3C/svg%3E");
background-repeat: no-repeat;
background-position: right 15px center;
background-size: 16px;
padding-right: 45px;
```

```
}

.form-select option {
background-color: var(--darker);
color: var(--text-dark);
padding: 12px;
}

.form-control:focus, .form-select:focus {
outline: none;
border-color: var(--primary);
box-shadow: 0 0 0 3px rgba(255, 90, 31, 0.3);
background-color: rgba(255,255,255,0.08);
}

.form-control::placeholder {
color: rgba(224, 224, 224, 0.6);
}

.form-control[type="date"] {
color-scheme: dark;
}

textarea.form-control {
resize: vertical;
min-height: 100px;
}

.form-row {
```

```
display: grid;
grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
gap: 1.5rem;
margin-bottom: 1.5rem;
}

.form-group {
margin-bottom: 1.5rem;
}

.alert {
padding: 1.5rem;
border-radius: 10px;
margin-bottom: 2rem;
display: flex;
align-items: center;
gap: 15px;
border: 2px solid transparent;
animation: slideIn 0.5s ease-out;
}

@keyframes slideIn {
from {
opacity: 0;
transform: translateY(-20px);
}
to {
opacity: 1;
transform: translateY(0);
```

```
}
}

.alert-danger {
background-color: rgba(220, 53, 69, 0.15);
border-color: rgba(220, 53, 69, 0.3);
color: #f8d7da;
}

.alert-success {
background-color: rgba(40, 167, 69, 0.15);
border-color: rgba(40, 167, 69, 0.3);
color: #d4edda;
}

.alert-info {
background-color: rgba(23, 162, 184, 0.15);
border-color: rgba(23, 162, 184, 0.3);
color: #d1ecf1;
}

.alert-icon {
font-size: 1.5rem;
flex-shrink: 0;
}

.btn-group {
display: grid;
grid-template-columns: 1fr 1fr;
```

```
gap: 1rem;
margin-top: 2rem;
}

@media (max-width: 768px) {
.btn-group {
grid-template-columns: 1fr;
}

.page-header {
flex-direction: column;
align-items: flex-start;
gap: 1rem;
}

.booking-form {
padding: 1.5rem;
}

.alert {
padding: 1.2rem;
flex-direction: column;
text-align: center;
}
}

/* Trainer cards preview */
.trainer-preview {
display: grid;
```

```
grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
gap: 1.5rem;
margin-top: 2rem;
}

.trainer-card {
background-color: var(--darker);
border-radius: 10px;
overflow: hidden;
transition: all 0.3s ease;
border-left: 4px solid var(--primary);
}

.trainer-card:hover {
transform: translateY(-5px);
box-shadow: 0 10px 25px rgba(0,0,0,0.3);
}

.trainer-info {
padding: 1.5rem;
}

.trainer-specialization {
color: var(--primary);
font-weight: 600;
margin-bottom: 1rem;
display: block;
}
```

```
.select-trainer-btn {
background: transparent;
border: 2px solid var(--primary);
color: var(--primary);
padding: 8px 15px;
border-radius: 50px;
font-weight: 600;
cursor: pointer;
transition: all 0.3s ease;
margin-top: 1rem;
}

.select-trainer-btn:hover {
background: var(--primary);
color: white;
}

/* Success bookings section */
.success-bookings {
margin-top: 3rem;
padding-top: 2rem;
border-top: 1px solid rgba(255,255,255,0.1);
}

.booking-card {
background-color: var(--darker);
border-radius: 10px;
padding: 1.5rem;
margin-bottom: 1rem;
```

```
border-left: 4px solid #28a745;
transition: all 0.3s ease;
}

.booking-card:hover {
transform: translateY(-2px);
box-shadow: 0 5px 15px rgba(0,0,0,0.3);
}

.booking-header {
display: flex;
justify-content: space-between;
align-items: flex-start;
margin-bottom: 1rem;
}

.booking-trainer {
color: var(--primary);
font-weight: 600;
font-size: 1.1rem;
}

.booking-package {
color: var(--text-light);
font-weight: 600;
}

.booking-details {
display: grid;
```

```
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
gap: 1rem;
margin-top: 1rem;
}

.booking-detail {
display: flex;
flex-direction: column;
}

.detail-label {
font-size: 0.8rem;
color: var(--text-dark);
margin-bottom: 0.2rem;
}

.detail-value {
color: var(--text-light);
font-weight: 600;
}

.status-badge {
background-color: #28a745;
color: white;
padding: 4px 12px;
border-radius: 20px;
font-size: 0.8rem;
font-weight: 600;
}
```

```
/* Additional dropdown improvements */
.form-select:hover {
background-color: rgba(255,255,255,0.08);
border-color: rgba(255,255,255,0.2);
}

.form-select.focused {
background-color: rgba(255,255,255,0.08);
transform: translateY(-1px);
}

.form-select.has-value {
border-color: rgba(255, 90, 31, 0.3);
}
</style>
</head>
<body>
<div class="container">
<div class="page-header">
<h1>Book a Trainer</h1>
<a href="dashboard.php" class="btn btn-outline">Back to Dashboard</a>
</div>

<?php if(isset($error)): ?>
<div class="alert alert-danger">
<div class="alert-icon">⚠</div>
<div>
<strong>Booking Failed</strong><br>
```

```
<?php echo $error; ?>
</div>
</div>
<?php endif; ?>

<?php if(isset($_SESSION['success'])): ?>
<div class="alert alert-success">
<div class="alert-icon">✓</div>
<div>
<strong>Success!</strong><br>
<?php echo $_SESSION['success']; unset($_SESSION['success']); ?>
</div>
</div>
<?php endif; ?>

<?php if(empty($memberPackages)): ?>
<div class="alert alert-info">
<div class="alert-icon">i</div>
<div>
<h3>No Active Packages Found</h3>
<p>You need to have an active package to book a trainer.</p>
<a href="book_package.php" class="btn" style="margin-top: 1rem;">Book a Package First</a>
</div>
</div>
<?php else: ?>
<div class="booking-form">
<form method="POST" id="bookingForm">
<div class="form-row">
```

```
<div class="form-group">
<label for="trainer_id" class="form-label">Select Trainer</label>
<select class="form-select" id="trainer_id" name="trainer_id" required>
<option value="">Choose Trainer</option>
<?php foreach($trainers as $trainer): ?>
<option value="<?= $trainer['trainer_id'] ?>">
<?= htmlspecialchars($trainer['full_name']) ?> - <?= htmlspecialchars($trainer['specialization']) ?>
</option>
<?php endforeach; ?>
</select>
</div>

<div class="form-group">
<label for="package_id" class="form-label">Select Your Package</label>
<select class="form-select" id="package_id" name="package_id" required>
<option value="">Choose Your Package</option>
<?php foreach($memberPackages as $package): ?>
<option value="<?= $package['package_id'] ?>">
<?= htmlspecialchars($package['name']) ?>
(₹<?= number_format($package['price'], 2) ?> -
<?= $package['duration_months'] ?> months)
</option>
<?php endforeach; ?>
</select>
</div>
</div>

<div class="form-group">
```

```
<label for="start_date" class="form-label">Training Start Date</label>

<input type="date" class="form-control" id="start_date" name="start_date" min="<?= date('Y-m-d') ?>" required>

</div>


<div class="form-group">

<label for="notes" class="form-label">Special Requests (Optional)</label>

<textarea class="form-control" id="notes" name="notes" rows="3" placeholder="Any specific goals or focus areas?"></textarea>

</div>


<div class="btn-group">

<button type="submit" class="btn">Book Trainer</button>

<a href="dashboard.php" class="btn btn-outline">Cancel</a>

</div>

</form>

</div>


<h2>Available Trainers</h2>

<div class="trainer-preview">

<?php foreach($trainers as $trainer): ?>

<div class="trainer-card">

<div class="trainer-info">

<h3><?= htmlspecialchars($trainer['full_name']) ?></h3>

<span class="trainer-specialization"><?= htmlspecialchars($trainer['specialization']) ?></span>

<p><?= htmlspecialchars($trainer['bio']) ?></p>

<button type="button" class="select-trainer-btn" data-trainer-id="<?= $trainer['trainer_id'] ?>">

Select This Trainer

</button>
```

```
</div>
</div>
<?php endforeach; ?>
</div>
<?php endif; ?>

<!-- Successfully Booked Packages Section -->
<?php if(!empty($successfulBookings)): ?>
<div class="success-bookings">
<h2>Your Active Trainer Sessions</h2>
<?php foreach($successfulBookings as $booking): ?>
<div class="booking-card">
<div class="booking-header">
<div>
<div class="booking-trainer"><?= htmlspecialchars($booking['trainer_name']) ?></div>
<div class="booking-package"><?= htmlspecialchars($booking['package_name']) ?></div>
</div>
<span class="status-badge">Active</span>
</div>
<div class="booking-details">
<div class="booking-detail">
<span class="detail-label">Specialization</span>
<span class="detail-value"><?= htmlspecialchars($booking['specialization']) ?></span>
</div>
<div class="booking-detail">
<span class="detail-label">Start Date</span>
<span class="detail-value"><?= date('M j, Y', strtotime($booking['start_date'])) ?></span>
</div>
<div class="booking-detail">
```

```
<span class="detail-label">End Date</span>
<span class="detail-value"><?= date('M j, Y', strtotime($booking['end_date'])) ?></span>
</div>
<div class="booking-detail">
<span class="detail-label">Package Duration</span>
<span class="detail-value"><?= $booking['duration_months'] ?> months</span>
</div>
</div>
<?php if(!empty($booking['notes'])): ?>
<div style="margin-top: 1rem; padding-top: 1rem; border-top: 1px solid rgba(255,255,255,0.1);">
<span class="detail-label">Notes</span>
<p style="color: var(--text-light); margin-top: 0.5rem;"><?= htmlspecialchars($booking['notes']) ?></p>
</div>
<?php endif; ?>
</div>
<?php endforeach; ?>
</div>
<?php endif; ?>
</div>

<script>
// Function to set the selected trainer
document.querySelectorAll('.select-trainer-btn').forEach(button => {
button.addEventListener('click', function() {
const trainerId = this.getAttribute('data-trainer-id');
document.getElementById('trainer_id').value = trainerId;

// Scroll to form
```

```
document.getElementById('bookingForm').scrollIntoView({
behavior: 'smooth'
});
});
});

// Add form validation
document.getElementById('bookingForm').addEventListener('submit', function(e) {
const startDate = new Date(document.getElementById('start_date').value);
const today = new Date();
today.setHours(0, 0, 0, 0);

if (startDate < today) {
e.preventDefault();
alert('Please select a future date for your training start.');
}
});
</script>
</body>
</html>
```

Logout

```
<?php
// Start session if not already started
if (session_status() === PHP_SESSION_NONE) {
session_start();
}
```

```
// Unset all session variables
$_SESSION = [];

// Destroy the session
session_destroy();

// Clear session cookie
if (ini_get("session.use_cookies")) {
$params = session_get_cookie_params();
setcookie(
session_name(),
",
time() - 42000,
$params["path"],
$params["domain"],
$params["secure"],
$params["httponly"]
);
}

// Redirect to index page with absolute URL
header("Location: http://" . $_SERVER['HTTP_HOST'] . "/crossfit/index.php");
exit();
?>
```

5.2.2 CODE VALIDATION AND OPTIMIZATION

Validation is the process of evaluating software at the end of the software development to ensure compliance with software requirements. Testing is a common method for validation. Validation succeeds when the software functions in a manner that can be reasonably expected

by the customer. Form data validation comes in a couple different forms. Data can be validated at the field level when it is entered by the user, and it can be validated at the form level (i.e., all fields) when the form is submitted or printed. These types of validation have different, complimentary purposes and for a complete form design it is a good practice to use a combination of the two methods.

Field level validation

The purpose of Field Level Validation is to verify that the input to a single field is entered correctly. For example, for an e-mail field, the job of the validation script is to make sure the entered text matches the standard email format, i.e., two sets of strings separated by an "@" symbol and to check whether there is a dot(.) separating the domain name. The most common way to implement a text pattern test like this is to use a regular expression. Most of the time validation scripts are used to match input text against a pattern using a regular expression.

Form level validation

Form level validation is used to ensure all the required form data is filled in, and or to make sure that any data dependencies between fields are met before the form is submitted.

5.3 UNIT TESTING

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing. It is performed by using the White Box Testing method. Unit Testing is the first level of software testing and is performed prior to Integration Testing. It is normally performed by software developers themselves or their peers. In rare cases, it may also be performed by independent software testers.

Unit Testing Benefits:

• Unit testing increases confidence in changing/ maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.

• Codes are more reusable. In order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.

• Development is faster. Writing tests takes time but the time is compensated by the less amount of time it takes to run the tests. Unit tests are more reliable than 'developer tests'.

Development is faster in the long run too. The effort required to find and fix defects found during unit testing is very less in comparison to the effort required to fix defects found during system testing or acceptance testing.

• The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels. Compare the cost (time, effort, destruction, humiliation) of a defect detected during acceptance testing or when the software is live.

• Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.

• Codes are more reliable

SYSTEM TESTING

CHAPTER 6

SYSTEM TESTING

6.1 INTRODUCTION

Once the source code of the CrossFit Gym Management System was developed and verified, the next crucial step was system testing. The objective of testing is to detect and correct as many errors as possible before the system is delivered to the end users. Testing ensures that the software performs according to specifications and meets user requirements effectively.The purpose of software testing is to uncover logic errors, design defects, and deviations from expected functionality. It involves designing test cases that have a high probability of revealing faults within the system.Testing is carried out using two major techniques:

- White Box Testing – focuses on the internal logic and structure of the code.
- Black Box Testing – focuses on the external behavior of the system, ensuring that the software meets the specified requirements.

Both approaches complement each other in detecting defects and ensuring a reliable, high-quality system. The ultimate goal is to ensure that the CrossFit Gym Management System operates smoothly under all possible conditions and user inputs.

6.2 INTEGRATION TESTING

Integration testing is performed to verify that different modules and subsystems of the CrossFit Gym Management System work correctly together. Once individual modules like Admin, Trainer, Member, and Supplement management are tested independently, they are integrated to ensure smooth data flow and communication between them.The main objective is to test the interaction between various components through their interfaces and identify any inconsistencies in data transfer or control logic. Test cases are designed to simulate both valid and invalid scenarios, ensuring that the system behaves as expected.

Integration testing in this project ensures:

- Proper communication between Admin, Trainer, and Member modules.
- Correct data flow between booking and payment modules.
- Seamless connectivity with the MySQL database for CRUD operations.
- Reliable login and authentication process across all user roles.

6.2.1 BIG BANG INTEGRATION TESTING

In this approach, all modules of the system—Admin, Trainer, Member, Package, Supplement, and Messaging—are integrated and tested simultaneously. This helps to quickly identify issues that occur due to the interaction between different components.The big bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of Integration testing. A type of Big Bang

Integration testing is called Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes on optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user like workloads for creating realistic scenarios in exercising the environment. This gives confident that the integrated environment will work as expected for the target customers. While this method saves time, it also requires careful documentation of test cases to ensure all possible errors are tracked and fixed. The CrossFit Gym Management System underwent Big Bang integration testing after all modules were fully developed and internally verified. This testing confirmed that all functionalities, including user login, booking, and communication, worked harmoniously in the integrated environment.

6.2.1 TOP-DOWN AND BOTTOM-UP

In **Top-Down Testing**, the higher-level modules such as Admin and Trainer panels were tested first, followed by the integration of lower-level modules like Member and Supplement sections. Stubs were used to simulate the behavior of unimplemented lower modules during early testing.In **Bottom-Up Testing**, lower-level modules such as Member registration, Package booking, and Supplement management were tested first, and then higher modules like Trainer management and Admin dashboard were integrated. Drivers were used to test interactions during early stages.This combined approach ensured that both top-level logic and low-level operations worked correctly when integrated, reducing the chance of undetected errors.

6.3 SYSTEM TESTING

System Testing is the type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specification or in the context of both.System testing tests the design and behaviour of the system and the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirement specification (SRS). System Testing is basically performed by a testing team that is independent of the development team that helps to best the quality of the system impartial. It has both functional and non-functional testing.

System Testing is a Black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

6.3.1 TEST PLAN AND TEST CASES

A **test plan** defines the strategy and scope of testing. It outlines what needs to be tested, how testing will be carried out, and the expected results. The **test cases** were designed to evaluate the functionalities of Admin, Trainer, and Member roles separately to ensure that all modules perform accurately.

Test Case 1 Admin Module

Test Case ID	Input Specification	Input Data	Expected Result	Test Result
1	Run the application in browser	URL of the system	Application home screen displayed	Pass
2	Enter valid username and password	Admin credentials	Admin dashboard displayed	Pass
3	Enter invalid username or password	Invalid credentials	Error message displayed	Pass
4	Add new Trainer	Trainer details	Trainer added successfully	Pass
5	Update supplement stock	Supplement details	Stock updated successfully	Pass
6	View Member list	Click “View Members”	Member list displayed	Pass

Test Case 2 – Trainer Module

Test Case ID	Input Specification	Input Data	Expected Result	Test Result
1	Enter valid Trainer credentials	Username & Password	Trainer dashboard displayed	Pass
2	View assigned members	Trainer login	List of assigned members displayed	Pass
3	Send message to member	Select member and message	Message sent successfully	Pass
4	Update personal details	Edit profile section	Profile updated successfully	Pass
5	Check client progress	Member details	Progress report displayed	Pass

Test **Case 3 – Member Module**

Test Case ID	Input Specification	Input Data	Expected Result	Test Result
1	Register a new account	Name, email, password	Member account created	Pass
2	Log in with valid credentials	Member username & password	Member dashboard displayed	Pass
3	Book a package	Select package and submit	Booking confirmed	Pass
4	Buy supplement	Choose supplement and confirm	Purchase successful	Pass
5	Message trainer	Select trainer and send message	Message delivered	Pass
6	Log out	Click logout button	Redirects to login page	Pass

SYSTEM MAINTENANCE

CHAPTER 7

SYSTEM MAINTENANCE

7.1 INTRODUCTION

System Maintenance is an essential phase in the Software Development Life Cycle (SDLC) that ensures the system continues to function efficiently after deployment. It involves ongoing monitoring, debugging, and enhancement of the system to maintain its performance, reliability, and security over time.

In the CrossFit Gym Management System, maintenance plays a crucial role in ensuring that all modules — including user registration, trainer management, package booking, supplement stock updates, and communication between members and trainers — continue to operate smoothly. As user needs evolve and the system scales, continuous maintenance helps in adapting to these changes without affecting performance.

Maintenance also addresses issues discovered during real-world usage, ensures database integrity, and optimizes system performance. This guarantees that the system remains user-friendly, secure, and efficient throughout its operational life.

The maintenance phase typically involves the following types of activities:

- Corrective Maintenance – Fixing software bugs, functional errors, or UI issues identified after deployment.
- Adaptive Maintenance – Adjusting the system to function correctly in changing environments such as new operating systems, browsers, or hardware.
- Perfective Maintenance – Enhancing performance, user experience, and interface design based on user feedback.
- Preventive Maintenance – Implementing updates and code optimizations to prevent potential future issues.
- Regular maintenance ensures that the CrossFit Gym Management System remains stable, secure, and capable of handling an increasing number of members, trainers, and data transactions efficiently..

7.2 MAINTENANCE ACTIVITIES

After successful implementation and testing, the CrossFit Gym Management System entered the maintenance stage. This phase focused on ensuring system stability, managing feedback, and improving overall performance.The key maintenance activities performed are as follows:

- **Error Correction**

  Despite thorough testing, minor issues like form validation errors, login failures, or page redirects occasionally appeared during real usage. These were promptly identified and resolved to maintain system integrity and functionality.

- **Database Maintenance**

Periodic reviews were conducted to ensure data accuracy and efficiency in the MySQL database. Regular backups were scheduled to prevent data loss, and indexing techniques were used to enhance query performance.

- **User Feedback Handling**

Feedback from admins, trainers, and members was collected to identify areas for improvement. Based on this input, UI enhancements, faster navigation paths, and improved accessibility features were implemented.

- **Security Updates**

Security was a major focus during maintenance. Measures such as password hashing, input validation, and prevention of SQL injection and XSS attacks were applied to protect user data and system operations.

- **Performance Optimization**

The system's speed and responsiveness were improved by optimizing SQL queries, compressing images, and implementing efficient caching mechanisms. This ensured a smoother user experience even during peak usage hours.

- **Scalability Enhancements**

The architecture of the system was designed to allow future modules — such as online trainer consultation, digital payment gateways, and AI-driven fitness tracking — to be added without disrupting existing functionalities.

- **Regular Monitoring**

Continuous monitoring was performed to ensure all features — such as login authentication, package booking, and supplement management — operated correctly. Any irregularities were logged and resolved proactively.

Maintenance Tools Used

- PHPMyAdmin – For managing and optimizing the MySQL database.
- Error Logs – To track runtime and backend issues.
- Browser Developer Console – To detect and resolve JavaScript or UI-related errors.
- Git Version Control – To manage code updates and maintain project history.
- XAMPP Server Logs – For monitoring server-side operations and performance.
- Benefits of Maintenance

- Improved system performance and response time.
- Enhanced security and data integrity.
- Increased user satisfaction through better usability.
- Reduced downtime and smoother operation.
- Ensured scalability for future enhancements.
- Long-term system sustainability and reliability.

Future Maintenance Plans

The CrossFit Gym Management System is designed with flexibility and long-term maintenance in mind. Future updates will focus on enhancing both functionality and user engagement. Planned maintenance tasks include:

Integration of online payment systems for package and supplement purchases.

Development of a mobile-friendly or app-based version of the system.Incorporation of AI-based workout tracking and progress analytics.Expansion of database capabilities to support more trainers, members, and supplement categories.Implementing automated email and SMS notifications for bookings and trainer communications..

FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

CHAPTER 8

FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

8.1 MERITS OF THE SYSTEM

The *CrossFit Gym Management System* provides an efficient and user-friendly solution for managing gym operations, improving communication, and enhancing member engagement. The system integrates multiple roles—Admin, Trainer, and Member—into a single platform, ensuring smooth coordination and effective management. The following are the key merits of the system:

- **User Convenience:**
- Members can easily book packages, trainers, and supplements online, eliminating the need for manual registration and payment queues.
- **Centralized Management:**
- The Admin can efficiently manage trainers, members, and supplement stock through a unified dashboard, ensuring operational transparency.
- **Enhanced Communication:**
- The system allows real-time messaging between trainers and members, promoting effective communication and better workout guidance.
- **Time Efficiency:**
- Automated record management and online transactions save time for both staff and members, leading to faster service delivery.
- **Improved Data Accuracy:**
- With automated data storage in MySQL, errors due to manual entry are minimized, ensuring reliable and accurate records.
- **Scalability:**
  The system is designed to accommodate future growth, such as an increase in the number of trainers, members, and new gym branches.

8.2 DEMERITS OF THE SYSTEM

While the *CrossFit Gym Management System* offers numerous benefits, certain limitations exist that can be addressed in future versions.

- **Dependence on Internet Connectivity:** As the system is web-based, uninterrupted internet access is necessary for smooth operation.Always dependent on internet connectivity and interrupted connection cause problems duing the operation of the project.

- **Technical Familiarity:** Some users may find it challenging to navigate the digital platform if they are not familiar with online systems.
- **Data Security Risks:** Since the system stores sensitive user information such as contact details and membership history, strong cybersecurity measures must be maintained.
- **Limited Personalization:** The current system does not include AI-based recommendations or personalized fitness tracking features.
- **Maintenance Effort:** Continuous updates and database backups are required to maintain performance and ensure security.

8.3 FUTURE ENHANCEMENTS OF THE SYSTEM

The *CrossFit Gym Management System* has a strong foundation for future expansion. Several features can be integrated to enhance its functionality, user experience, and technological adaptability.

- **AI-Powered Fitness Recommendations:** Artificial Intelligence can be implemented to provide customized workout and diet plans based on user goals and progress tracking.
- **Mobile Application Integration:** Developing a mobile app version will increase accessibility, allowing members to book sessions and track progress on the go.
- **Online Payment Gateway:** Integration of secure online payment systems such as UPI, Paytm, or credit/debit cards can make transactions more seamless.
- **Wearable Device Integration:** Connecting fitness trackers and smartwatches can help in monitoring real-time workout data such as heart rate, calories burned, and performance analytics.
- **Virtual Training Sessions:** Incorporating video conferencing features will enable trainers to conduct online sessions, expanding the gym's reach beyond physical boundaries.
- **Feedback Analytics Dashboard:** Adding automated feedback analysis will help the admin and trainers understand member satisfaction and improve services accordingly.
- **Notification and Reminder System:** Automatic alerts for upcoming bookings, subscription renewals, or supplement restocks can improve engagement and user experience.
- **Multi-Language Support:** Implementing multiple language options can make the platform more inclusive for users from diverse regions.
- **Data Analytics for Performance Insights:** Analytical reports can be added to provide insights into trainer performance, attendance trends, and business growth metrics.

CONCLUTION

CHAPTER 9

CONCLUSION

The *CrossFit Gym Management System* has been developed to simplify and automate the various administrative, operational, and communication processes within a fitness center. This project addresses the challenges faced by gym administrators, trainers, and members by providing a unified digital platform that ensures efficiency, transparency, and better user engagement.The system successfully integrates multiple modules—such as **user management, trainer allocation, package booking, supplement inventory management, and member-trainer communication**—to create a complete solution for modern gym operations. By computerizing these processes, it reduces manual effort, minimizes errors, and enhances overall service delivery.From a user's perspective, the system provides convenience and accessibility, enabling members to register, book packages, and communicate with trainers from anywhere through an interactive interface. For administrators, it offers effective tools to manage resources, update supplement stock, track bookings, and monitor staff activities—all from a single dashboard. Trainers also benefit by managing their clients efficiently and providing personalized guidance through messaging features.This project demonstrates the potential of technology in transforming traditional gym management into a streamlined, data-driven process. It lays the foundation for improved coordination between all users, ensures better utilization of gym resources, and promotes a more engaging fitness experience for members.

In the future, the system can be further enhanced by incorporating **AI-based recommendations, data analytics, and mobile app integration** to improve scalability and personalization. With such advancements, the *CrossFit Gym Management System* can evolve into a comprehensive fitness ecosystem that supports long-term health goals and strengthens the relationship between gyms and their members.Ultimately, the project not only fulfills the objectives of digital gym management but also emphasizes the growing importance of **automation and technology in the fitness industry**, paving the way for smarter, more efficient, and user-friendly management solutions.

BIBLIOGRAPHY

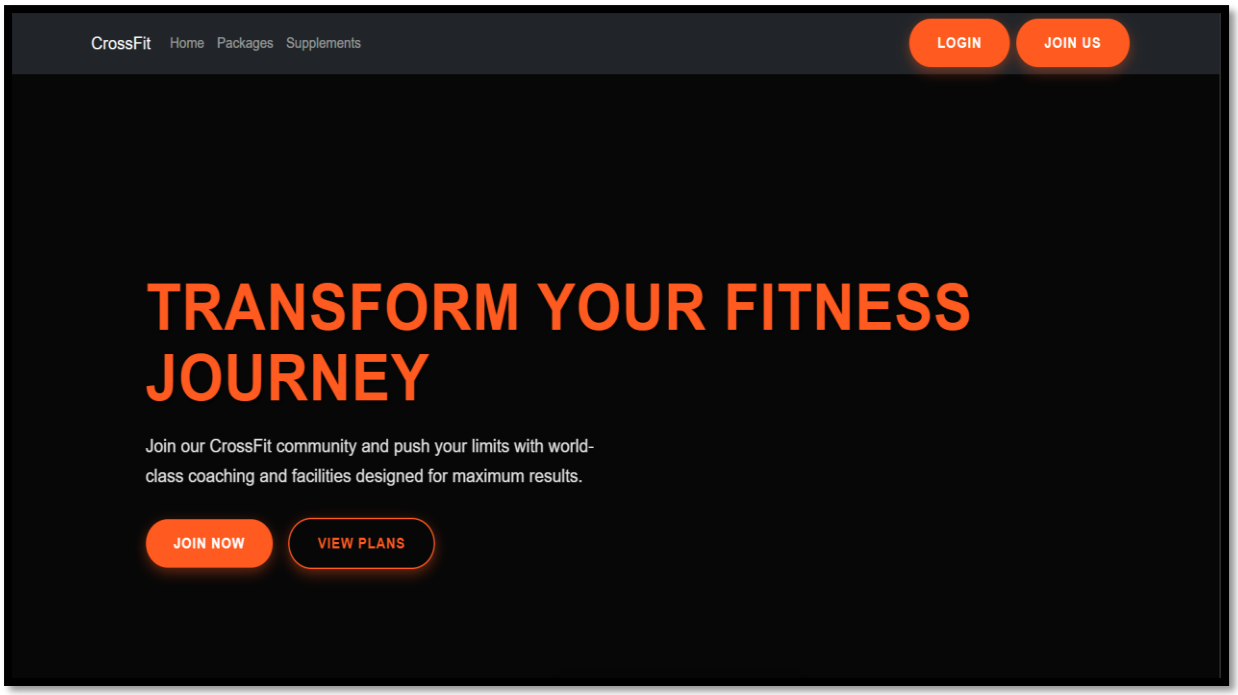
CHAPTER 10

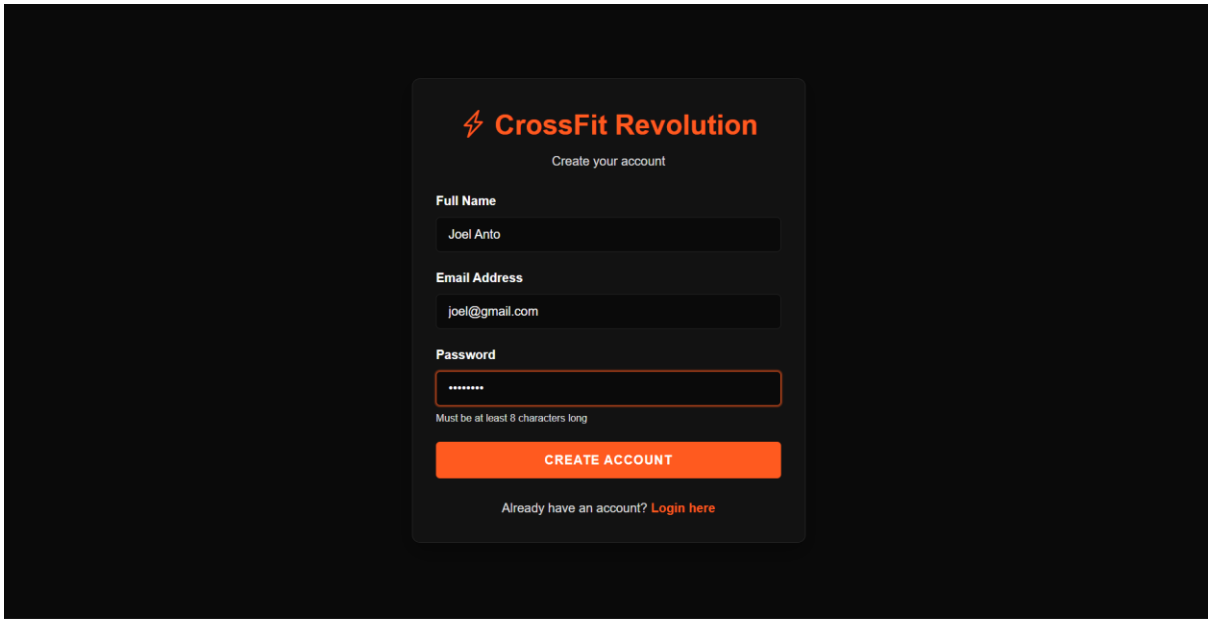
BIBILOGRAPHY

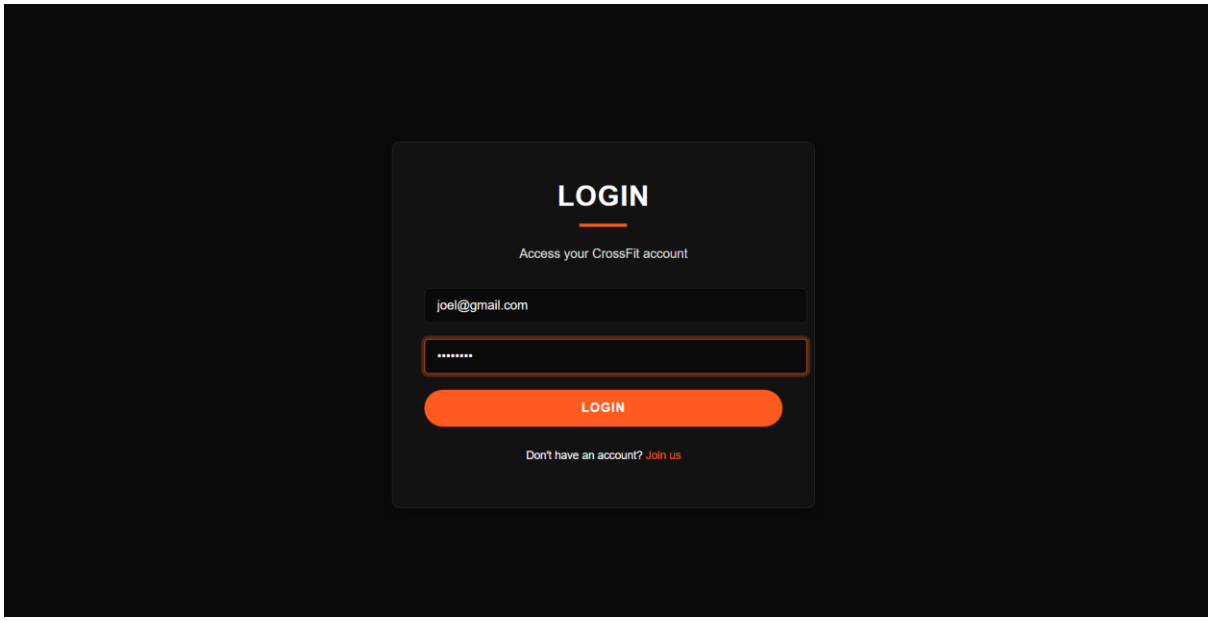
Web References:

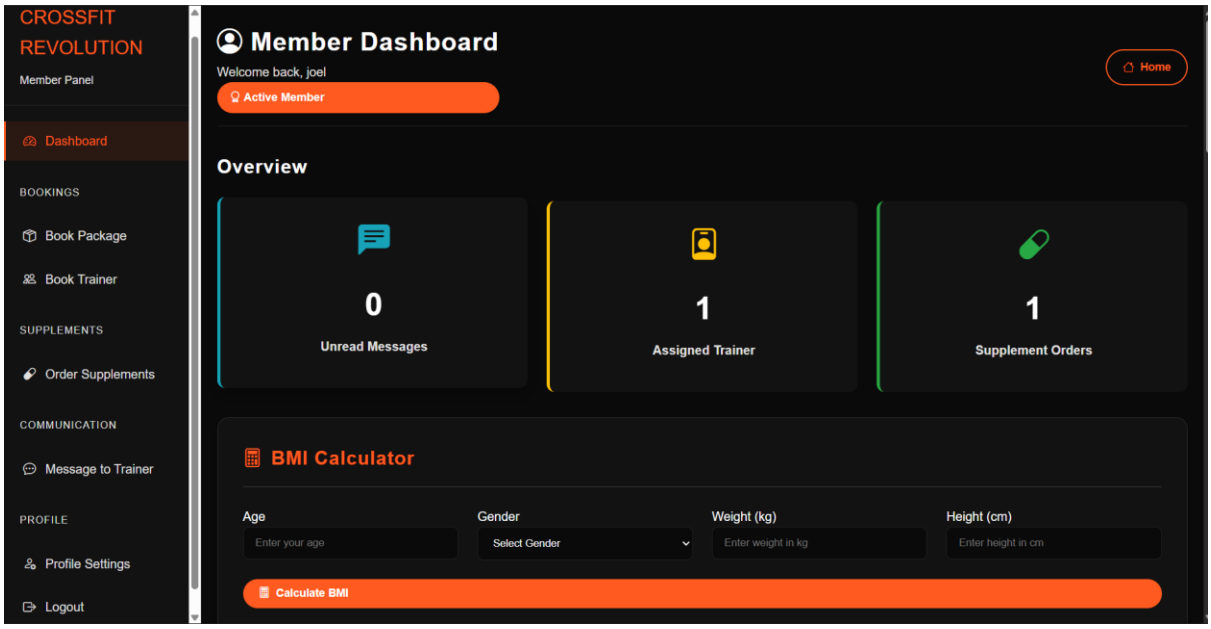
- https://www.php.net – Official PHP Documentation.
- https://www.mysql.com – MySQL Database Documentation and Tutorials.
- https://www.w3schools.com – Tutorials and examples for web technologies including HTML, CSS, JavaScript, and PHP.
- https://openrouter.ai – Official documentation for Open Router API integration for AI based chat features.
- https://mistral.ai – Mistral AI Model documentation and usage guidelines.
- https://www.geeksforgeeks.org – Reference for PHP, MySQL connectivity, and real time message implementation examples.
- https://www.stackoverflow.com – Community-based solutions for common web development errors and optimizations.
- https://leafletjs.com – Documentation for integrating map and location features in web applications.
- https://chat.openai.com – For conceptual and syntax references in building AI and NLP driven systems.
- https://chat.deepseek.com - For conceptual and syntax references

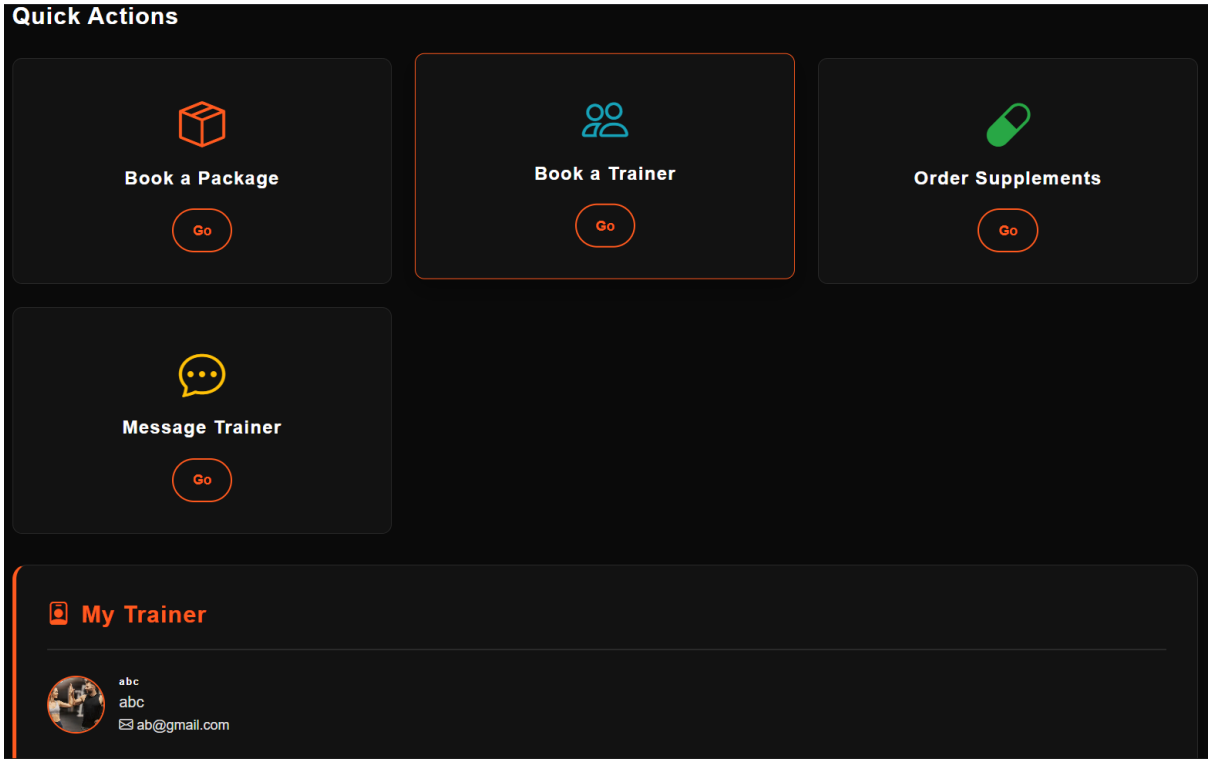
APPENDIX

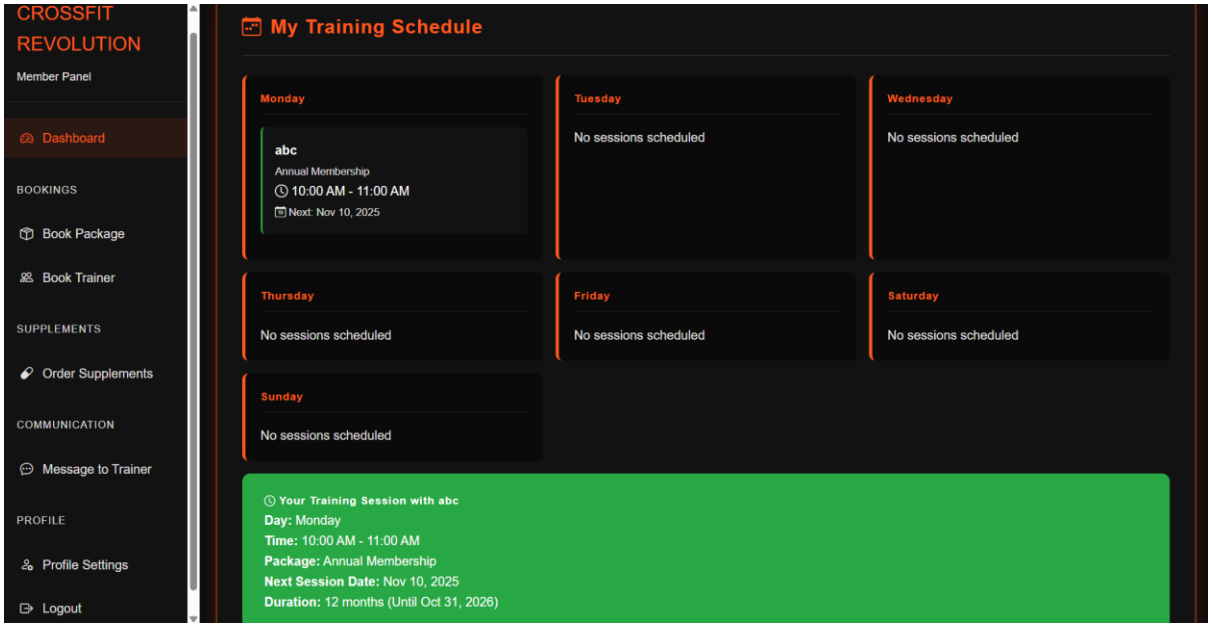

CrossFit
Home
Packages
Supplements
LOGIN
JOIN US
TRANSFORM YOUR FITNESS JOURNEY
Join our CrossFit community and push your limits with world-class coaching and facilities designed for maximum results.
JOIN NOW
VIEW PLANS


CrossFit Revolution
Create your account
Full Name
Joel Anto
Email Address
joel@gmail.com
Password
Must be at least 8 characters long
CREATE ACCOUNT
Already have an account? Login here


LOGIN
Access your CrossFit account
joel@gmail.com
LOGIN
Don't have an account? Join us


CROSSFIT REVOLUTION
Member Panel
Dashboard
BOOKINGS
Book Package
Book Trainer
SUPPLEMENTS
Order Supplements
COMMUNICATION
Message to Trainer
PROFILE
Profile Settings
Logout
Member Dashboard
Welcome back, joel
Active Member
Home
Overview
0
Unread Messages
1
Assigned Trainer
1
Supplement Orders
BMI Calculator
Age
Enter your age
Gender
Select Gender
Weight (kg)
Enter weight in kg
Height (cm)
Enter height in cm
Calculate BMI


Quick Actions
Book a Package
Go
Book a Trainer
Go
Order Supplements
Go
Message Trainer
Go
My Trainer
abc
abc
ab@gmail.com


CROSSFIT REVOLUTION
Member Panel
Dashboard
BOOKINGS
Book Package
Book Trainer
SUPPLEMENTS
Order Supplements
COMMUNICATION
Message to Trainer
PROFILE
Profile Settings
Logout
My Training Schedule
Monday
abc
Annual Membership
10:00 AM - 11:00 AM
Next: Nov 10, 2025
Tuesday
No sessions scheduled
Wednesday
No sessions scheduled
Thursday
No sessions scheduled
Friday
No sessions scheduled
Saturday
No sessions scheduled
Sunday
No sessions scheduled
Your Training Session with abc
Day: Monday
Time: 10:00 AM - 11:00 AM
Package: Annual Membership
Next Session Date: Nov 10, 2025
Duration: 12 months (Until Oct 31, 2026)

Active Memberships

Package Memberships

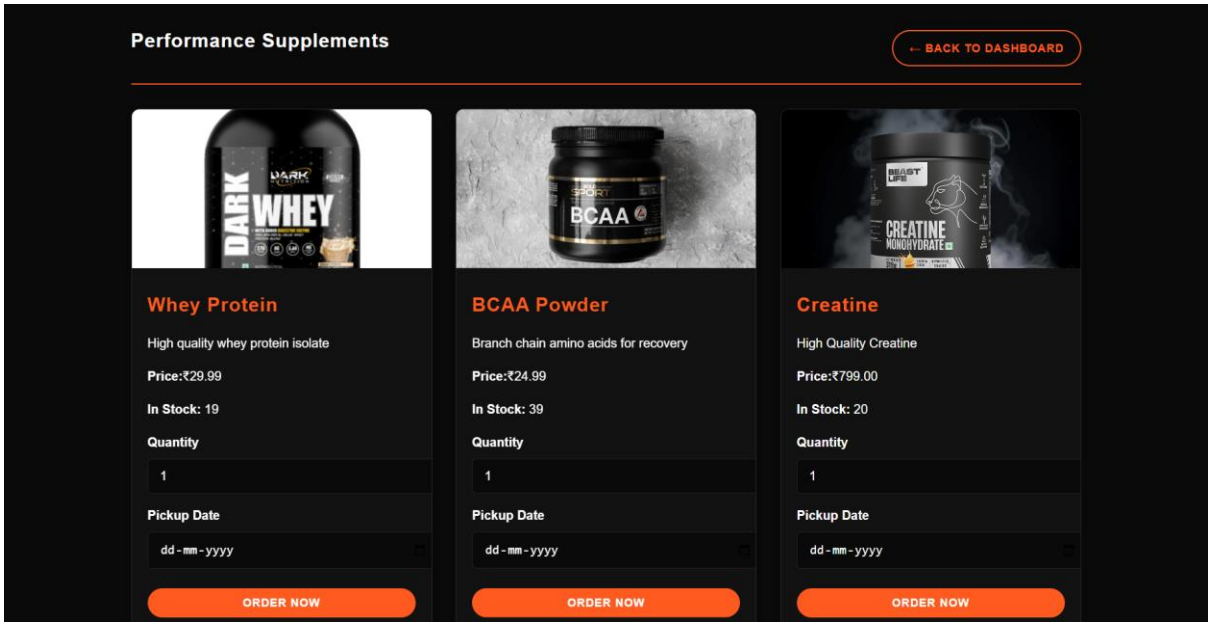
Package	Duration	Price	Start Date	End Date
Annual Membership	12 months	₹500.00	Oct 31, 2025	Oct 31, 2026

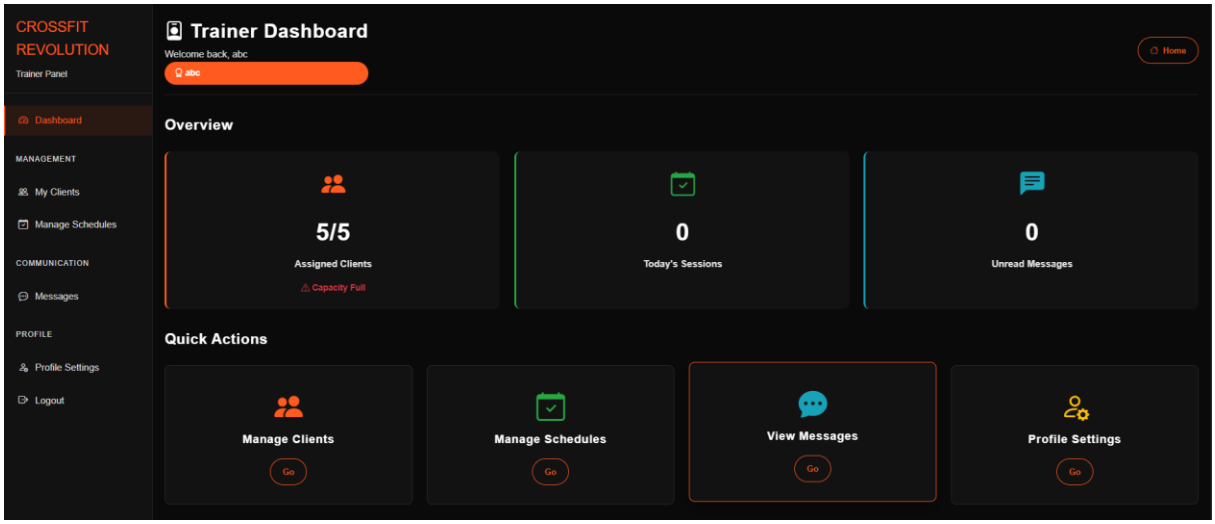
Trainer Sessions

Package	Duration	Price	Start Date	End Date	Trainer
Annual Membership	12 months	₹500.00	Oct 31, 2025	Oct 31, 2026	abc abc

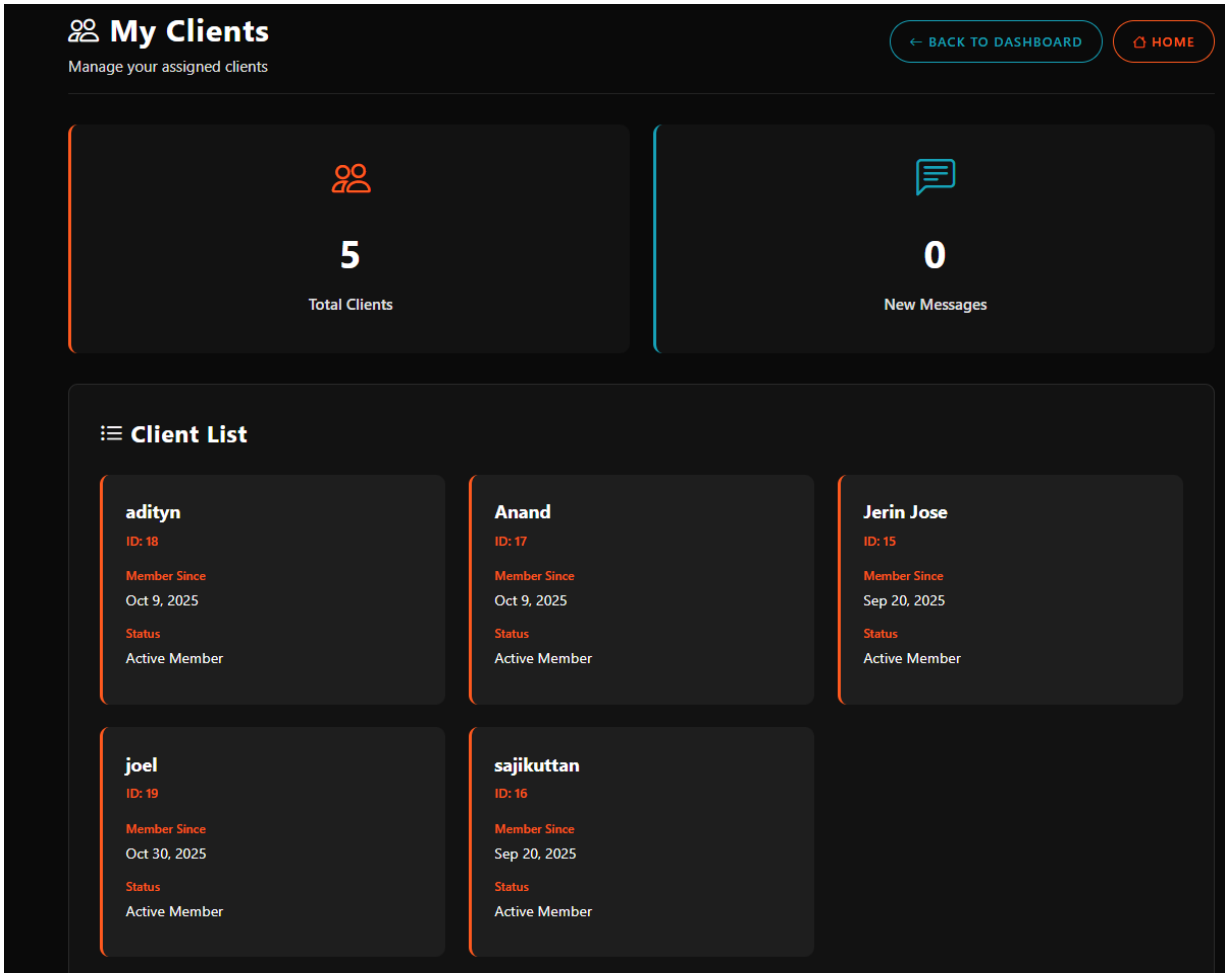
Recent Supplement Orders

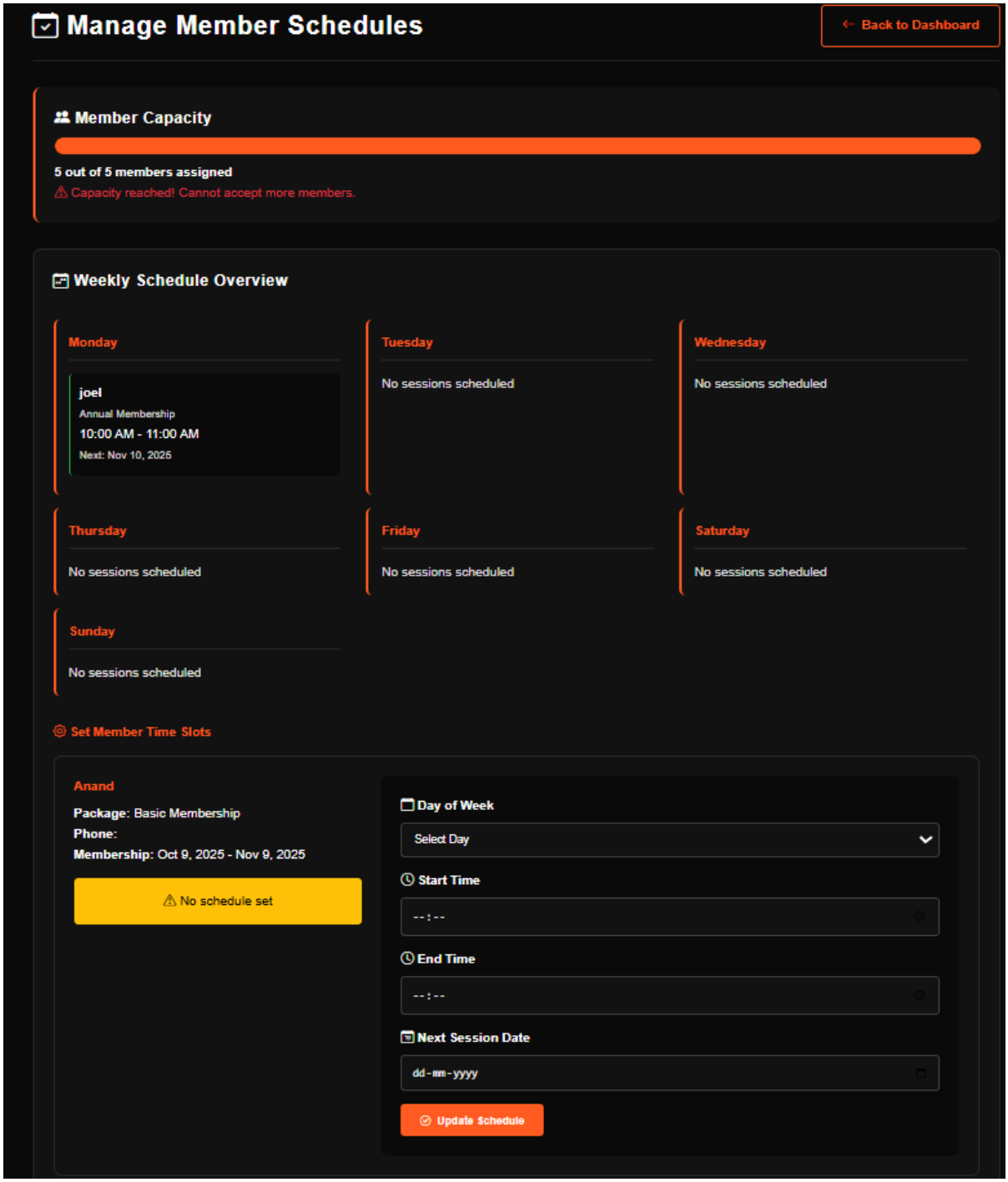
Supplement	Quantity	Price	Total	Order Date	Pickup Date
Whey Protein	1	₹29.99	₹29.99	Oct 30, 2025	Oct 31, 2025

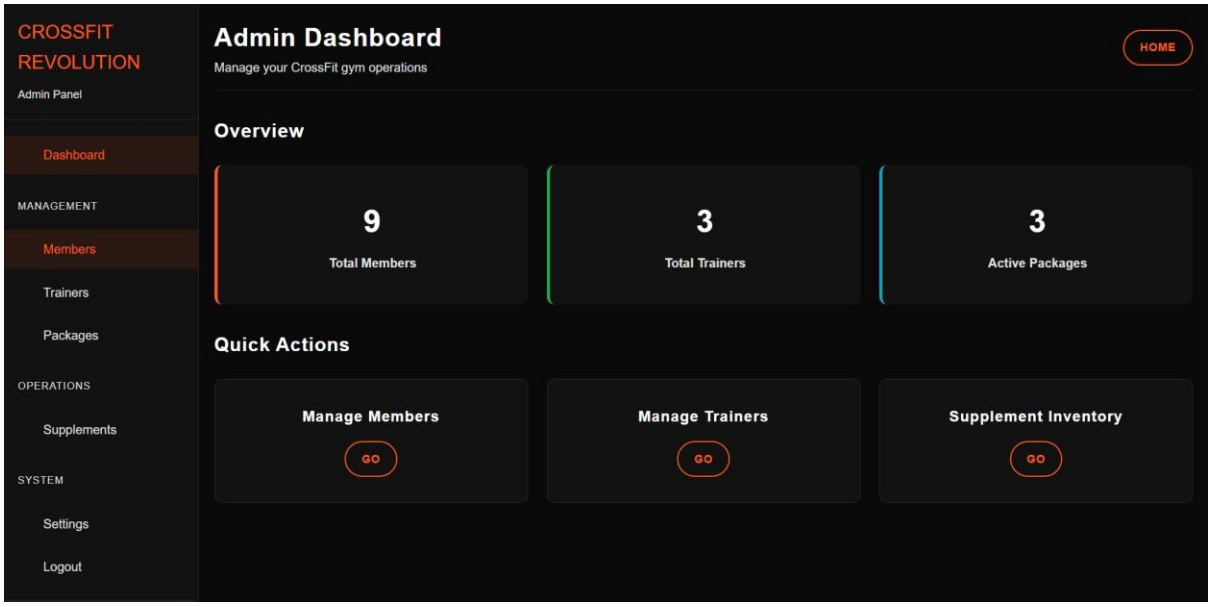


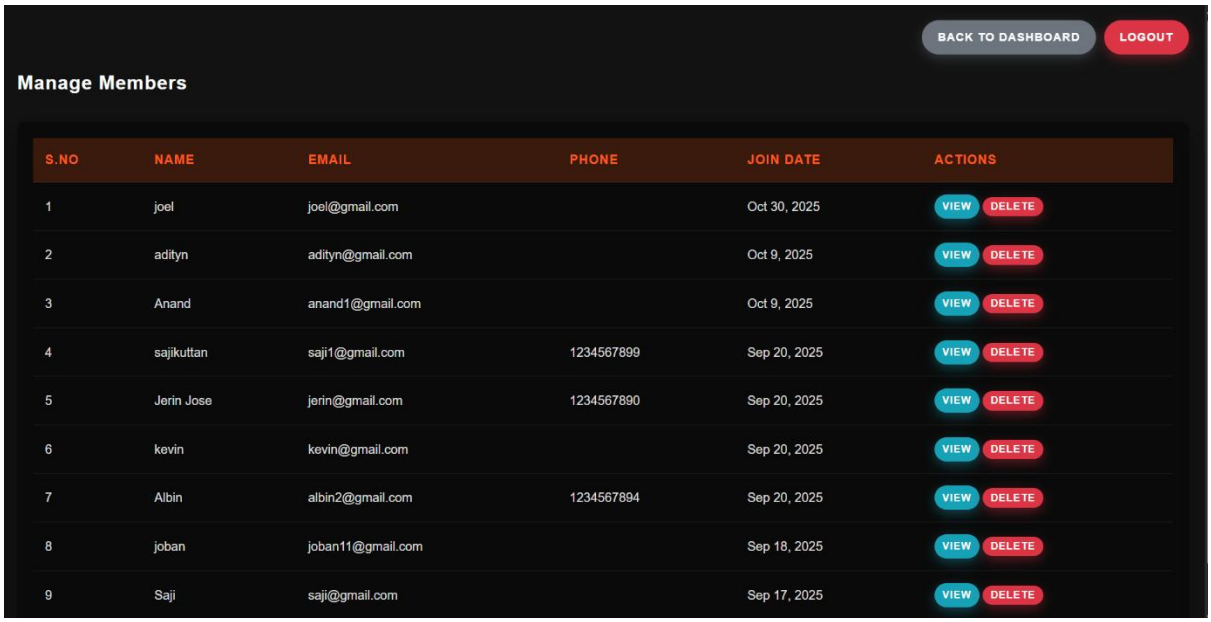

CROSSFIT REVOLUTION
Trainer Panel
Dashboard
MANAGEMENT
My Clients
Manage Schedules
COMMUNICATION
Messages
PROFILE
Profile Settings
Logout
Trainer Dashboard
Welcome back, abc
abc
Home
Overview
5/5
Assigned Clients
Capacity Full
0
Today's Sessions
0
Unread Messages
Quick Actions
Manage Clients
Go
Manage Schedules
Go
View Messages
Go
Profile Settings
Go

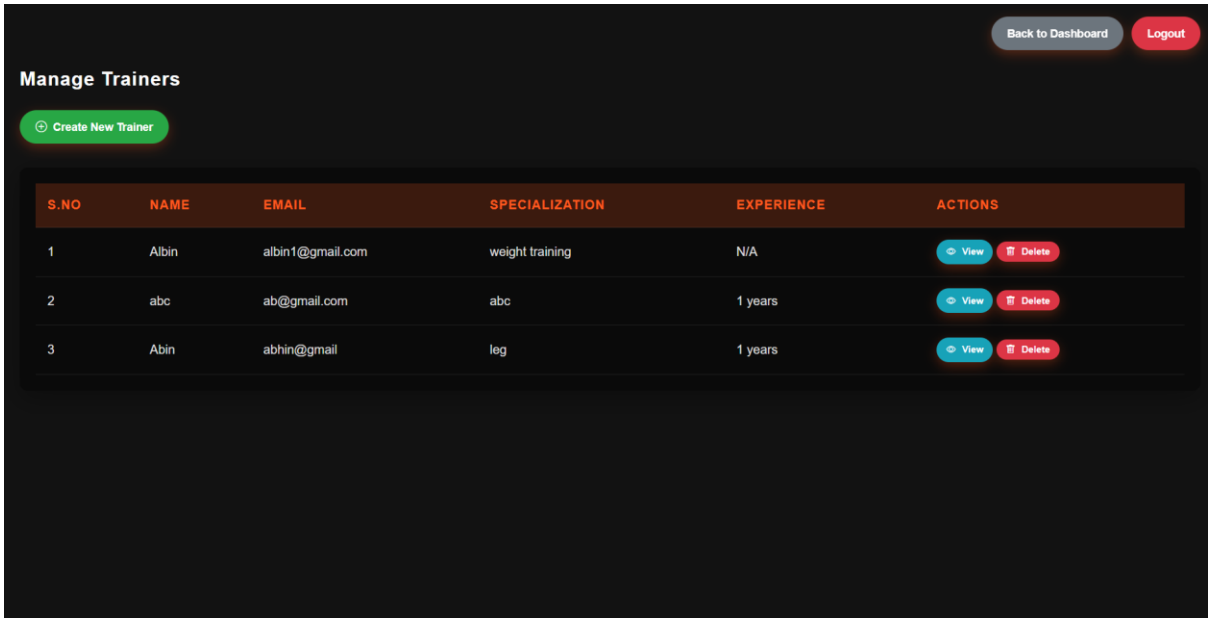

Quick Messaging
Recent Messages
Oct 30, 2025 9:23 PM
sajikuttan
im fine
Oct 9, 2025 11:33 AM
sajikuttan
hi
Oct 9, 2025 11:02 AM
Send New Message
Select Client
Choose a client...
Your Message
Type your message here...
Send Message

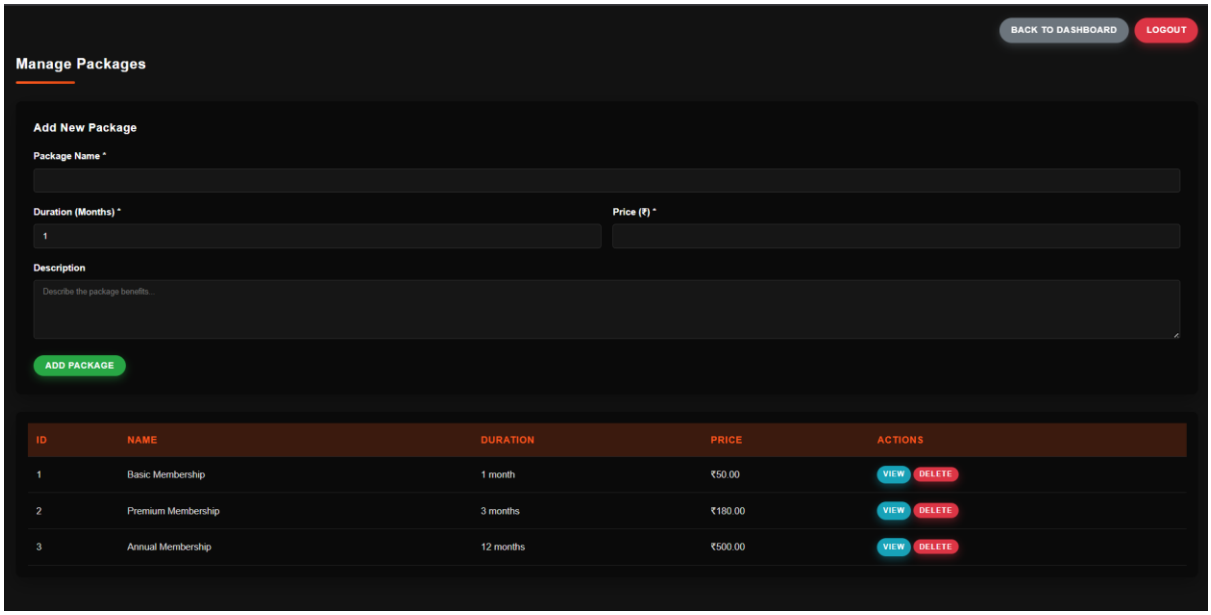

My Clients
Manage your assigned clients
← BACK TO DASHBOARD
HOME
5
Total Clients
0
New Messages
Client List
adityn
ID: 18
Member Since
Oct 9, 2025
Status
Active Member
Anand
ID: 17
Member Since
Oct 9, 2025
Status
Active Member
Jerin Jose
ID: 15
Member Since
Sep 20, 2025
Status
Active Member
joel
ID: 19
Member Since
Oct 30, 2025
Status
Active Member
sajikuttan
ID: 16
Member Since
Sep 20, 2025
Status
Active Member

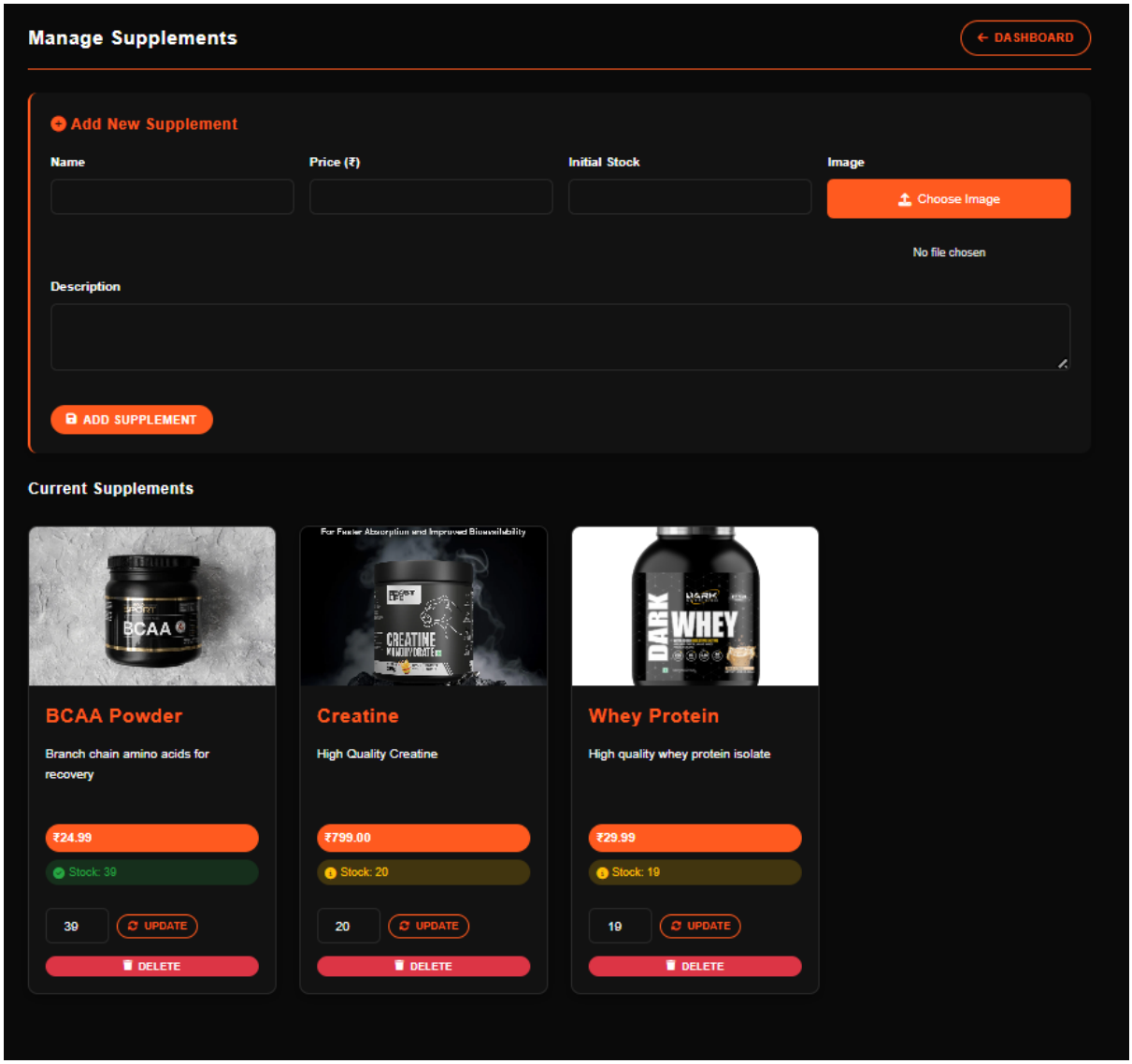

Manage Member Schedules
← Back to Dashboard
Member Capacity
5 out of 5 members assigned
⚠ Capacity reached! Cannot accept more members.
Weekly Schedule Overview
Monday
joel
Annual Membership
10:00 AM - 11:00 AM
Next: Nov 10, 2025
Tuesday
No sessions scheduled
Wednesday
No sessions scheduled
Thursday
No sessions scheduled
Friday
No sessions scheduled
Saturday
No sessions scheduled
Sunday
No sessions scheduled
Set Member Time Slots
Anand
Package: Basic Membership
Phone:
Membership: Oct 9, 2025 - Nov 9, 2025
⚠ No schedule set
Day of Week
Select Day
Start Time
--:--
End Time
--:--
Next Session Date
dd-mm-yyyy
Update Schedule


CROSSFIT REVOLUTION
Admin Panel
Dashboard
MANAGEMENT
Members
Trainers
Packages
OPERATIONS
Supplements
SYSTEM
Settings
Logout
Admin Dashboard
Manage your CrossFit gym operations
HOME
Overview
9
Total Members
3
Total Trainers
3
Active Packages
Quick Actions
Manage Members
GO
Manage Trainers
GO
Supplement Inventory
GO


BACK TO DASHBOARD
LOGOUT
Manage Members
S.NO NAME EMAIL PHONE JOIN DATE ACTIONS
1 joel joel@gmail.com Oct 30, 2025 VIEW DELETE
2 adityn adityn@gmail.com Oct 9, 2025 VIEW DELETE
3 Anand anand1@gmail.com Oct 9, 2025 VIEW DELETE
4 sajikuttan saji1@gmail.com 1234567899 Sep 20, 2025 VIEW DELETE
5 Jerin Jose jerin@gmail.com 1234567890 Sep 20, 2025 VIEW DELETE
6 kevin kevin@gmail.com Sep 20, 2025 VIEW DELETE
7 Albin albin2@gmail.com 1234567894 Sep 20, 2025 VIEW DELETE
8 joban joban11@gmail.com Sep 18, 2025 VIEW DELETE
9 Saji saji@gmail.com Sep 17, 2025 VIEW DELETE


Back to Dashboard
Logout
Manage Trainers
Create New Trainer
S.NO
NAME
EMAIL
SPECIALIZATION
EXPERIENCE
ACTIONS
1
Albin
albin1@gmail.com
weight training
N/A
View
Delete
2
abc
ab@gmail.com
abc
1 years
View
Delete
3
Abin
abhin@gmail
leg
1 years
View
Delete


BACK TO DASHBOARD
LOGOUT
Manage Packages
Add New Package
Package Name *
Duration (Months) *
1
Price (₹) *
Description
Describe the package benefits...
ADD PACKAGE
ID
NAME
DURATION
PRICE
ACTIONS
1
Basic Membership
1 month
₹50.00
VIEW
DELETE
2
Premium Membership
3 months
₹180.00
VIEW
DELETE
3
Annual Membership
12 months
₹500.00
VIEW
DELETE


Manage Supplements
← DASHBOARD
Add New Supplement
Name
Price (₹)
Initial Stock
Image
Choose Image
No file chosen
Description
ADD SUPPLEMENT
Current Supplements
BCAA
BCAA Powder
Branch chain amino acids for recovery
₹24.99
Stock: 39
39
UPDATE
DELETE
For Faster Absorption and Improved Bioavailability
CREATINE
Creatine
High Quality Creatine
₹799.00
Stock: 20
20
UPDATE
DELETE
DARK
WHEY
Whey Protein
High quality whey protein isolate
₹29.99
Stock: 19
19
UPDATE
DELETE