

UNIVERSITY OF MUMBAI

PROJECT ENTITLED

“CrossFit Online Gym Website”

SUBMITTED BY

Aman PavanKumar Mishra

UNDER GUIDANCE OF

Miss. Simran Shinde

SUBMITTED FOR THE FULFILLMENT OF THE CURRICULUM OF DEGREE
OF BACHELOR OF SCIENCE
IN COMPUTER SCIENCE



**PILLAI COLLEGE OF ARTS, COMMERCE &
SCIENCE (AUTONOMOUS), NEW PANVEL 2023-2024**

**MAHATMA EDUCATION SOCIETY'S
PILLAI COLLEGE OF ARTS COMMERCE & SCIENCE
(AUTONOMOUS)**

(Affiliated to University of Mumbai)

NEW PANVEL -410206 MAHARASHTRA

DEPARTMENT OF COMPUTER SCIENCE



CERTIFICATE

This is to certify that the project entitled “**CrossFit Online Gym Website**”, is bonafied work of **Mr Aman PavanKumar Mishra** bearing Seat No: **8865** submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in COMPUTER SCIENCE from University of Mumbai.

Date:

External Examiner

Internal Guide

Coordinator/HOD

College Seal

ACKNOWLEDGEMENT

Apart from the efforts of myself, the success of the project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I would like to express a deep sense of gratitude to my Project Guide Professor Simran Shinde for her cordial support and for her guidance and supervision as well as for providing necessary information regarding the project and also for the support in completing the project.

My thanks and appreciation also goes to my College Computer Laboratory Administrators for their support for making the right resources available in time for my project.

Finally, I also extend my heartiest thanks to my parents, friends and wellwishers for being with me and extending encouragement throughout the project.

Yours faithfully,

Aman PavanKumar Mishra

(Final Year Computer Science)

Table Of Content

SR No.	Content	Page No.	Signature.
1	Chapter - 1 (Overview)		
	1.1 Introduction		
	1.2 Background		
	1.3 Objective		
	1.4 Purpose		
	1.5 Scope & Limitations		
	1.6 Applicability		
2	Chapter - 2 (Feasibility Study and Stakeholders)		
	2.1 Feasibility Study		
	1. Technical Feasibility		
	2. Economic Feasibility		
	3. Financial Feasibility		
	4. Operational Feasibility		
	2.2 Stakeholders		
3	Chapter - 3 (Gantt Chart)		
4	Chapter – 4 (Software Requirement Specification)		
	4.1 Software and Hardware Requirements		
	4.2 Functional Requirements		
	4.3 Non-Functional Requirements		
	4.4 Conceptual Models(Data Flow Diagrams)		
	1. E-R Diagram		
	2. Flowchart Diagram		
	3. Class Diagram		
	4. Use-case Diagram		
	5. Activity Diagram		
	6. Package Diagram		
	7. Deployment Diagram		

5	Chapter – 5 (Software Development Life Cycle Phases and Models)		
	5.1 What is SDLC?		
	5.2 SDLC Models		
	- Agile Model		
	5.3 SDLC Phases		
6	Chapter – 6 (Designing)		
	6.1 System Design		
	6.2 Data Design		
	6.3 Data Integrity and Constraints		
	6.4 User Interface and Design		
7	Chapter – 7 (Coding/Implementation)		
	7.1 Coding		
8	Chapter – 8 (Testing)		
	8.1 Test Cases		
	8.2 Testing Outputs		
	8.3 Testing Approach		
	8.4 Unit testing		
	8.5 Integrated Testing		
	8.6 Security Issues		
9	Chapter – 9 (Maintenance)		
	9.1 Modifications and Improvements		
10	Chapter – 10 (Conclusion)		
	10.1 Conclusion And Future Enhancement		
11	Chapter – 11 (References)		

CHAPTER – 1 (OVERVIEW)

1.1 INTRODUCTION

The CrossFit Gym Management System is a comprehensive and user-friendly platform designed to streamline the operations of a fitness center. Developed using JSP, HTML, CSS, Java, and MySQL, this system is tailored to meet the specific needs of gym administrators and members alike. The system is divided into two main modules: Admin and User (Member).

Advantages of the CrossFit Gym Management System:

Efficient Administration:

The Admin module provides a secure login for gym administrators to manage various aspects of the gym, including trainers, members, receptionists, and workers.

Admins can easily add, update, delete, and view information related to trainers, members, receptionists, and workers, ensuring efficient management of staff and resources.

Member Self-Service:

The User (Member) module empowers gym members to take control of their information. Members can log in to add and view their own details, making the system interactive and user-centric.

Enquiry Tracking:

Admins can keep track of gym enquiries, enabling them to follow up and convert potential leads into members. This feature enhances customer relationship management and contributes to business growth.

Database Integration:

The system utilizes MySQL as a backend database, ensuring data integrity, security, and scalability. This database-driven approach allows for efficient storage and retrieval of information.

User-Friendly Interface:

The system boasts a user-friendly interface with a combination of HTML, CSS, and JSP, providing a visually appealing and intuitive experience for both administrators and members.

Centralized Information:

All essential information, such as member details, trainer profiles, and receptionist data, is centralized within the system. This centralized approach simplifies data management and reduces the likelihood of errors.

Scalability and Customization:

The system is designed to be scalable, allowing the addition of new features or modules as the gym expands. Additionally, it can be customized to meet the unique requirements of the CrossFit gym.

1.2 BACKGROUND

In the fast-paced fitness industry, the need for an integrated and efficient gym management system has become increasingly paramount. The inception of the CrossFit Gym Management System was rooted in the recognition of challenges faced by gym administrators in balancing the dynamic elements of member engagement, staff coordination, and overall operational efficiency.

Traditionally, managing a fitness center involved manual tracking, disparate systems, and the potential for data discrepancies. Recognizing these hurdles, the idea of a comprehensive and technologically advanced solution took shape. The amalgamation of JSP, HTML, CSS, Java, and MySQL technologies paved the way for a robust and flexible platform capable of meeting the diverse needs of both administrators and gym members.

The background of this system encapsulates the evolution of fitness management practices, mirroring the industry's demand for streamlined processes and enhanced member experiences. The focus on two core modules, Admin and User (Member), was a strategic decision to create a holistic solution that addresses the intricate balance between administrative control and member self-service.

The incorporation of MySQL as the backend database was a deliberate choice, ensuring a secure, scalable, and organized storage solution for the wealth of information generated within a gym environment. This database-centric approach aligns with contemporary data management standards, fostering data integrity and accessibility.

The user-friendly interface, achieved through HTML, CSS, and JSP, reflects a commitment to providing an engaging and intuitive experience for all users. The background development process considered not only the current requirements of a CrossFit gym but also anticipated future scalability and customization needs to accommodate the evolving landscape of fitness management.

1.3 OBJECTIVE

The primary objective of the CrossFit Gym Management System is to revolutionize and optimize the operational landscape of fitness centers, specifically tailored for CrossFit gyms. The system aims to empower administrators with a comprehensive suite of tools for seamless management of trainers, members, receptionists, and workers. By providing a user-friendly and intuitive interface, the system enhances administrative efficiency, allowing for the effortless addition, updating, and deletion of crucial data. Simultaneously, the Member module is designed to empower gym enthusiasts by offering a self-service platform where they can autonomously add and view their information. The integration of MySQL as the backend database ensures data integrity, security, and scalability, laying the foundation for efficient information storage and retrieval. The system's objective is to foster member engagement, improve administrative processes, and contribute to the overall growth and success of CrossFit gyms. Through scalability and customization, the system anticipates and adapts to the evolving needs of the fitness industry, positioning itself as a forward-looking and indispensable tool for gym management.

1.4 PURPOSE

The CrossFit Gym Management System serves as a pivotal solution with the overarching purpose of revolutionizing the way fitness centers operate and engage with their members. At its core, the system aims to streamline and enhance the administrative processes within a CrossFit gym, empowering administrators with efficient tools to manage trainers, members, receptionists, and workers seamlessly. Simultaneously, the purpose extends to member empowerment, allowing individuals to take an active role in managing their own information through a user-friendly interface. The integration of MySQL as the backend database aligns with the purpose of establishing a secure and scalable data management system, ensuring the integrity and accessibility of critical gym-related information. This system is purposefully designed not only to meet the current needs of gym management but also to anticipate and accommodate future growth and customization, reflecting a commitment to long-term viability and adaptability in the ever-evolving fitness landscape. Ultimately, the purpose of the CrossFit Gym Management System is to foster a cohesive and technologically advanced environment where administrators can efficiently operate, and members can actively engage in their fitness journey.

1.5 SCOPE

The CrossFit Gym Management System encompasses a broad and dynamic scope, addressing the multifaceted challenges faced by modern fitness centers. At its core, the system provides administrators with a powerful toolset to efficiently oversee and manage various aspects of gym operations, including the seamless addition, updating, and deletion of crucial data related to trainers, members, receptionists, and workers. The scope extends to empowering gym members through a user-centric module, enabling them to take control of their information with features allowing for self-service functionalities. Beyond member engagement, the system facilitates effective inquiry tracking, enabling administrators to nurture leads and drive business growth. The integration of MySQL as the backend database expands the scope to robust data management, ensuring security, integrity, and scalability. The user-friendly interface, achieved through HTML, CSS, and JSP, amplifies the scope by providing an intuitive platform for both administrators and members. Additionally, the system's scalability and customization features anticipate future needs, positioning it as a comprehensive solution poised to adapt to the evolving landscape of fitness management. The CrossFit Gym Management System's scope extends far beyond conventional management tools, offering a holistic approach to elevate the operational efficiency and member experience within fitness centers.

LIMITATIONS

Despite its many advantages, the CrossFit Gym Management System is not without limitations. One notable constraint lies in the dependency on internet connectivity; the system's optimal functionality may be compromised in the absence of a stable internet connection, hindering real-time data access. Additionally, the system's effectiveness hinges on the accurate input of information by administrators and members alike, making data integrity vulnerable to human error. Another limitation arises in the scalability aspect; while the system is designed with scalability in mind, unforeseen technological advancements or drastic increases in data volume may necessitate further adaptations. Furthermore, the system primarily focuses on basic administrative and member functionalities, potentially lacking more advanced features sought by larger or specialized fitness centers. As technology evolves, ongoing maintenance and updates may be required to ensure compatibility with the latest software and security standards. It's essential to acknowledge these limitations as inherent aspects of any software solution, prompting a proactive approach to address challenges and foster continuous improvement.

1.6 APPLICABILITY

The CrossFit Gym Management System is strategically designed to bring professionalism and efficiency to the management of fitness centers. Its features cater to the specific needs of gym administrators, enhancing their ability to streamline operations and provide a seamless experience for members. Here's how the system is professionally applicable:

Effective Staff Management:

Professional gym administrators can efficiently manage their staff, including trainers, receptionists, and workers, by easily adding, updating, and deleting their profiles through the Admin module.

Member Engagement and Satisfaction:

The system contributes to professional member engagement by allowing members to manage their profiles, fostering a sense of control and satisfaction in their fitness journey.

Lead Conversion and Business Growth:

Professional gym managers can use the system to effectively track and manage inquiries, facilitating timely follow-ups and lead conversions, thereby contributing to the overall growth of the fitness center.

Secure Data Management:

The integration of MySQL as the backend database ensures a secure and professional approach to data management, aligning with industry standards for confidentiality and integrity.

Scalability for Business Expansion:

Designed with scalability in mind, the system accommodates the growth of the fitness center professionally, allowing for the addition of new features and customization to meet evolving business requirements.

Adaptability to Industry Trends:

Professional fitness establishments can leverage the system to stay abreast of industry trends and technology standards, showcasing a commitment to adopting modern solutions for enhanced management practices.

Chapter - 2 (Feasibility Study and Stakeholders)

2.1 FEASIBILITY STUDY

An important outcome of preliminary investigation is to determine whether the proposed system is feasible or not. feasibility study is an assessment conducted during the initial stages of a project to evaluate its viability, practicality, and potential for success. It involves analyzing various factors, including technical, economic, legal, operational, and scheduling considerations, to determine whether the project is feasible or achievable within the constraints of the available resources and objectives. The primary goal of a feasibility study is to provide stakeholders with valuable insights and information to make informed decisions about whether to proceed with the project, modify its scope or approach, or abandon it altogether.

By conducting a comprehensive feasibility study, project stakeholders can gain a thorough understanding of the project's opportunities, challenges, risks, and constraints, enabling them to make well-informed decisions about its feasibility and potential for success. This information serves as a crucial foundation for the project planning and decision-making process, guiding subsequent actions and ensuring the project's alignment with organizational goals and objectives.

There are Some types of Feasibility Study:

- Technical Feasibility Study
- Economic Feasibility Study
- Financial Feasibility Study
- Operational Feasibility Study

1. Technical Feasibility Study

Technology Stack:

JSP, HTML, CSS, Java, and MySQL: The choice of these technologies is sound and widely accepted in web development. They provide a stable foundation for creating dynamic and interactive web applications, ensuring compatibility and support across various platforms.

Development Environment:

NetBeans IDE: NetBeans is a robust and user-friendly integrated development environment. Its support for Java and web development, along with debugging and version control features, enhances the development process. The familiarity and community support for NetBeans contribute to technical feasibility.

Scalability:

Designed for Growth: The system is designed with scalability in mind, allowing for future expansion and addition of features. This ensures that the system can accommodate the growing needs of the gym, making it technically feasible for long-term use.

Compatibility and Cross-Browser Support:

Browser Compatibility: The system is developed with cross-browser compatibility in mind, ensuring a consistent user experience across different web browsers. This technical consideration enhances usability and accessibility for both administrators and members.

Security Measures:

MySQL Database Security: The utilization of MySQL as the backend database provides robust security features. Measures such as encryption and user authentication are implemented to safeguard sensitive data, ensuring technical feasibility by addressing potential security concerns.

Integration of Frontend and Backend:

Effective Integration: The seamless integration of frontend technologies (JSP, HTML, CSS) with backend technologies (Java, MySQL) ensures smooth data flow and user interaction. This technical cohesion contributes to the overall functionality and user experience of the system.

Efficient Code Practices:

The development follows best practices for code optimization, ensuring efficient performance of the system. Techniques such as caching and minimizing server requests contribute to technical feasibility by optimizing resource utilization.

In conclusion, the technical feasibility of the CrossFit Gym Management System is underpinned by a robust technology stack, thoughtful design considerations, scalability features, security measures, and effective integration of frontend and backend components. This study affirms the system's technical soundness for successful implementation and sustained use.

2. Economic Feasibility Study

Development Costs:

Software and Hardware: The initial investment in software licenses, development tools, and hardware resources is considered. NetBeans IDE and other development tools are generally open-source, reducing upfront costs. Hardware requirements are evaluated for optimal system performance.

Human Resource Costs:

Development Team: The salaries or hourly rates of developers, designers, and other team members involved in the project are factored into the economic feasibility analysis. The skill set and experience of the team contribute to the overall cost.

Operational Costs:

Maintenance and Support: Ongoing operational costs, including maintenance, support, and updates, are considered. Regular updates to accommodate changing requirements and technologies contribute to the economic sustainability of the system.

Benefits and Revenue Generation:

Efficiency Gains: The system's potential to streamline administrative tasks, enhance member engagement, and improve lead management contributes to efficiency gains. These gains may translate into time savings for administrators and increased member satisfaction, indirectly impacting revenue.

Membership Fees and Growth: The system's capability to attract and retain members through improved services may result in increased membership fees and overall business growth. The economic feasibility is influenced by the potential for a return on investment through enhanced revenue streams.

Risk Analysis:

Mitigation Strategies: Economic feasibility considers potential risks such as project delays, scope changes, or unforeseen challenges. Mitigation strategies are evaluated to minimize the impact of these risks on the overall economic viability of the project.

In conclusion, the economic feasibility of the CrossFit Gym Management System is evaluated through a comprehensive analysis of development costs, operational expenses, revenue generation potential, risk mitigation, and comparison with alternative solutions. This study provides insights into the financial viability and sustainability of the project.

1. Financial Feasibility Study

A financial feasibility study assesses the financial viability of a proposed project by analyzing its costs, revenue projections, and financial metrics. A study on whether a project is viable after taking into consideration its total costs and probable revenues. If the revenues cover the costs of the project, then the project is viable. For this project, a financial feasibility study would examine various financial aspects to determine whether the project is financially feasible and has the potential to generate sufficient returns on investment. Here's an outline of what would typically be included in a financial feasibility study for this project:

Cost Estimation:

Identify and estimate the initial costs associated with developing and launching the online gym website. This includes costs such as:

Software development, Website design and hosting, Infrastructure setup, Marketing and promotional expenses.

Revenue Projections:

Forecast the potential revenue streams generated by the online bookstore website.

Financial Metrics Analysis:

Calculate financial metrics to evaluate the project's financial performance and viability.

Risk Assessment:

Identify and evaluate potential financial risks and uncertainties that may affect the project's financial feasibility. This includes risks related to market conditions, competition, technological changes, regulatory compliance, and financial constraints.

Conclusion and Recommendations:

Summarize the findings of the financial feasibility study and provide recommendations regarding the viability of the project.

Determine whether the project is financially feasible, considering factors such as the projected returns on investment, risk factors, and sensitivity to key variables.

Make recommendations for adjustments to the project plan, financial assumptions, or risk mitigation strategies to improve its financial feasibility and likelihood of success.

By conducting a financial feasibility study for the "CrossFit– an Online Gym Website" project, stakeholders can gain valuable insights into its financial viability, risks, and potential returns, enabling informed decision-making and strategic planning for the project's development and implementation.

4.Operational Feasibility Study

Operational feasibility is a critical dimension in assessing the viability of the CrossFit Gym Management System, encompassing its seamless integration into the existing operational framework of the gym. The system's design is inherently user-centric, aligning with the diverse needs of both administrators and members. With an intuitive and user-friendly interface, administrators can easily navigate through functionalities to add, update, delete, and view information about trainers, members, receptionists, and workers. The ability to view inquiries provides administrators with a comprehensive overview, enabling effective lead management and business growth.

For gym members, the operational feasibility is evident in the self-service capabilities embedded within the User (Member) module. Members can effortlessly add and view their own details, fostering autonomy and engagement in their fitness journey. The system's design takes into account the need for minimal training, ensuring a smooth transition for both administrators and members. This user-centric operational approach not only enhances the overall efficiency of gym management but also contributes to a positive experience for all stakeholders involved.

Moreover, the centralized information hub within the system aids in eliminating data silos, reducing the likelihood of errors, and streamlining data management. The operational feasibility is further enhanced by the system's scalability, allowing for the addition of new features and modules as the gym expands. The incorporation of familiar technologies such as JSP, HTML, CSS, Java, and MySQL, along with the use of NetBeans as the development environment, ensures compatibility and ease of integration with existing systems.

In conclusion, the operational feasibility of the CrossFit Gym Management System is evident in its user-centric design, intuitive functionalities, and seamless integration into the day-to-day operations of the gym. The system's adaptability, scalability, and centralized information management contribute to its effectiveness in enhancing operational efficiency and promoting a positive experience for both administrators and gym members.

2.2 STAKEHOLDERS

Stakeholders for the CrossFit Gym Management System can be identified at various levels, representing those who are directly involved or impacted by the project. Here is a list of key stakeholders:

Gym Administrators:

Gym administrators are primary stakeholders responsible for overseeing the implementation, use, and maintenance of the system. Their input is crucial in shaping the system's features, ensuring it aligns with the specific needs and workflows of the gym.

Members (Users):

Gym members are direct users of the system and, therefore, significant stakeholders. Their needs and expectations influence the user interface, functionality, and overall user experience. Member feedback is valuable for ongoing system improvements.

Trainers:

Trainers play a vital role in the gym ecosystem. They are stakeholders with a vested interest in features related to class scheduling, member attendance tracking, and member progress monitoring. Involving trainers in the system design ensures it supports their daily activities.

Receptionists:

Receptionists are essential for managing inquiries, member registrations, and ensuring smooth day-to-day operations. Their input is valuable for features related to member enrollment, visitor management, and communication tools within the system.

Workers/Staff:

Other staff members, such as cleaning staff or maintenance personnel, are stakeholders with specific operational needs. The system should consider features related to staff management, task assignments, and communication channels to streamline internal processes.

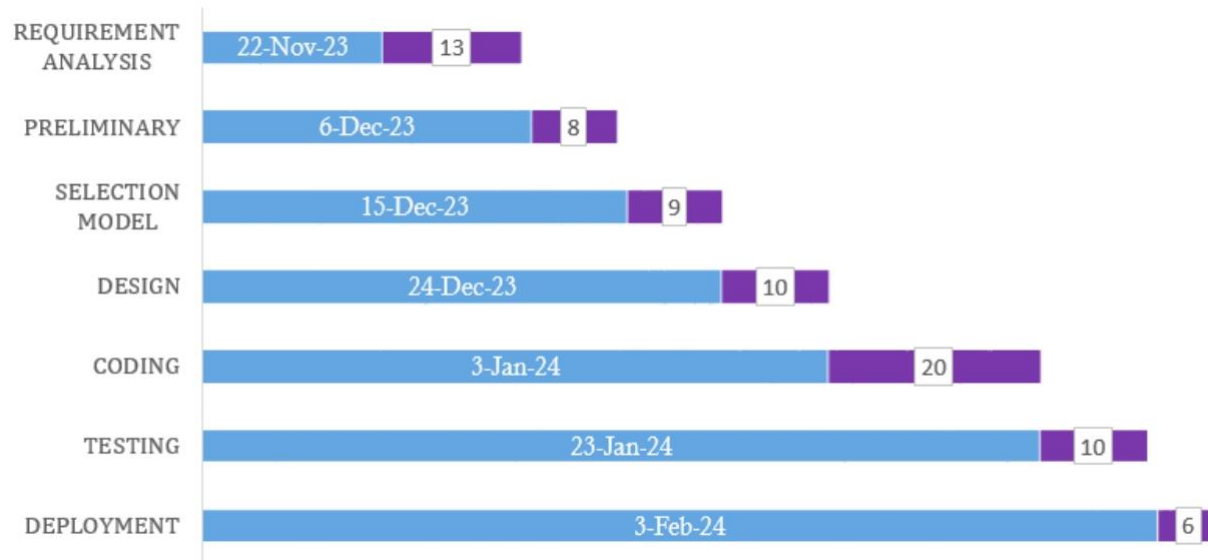
Business Owners/Managers:

Owners or managers of the gym are ultimate decision-makers and key stakeholders. They are concerned with the system's impact on the gym's overall performance, business growth, and return on investment. Involving them in decision-making ensures alignment with business goals.

Enquiry Management Team:

Individuals or teams responsible for handling gym inquiries are stakeholders with a focus on lead management. The system's features related to inquiry tracking, follow-ups, and conversion analytics directly impact their responsibilities.

Chapter - 3 (Gantt Chart)



Chapter – 4 (Software Requirement Specification)

4.1 SOFTWARE AND HARDWARE REQUIREMENTS

Hardware Requirements

Component	Description
Computer/Laptop	Minimum: Dual-core processor, 4GB RAM, 100GB HDD Recommended: Quad-core processor, 8GB RAM, SSD
Monitor	Resolution: 1920x1080 or higher
Internet Connection	Broadband or high-speed internet

Software Requirements

Component	Description
Operating System	Windows, MacOS or any type of OS
Java Development Kit	JDK 8 or later
NetBeans IDE	Netbeans 8.2 or later , here I'm using NetBeans IDE 8.0.2
Application Server	Apache Tomcat 9 or later / Glassfish Server
Database	MySQL 5.7 or later

4.2 FUNCTIONAL REQUIREMENTS

Functional Requirements:

User Authentication and Authorization: Enable secure login for admins and member self-registration.

Admin Module: Add, update, delete, and view trainers, members, receptionists, and workers.

View and manage gym inquiries.

User (Member) Module: Allow members to add and view personal details, fitness progress, and attendance history.

Enquiry Management: Facilitate tracking and management of gym inquiries.

Responsive Design: Ensure a seamless user experience across devices.

Centralized Information Hub: Provide a centralized repository for storing and managing information.

4.3 Non-Functional Requirements:

Performance: Respond promptly to user actions with minimal latency.

Security: Implement robust security measures, including data encryption and secure login processes.

Scalability: Accommodate an increasing number of users, trainers, and members.

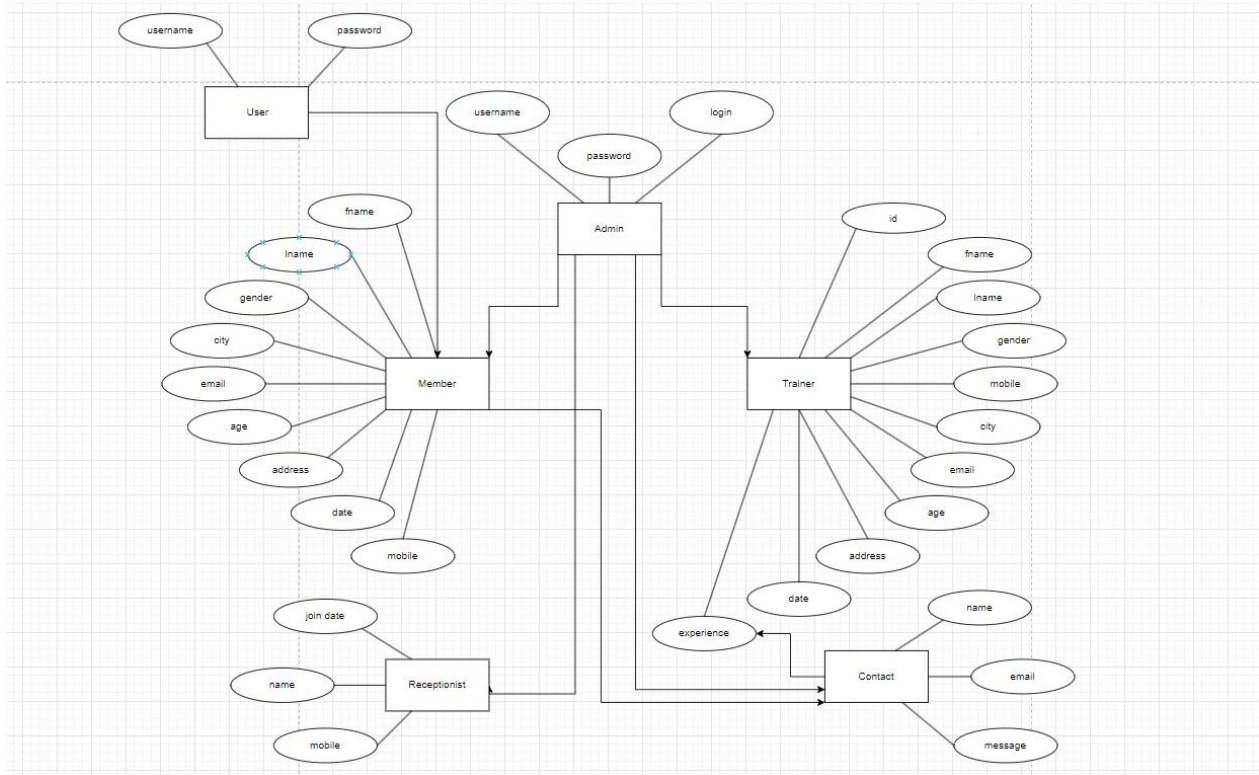
Reliability: Ensure minimal downtime for maintenance and daily automated backups.

Usability: Design an intuitive and user-friendly interface requiring minimal training

4.4 CONCEPTUAL MODELS (DATA FLOW DIAGRAMS)

1. E-R Diagram

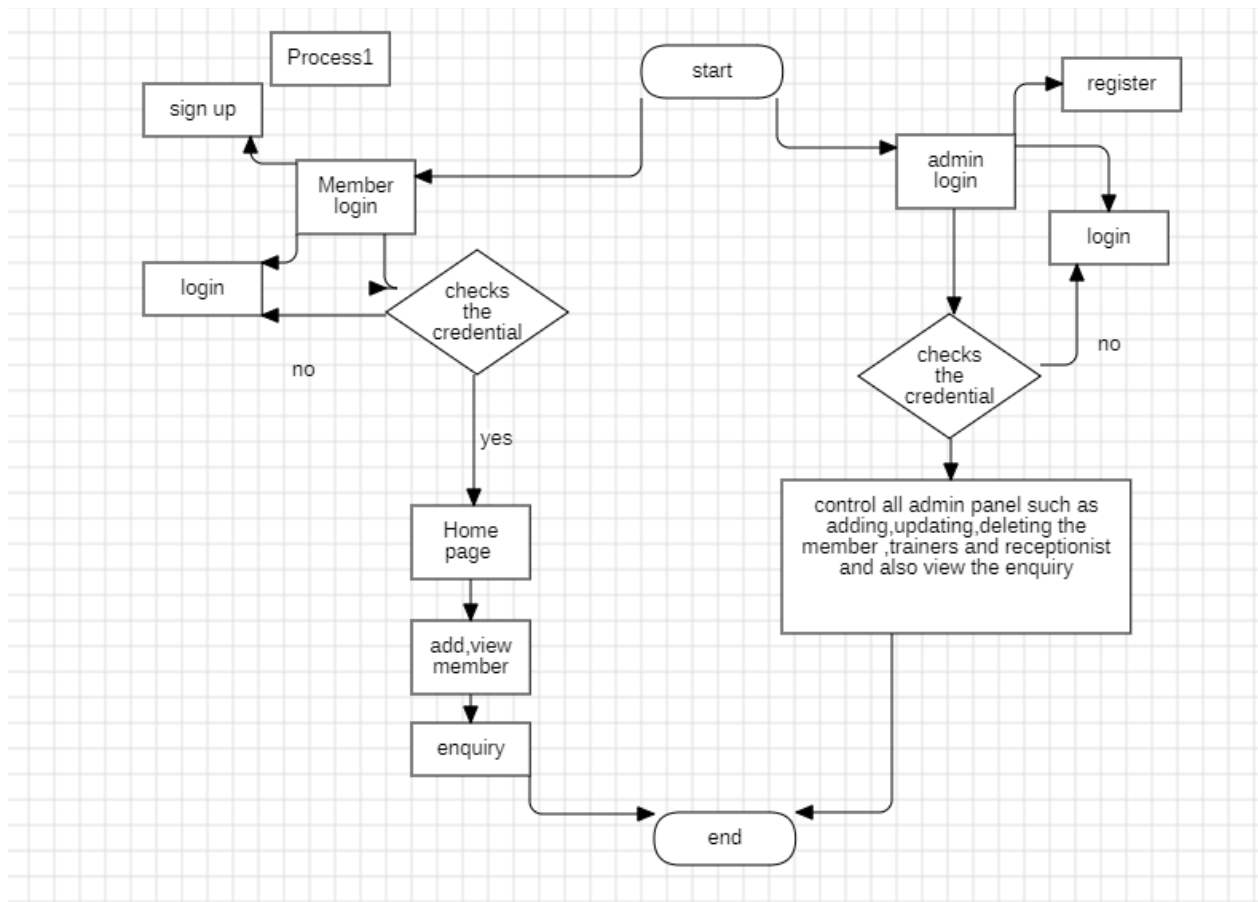
An Entity-Relationship (ER) diagram is a visual representation of the entities, attributes, relationships, and constraints within a database schema. It is a conceptual modeling technique used in database design to depict the structure and organization of data in a system. ER diagrams are essential for designing databases, as they provide a clear and concise overview of the data model and help stakeholders understand the relationships between different entities.



2. Flowchart Diagram

A flowchart diagram is a graphical representation of a process or algorithm, depicting the sequence of steps and decisions involved in completing a task or solving a problem. Flowcharts use standardized symbols and shapes to represent different types of actions, processes, and decision points, making them easy to understand and interpret by various stakeholders.

Overall, flowchart diagrams are versatile tools used in a wide range of fields and industries to visually represent processes, algorithms, and workflows, facilitating communication, analysis, and problem-solving.



3. Class Diagram

A Class Diagram is a type of static structure diagram in the Unified Modeling Language (UML) that illustrates the structure of a system by showing the classes of objects, their attributes, methods, and relationships among the classes. Class diagrams are widely used in software development to visualize the object-oriented design of a system.

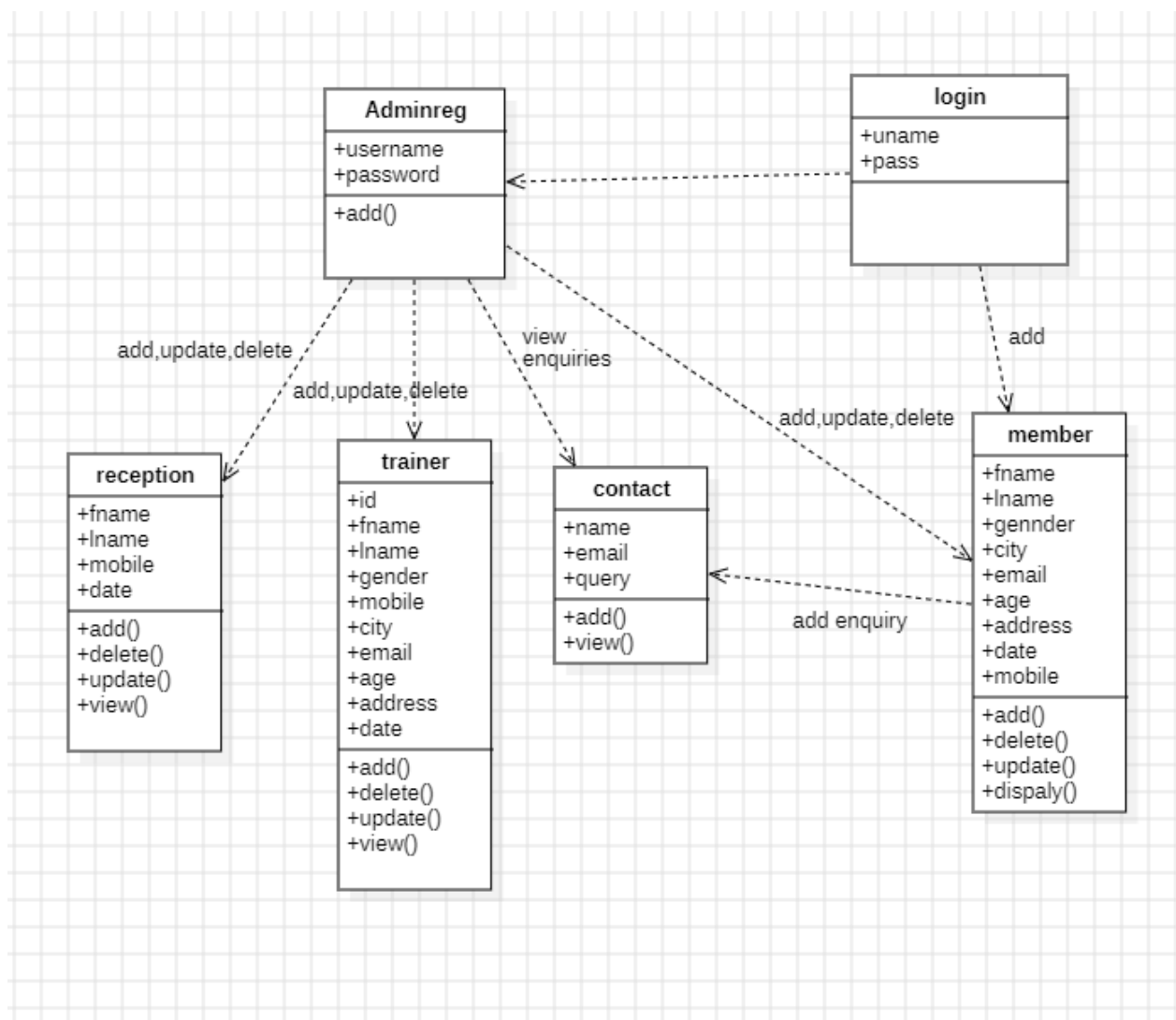
Class diagrams are **useful** for:

Understanding the structure and organization of a system's classes and their relationships.

Communicating the design of a system to stakeholders, including developers, designers, and clients.

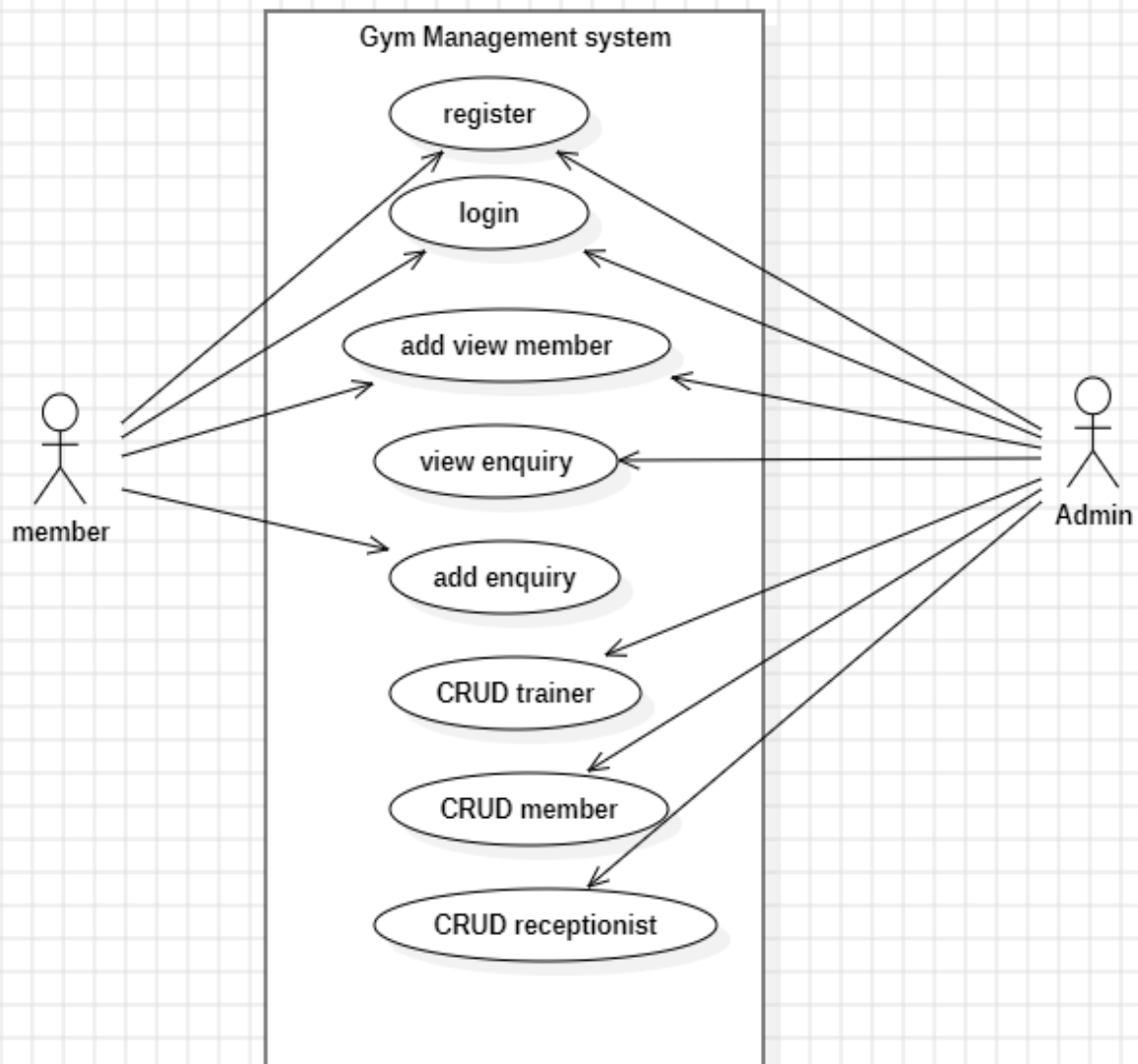
Serving as a blueprint for implementing the system in code, especially in object-oriented programming languages like Java, C++, and Python.

Overall, Class Diagrams provide a high-level overview of the static structure of a system, focusing on the classes, their attributes, methods, and relationships, and are an essential tool in software development for designing and documenting object-oriented systems.



4. Use-case Diagram

A use case diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that illustrates the interactions between actors (users or external systems) and a system under consideration. It provides a high-level view of the functionalities or features of a system from the perspective of its users.



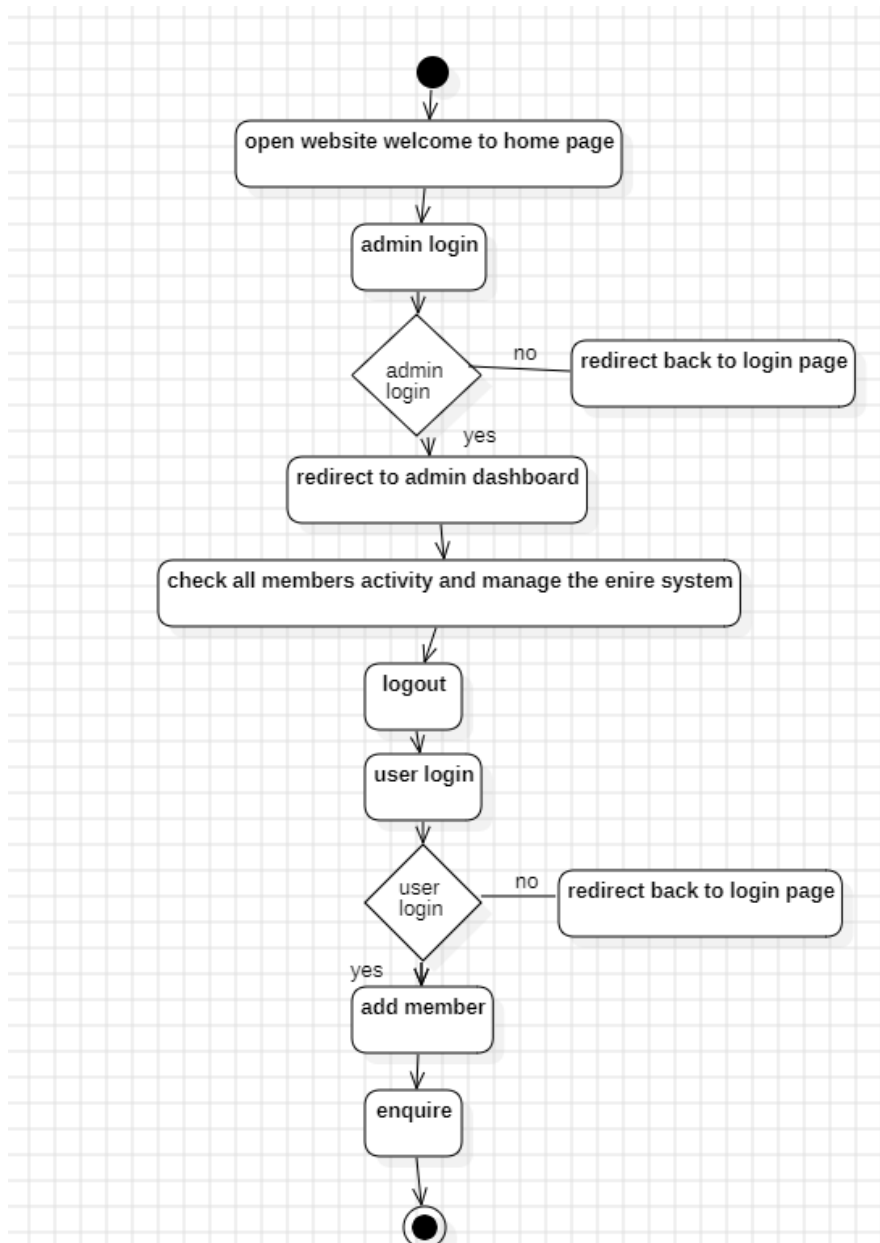
5. Activity Diagram

An activity diagram is a behavioral diagram in the Unified Modeling Language (UML) that illustrates the flow of control or behavior of a system or business process. Activity diagrams are used to model the dynamic aspects of a system, showing the sequence of activities, actions, decisions, and transitions between them.

Key components of an activity diagram include:

Activity: Represents a specific action or task performed as part of a process or workflow. Activities are depicted as rounded rectangles and can include both simple actions (e.g., sending an email) and complex processes (e.g., order processing).

Control Flow: Represents the flow of control between activities, indicating the sequence in which activities are executed. Control flow arrows connect activities, showing the order in which they are performed.



6. Package Diagram

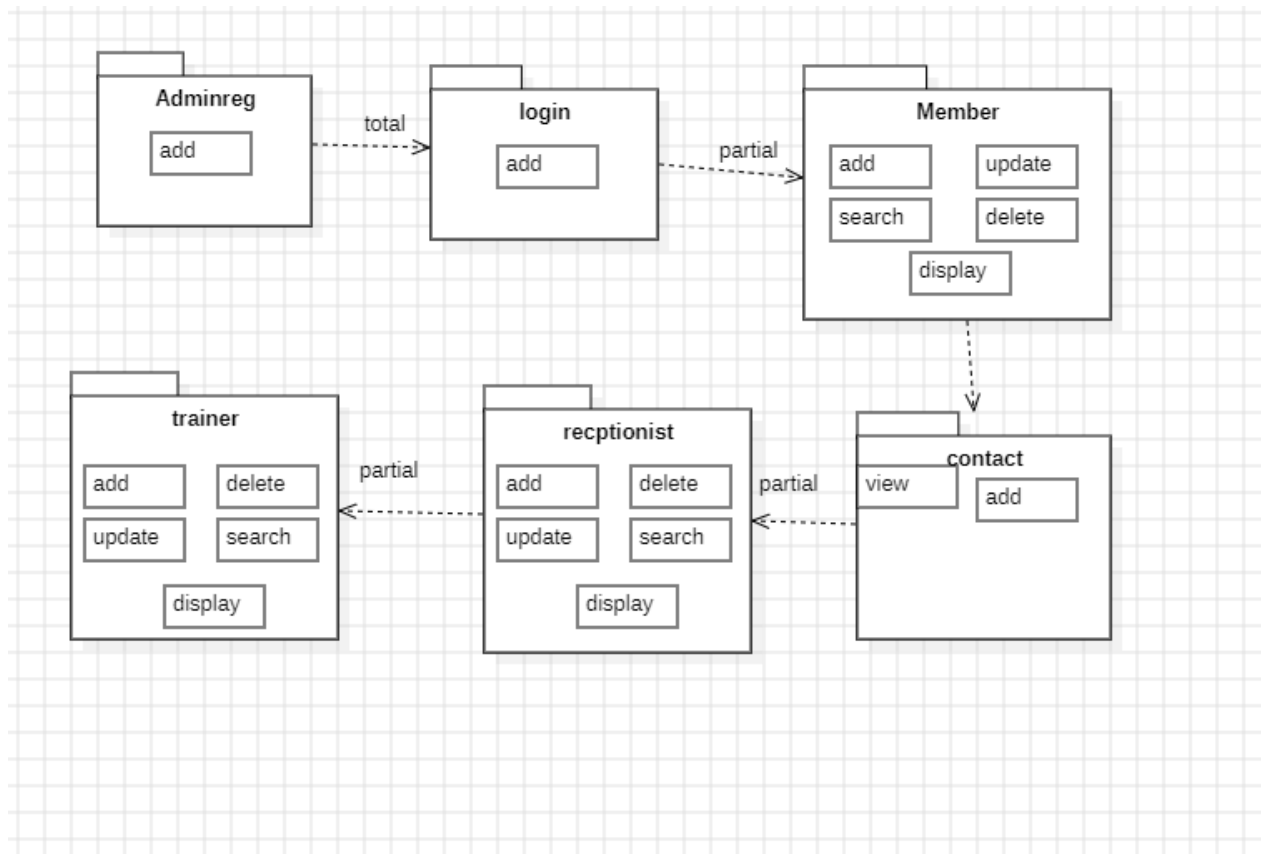
A package diagram is a type of UML (Unified Modeling Language) diagram that represents the organization and arrangement of various elements in a software system. It provides a high-level view of the system's structure, showcasing how different components and artifacts are grouped into packages. In software development, a package is a mechanism for organizing and controlling access to a set of classes, interfaces, and other elements.

Here are key components and concepts related to package diagrams:

Package: A package is a container that holds a collection of related elements, such as classes, interfaces, and other packages. It serves as a namespace, allowing you to organize and modularize the system.

Dependency: Dependencies between packages represent the relationships and connections between them. A dependency indicates that changes in one package may affect another package.

Association: Associations may be used to represent relationships between classes or components within the packages.



7. Deployment Diagram

A deployment diagram in software engineering and systems engineering illustrates the physical deployment of software components and the infrastructure on which they run. It provides a high-level view of the system's hardware and software architecture, showing how software components are distributed across nodes (physical devices or computing resources).

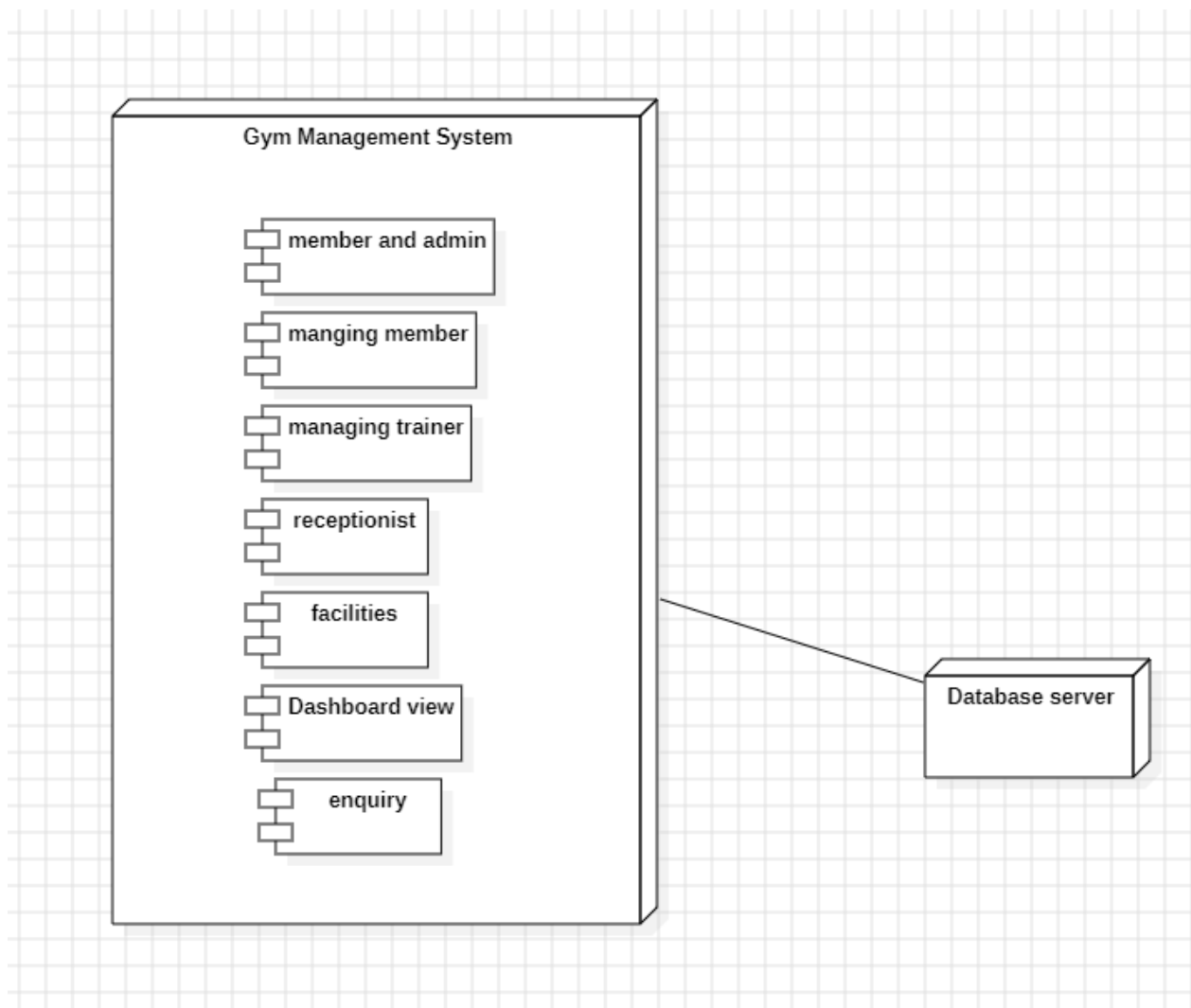
Here are key elements and information about deployment diagrams:

Components:

Nodes: Nodes represent physical entities in the deployment diagram. These can be hardware devices like servers, personal computers, or any device capable of executing software.

Components: Components represent the software elements deployed to the nodes. These could be application modules, databases, or other executable files.

Artifacts: Artifacts represent files or data that are used or produced during the execution of a software component. They are often associated with components.ts within the packages.



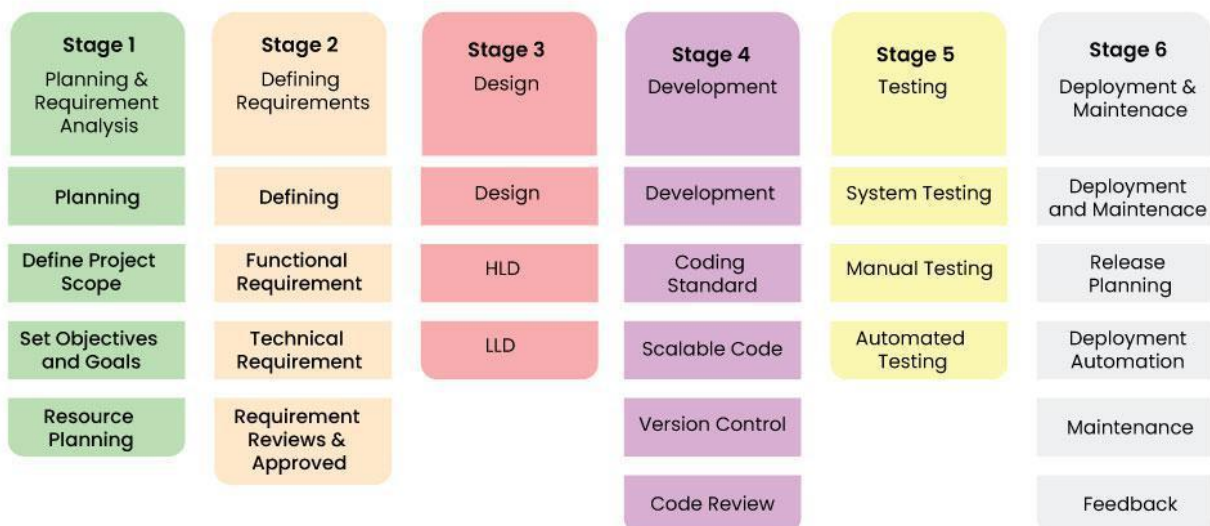
Chapter – 5 (Software Development Life Cycle)

5.1 What is SDLC?

Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.

SDLC is a process followed for software building within a software organization. SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software. The life cycle defines a method for improving the quality of software and the all-around development process.

SDLC specifies the task(s) to be performed at various stages by a software engineer or developer. It ensures that the end product is able to meet the customer's expectations and fits within the overall budget. Hence, it's vital for a software developer to have prior knowledge of this software development process.



6 Stages of Software Development Life Cycle



5.2 SDLC MODELS

To this day, we have more than 50 recognized SDLC models in use. But None of them is perfect, and each brings its favourable aspects and disadvantages for a specific software development project or a team.

In this article, I've listed the most popular SDLC model below.

Agile Model

The agile model was mainly designed to adapt to changing requests quickly. The main goal of the Agile model is to facilitate quick project completion. The agile model refers to a group of development processes. These processes have some similar characteristics but also possess certain subtle differences among themselves.

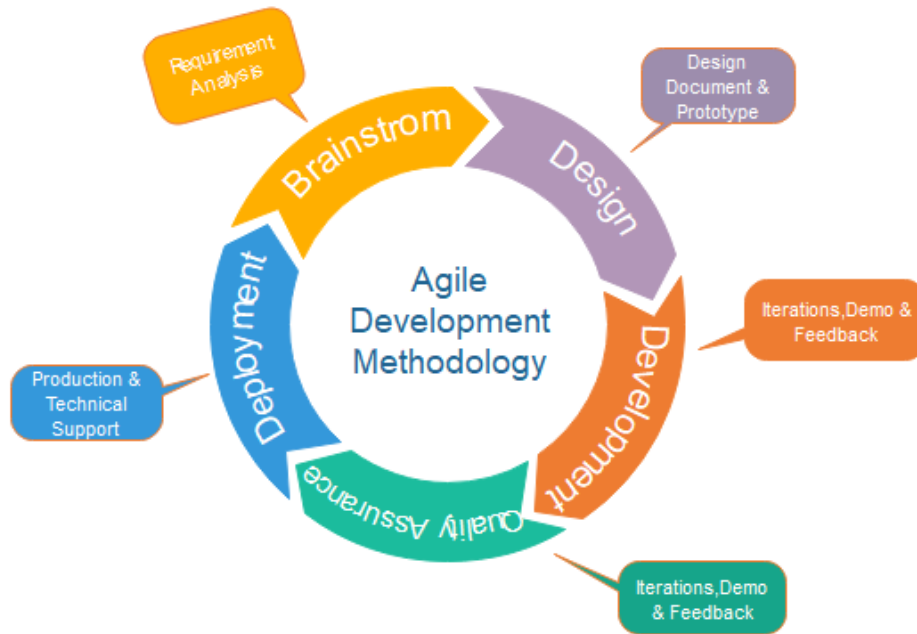


Fig. Agile Model

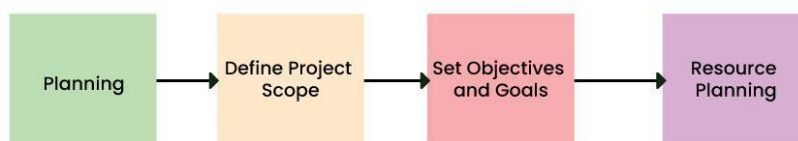
5.3 PHASES OR STAGES OF SDLC ARE AS FLLOWS:

Stage-1: Planning and Requirement Analysis

Planning is a crucial step in everything, just as in software development. In this same stage, requirement analysis is also performed by the developers of the organization. This is attained from customer inputs, and sales department/market surveys.

The information from this analysis forms the building blocks of a basic project. The quality of the project is a result of planning. Thus, in this stage, the basic project is designed with all the available information.

Stage-1: Planning and Requirement Analysis

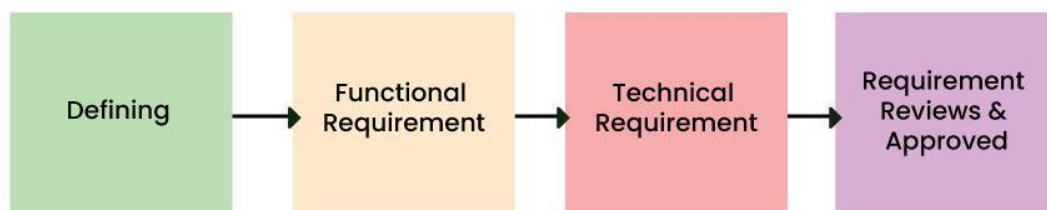


Stage-2: Defining Requirements

In this stage, all the requirements for the target software are specified. These requirements get approval from customers, market analysts, and stakeholders.

This is fulfilled by utilizing SRS (Software Requirement Specification). This is a sort of document that specifies all those things that need to be defined and created during the entire project cycle.

Stage-2: Defining Requirements



Stage-3: Designing Architecture

SRS is a reference for software designers to come up with the best architecture for the software. Hence, with the requirements defined in SRS, multiple designs for the product architecture are present in the Design Document Specification (DDS).

This DDS is assessed by market analysts and stakeholders. After evaluating all the possible factors, the most practical and logical design is chosen for development.

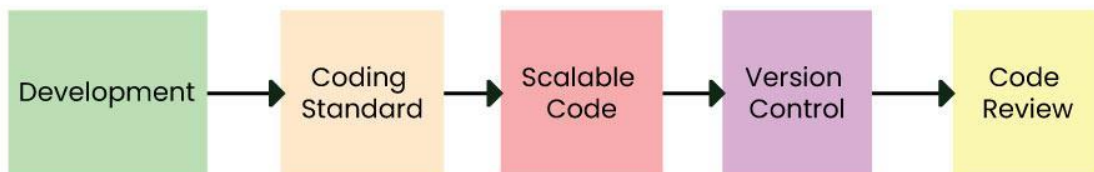
Stage-3: Designing Architecture



Stage-4: Developing Website(product)

At this stage, the fundamental development of the product starts. For this, developers use a specific programming code as per the design in the DDS. Hence, it is important for the coders to follow the protocols set by the association. Conventional programming tools like compilers, interpreters, debuggers, etc. are also put into use at this stage. Some popular languages like C/C++, Python, Java, etc. are put into use as per the software regulations.

Stage-4: Developing Product

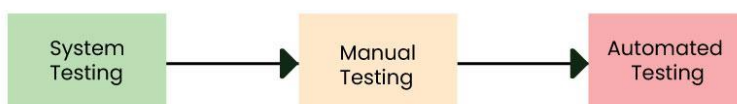


Stage-5: Product Testing and Integration

After the development of the website or product, testing of the software is necessary to ensure its smooth execution. Although, minimal testing is conducted at every stage of SDLC. Therefore, at this stage, all the probable flaws are tracked, fixed, and retested. This ensures that the product confronts the quality requirements of SRS.

Documentation, Training, and Support: Software documentation is an essential part of the software development life cycle. A well-written document acts as a tool and means to information repository necessary to know about software processes, functions, and maintenance. Documentation also provides information about how to use the product. Training in an attempt to improve the current or future employee performance by increasing an employee's ability to work through learning, usually by changing his attitude and developing his skills and understanding.

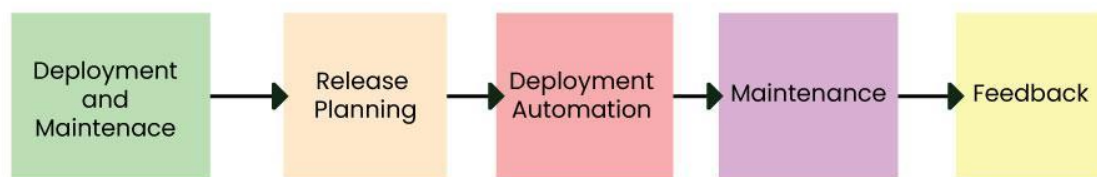
Stage-5: Product Testing and Integration



Stage 6: Deployment and Maintenance of Products

After detailed testing, the conclusive product is released in phases as per the organization's strategy. Then it is tested in a real industrial environment. It is important to ensure its smooth performance. If it performs well, the organization sends out the product as a whole. After retrieving beneficial feedback, the company releases it as it is or with auxiliary improvements to make it further helpful for the customers. However, this alone is not enough. Therefore, along with the deployment, the product's supervision.

Stage 6: Deployment and Maintenance of Products



Stakeholders, including clients, end-users, project managers, and developers, play crucial roles throughout the SDLC. They contribute to requirements gathering, provide feedback during development, and ensure that the final product aligns with business objectives.

Chapter – 6 (Designing)

6.1 SYSTEM DESIGN

Creating a comprehensive system design involves defining the architecture, components, and interactions within the CrossFit Gym Management System. Below are key elements of the system design:

System Architecture:

Three-Tier Architecture:

Implement a three-tier architecture comprising a presentation layer (user interface), application layer (business logic), and data layer (database).

Presentation Layer:

Develop the user interface using JSP, HTML, and CSS for an interactive and responsive design. Utilize JavaScript for client-side validations and dynamic content rendering.

Application Layer:

Implement the application layer using Java to handle business logic. Use servlets and JSP for server-side processing and dynamic content generation.

Data Layer:

Employ MySQL as the relational database management system (RDBMS) for efficient data storage. Utilize JDBC (Java Database Connectivity) for seamless interaction between the application and the database.

Components and Modules:

Admin Module:

Admin Management: Add, update, delete, and view administrators.

Trainer Management: Add, update, delete, and view trainers.

Member Management: Add, update, delete, and view members.

Receptionist Management: Add, update, delete, and view receptionists.

Worker Management: Add, update, delete, and view workers.

Enquiry Management: View and manage gym inquiries.

User (Member) Module:

Member Profile: Add and view personal details, fitness progress, and attendance history.

System Flow:

User Authentication and Authorization: Users log in with unique credentials.

Admins can access all modules, while members have restricted access.

Admin Actions:Admins perform CRUD operations on trainers, members, receptionists, workers, and inquiries through the Admin module.

Member Actions:Members add and view personal details, fitness progress, and attendance history through the User (Member) module.

Enquiry Management:Admins view inquiries, categorize them, and track follow-ups.

Responsive Design:Ensure a seamless and responsive user interface for access across various devices.

Security Measures:

Secure Login:Implement secure login mechanisms with password hashing for user authentication.

Role-Based Access Control:Apply role-based access control to restrict functionalities based on user roles (Admin, Member).

Data Encryption:Encrypt sensitive data stored in the database to enhance security.

This system design serves as a foundational blueprint, and you can adapt and expand it based on specific project requirements and constraints.

6.2 DATA DESIGN

Admin Table:

adminreg
username
password

Contact Table:

contact
name
email
query

loginTable:

Login
username
password

Member Table:

member
fname
lname

Gender
City
email
age
address
date
mobile

Trainer Table:

member
id
fname
lname
Gender
mobile
City
email
age
address
date
experience

Recp Table:

contact
fname
lname
mobile
date

6.3 DATA INTEGRITY AND CONSTRAINT

Admin TABLE:

ATTRIBUTE NAME	DATATYPE	CONSTRAINTS
username	Varchar(30)	
Password	Varchar(30)	

Contact TABLE:

ATTRIBUTE NAME	DATATYPE	CONSTRAINTS
name	Varchar(30)	
email	Varchar(30)	

query	Varchar(30)	
-------	-------------	--

Login TABLE:

ATTRIBUTE NAME	DATATYPE	CONSTRAINTS
password	Varchar(15)	
Quantity	Varchar(15)	

Member TABLE:

ATTRIBUTE NAME	DATATYPE	CONSTRAINTS
fname	Varchar(30)	
lname	Varchar(30)	
gender	Varchar(25)	
City	Varchar(30)	
Email	Varchar(30)	
age	Varchar(30)	
Address	Varchar(30)	
Date	Varchar(30)	
mobile	Varchar(30)	PrimaryKey

Recp TABLE:

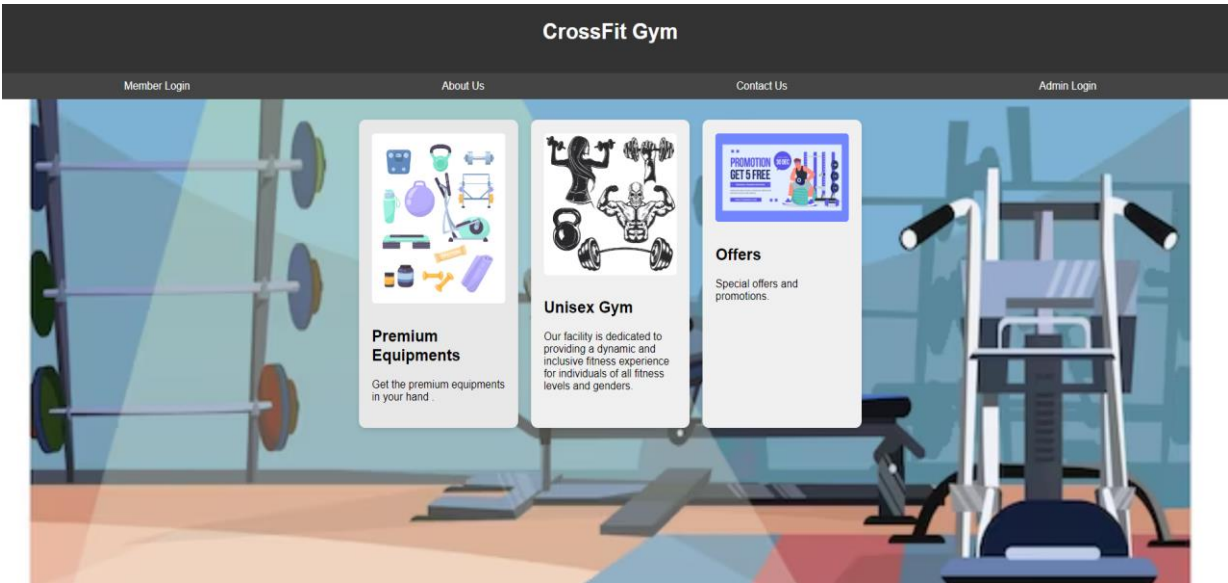
ATTRIBUTE NAME	DATATYPE	CONSTRAINTS
fname	Varchar(30)	
lname	Varchar(30)	
date	Varchar(10)	
mobile	Varchar(30)	Primarykey

Trainer TABLE:

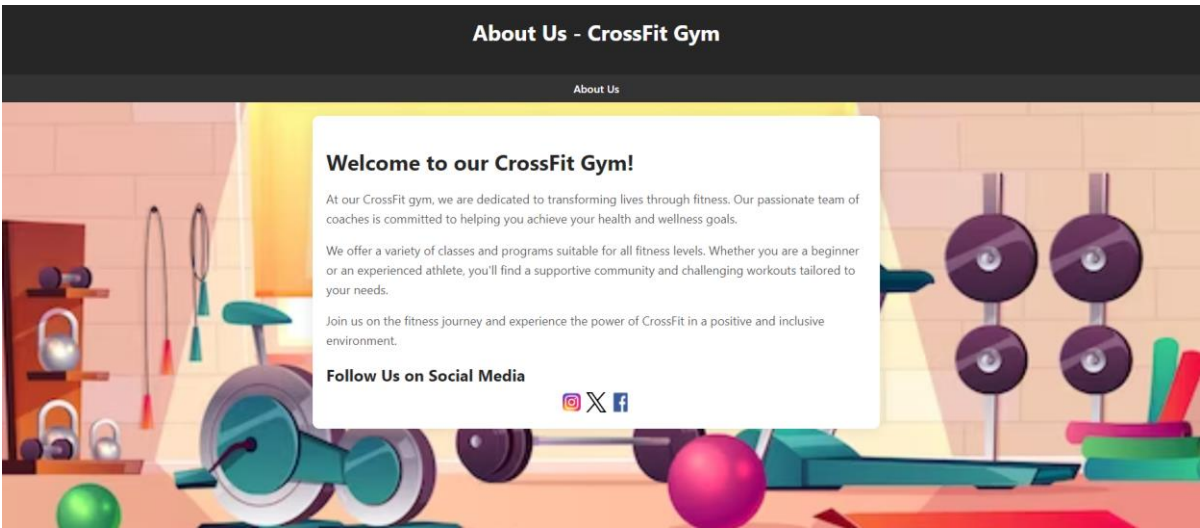
ATTRIBUTE NAME	DATATYPE	CONSTRAINTS
id	Int	Primarykey
fname	Varchar(30)	
lname	Varchar(30)	
gender	Varchar(25)	
City	Varchar(30)	
Email	Varchar(30)	
age	Varchar(30)	
Address	Varchar(30)	
Date	Varchar(30)	
mobile	Varchar(30)	
experience	Varchar(30)	

6.4USER INTERFACE AND DESIGN

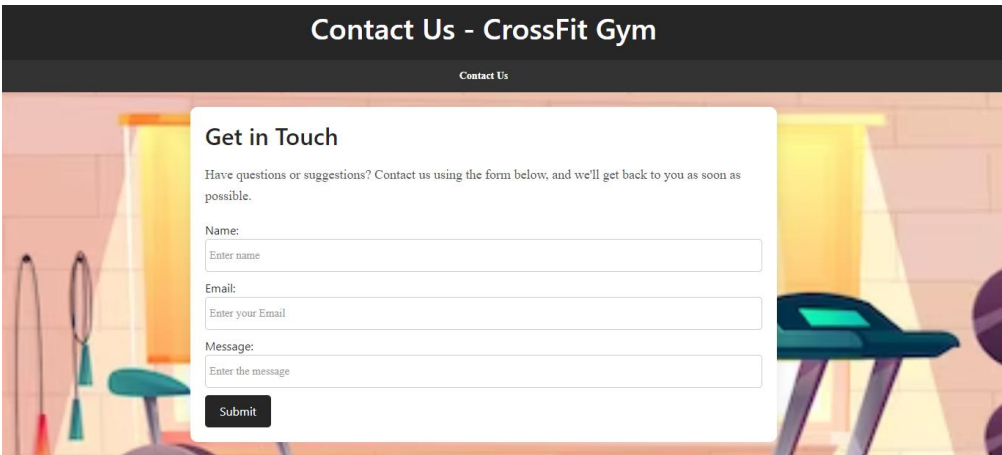
Main Page:



Aboutus page:

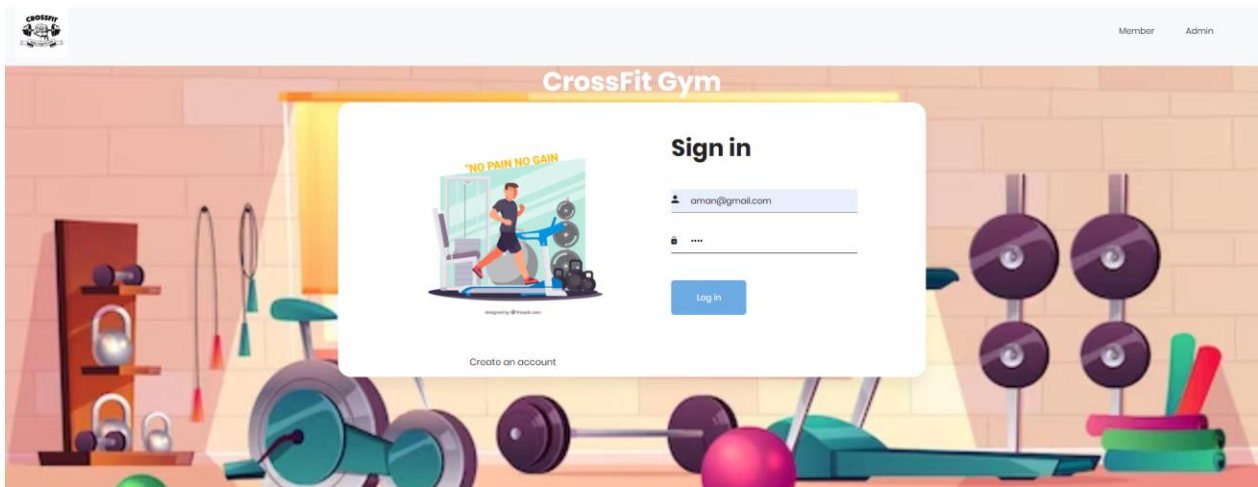


ContactUs page:

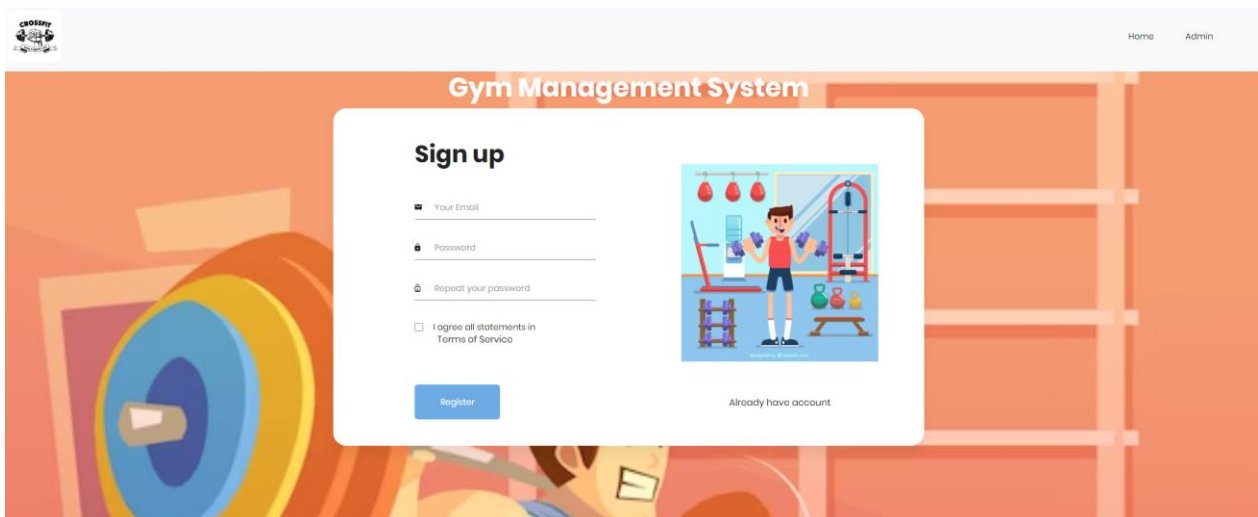


Admin:

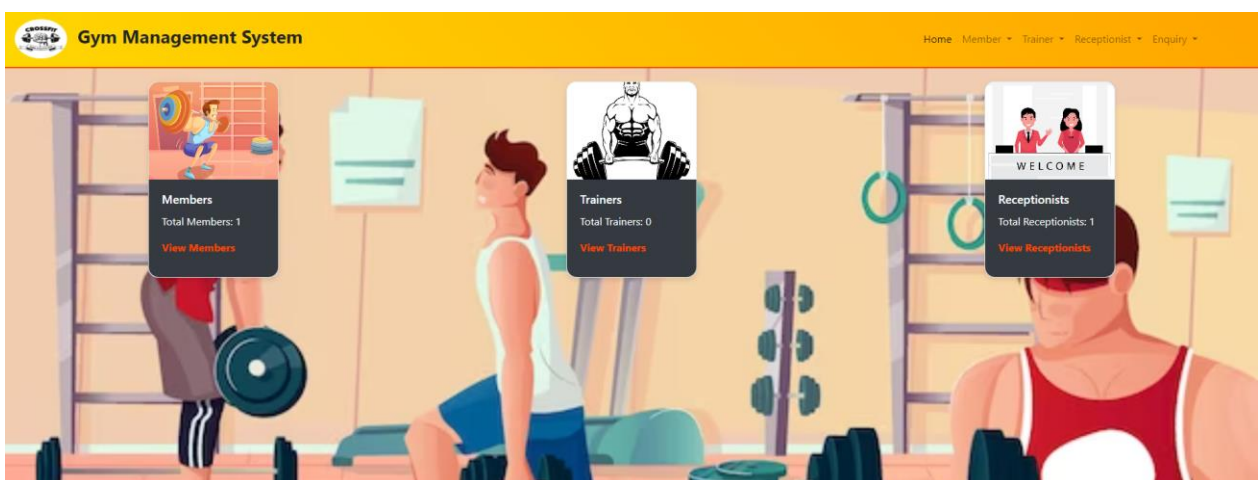
Admin login :



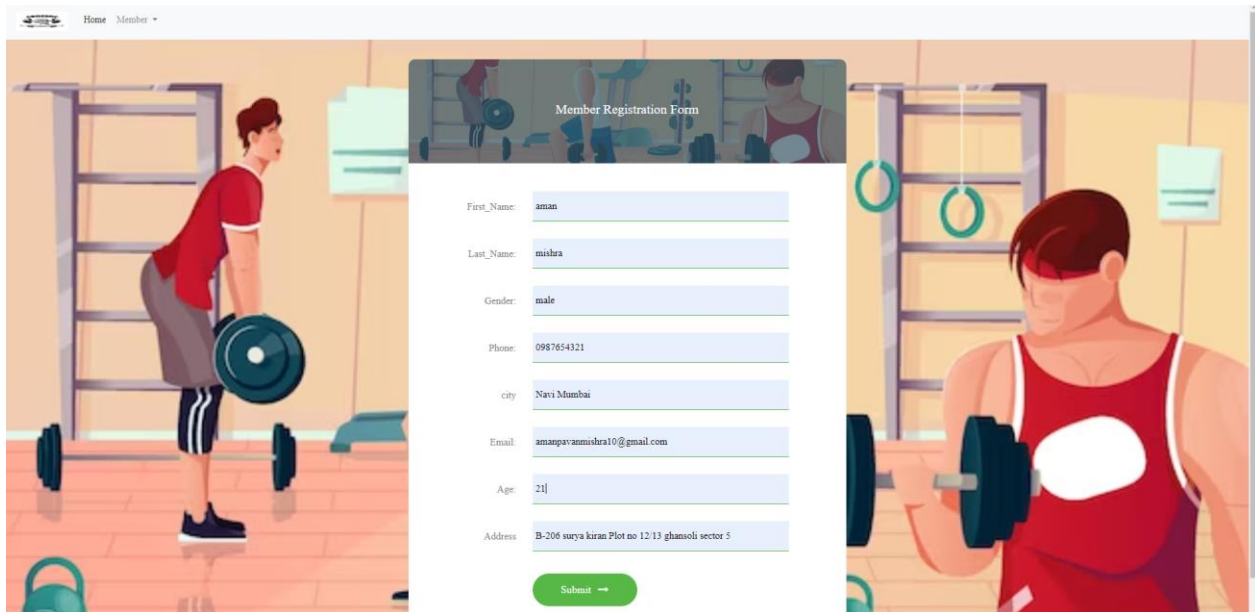
Create admin account:



Admin Dashboard:



Add member:



Member Registration Form

First_Name: aman

Last_Name: mishra

Gender: male

Phone: 0987654321

City: Navi Mumbai

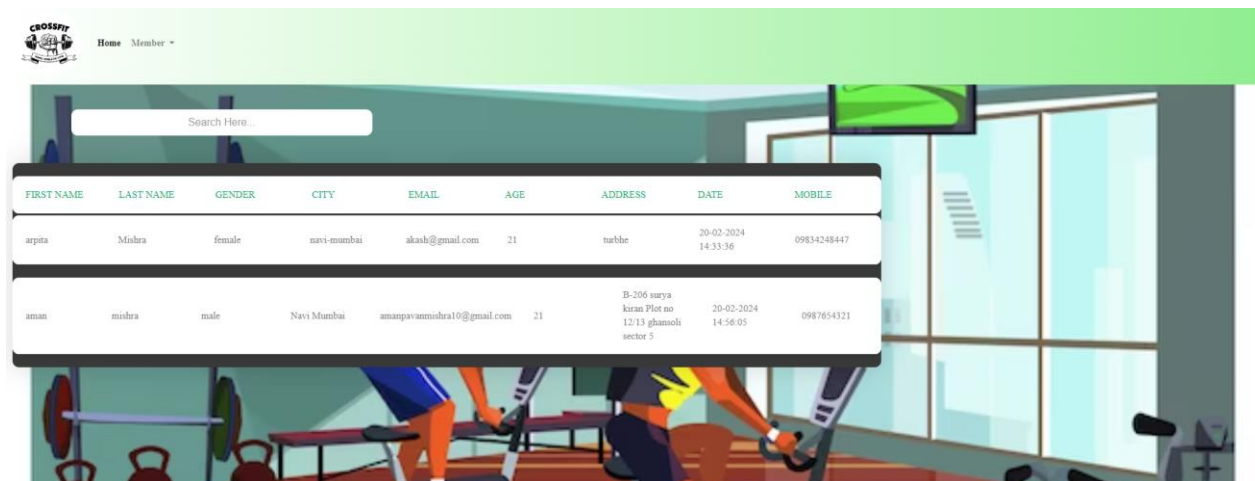
Email: amanpavammishra10@gmail.com

Age: 21

Address: B-206 surya kiran Plot no 12/13 ghanoli sector 5

Submit

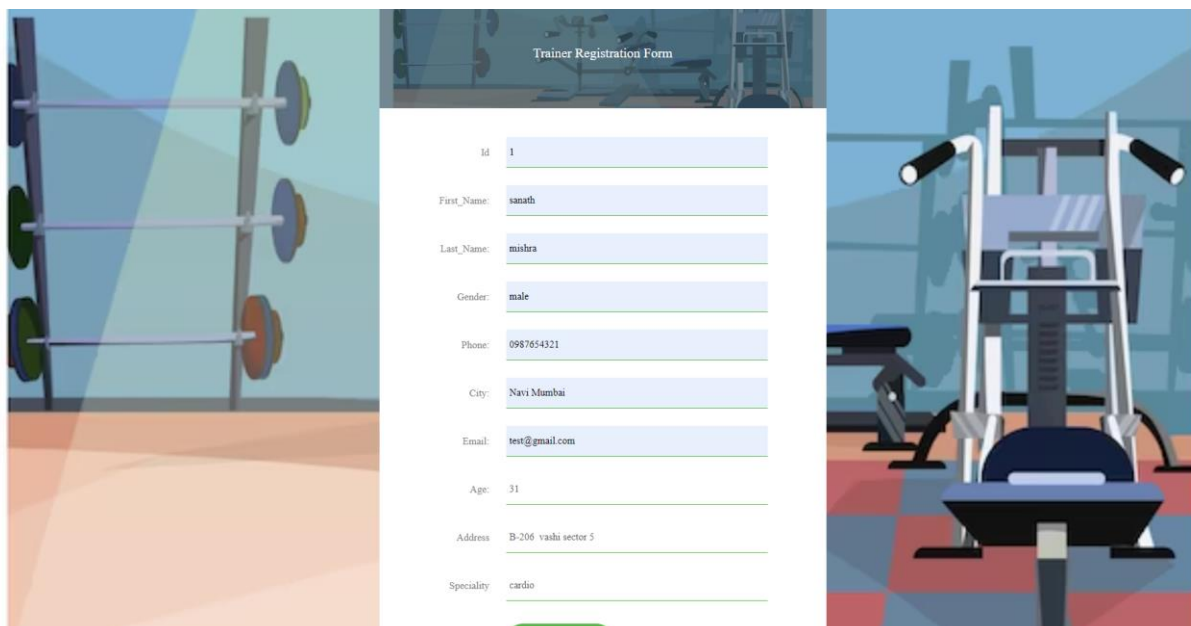
View Member Page:



Search Here...

FIRST NAME	LAST NAME	GENDER	CITY	EMAIL	AGE	ADDRESS	DATE	MOBILE
arpita	Mishra	female	navi-mumbai	akash@gmail.com	21	turbhe	20-02-2024 14:33:36	09834248447
aman	mishra	male	Navi Mumbai	amanpavammishra10@gmail.com	21	B-206 surya kiran Plot no 12/13 ghanoli sector 5	20-02-2024 14:56:05	0987654321

Add trainer:



Trainer Registration Form

Id: 1

First_Name: sanath

Last_Name: mishra

Gender: male

Phone: 0987654321

City: Navi Mumbai

Email: test@gmail.com

Age: 31

Address: B-206 vashi sector 5

Speciality: cardio

View trainer:

Gym Management System Home Member Trainer Receptionist

Search Here...

ID	FIRST NAME	LAST NAME	GENDER	MOBILE	CITY	EMAIL	AGE	ADDRESS	DATE	QUALIFICATION	
1	sanzath	mishra	male	0987654321	Navi Mumbai	test@gmail.com	31	B-206 vashi sector 5	20-02-2024 15:01:11	null	Edit Delete

Add receptionist:

Gym Management System Home Member Trainer Receptionist

Receptionist Registration Form

First_Name: mayuri

Last_Name: kale

Phone: 9873453623

Submit →

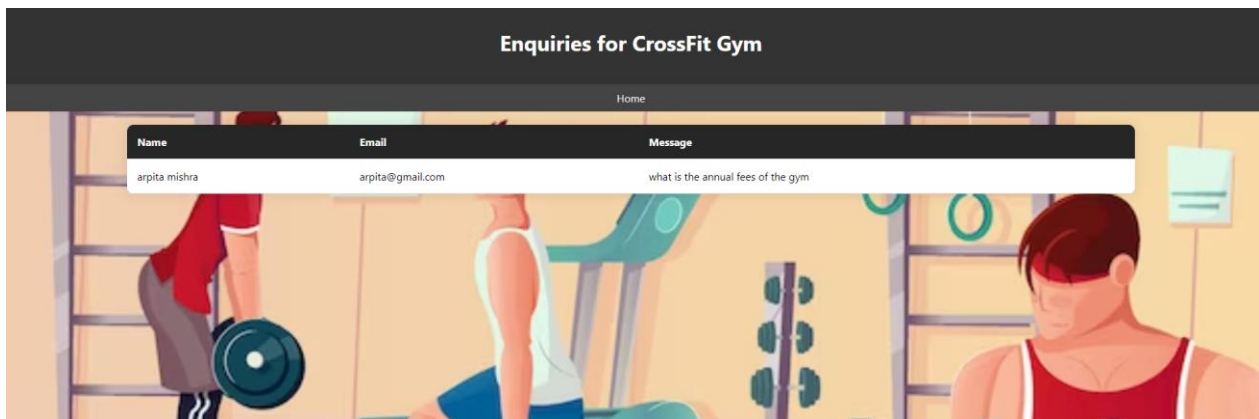
View Receptionist:

Gym Management System Home Member Trainer Receptionist

Search Here...

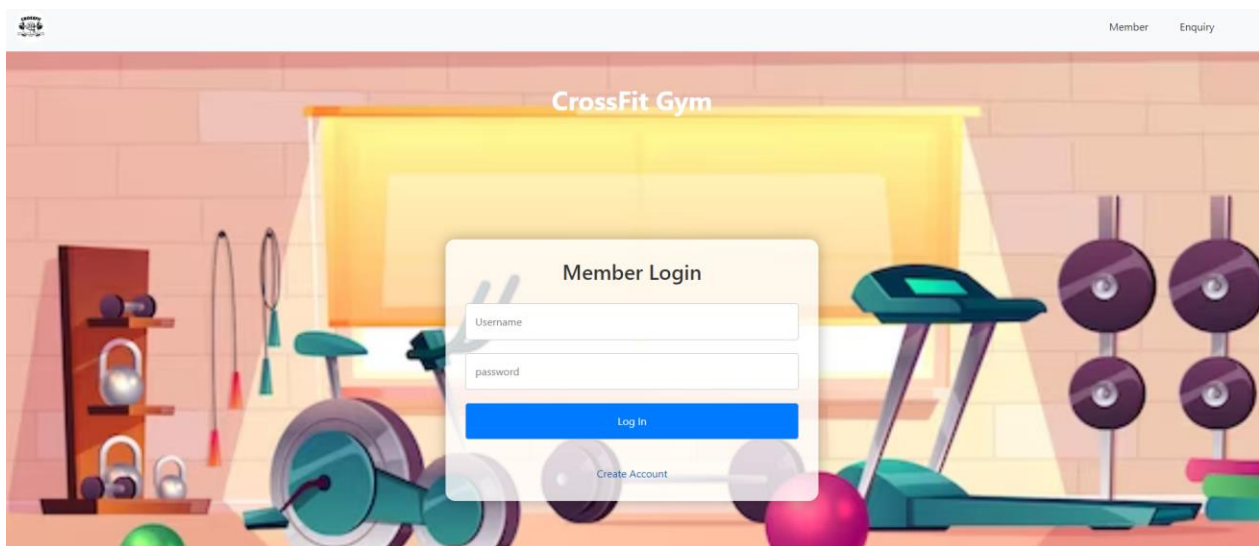
FIRST NAME	LAST NAME	MOBILE	DATE	ACTION
Akash	mishra	09876540393	20-02-2024 14:41:18	Update Delete
mayuri	kale	9873453623	20-02-2024 15:02:34	Update Delete

Enquiries:

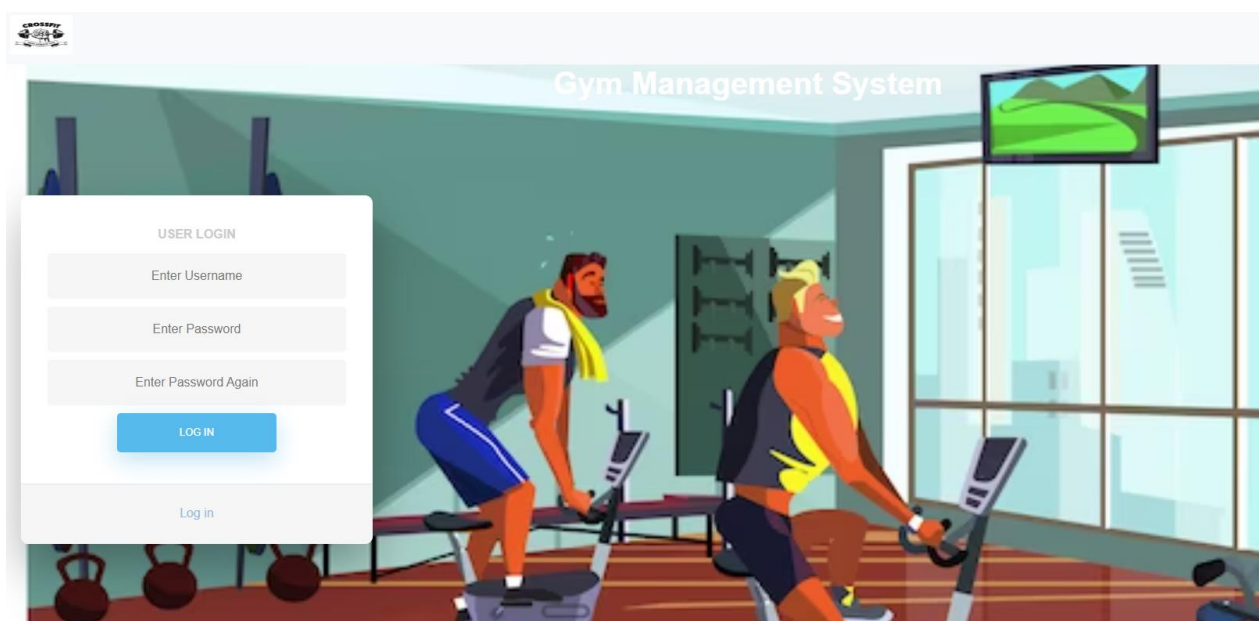


Member:

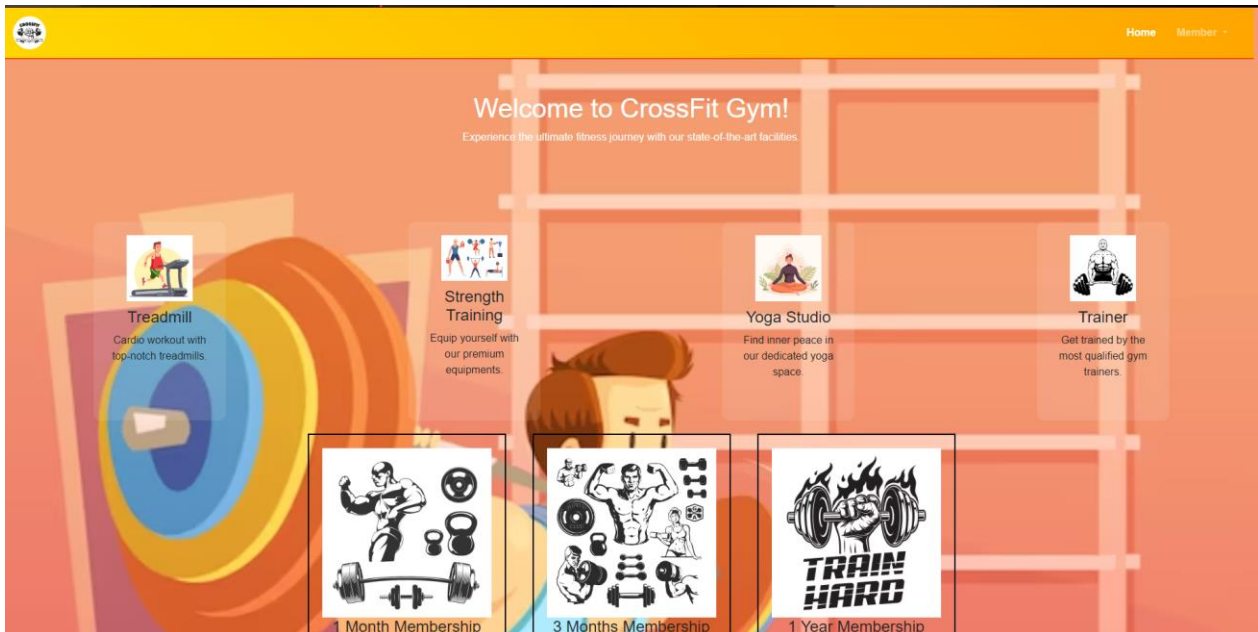
Member login page:



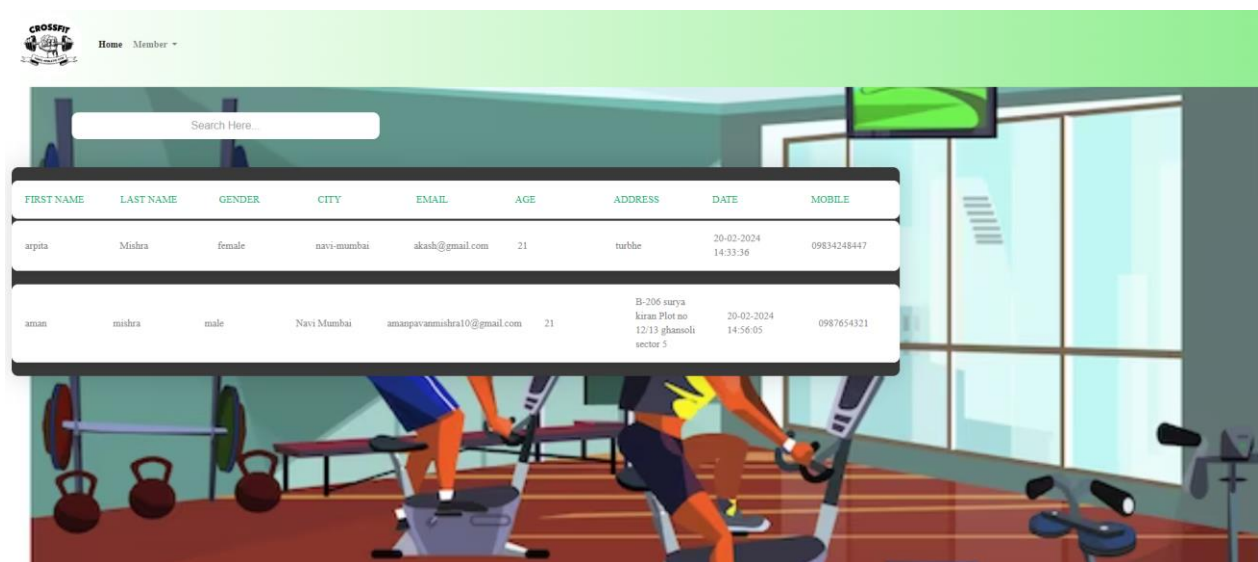
Member account creation page:



Member dashboard:



Member list page:



Database :

Member:

```
SELECT * FROM member LIMIT 100;
```

frame	lname	gender	city	email	age	address	date	mobile
arpita	Mishra	female	navi-mumbai	akash@gmail.com	21	turbhe	20-02-2024 14:33:36	09834248447
aman	mishra	male	Navi Mumbai	amanpavammishra10@gmail.com	21	B-206 surya kiran Plot no 12/13 ghansoli sector 5	20-02-2024 14:56:05	0987654321

Trainer:

```
SELECT * FROM trainer LIMIT 100;
```

#	id	fname	lname	gender	mobile	city	email	age	address
1	1	sarath	mishra	male	0987654321	Navi Mumbai	test@gmail.com	31	B-206 vashi s

Contact:

SELECT * FROM contact LIMIT 100;		
ECT * FROM contact LIM... X		
Max. rows: 100 Fetched Rows: 1		
name	email	query
arpita mishra	arpita@gmail.com	what is the annual fees of the gym

Receptionist:

SELECT * FROM recp LIMIT 100;			
ECT * FROM recp LIMIT ... X			
Max. rows: 100 Fetched Rows: 2			
fname	lname	mobile	date
Akash	mishra	09876540393	20-02-2024 14:41:18
mayuri	kale	9873453623	20-02-2024 15:02:34

Chapter – 7 (Coding/Implementation)

7.1 CODING

Index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Gym Management System</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      margin: 0;

      padding: 0;

      background-image: url('img/gym5.jpg');

      background-size: cover;

    }

    header {

      background-color: #333;

      color: #fff;

      text-align: center;

      padding: 20px; }

    nav {

      display: flex;

      justify-content: space-around;

      background-color: #444;

      padding: 10px;

    }
```



```
nav a {
    color: #fff;
    text-decoration: none;
}
nav a:hover {
    text-decoration: underline;
}
.container {
    max-width: 800px;
    margin: 20px auto;
    display: flex;
    justify-content: space-between;
}
.box {
    width: 30%;
    padding: 20px;
    background-color: #f0f0f0;
    margin: 10px;
    border-radius: 10px;
    transition: background-color 0.3s ease-in-out;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    overflow: hidden;
}
.box:hover {
    background-color: #ddd;
}
.box img {
    max-width: 100%;
    height: auto;
    border-radius: 5px;
    margin-bottom: 10px;
```

```

    }
</style>
</head>
<body>
    <header>
        <h1>CrossFit Gym </h1>
    </header>
    <nav>
        <a href="index.jsp">Member Login</a>
        <a href="about.jsp">About Us</a>
        <a href="contact.jsp">Contact Us</a>
        <a href="adminLogin.jsp">Admin Login</a>
    </nav>
    <div class="container">
        <div class="box">
             <!-- Add your image path for Box 1 -->
            <h2>Premium Equipments</h2>
            <p>Get the premium equipments in your hand .</p>
        </div>
        <div class="box">
             <!-- Add your image path
for Box 2 -->
            <h2>Unisex Gym</h2>
            <p>Our facility is dedicated to providing a dynamic and inclusive fitness experience for
individuals of all fitness levels and genders.</p>
        </div>
        <div class="box">
             <!-- Add your image path for Box 3 -->
            <h2>Offers</h2>
            <p>Special offers and promotions.</p>
        </div>

```

</div>

</body>

</html>

Database connection.java

```
package Database;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```

```
public class DatabaseConnection {
```

```
    public static Connection initializeDatabase()
```

```
        throws SQLException, ClassNotFoundException
```

```
    {
```

```
        String dbDriver = "com.mysql.jdbc.Driver";
```

```
        String dbURL = "jdbc:mysql://localhost:3306/";
```

```
        String dbName = "gym";
```

```
        String dbUsername = "root";
```

```
        String dbPassword = "root";
```

```
        Class.forName(dbDriver);
```

```
        Connection con =
```

```
        DriverManager.getConnection(dbURL+dbName,dbUsername,dbPassword);
```

```
        return con;
```

```
    }
```

```
}
```

AdminHome.jsp:

```
<% @page import="java.sql.Connection"%>
```

```
<% @page import="java.sql.ResultSet"%>
```

```

<% @page import="java.sql.Statement"%>
<% @page import="Database.DatabaseConnection"%>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Admin Home</title>
    <link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css">
    <script
      src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
    <link
      href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
      rel="stylesheet" id="bootstrap-css">
    <script
      src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <script
      src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

    <style>
      body {
        background-image: url("img/gym7.jpg");
        background-size: cover;

```

```
background-color: #cccccc;
height: 100%;
}

nav {
background: linear-gradient(45deg, #FFD700, #FFA500);
border-bottom: 2px solid #FF4500;
}

nav .navbar-brand img {
border-radius: 50%;
}

.navbar-toggler-icon {
background-color: #FF4500;
}

.navbar-nav {
margin-right: 70px;
}

.navbar-nav a {
color: black;
}

.card-container {
margin-top: 20px;
display: flex;
justify-content: space-around;
}

.card {
```

```
width: 200px;
height: 300px;
overflow: hidden;
border-radius: 15px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
transition: transform 0.3s ease-in-out;
}
```

```
.card:hover {
  transform: scale(1.1);
}
```

```
.card img {
  width: 100%;
  height: 50%;
  object-fit: cover;
  border-radius: 15px 15px 0 0;
}
```

```
.card-body {
  padding: 20px;
  background-color: #343a40;
  color: white;
  border-radius: 0 0 15px 15px;
}
```

```
.card-title {
  font-size: 18px;
  margin-bottom: 10px;
}
```

```

.card-link {
    color: #FF4500;
    font-weight: bold;
    text-decoration: none;
}

.card-link:hover {
    text-decoration: underline;
}
</style>
</head>
<body>
    <%
        Connection con = null;
    %>

    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a href="#" class="navbar-brand"> 

        </a>

        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">

            <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <h3>
                <b>Gym Management System</b>
            </h3>
            <ul class="navbar-nav ml-auto" style="margin-right: 70px;">

```

```

<li class="nav-item active">
    <a class="nav-link" href="index.html">Home <span class="sr-
only">(current)</span></a>
</li>

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        Member
    </a>
    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
        <a class="dropdown-item" href="addMember.jsp">Add Member</a>
        <a class="dropdown-item" href="adminMemberList.jsp">Member List</a>
    </div>
</li>

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        Trainer
    </a>
    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
        <a class="dropdown-item" href="addTrainer.jsp">Add Trainer</a>
        <a class="dropdown-item" href="adminTrainerList.jsp">View Trainer</a>
    </div>
</li>

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        Receptionist
    </a>
    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
        <a class="dropdown-item" href="addRecp.jsp">Add Receptionist</a>

```



```

        <a class="dropdown-item" href="adminRecpList.jsp">View Receptionist</a>
    </div>
</li>

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        Enquiry
    </a>
    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
        <a class="dropdown-item" href="enquiry.jsp">View Enquiry</a>
    </div>
</li>
</ul>
</div>
</nav>
<%
    try {
        con = DatabaseConnection.initializeDatabase();
        Statement st = (Statement) con.createStatement();
        String query = "select count(*) from member";
        ResultSet rs = st.executeQuery(query);
        while (rs.next()) {
            int member = rs.getInt(1);
        }
    }
    %>
<div class="card-container">
<div class="card card-inverse card-success">
    
    <div class="card-body">
        <h5 class="card-title">Members</h5>
        <p class="card-text">Total Members: <%= member %></p>

```

```

        <a href="adminMemberList.jsp" class="card-link">View Members</a>
    </div>
</div>

<%
    }
    con.close();
} catch (Exception e) {
    e.printStackTrace();
}
%>

<%
    try {
        con = DatabaseConnection.initializeDatabase();
        Statement st = (Statement) con.createStatement();
        String query = "select count(*) from trainer";
        ResultSet rs = st.executeQuery(query);
        while (rs.next()) {
            int trainer = rs.getInt(1);
        }
    %>

    <div class="card card-inverse card-danger">
    
    <div class="card-body">
        <h5 class="card-title">Trainers</h5>
        <p class="card-text">Total Trainers: <%= trainer %></p>
        <a href="adminTrainerList.jsp" class="card-link">View Trainers</a>
    </div>
</div>

<%
    }
    con.close();
} catch (Exception e) {

```

```

        e.printStackTrace();
    }
%>
<%
    try {
        con = DatabaseConnection.initializeDatabase();
        Statement st = (Statement) con.createStatement();
        String query = "select count(*) from recp";
        ResultSet rs = st.executeQuery(query);
        while (rs.next()) {
            int recp = rs.getInt(1);
%>
            <div class="card card-inverse card-info">

<div class="card-body">
    <h5 class="card-title">Receptionists</h5>
    <p class="card-text">Total Receptionists: <%= recp %></p>
    <a href="adminRecpList.jsp" class="card-link">View Receptionists</a>
</div>
</div>
</div>
<%
    }
    con.close();
} catch (Exception e) {
    e.printStackTrace();
}
%>
</body>
</html>

```

UserHome.jsp:

```
<!DOCTYPE html>
```

```

<html lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>User Home</title>
  <link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
  <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet"
id="bootstrap-css">
  <script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
  <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>Gym Management System</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <style>
    body {
      background-image: url("img/gym2.jpg");
      background-size: cover;
      background-color: #cccccc;
      margin: 0;
      font-family: 'Arial', sans-serif;
    }
    nav {
      background: linear-gradient(45deg, #FFD700, #FFA500);
      margin-right: 10px;

```

```
        border-bottom: 2px solid #FF4500;
    }
    nav .navbar-brand img {
        border-radius: 50%;
    }
    .navbar-toggler-icon {
        background-color: #FF4500;
    }
    .navbar-nav .nav-item {
        margin-right: 15px;
    }
    .nav-link {
        color: black;
        font-weight: bold;
    }
    .nav-link:hover {
        color: #FF4500 !important;
    }
    .dropdown-menu {
        background: linear-gradient(45deg, #FFD700, #FFA500);
        border: 2px solid #FF4500;
    }
    .dropdown-item {
        color: black;
        font-weight: bold;
    }
    .dropdown-item:hover {
        background-color: #FF4500;
        color: white !important;
    }
    .gym-details {
```

```

padding: 50px;
text-align: center;
color: white;
}
.facilities-section {
display: flex;
justify-content: space-around;
margin-top: 50px;
}
.facility {
height: 300px;
width: 200px;
text-align: center;
padding: 20px;
border-radius: 10px;
background: rgba(255, 255, 255, 0.1);
position: relative;
overflow: hidden;
transform: perspective(1000px);
transition: transform 0.5s;
}
.facility:hover {
transform: perspective(1000px) rotateX(40deg) rotateY(20deg);
}
.facility img {
max-width: 100px;
max-height: 100px;
margin-bottom: 10px;
}
.membership-box-section {
display: flex;

```

```
    flex-wrap: wrap;
    justify-content: center;
}
.membership-box {
    flex: 0 0 300px;
    border: 2px solid #000;
    padding: 20px;
    margin: 20px;
    text-align: center;
}
.membership-box img {
    max-width: 100%;
}
.membership-box h3 {
    margin-top: 10px;
}
.membership-box p {
    margin-top: 5px;
}
.facilities-list {
    text-align: left;
    margin-top: 15px;
}
.facilities-list ul {
    list-style-type: none;
    padding: 0;
}
.facilities-list li {
    margin-bottom: 5px;
}
.button-container {
```

```

        margin-top: 10px;
    }
    .button-container a {
        display: inline-block;
        padding: 8px 16px;
        background-color: #4CAF50;
        color: #fff;
        text-decoration: none;
        border-radius: 5px;
    }
</style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark">
        <a href="#" class="navbar-brand">
            
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="index.html">Home <span class="sr-
only">(current)</span></a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" id="memberDropdown"
role="button" data-toggle="dropdown">

```



```

        aria-haspopup="true" aria-expanded="false">
        Member
    </a>

    <div class="dropdown-menu" aria-labelledby="memberDropdown">
        <a class="dropdown-item" href="addMember.jsp">Add Member</a>
        <a class="dropdown-item" href="listMember.jsp">Member List</a>
    </div>
</li>
</ul>
</div>
</nav>
<div class="gym-details">
    <h1>Welcome to CrossFit Gym!</h1>
    <p>Experience the ultimate fitness journey with our state-of-the-art facilities.</p>
</div>
<!-- Facilities section -->
<div class="facilities-section">
    <div class="facility">
        
        <h4>Treadmill</h4>
        <p>Cardio workout with top-notch treadmills.</p>
    </div>
    <div class="facility">
        
        <h4>Strength Training</h4>
        <p>Equip yourself with our premium equipments.</p>
    </div>
    <div class="facility">
        
        <h4>Yoga Studio</h4>
        <p>Find inner peace in our dedicated yoga space.</p>
    </div>

```

```
</div>

<div class="facility">

  <h4>Trainer</h4>

  <p>Get trained by the most qualified gym trainers.</p>

</div>

</div>
```

```
<!-- Membership plan boxes -->

<div class="membership-box-section">

  <div class="membership-box">

    <h4>1 Month Membership</h4>

    <div class="facilities-list">

      <h5>Facilities:</h5>

      <ul>

        <li>Access to gym equipment</li>

        <li>Group fitness classes</li>

        <li>Personalized training sessions</li>

        <!-- Add more facilities as needed -->

      </ul>

    </div>

  </div>

  <div class="button-container">

    <a href="addMember.jsp">Get Membership</a>

  </div>

</div>

<div class="membership-box">

  <h4>3 Months Membership</h4>

  <div class="facilities-list">
```

```

    <h5>Facilities:</h5>

    <ul>

        <li>Access to gym equipment</li>

        <li>Group fitness classes</li>

        <li>Nutritional counseling</li>

        <!-- Add more facilities as needed -->

    </ul>

</div>

<div class="button-container">

    <a href="addMember.jsp">Get Membership</a>

</div>

</div>

<div class="membership-box">

    <h4>1 Year Membership</h4>

    <div class="facilities-list">

        <h5>Facilities:</h5>

        <ul>

            <li>Access to gym equipment</li>

            <li>Group fitness classes</li>

            <li>Exclusive events and workshops</li>

            <!-- Add more facilities as needed -->

        </ul>

    </div>

    <div class="button-container">

        <a href="addMember.jsp">Get Membership</a>

    </div>

</div>

</div>

</body>

```

</html>

addMember.jsp:

<% @page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>User Home</title>

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

<link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

<script

src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script

src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>

<script

src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

<link

href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"

rel="stylesheet" id="bootstrap-css">

<script

src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

<script

src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<link rel="stylesheet" type="text/css" href="css/adddataform.css">

<link rel="stylesheet" type="text/css" href="css/adddatafrm1.css">

<style>

body {

background-image: url("img/gym7.jpg");

```

background-size: cover;

background-color: #cccccc;

height: 100vh; /* Full viewport height */

perspective: 1000px; /* Adjust the perspective value for more or less depth */
}

</style>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

  <a href="#" class="navbar-brand"> 

  </a>

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">

    <span class="navbar-toggler-icon"></span>

  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">

    <ul class="navbar-nav mr-auto">

      <li class="nav-item active">

        <a class="nav-link" href="index.html">Home <span class="sr-
only">(current)</span></a>

      </li>

      <li class="nav-item dropdown">

        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

          Member

        </a>

        <div class="dropdown-menu" aria-labelledby="navbarDropdown">

          <a class="dropdown-item" href="addMember.jsp">Add Member</a>

          <a class="dropdown-item" href="listMember.jsp">Member List</a>

        </div>

```

```

        </li>
    </div>
</nav>
<div class="container-contact100">
    <div class="wrap-contact100">
        <div class="contact100-form-title" style="background-image: url(img/gym7.jpg);">
            <span class="contact100-form-title-1">
                Member Registration Form
            </span>
        </div>
        <form class="contact100-form validate-form"
action="<%=request.getContextPath()%>/AddMember" method="post">
            <div class="wrap-input100 validate-input" data-validate="First Name is required">
                <span class="label-input100">First_Name:</span>
                <input class="input100" type="text" name="fname" placeholder="Enter First
name">
                <span class="focus-input100"></span>
            </div>
            <div class="wrap-input100 validate-input" data-validate="Last Name is required">
                <span class="label-input100">Last_Name:</span>
                <input class="input100" type="text" name="lname" placeholder="Enter Last
name">
                <span class="focus-input100"></span>
            </div>
            <div class="wrap-input100 validate-input" data-validate = "gender is required">
                <span class="label-input100">Gender:</span>
                <input class="input100" type="text" name="gender" placeholder="Enter
Gender">
                <span class="focus-input100"></span>
            </div>
            <div class="wrap-input100 validate-input" data-validate="Phone is required">
                <span class="label-input100">Phone:</span>

```

```

number">
    <input class="input100" type="text" name="Mobile" placeholder="Enter phone
number">
    <span class="focus-input100"></span>
</div>
<div class="wrap-input100 validate-input" data-validate="city is required">
    <span class="label-input100">city</span>
    <input class="input100" type="text" name="city" placeholder="Enter city">
    <span class="focus-input100"></span>
</div>
<div class="wrap-input100 validate-input" data-validate="Valid email is required:
ex@abc.xyz">
    <span class="label-input100">Email:</span>
    <input class="input100" type="text" name="email" placeholder="Enter email">
    <span class="focus-input100"></span>
</div>
<div class="wrap-input100 validate-input" data-validate="Age is required">
    <span class="label-input100">Age:</span>
    <input class="input100" type="text" name="age" placeholder="Enter Age">
    <span class="focus-input100"></span>
</div>
<div class="wrap-input100 validate-input" data-validate="Address is required">
    <span class="label-input100">Address</span>
    <input class="input100" type="text" name="address" placeholder="Enter
Address">
    <span class="focus-input100"></span>
</div>
<div class="container-contact100-form-btn">
    <button class="contact100-form-btn">
        <span>
            Submit
            <i class="fa fa-long-arrow-right m-l-7" aria-hidden="true"></i>
        </span>
    </button>

```

```

        </button>

    </div>

</form>

</div>

</div>

<div id="dropDownSelect1"></div>

<script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAKFWBqlKAGCeS1rMVoaNlwyayu
0e0YRes"></script>

<script src="js/map-custom.js"></script>

<script src="js/main.js"></script>

<script async src="https://www.googletagmanager.com/gtag/js?id=UA-23581568-
13"></script>

<script>

    window.dataLayer = window.dataLayer || [];

    function gtag() {

        dataLayer.push(arguments);

    }

    gtag('js', new Date());

    gtag('config', 'UA-23581568-13');

</script>

</body>

</html>

```

UpdateMember.jsp:

```

<%--

    Document   : updatePatient

    Created on : 19 Aug, 2020, 5:19:02 PM

    Author    : Admin

--%>

<% @page import="java.sql.Statement"%>

<% @page import="java.sql.ResultSet"%>

```



```

<% @page import="java.sql.PreparedStatement"%>
<% @page import="Database.DatabaseConnection"%>
<% @page import="java.sql.Connection"%>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Update Member</title>
    <link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css">
    <script
      src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
    <link
      href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
      rel="stylesheet" id="bootstrap-css">
    <script
      src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <script
      src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <link rel="stylesheet" type="text/css" href="css/adddataform.css">
    <link rel="stylesheet" type="text/css" href="css/adddatafrm1.css">
    <style>
      body {

```

```

        background-image: url("img/gym11.jpg");
        background-size: cover;
        background-color: #cccccc;
    }
</style>
</head>
<body>

<%
    String mob = request.getParameter("mob");
    Connection con = DatabaseConnection.initializeDatabase();
    String s = "SELECT * FROM member WHERE mobile = '"+mob+"' ";
    PreparedStatement pstmt = con.prepareStatement(s);
    ResultSet rs = pstmt.executeQuery();
    while (rs.next()) {
%>
<div class="container-contact100">

<div class="wrap-contact100">
    <div class="contact100-form-title" style="background-image: url(img/gym6.jpg);">
        <span class="contact100-form-title-1">
            Member Registration Form
        </span>
    </div>

    <form class="contact100-form validate-form"
action="<%=request.getContextPath()%>/updateMember" method="post">
        <div class="wrap-input100 validate-input" data-validate="First Name is required">
            <span class="label-input100">First_Name:</span>

```

```
        <input class="input100" type="text" value="<%= rs.getString(1)%>"
name="fname" placeholder="Enter First name">
```

```
        <span class="focus-input100"></span>
```

```
    </div>
```

```
    <div class="wrap-input100 validate-input" data-validate="Last Name is required">
```

```
        <span class="label-input100">Last_Name:</span>
```

```
        <input class="input100" type="text" value="<%= rs.getString(2)%>"
name="lname" placeholder="Enter Last name">
```

```
        <span class="focus-input100"></span>
```

```
    </div>
```

```
    <div class="wrap-input100 validate-input" data-validate = "gender is required">
```

```
        <span class="label-input100">Gender:</span>
```

```
        <input class="input100" type="text" value="<%= rs.getString(3)%>"
name="gender" placeholder="Enter Gender">
```

```
        <span class="focus-input100"></span>
```

```
    </div>
```

```
    <div class="wrap-input100 validate-input" data-validate="Phone is required">
```

```
        <span class="label-input100">Phone:</span>
```

```
        <input class="input100" type="text" value="<%= rs.getString(9)%>"
name="Mobile" placeholder="Enter phone number">
```

```
        <span class="focus-input100"></span>
```

```
    </div>
```

```
    <div class="wrap-input100 validate-input" data-validate="Package is required">
```

```
        <span class="label-input100">City</span>
```

```
        <input class="input100" type="text" value="<%= rs.getString(4)%>" name="city"
placeholder="Enter city">
```

```
        <span class="focus-input100"></span>
```

```
    </div>
```

```
<div class="wrap-input100 validate-input" data-validate="Valid email is required:
ex@abc.xyz">
```

```
<span class="label-input100">Email:</span>
```

```
<input class="input100" type="text" value="<%= rs.getString(5)%>"
name="email" placeholder="Enter email">
```

```
<span class="focus-input100"></span>
```

```
</div>
```

```
<div class="wrap-input100 validate-input" data-validate="Age is required">
```

```
<span class="label-input100">Age:</span>
```

```
<input class="input100" type="text" value="<%= rs.getString(6)%>" name="age"
placeholder="Enter Age">
```

```
<span class="focus-input100"></span>
```

```
</div>
```

```
<div class="wrap-input100 validate-input" data-validate="Address is required">
```

```
<span class="label-input100">Address</span>
```

```
<input class="input100" type="text" value="<%= rs.getString(7)%>"
name="address" placeholder="Enter Address">
```

```
<span class="focus-input100"></span>
```

```
</div>
```

```
<div class="container-contact100-form-btn">
```

```
<button class="contact100-form-btn">
```

```
<span>
```

```
Submit
```

```
<i class="fa fa-long-arrow-right m-l-7" aria-hidden="true"></i>
```

```
</span>
```

```
</button>
```

```
</div>
```

```
</form>
```

```
</div>
```

```
</div>

<%
    }
%>

</body>

</html>
```

deleteMember.jsp

```
<%--
```

Document : deleteMember

Created on : 19 Aug, 2020, 5:52:13 PM

Author : Admin

```
--%>
```

```
<% @page import="java.sql.Statement"%>
<% @page import="java.sql.Connection"%>
<% @page import="Database.DatabaseConnection"%>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Delete Member</title>

</head>

<body>

<%

    String mob = request.getParameter("mob");

    Connection con = null;

    Statement stmt = null;

    con = DatabaseConnection.initializeDatabase();

    stmt = (Statement) con.createStatement();

    String query = "delete from member " + "where mobile = '"+mob+"'";
```

```

        stmt.executeUpdate(query);

        con.close();

        RequestDispatcher rd = request.getRequestDispatcher("AdminHome.jsp");
        rd.forward(request, response);

    %>

</body>

</html>

```

AddMember.java

```

package Controller;

import Database.DatabaseConnection;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;

@WebServlet("/AddMember")

public class AddMember extends HttpServlet {

    private int i;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)

```

```

        throws ServletException, IOException {
    PrintWriter pw = response.getWriter();
    try {
        Date todaysDate = new Date();
        DateFormat df2 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
        String fname = request.getParameter("fname");
        String lname = request.getParameter("lname");
        String gender = request.getParameter("gender");
        String phone = request.getParameter("Mobile");
        String city = request.getParameter("city");
        String email = request.getParameter("email");
        String age = request.getParameter("age");
        String address = request.getParameter("address");
        String DateAndTime = df2.format(todaysDate);
        Connection con = DatabaseConnection.initializeDatabase();
        PreparedStatement pst = con.prepareStatement("insert into member
values(?,?,?,?,?,?,?,?,?)");
        pst.setString(9, phone);
        pst.setString(1, fname);
        pst.setString(2, lname);
        pst.setString(3, gender);
        pst.setString(4, city);
        pst.setString(5, email);
        pst.setString(6, age);
        pst.setString(7, address);
        pst.setString(8, DateAndTime);
        i = pst.executeUpdate();
        if (i > 0) {
            pw.println("<script type=\"text/javascript\">");
            pw.println("alert('Added Data Successfully..!');");
            pw.println("window.location.href = \"listMember.jsp\";");
        }
    }
}

```

```

        pw.println("</script>");    } else {
        pw.println("<script type=\"text/javascript\">");
        pw.println("alert('Incorrect Data..!');");
        pw.println("window.location.href = \"addMember.jsp\";");
        pw.println("</script>"); }
    } catch (SQLException ex) {
        Logger.getLogger(AddMember.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(AddMember.class.getName()).log(Level.SEVERE, null, ex);
    } } }

```

UpdateMember.java

```

package Controller;

import Database.DatabaseConnection;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/updateMember")

public class UpdateMember extends HttpServlet {

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

        PrintWriter pw = response.getWriter();

        String fname = request.getParameter("fname");
        String lname = request.getParameter("lname");

```



```

String gender = request.getParameter("gender");
String phone = request.getParameter("Mobile");
String city = request.getParameter("city");
String email = request.getParameter("email");
String age = request.getParameter("age");
String address = request.getParameter("address");
try {
    Connection con = DatabaseConnection.initializeDatabase();

    PreparedStatement pst = con.prepareStatement("update member set fname = ? , lname = ?
, gender = ? , package = ? , email = ? , age = ? , address = ? where mobile = '" + phone + "'");

    pst.setString(1, fname);
    pst.setString(2, lname);
    pst.setString(3, gender);
    pst.setString(4, city);
    pst.setString(5, email);
    pst.setString(6, age);
    pst.setString(7, address);
    int i = pst.executeUpdate();
    if (i > 0) {
        pw.println("<script type='\"text/javascript\"'>");
        pw.println("alert('Update Successfully..!');");
        pw.println("window.location.href = '\"AdminHome.jsp\"';");
        pw.println("</script>");
    } else {
        pw.println("<script type='\"text/javascript\"'>");
        pw.println("alert('Failed..! Try Again Later...');");
        pw.println("window.location.href = '\"updateMember.jsp\"';");
        pw.println("</script>");    }
    con.close();
} catch (Exception e) {
} }}

```

CHAPTER – 8 (TESTING)

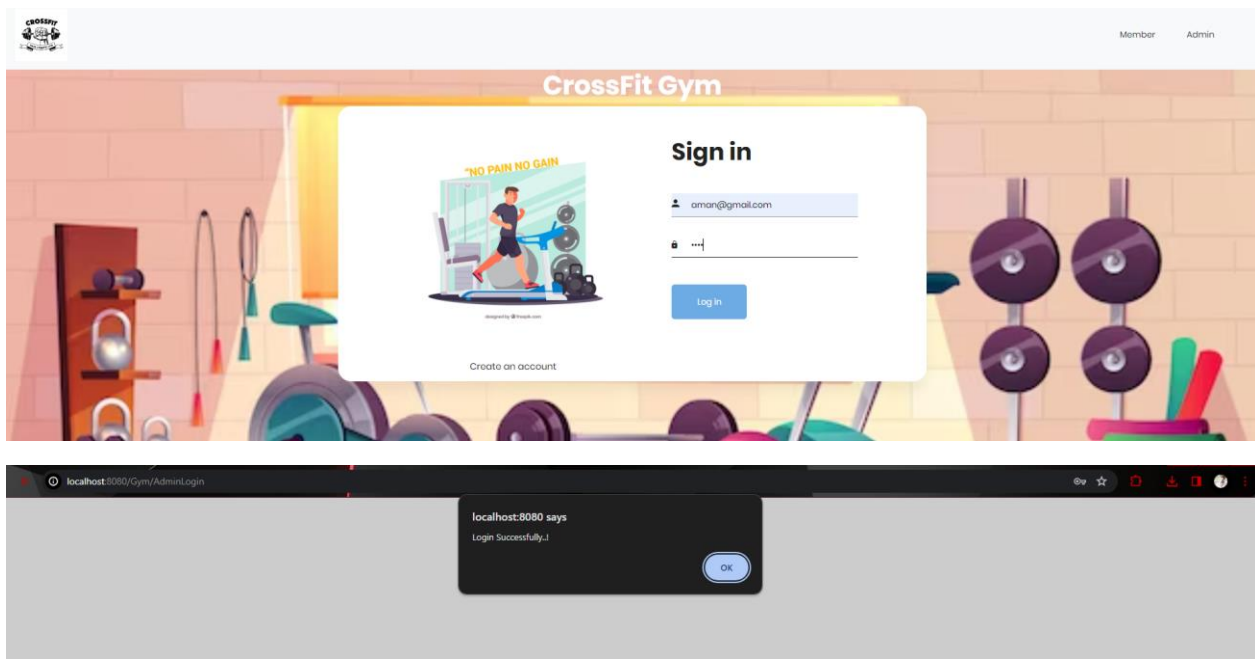
8.1 TEST CASE

Test Scenario ID:	1					
Tester Name	Aman Mishra					
Test Page	admin login(crossfit gym)					
Date	31/01/2024					
TestID	Test Scenario	Test Cases	Test Data	Actual output	Excepted Output	Status
1	Enter correct email Enter correct password	1.open website 2.open login page 3.Enter email id 4.Enter password 5.click on login button	http://localhost:8080/Gym/adminLogin.jsp admin login aman@gmail.com aman	login successfully	login successfully	Pass
2	Enter correct email Enter incorrect password	1.open website 2.open login page 3.Enter email id 4.enter password	http://localhost:8080/Gym/adminLogin.jsp admin login aman@gmail.com aaaaam	email or password is incorrect	login successfully	Fail
3	Enter InCorrect Email Enter Correct password	1.open website 2.open login page 3.Enter email id 4.Enter Password	http://localhost:8080/Gym/adminLogin.jsp admin login admin@gmail.com aman	email or password is incorrect	login successfully	Fail
4	Enter incorrect Email Enter incorrect password	1.open website 2.open login page 3.Enter email Id 4.enter password	http://localhost:8080/Gym/adminLogin.jsp admin login Admin@gmail.com aa21	email or password is incorrect	login successfully	Fail

8.2 TESTING OUTPUTS

1.Admin login page

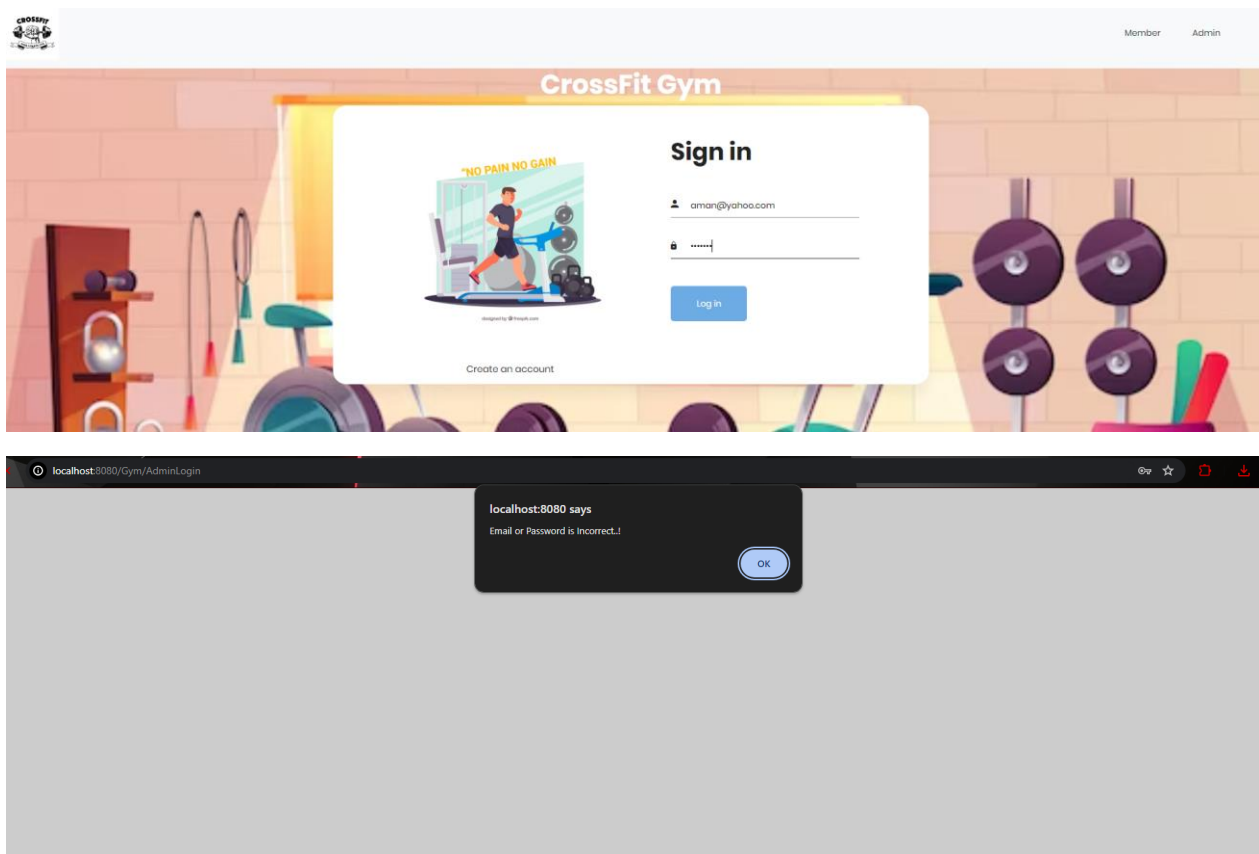
Correct mail and password



Running 'test1'

1. open on /Gym/adminLogin.jsp OK
 2. setWindowSize on 984x1023 OK
 3. click on css=.signin-form OK
 4. click on id=your_name OK
 5. type on id=your_name with value aman@gmail.com OK
 6. click on id=your_pass OK
 7. type on id=your_pass with value aman OK
 8. click on id=signin OK
- 'test1' completed successfully

Incorrect mail and correct password



Running 'test 2'

1. open on /Gym/adminLogin.jsp OK
2. setWindowSize on 984x1023 OK
3. click on id=your_name OK
4. type on id=your_name with value aman@yahoo.com OK
5. click on id=your_pass OK
6. type on id=your_pass with value aman OK
7. click on id=signin OK

'test 2' completed successfully

8.3 TESTING APPROACH

A testing approach is a systematic and structured method used to plan, design, and execute software testing activities. It is an essential part of the software development process that ensures the quality, reliability, and performance of the software product. Different approaches may be used based on the context, objectives, and constraints of the project.

There are several types of testing approaches, including:

Waterfall Testing: This approach involves sequential phases of development and testing. Each phase must be completed before moving on to the next phase. This is a traditional approach and is not very flexible.

Agile Testing: Agile testing is based on the principles of the Agile software development methodology. It is an iterative and incremental approach that emphasizes collaboration, flexibility, and responsiveness to change. Testing activities are integrated into the development process, and testing is performed continuously throughout the development cycle.

Risk-Based Testing: Risk-based testing is an approach that prioritizes testing activities based on the risk associated with different features or components of the software application. This approach helps focus testing efforts on critical areas of the application.

Model-Based Testing: Model-based testing is an approach that uses models to define the behavior of the software application and generate test cases automatically. Models can be used to represent the expected behavior of the software application, and test cases can be generated from these models automatically.

User Acceptance Testing: User acceptance testing is a testing approach that involves testing the software application from the end-user's perspective. This approach helps ensure that the software meets the requirements and expectations of the end-users.

Performance Testing: Performance testing is an approach that involves testing the performance of the software application under different load conditions. This approach helps identify performance bottlenecks and ensure that the software application can handle the expected load.

The choice of testing approach depends on various factors, including the type of software application being developed, the development methodology used, the project timeline and budget, and the testing goals and objectives.

8.4 UNIT TESTING

unit testing is a software testing technique where individual units or components of a program, typically the smallest testable parts, are tested in isolation from the rest of the application. It is performed during the development phase by developers to validate the correctness of each unit of code. Here are three advantages of unit testing:

Early Detection of Defects: Unit tests are written and executed during the development process, which helps identify defects and issues at an early stage. This allows developers to fix problems before they become more complex and costly to resolve.

Improved Code Quality: Writing unit tests forces developers to think more deeply about how their code should behave under different conditions. This results in better code quality as it encourages the use of good coding practices, such as modular and reusable code.

Regression Testing: Unit tests can be easily automated and run repeatedly whenever code changes are made. This ensures that new updates or changes do not introduce new bugs or break existing functionality, thereby providing a form of regression testing.

8.5 INTEGRATED TESTING

Integration testing is a software testing technique used to test the interaction between different software components or modules after they have been integrated. The main purpose of integration testing is to detect and resolve any issues that may arise as a result of integrating these components. Here are three advantages of integration testing:

- 1. Early Identification of Integration Problems:** Integration testing can help identify issues that arise when different components or modules are combined, such as incompatible interfaces, incorrect data exchanges, or missing functionality.
- 2. Improved Software Reliability:** By ensuring that different parts of the software work together as expected, integration testing helps improve the overall reliability and stability of the software.
- 3. Better Understanding of System Behavior:** Integration testing allows testers to gain a better understanding of how different components interact with each other and how the system as a whole behaves.

8.6 SECURITY

Access Control: Implement role-based access control for Admin and Member modules. Assign specific permissions to each role to ensure that users have access only to the functionalities necessary for their roles. Regularly review and update access permissions based on personnel changes.

Encryption: Use HTTPS to encrypt data transmitted between the client and the server, especially during login and data submission. Employ encryption algorithms to protect sensitive data stored in the database, such as user credentials and personal information.

Authentication and Authorization: Implement strong authentication mechanisms for both Admin and Member logins, including secure password policies and multi-factor authentication (MFA). Enforce fine-grained authorization rules to control access to specific functionalities within the system based on user roles.

Data Integrity: Use hashing algorithms and digital signatures to ensure the integrity of data stored in the MySQL database, preventing unauthorized modifications. Regularly validate and sanitize user inputs to prevent injection attacks and data tampering.

Secure Coding Practices: Adhere to secure coding practices, ensuring that the code is resilient against common vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS). Conduct regular code reviews and employ security testing techniques (static code analysis, dynamic application security testing) to identify and address vulnerabilities.

Data Backup and Recovery: Implement a robust backup strategy for the MySQL database, ensuring regular backups of critical data. Develop and test a disaster recovery plan to quickly restore functionality in case of system failures, natural disasters, or cyberattacks.

Network Security: Protect the network infrastructure with firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS). Consider implementing Virtual Private Network (VPN) for secure remote access, especially if there are remote admin functionalities.

CHAPTER – 9 (MAINTENANCE)

9.1 MODIFICATIONS AND IMPROVEMENTS

Certainly! Here are some detailed modifications and improvements you can consider for your CrossFit gym management system:

- 1. Enhanced User Authentication:** Implement a secure authentication system using hashing algorithms (e.g., bcrypt) to store and verify passwords. Introduce session management to ensure secure and persistent user sessions.
- 2. User Roles and Permissions:** Extend the user roles to include different levels of access (e.g., Super Admin, Trainer, Receptionist). Implement role-based access control (RBAC) to restrict certain functionalities based on user roles.
- 3. Responsive Design:** Ensure that your front-end is responsive and mobile-friendly using CSS media queries. Optimize the layout for various screen sizes, considering the prevalence of mobile devices.
- 4. Improved UI/UX:** Enhance the overall user interface by using modern design principles and frameworks. Implement a clean and intuitive dashboard for both admins and members.
- 5. Email Notifications:** Set up email notifications for important events, such as account creation, password resets, and membership renewals.
- 6. Trainer Management:** Include features for trainers to update their profiles, including qualifications, certifications, and schedules. Implement a system for assigning trainers to specific classes or sessions.
- 7. Membership Plans:** Introduce different membership plans with varying durations and features. Implement a system for members to upgrade or downgrade their membership plans.
- 8. Payment Integration:** Integrate a secure payment gateway to handle membership fees and other transactions. Provide an invoice generation system for members to view their payment history.
- 9. Attendance Tracking:** Implement a system for tracking member attendance in classes or training sessions. Allow trainers and admins to view attendance reports.
- 10. Enquiry Tracking:** Expand the enquiry system to track the status of enquiries, including follow-ups and conversions. Implement a feature for admins to analyze and generate reports on enquiry data.
- 11. Document Uploads:** Allow members to upload necessary documents such as health forms, waivers, or medical certificates. Implement a secure document storage system.
- 12. Improved Search and Filtering:** Enhance the search and filtering capabilities in the admin module for better data management. Implement sorting and filtering options for member lists and trainer schedules.
- 13. Data Validation and Error Handling:** Implement robust input validation on both client and server sides to prevent security vulnerabilities. Enhance error handling to provide meaningful and user-friendly error messages.

CHAPTER – 10

10.1 CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

Your CrossFit gym management system seems to cover the essential functionalities required for both Admin and Member modules. The use of JSP, CSS, HTML, Java, and MySQL for the backend provides a well-rounded technology stack for web application development. The system allows the Admin to manage trainers, members, receptionists, and enquiries efficiently. Members, on the other hand, have the ability to add and view their own information.

The CrossFit gym management system developed using JSP, CSS, HTML, Java, and MySQL for the backend, implemented through NetBeans, comprises two primary modules: Admin and User (Member). The Admin module enables administrators to log in, create an account, and exercise comprehensive control over the system. Admins can perform operations such as adding, updating, deleting, and viewing information related to trainers, members, and receptionists. Additionally, the admin module facilitates the management of member enquiries. On the other hand, the Member module provides users with the ability to add and view their personal information. This system streamlines the administrative tasks of the gym, allowing administrators to efficiently handle user data, while members can conveniently manage and access their own information within the platform. The integration of various technologies and the utilization of a robust database contribute to the overall effectiveness and functionality of the CrossFit gym management system. Stakeholders can refine and improve the system based on evolving needs and technological advancements.

Future Enhancements:

Payment and Subscription Management: Integrate a payment system to manage member subscriptions, track payments, and automate billing processes.

Workout Tracking: Implement a feature for members to log and track their workout routines, achievements, and progress over time.

Class Scheduling: Introduce a class scheduling system, allowing members to book classes and trainers to manage their schedules.

Messaging System: Implement a communication platform for members to interact with trainers, receive updates, and get personalized advice.

Attendance Tracking: Create a system to track member attendance in classes or gym sessions, providing insights for both members and trainers.

Mobile Application: Develop a mobile app for better accessibility, allowing members to manage their accounts, receive notifications, and interact with the gym community on the go.

Nutrition and Wellness Tracking: Include features for members to log their nutritional intake and wellness activities, creating a holistic health management system.

Multi-Language Support: Provide support for multiple languages to cater to a diverse member base.

CHAPTER – 11 (REFERENCES)

- <https://chat.openai.com/>
- <https://github.com/topics/gym-website>
- <https://www.youtube.com/>
- <https://www.tutorialspoint.com/index.html>
- <https://www.javatpoint.com/>
- <https://www.geeksforgeeks.org/css-tutorial/>
- <https://www.blackbox.ai/>