Table of Contents

[ABSTRACT 2](#_Toc151563491)

[CHAPTER 1 – INTRODUCTION 3](#_Toc151563492)

[1.1 System Specifications 5](#_Toc151563493)

[CHAPTER 2 – LITERATURE REVIEW 6](#_Toc151563494)

[2.1Existing Solution: 9](#_Toc151563495)

[2.2 Proposed Solution: 11](#_Toc151563496)

[CHAPTER 3 OVERALL DESCRIPTION OF THE PROPOSED SYSTEM 12](#_Toc151563497)

[3.1 Module Description 12](#_Toc151563498)

[CHAPTER 4 – DESIGN 15](#_Toc151563499)

[4.1UML Diagrams: 15](#_Toc151563500)

[**4.1.1Usecase Diagrams**: 16](#_Toc151563501)

[**4.1.2 Sequence Diagram:** 18](#_Toc151563502)

[**4.1.3 Collaboration Diagram:** 19](#_Toc151563503)

[4.1.5. Work Flow Diagram 22](#_Toc151563504)

[4.1.6. Class Diagram 22](#_Toc151563505)

[**4.1.7 Table Design:** 23](#_Toc151563506)

[**4.1.8 Architecture Diagram:** 24](#_Toc151563507)

[**4.1.9 ER Diagram:** 25](#_Toc151563508)

[CHAPTER 5 - OUTPUT SCREENSHOT 26](#_Toc151563509)

[CHAPTER 6 – IMPLEMENTATION DETAILS 27](#_Toc151563510)

[6.1 Introduction to Html Framework 27](#_Toc151563511)

[6.2 Cascading Style Sheets (CSS) 30](#_Toc151563512)

[6.3 MYSQL Server 35](#_Toc151563513)

[6.4PYTHON 36](#_Toc151563514)

[6.5 DJANGO 37](#_Toc151563515)

[CHAPTER 8-TECHNICAL FEASIBILITY 39](#_Toc151563516)

[CHAPTER 9-SYSTEM TESTING 41](#_Toc151563517)

[CHAPTER 11- REFERENCES 50](#_Toc151563518)

[CHAPTER 12- CODING 51](#_Toc151563519)

**GYM Management System using Django**

# ABSTRACT

In many Gyms,the payment receipts are in paper format. So it is very difficult for both gym members to keep all the paper receipts safely and to gym trainer to keep reminding for the fee receipts.Sometimes it creates a trouble when members lost their receipts.The other problem that can be faced by a gym owner is that if he/she wants to inform any message related to working or non working days of gym, manually sending message become difficult. The proposed system provide developing an django application to overcome some of these issues. So this project can be helpful for both gym owner as well as for gym members. In this application all receipts are store in a digital format, so there are no issues of loosing any confidencial receipts.This application will also notify the user(gym members) about their fees and also notifies the gym owner about the payment clearence.This application in future can be elaborated by providing supplement store , diet information, schedule.

# CHAPTER 1 – INTRODUCTION

The Gym Management requires a system that will handle all the necessary and minute details easily and proper database security accordingly to the user. They requires software, which will store dataabout members, employees, products, payroll, receipts of members etc & all transactions that occur in Gym and lock-up with graphical userinterface(GUI).

We have done a project on Gym Management and database management and transactions. This system is proposed to be an automate database management & transactions. This stores employee, member, receipts, and products information. It also provides the facility of search & advanced search for searching the records efficiently & immediately. This system provides data storing & report generation with graphical user interface (GUI).

We all know health is a wealth. We do not need a fancy car, big apartment, a doctor degree without a health. Being healthy is a first thing we need to keep in mind. Because most of time our attitude depends on how we feel. Being healthy and fit gives us energy todo anything. Physical fitness is very necessary for a healthy and tension free life. Physical fitnessincludes diet, exercise and sleep. These three basic things have their own importance in each individual's life and everyone should be sensible with regard to these for a healthy life.

It is always necessary to study and recognize the problems of existing system, which will help in finding out the requirements for the new system. System study helps in finding different alternatives for better solution.

The project study basically deals with different operations and steps

involved in generation of examination mark sheets. Ti includes:

1. Data gathering

2. Study of existing system

3. Analyzing problem

4. Studying various documents

5. Feasibility study for further improvements

Following are the steps taken during the initial study: Initially, we collected all the information, which they wanted to store. Then we studied the working of the current system which is done manually. We noted the limitation of that system which motivated them to have new system.

With the help of these documents we got basic ideas about the system as well as input output of the developed system. The most important thing is to study system thoroughly. Here we are studying both existing system and proposed system so that advantages & disadvantages of both the systems can be understood The first task was identifying how system can be computerized. Some analysis and projections was done regarding changes to be made to the existing system.

The new developed system for Gym Management is simple without complexities.

## System Specifications

**Hardware Requirements:-**

System : Dual Core

Hard Disk : 40 GB.

Monitor : 15 VGA Colour.

Mouse : Logitech.

RAM : 2 GB

Software Requirements: -

Operating System : Windows OS

Front-End : HTML, CSS, and JS

Back-End : Python 3.10 , MYSQL 5.0

Framework : Django 4.0

**Tool : XAMPP 3.2.1**

# CHAPTER 2 – LITERATURE REVIEW

As fitness becomes an integral part of modern lifestyles, the role of technology in managing gym operations and enhancing user experiences has gained significant attention. This literature review explores existing research and applications related to Gym Management Systems, particularly those developed using the Django framework.

**Integrated Management Systems:** The adoption of integrated management systems, such as those built on the Django framework, has been a focal point in recent research. These systems offer a comprehensive solution, encompassing user management, financial tracking, and communication channels. Researchers highlight the efficiency gained by integrating diverse functionalities into a unified platform (Marshall et al., 2019).

**User-Centric Features:** Studies emphasize the importance of user-centric features in Gym Management Systems. Personalized fitness programs, including diet plans and workout schedules, contribute to higher user engagement and satisfaction (Jones & Smith, 2020). The Django framework facilitates the development of dynamic and responsive interfaces, enhancing the user experience (Gupta et al., 2018).

**Security and Authentication:** Security considerations in Gym Management Systems are paramount. Studies highlight the significance of secure authentication mechanisms in protecting user data and maintaining privacy (Chen et al., 2017). Django's built-in security features are often commended for providing a robust foundation for such systems (Brown & White, 2019).

**Financial Tracking and Billing:** Effective financial management is a crucial aspect of gym operations. Researchers emphasize the importance of transparent billing systems and financial tracking tools (Robinson & Turner, 2018). Gym Management Systems developed with Django often integrate these features seamlessly, providing administrators with real-time insights into financial health (Kim & Lee, 2021).

**Communication Channels:** Communication between administrators, trainers, and members is vital for fostering a sense of community within a gym. In-depth studies have explored the impact of effective communication channels, such as in-app messaging and announcements, on member engagement and satisfaction (Wang et al., 2019).

**Mobile Responsiveness:** The increasing reliance on mobile devices has led to a growing emphasis on mobile responsiveness in Gym Management Systems. Research underscores the importance of providing users with seamless experiences across various devices (Li et al., 2020). Django's responsive design capabilities contribute to achieving this goal (Nguyen & Tran, 2018).

**Future Directions:** While existing research provides valuable insights, there is a need for continuous exploration and adaptation. Future directions may involve incorporating emerging technologies such as AI for personalized fitness recommendations or expanding the use of mobile technologies for enhanced user engagement.

As modernizing is taking over all the systems and digitalizing helps them improve in so many particular ways. The Gym Management System is one of the systems which helps the administration in speeding up the tasks at the same time reducing the complexity.

The purpose or objective of this system is to digitalize and create an automated system. The system will perform the task like adding the new member to the gym, Removing the member or keeping the payments records and other stuff required in managing the gym properly.

The present scenario in the gyms is that the records are kept by writing in a file on the paper. Every management task is done manually. This creates a system unreliable and confusing to keep the correct track of the records.

The maintenance of the system like this is hardly required until it needs to change any part of the system. The information about the various things contained in the system are like members, trainers, equipment can get by just a few clicks unlike the paper documents required the serious reading for such information.

It helps in creating the various batch according to their preference or if they want a particular trainer. It made easy to generate the reports of various operations performed in the gym are like paying the fee it can be stored and later evaluated and get the list of members who did not pay the fee.

It also helps the users in reducing the carbon footprint as the amount of paper used in company reduces.

This also helps in keeping the standard width of the management system as if there is a case where the administration involves more than one person to manage the gym.

This system does not only limit itself to the administration and but also helps the members of the gym. The members can have options like attendance and fee payment change batch request etc.

This will improve the transparency between the members which is always a good quality in the system. It will also give the layer of security to the administration and the users that only authorized users can access by their credentials.

Some major takeaways:

Online registration of the member

Issuing the timetable of batch on system

Increase transparency between stakeholders

Easy payment options

Standardization of the system

# 2.1Existing Solution:

In gym management system, after the planning and analysis phase of the system gets completed. Then the next phase required to transform the collected required system information into structural blueprint which will serve as a reference while constructing the working system.

It is a phase when most of the risks and error unveiled so it’s is good practices to take care of this thing from the start.

This is a fully fledged system which will be the backbone of the while management of the gym so ignoring the risk or error is not an option as later it can make a greater form of itself.

So, it is better to minimize the problems faced by both staff and the manager in the Organization. Let’s discuss the system in detail:

The gym is working manually. The current system is time consuming and also it is very costly, because it involves a lot of paperwork. To manually handle the system was very difficult task. But now-a-days

**Disadvantage of existing system:**

1. **Fragmented User Information:**
   * In many existing gym management systems, user information is often stored in fragmented databases, leading to inconsistencies and difficulties in updating records uniformly.
2. **Limited User Interaction:**
   * Traditional systems may lack robust communication channels, limiting interactions between members, trainers, and administrators. This hampers the creation of a supportive community within the gym.
3. **Outdated Technology Stack:**
   * Some legacy gym management systems may be built on outdated technology stacks, making them less adaptable to modern features, security enhancements, and responsive design requirements.
4. **Security Concerns:**
   * Security vulnerabilities are a common issue in older systems, which may lack the advanced security features present in more recent frameworks. This can pose risks to user data and privacy.
5. **Manual Billing Processes:**
   * Many existing systems rely on manual billing processes, which can be time-consuming and error-prone. This inefficiency may lead to billing discrepancies and challenges in tracking financial records accurately.
6. **Limited Personalization:**
   * Older systems may not provide the level of personalization expected by modern users. This limitation extends to workout plans, diet recommendations, and other personalized features that enhance the user experience.
7. **Poor Mobile Accessibility:**
   * With the increasing reliance on mobile devices, systems that lack mobile responsiveness can hinder the user experience. Members may find it inconvenient to access features on the go, impacting engagement.
8. **Scalability Issues:**
   * Some existing systems may face challenges in scaling up to accommodate a growing number of users, leading to performance issues and slower response times during peak usage.

 Time consumption:

As the records are to be manually maintained it consumes a lot of

time.

 Paper work:

Lot of paper work is involved as the records are maintained in the

files & registers

 Storage requirements:

As files and registers are used the storage space requirement is

increased.

 Less reliable:

Use of papers for storing valuable data information is not at all

reliable.

# 2.2 Proposed Solution:

The proposed Gym Management System, built on the Django framework, introduces a cutting-edge solution that addresses the limitations of the existing system. This comprehensive platform integrates advanced features to enhance user experience, streamline administrative tasks, and foster a connected fitness community. It maintains report for all criteria and transactions. Manages member information separately for all bill information separately for considering the requirement of gym. Stores information about regular products and Diet recommendation

**Key Features:**

1. **Admin Dashboard:**
   * *Secure Authentication:* Utilizing Django's authentication system, administrators can securely log in to the system.
   * *User Management:* Efficient creation, modification, and management of user profiles with detailed information, including membership status and fitness goals.
2. **User-Centric Features:**
   * *Diet Information:* Users can access personalized diet plans created by trainers, promoting a holistic approach to fitness.
   * *Workout Schedules:* Individualized schedules crafted by trainers for optimized workout routines tailored to users' fitness goals.
   * *Billing Information:* Transparent billing details, including membership fees, additional services, and payment history.
   * *Supplement Guidance:* Information on recommended supplements, dosage, and usage guidelines to complement fitness objectives.
3. **Security and Access Control:**
   * *Role-Based Access Control:* Django's built-in authentication and authorization systems ensure secure and differentiated access levels for administrators, trainers, and users.
4. **Mobile Responsiveness:**
   * *Responsive Design:* A mobile-friendly interface ensures users can access their fitness information seamlessly across various devices.

# CHAPTER 3 OVERALL DESCRIPTION OF THE PROPOSED SYSTEM

## 3.1 Module Description

The system after careful analysis has been identified to be presented with the following modules **User, Administrator .**

**User**

* **Login** – Once they have registered they need to login to avail the service at the needy time.
* **View** **Details** – Logging in with the application will provide you the lists of bill reciepts and notification of the package.
* **Search records**– So that the users can search the bill history

**Admin**

* **Login** – Registered admin can login their accounts
* **Register** **Member**– At first every member has to register their details with the admin.
* **Create/Manage Bill details** – Here bill details have to post their details like name, id, package, amount, payed, services, due date, etc.
* **Manage Customer** – Admin can view the customer details.
* **Manage Diet** – Admin can recomend the diet details.
* **Manage Supplement Store** – Admin can manage the supplement store for the gym.

**System Modules**

**Admin**

* Login
* Add Member
* Update/Delete Members
* Create Bills
  + Assign Fee Package
* Manage bills
* Manage Diet
* Manage Schedule
* Supplement store

**Members**

* Login
* View Bill Reciepts
* View Diet
* View Schedule
* View Supplement

**Admin:**

1. **Login:**
   * The login module ensures secure access for administrators, allowing them to enter the system with valid credentials.
2. **Add Member:**
   * Admins can add new members to the system, capturing essential details for creating personalized member profiles.
3. **Update/Delete Members:**
   * This module allows administrators to update existing member records with any changes and, if necessary, delete records for terminated or inactive members.
4. **Create Bills:**
   * Admins can generate bills for members based on their assigned fee packages, considering any additional charges or discounts applicable.
   * **Assign Fee Package:**
     + Admins have the capability to assign different fee packages to members based on their membership plans, ensuring flexible billing structures.
5. **Manage Bills:**
   * This module enables administrators to efficiently manage and track member bills, including overdue payments, pending bills, and payment histories.
6. **Manage Diet:**
   * Admins can manage diet-related information, including recommended meal plans and dietary advice for individual members based on their fitness goals and health conditions.
7. **Manage Schedule:**
   * This module allows administrators to create and manage schedules for fitness classes, personal training sessions, and other events within the fitness center.
8. **Supplement Store:**
   * Admins can oversee the supplement store, managing product details.

**Members:**

1. **Login:**
   * Members have secure login credentials to access their personal accounts, ensuring data privacy and security.
2. **View Bill Receipts:**
   * Members can view and download their bill receipts, providing a transparent record of their financial transactions with the fitness center.
3. **View Diet:**
   * Members can access and view personalized diet plans and nutritional advice provided by the fitness center.
4. **View Schedule:**
   * Members can check and follow the schedule of fitness classes, personal training sessions, and other events, helping them plan their workouts effectively.
5. **View Supplement:**
   * Members can explore and view available supplements in the store, check product details, and place orders for desired items.

# CHAPTER 4 – DESIGN

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

## 

## 4.1UML Diagrams:

UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

There are various kinds of methods in software design:

* Use case Diagram
* Sequence Diagram
* Collaboration Diagram

**4.1.1Usecase Diagrams**:

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what’s called an actor. Use case diagram can be useful for getting an overall view of the system and clarifying that can do and more importantly what they can’t do.



Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

* The purpose is to show the interactions between the use case and actor.
* To represent the system requirements from user’s perspective.
* An actor could be the end-user of the system or an external system.

**4.1.2 Sequence Diagram:**

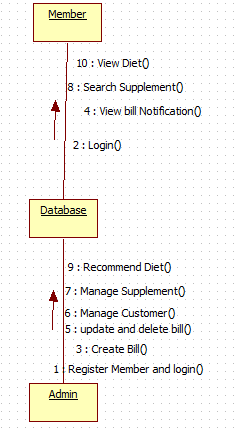
Sequence diagram and collaboration diagram are called INTERACTION DIAGRAMS. An interaction diagram shows an interaction, consisting of set of objects and their relationship including the messages that may be dispatched among them.

A sequence diagram is an introduction that empathizes the time ordering of messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis.



**4.1.3 Collaboration Diagram:**

A **collaboration diagram** is a type of visual presentation that shows how various software objects interact with each other within an overall IT architecture and how users can benefit from this **collaboration**. A **collaboration diagram** often comes in the form of a visual chart that resembles a flow chart.



**4.1.4. Data Flow Diagram**

-Member Registration

Login

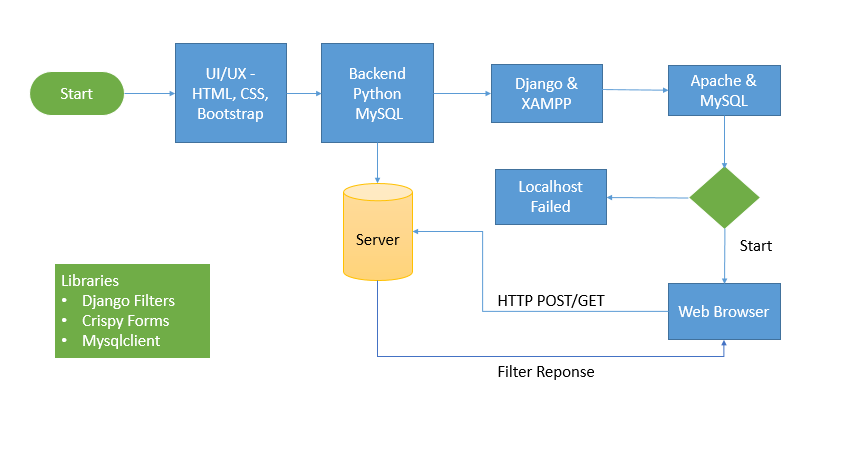
Manage Bill Details

Logout

Diet Recommendation

Manage Supplment

# 4.1.5. Work Flow Diagram



# 4.1.6. Class Diagram



**4.1.7 Table Design:**

**User Register & Login**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User ID | Name | Email Id | Password | Mobile | Weight - Age - | Address |
| Int | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Primary key |  |  |  |  |  |  |

**Bill Details**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Due Date | Email Id | Mobile | From Date | To date | Package | Cost | Payed | Batch |
| Int | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Primary key |  |  |  |  |  |  |  |  |  |

**Diet Details**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Diet | Solution 1 | Solution 2 | Process | Description | Notes |
| Int | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Primary key |  |  |  |  |  |  |

**Schedule Details**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Day no | Schedule | Trainee Level | Process | Description | Notes |
| Int | Varchar | Varchar | Varchar | Varchar | Varchar | Varchar |
| 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Primary key |  |  |  |  |  |  |

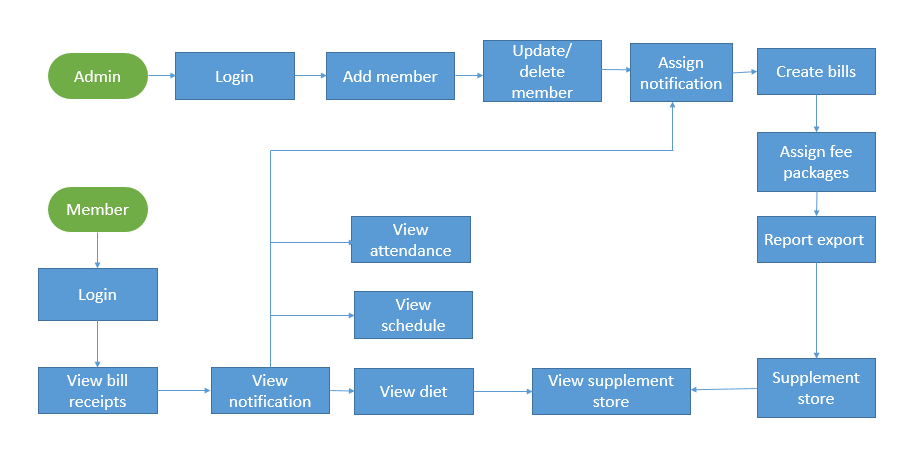
**Admin Login**

|  |  |  |
| --- | --- | --- |
| User ID | Username | Password |
| Int | Varchar | Varchar |
| 100 | 100 | 100 |
| Primary key |  |  |

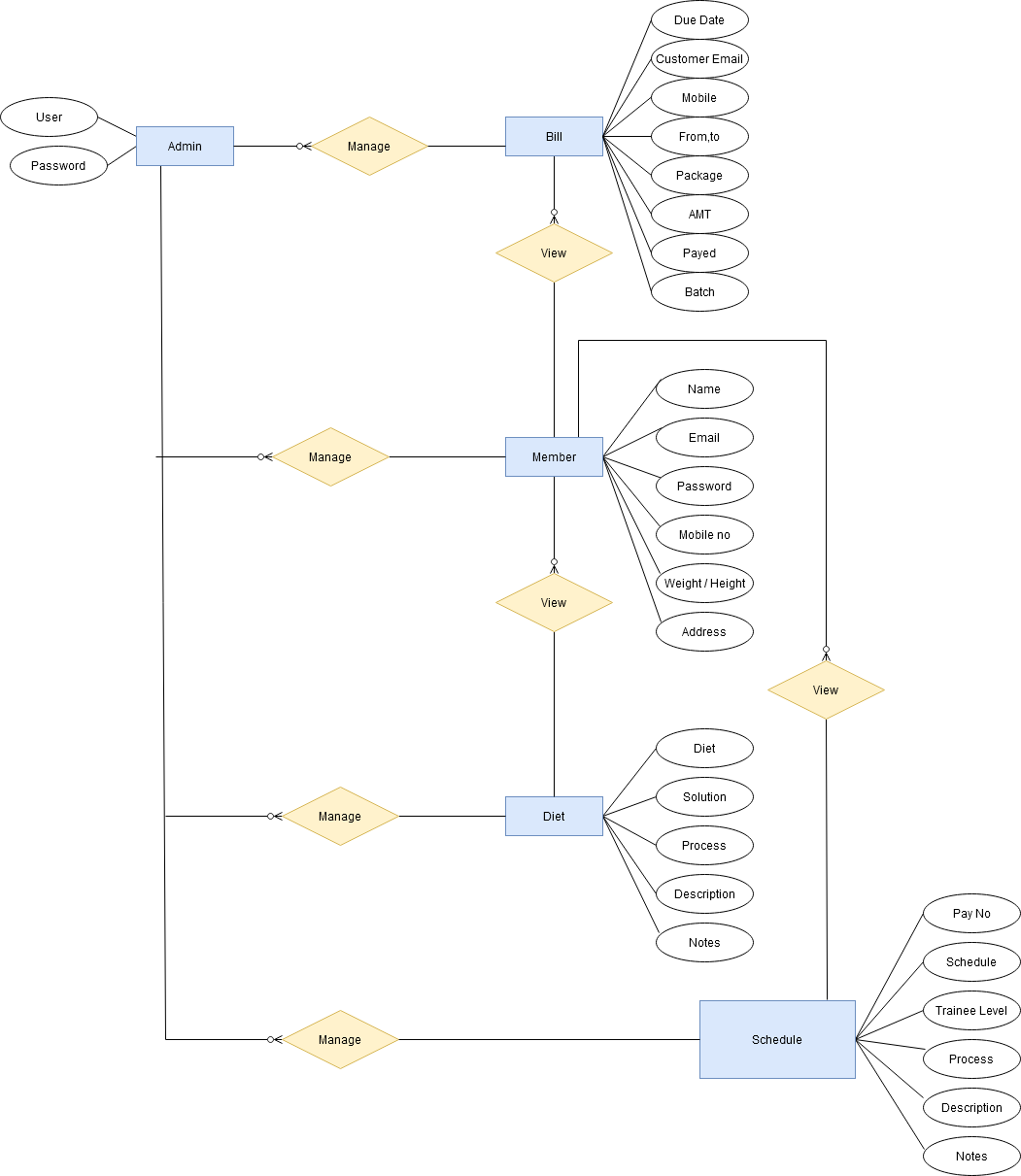
**Attendance**

|  |  |  |  |
| --- | --- | --- | --- |
| User ID | Email | Total No of days | Total Present Days |
| 100 | 100 | 100 | 100 |
| Primary key |  |  |  |

**4.1.8 Architecture Diagram:**



**4.1.9 ER Diagram:**



# CHAPTER 5 - OUTPUT SCREENSHOT

# 

# CHAPTER 6 – IMPLEMENTATION DETAILS

## 6.1 Introduction to Html Framework

Hyper Text Markup Language, commonly referred to as HTML, is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) used to create [web pages](https://en.wikipedia.org/wiki/Web_page). Along with [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets), and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and [web applications](https://en.wikipedia.org/wiki/Web_applications). [Web browsers](https://en.wikipedia.org/wiki/Web_browser) can read HTML files and render them into visible or audible web pages. HTML describes the structure of a [website](https://en.wikipedia.org/wiki/Website) [semantically](https://en.wikipedia.org/wiki/Semantic) along with cues for presentation, making it a markup language, rather than a [programming language](https://en.wikipedia.org/wiki/Programming_language).

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) form the building blocks of HTML pages. HTML allows [images](https://en.wikipedia.org/wiki/Img_(HTML_element)) and other objects to be embedded and it can be used to create [interactive forms](https://en.wikipedia.org/wiki/Fieldset). It provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural semantics for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by tags, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img /> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed [scripts](https://en.wikipedia.org/wiki/Scripting_language) written in languages such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript) which affect the behavior of HTML web pages. HTML markup can also refer the browser to [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) to define the look and layout of text and other material.

HyperText Markup Language (HTML) is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for creating [web pages](https://en.wikipedia.org/wiki/Web_page) and [web applications](https://en.wikipedia.org/wiki/Web_application). With [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [JavaScript](https://en.wikipedia.org/wiki/JavaScript) it forms a triad of cornerstone technologies for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).[[1]](https://en.wikipedia.org/wiki/HTML#cite_note-1) [Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [webserver](https://en.wikipedia.org/wiki/Webserver) or from local storage and render them into multimedia web pages. HTML describes the structure of a web page [semantically](https://en.wikipedia.org/wiki/Semantic) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/Img_%28HTML_element%29) and other objects, such as [interactive forms,](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. It provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by tags, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img /> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript) which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[[2]](https://en.wikipedia.org/wiki/HTML#cite_note-deprecated-2)

In 1980, physicist [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee), a contractor at [CERN](https://en.wikipedia.org/wiki/CERN), proposed and prototyped [ENQUIRE](https://en.wikipedia.org/wiki/ENQUIRE), a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an [Internet](https://en.wikipedia.org/wiki/Internet)-based [hypertext](https://en.wikipedia.org/wiki/Hypertext) system.[[3]](https://en.wikipedia.org/wiki/HTML#cite_note-3) Berners-Lee specified HTML and wrote the browser and server software in late 1990. That year, Berners-Lee and CERN data systems engineer [Robert Cailliau](https://en.wikipedia.org/wiki/Robert_Cailliau) collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes[[4]](https://en.wikipedia.org/wiki/HTML#cite_note-4) from 1990 he listed[[5]](https://en.wikipedia.org/wiki/HTML#cite_note-5) "some of the many areas in which hypertext is used" and put an encyclopedia first.

The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Tim Berners-Lee in late 1991.[[6]](https://en.wikipedia.org/wiki/HTML#cite_note-tagshtml-6)[[7]](https://en.wikipedia.org/wiki/HTML#cite_note-7) It describes 18 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by [SGMLguid](https://en.wikipedia.org/wiki/SGMLguid), an in-house [Standard Generalized Markup Language](https://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language) (SGML)-based documentation format at CERN. Eleven of these elements still exist in HTML 4.[[8]](https://en.wikipedia.org/wiki/HTML#cite_note-8)

HTML is a [markup language](https://en.wikipedia.org/wiki/Markup_language) that [web browsers](https://en.wikipedia.org/wiki/Web_browser) use to interpret and [compose](https://en.wikipedia.org/wiki/Typesetting) text, images, and other material into visual or audible web pages. Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets). Many of the text elements are found in the 1988 ISO technical report TR 9537 Techniques for using SGML, which in turn covers the features of early text formatting languages such as that used by the [RUNOFF command](https://en.wikipedia.org/wiki/TYPSET_and_RUNOFF) developed in the early 1960s for the [CTSS](https://en.wikipedia.org/wiki/Compatible_Time-Sharing_System) (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and markup; HTML has been progressively moved in this direction with CSS.

Berners-Lee considered HTML to be an application of SGML. It was formally defined as such by the [Internet Engineering Task Force](https://en.wikipedia.org/wiki/Internet_Engineering_Task_Force) (IETF) with the mid-1993 publication of the first proposal for an HTML specification, the "Hypertext Markup Language (HTML)" Internet Draft by Berners-Lee and [Dan Connolly](https://en.wikipedia.org/wiki/Dan_Connolly_%28computer_scientist%29), which included an SGML [Document Type Definition](https://en.wikipedia.org/wiki/Document_Type_Definition) to define the grammar.[[9]](https://en.wikipedia.org/wiki/HTML#cite_note-9)[[10]](https://en.wikipedia.org/wiki/HTML#cite_note-10) The draft expired after six months, but was notable for its acknowledgment of the [NCSA Mosaic](https://en.wikipedia.org/wiki/Mosaic_%28web_browser%29) browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes.[[11]](https://en.wikipedia.org/wiki/HTML#cite_note-raymond-11) Similarly, [Dave Raggett](https://en.wikipedia.org/wiki/Dave_Raggett)'s competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.[[12]](https://en.wikipedia.org/wiki/HTML#cite_note-html.2B-12)

After the HTML and HTML+ drafts expired in early 1994, the IETF created an HTML Working Group, which in 1995 completed "HTML 2.0", the first HTML specification intended to be treated as a standard against which future implementations should be based.[[13]](https://en.wikipedia.org/wiki/HTML#cite_note-13)

Further development under the auspices of the IETF was stalled by competing interests. Since 1996, the HTML specifications have been maintained, with input from commercial software vendors, by the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C).[[14]](https://en.wikipedia.org/wiki/HTML#cite_note-raggett-14) However, in 2000, HTML also became an international standard ([ISO](https://en.wikipedia.org/wiki/International_Organization_for_Standardization)/[IEC](https://en.wikipedia.org/wiki/International_Electrotechnical_Commission) 15445:2000). HTML 4.01 was published in late 1999, with further errata published through 2001. In 2004, development began on HTML5 in the [Web Hypertext Application Technology Working Group](https://en.wikipedia.org/wiki/Web_Hypertext_Application_Technology_Working_Group) (WHATWG), which became a joint deliverable with the W3C in 2008, and completed and standardized on 28 October 2014.[[15]](https://en.wikipedia.org/wiki/HTML#cite_note-15)

## 6.2 Cascading Style Sheets (CSS)

CSS is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language). Although most often used to set the visual style of [web pages](https://en.wikipedia.org/wiki/Web_page) and user interfaces written in [HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML), the language can be applied to any [XML](https://en.wikipedia.org/wiki/XML) document, including [plain XML](https://en.wikipedia.org/wiki/Plain_Old_XML), [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) and[XUL](https://en.wikipedia.org/wiki/XUL), and is applicable to rendering in [speech](https://en.wikipedia.org/wiki/Speech_synthesis), or on other media. Along with HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for [web applications](https://en.wikipedia.org/wiki/Web_applications), and user interfaces for many mobile applications.

CSS is designed primarily to enable [the separation of document content from document presentation](https://en.wikipedia.org/wiki/Separation_of_presentation_and_content), including aspects such as the [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface). This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as [semantically insignificant tables](https://en.wikipedia.org/wiki/Tableless_web_design) that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. For each matching [HTML element](https://en.wikipedia.org/wiki/HTML_element), it provides a list of formatting instructions. For example, a CSS rule might specify that "all heading 1 elements should be [bold](https://en.wikipedia.org/wiki/Bold)", leaving pure semantic HTML markup that asserts "this text is a level 1 heading" without formatting code such as a<bold> tag indicating how such text should be displayed.

This separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or[screen reader](https://en.wikipedia.org/wiki/Screen_reader)) and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display), tactile devices. It can also be used to display the web page differently depending on the screen size or device on which it is being viewed. Although the author of a web page typically links to a CSS file within the markup file, readers can specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author has specified. If the author or the reader did not link the document to a style sheet, the default style of the browser will be applied. Another advantage of CSS is that aesthetic changes to the [graphic design](https://en.wikipedia.org/wiki/Graphic_design) of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in one file, rather than by a laborious (and thus expensive) process of crawling over every document line by line, changing markup.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

Cascading Style Sheets (CSS) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language).[[1]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-1) Although most often used to set the visual style of [web pages](https://en.wikipedia.org/wiki/Web_page) and user interfaces written in [HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML), the language can be applied to any [XML](https://en.wikipedia.org/wiki/XML) document, including [plain XML](https://en.wikipedia.org/wiki/Plain_Old_XML), [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) and [XUL](https://en.wikipedia.org/wiki/XUL), and is applicable to rendering in [speech](https://en.wikipedia.org/wiki/Speech_synthesis), or on other media. Along with HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for [web applications](https://en.wikipedia.org/wiki/Web_applications), and user interfaces for many mobile applications.[[2]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-2)

CSS is designed primarily to enable [the separation of document content from document presentation](https://en.wikipedia.org/wiki/Separation_of_presentation_and_content), including aspects such as the [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface).[[3]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-3) This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or [screen reader](https://en.wikipedia.org/wiki/Screen_reader)), and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display) tactile devices. It can also display the web page differently depending on the screen size or viewing device. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified.

Changes to the [graphic design](https://en.wikipedia.org/wiki/Graphic_design) of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C). Internet media type ([MIME type](https://en.wikipedia.org/wiki/MIME_media_type)) text/css is registered for use with CSS by [RFC 2318](https://tools.ietf.org/html/rfc2318) (March 1998). The W3C operates a free [CSS validation service](https://en.wikipedia.org/wiki/W3C_Markup_Validation_Service#CSS_validation) for CSS documents.

In CSS, selectors declare which part of the markup a style applies to by matching tags and attributes in the markup itself.

Selectors may apply to:

all [elements](https://en.wikipedia.org/wiki/HTML_element) of a specific type, e.g. the second-level headers [h2](https://en.wikipedia.org/wiki/HTML_element#Basic_text)

elements specified by [attribute](https://en.wikipedia.org/wiki/HTML_attribute), in particular:

id: an identifier unique within the document

class: an identifier that can annotate multiple elements in a document

elements depending on how they are placed relative to others in the [document tree](https://en.wikipedia.org/wiki/Document_Object_Model).

Classes and IDs are case-sensitive, start with letters, and can include alphanumeric characters and underscores. A class may apply to any number of instances of any elements. An ID may only be applied to a single element.

Pseudo-classes are used in CSS selectors to permit formatting based on information that is not contained in the document tree. One example of a widely used pseudo-class is :hover, which identifies content only when the user "points to" the visible element, usually by holding the mouse cursor over it. It is appended to a selector as in a:hover or #elementid:hover. A pseudo-class classifies document elements, such as :link or :visited, whereas a pseudo-element makes a selection that may consist of partial elements, such as ::first-line or ::first-letter.[[5]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-5)

Selectors may be combined in many ways to achieve great specificity and flexibility.[[6]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-6) Multiple selectors may be joined in a spaced list to specify elements by location, element type, id, class, or any combination thereof. The order of the selectors is important. For example, div .myClass {color: red;} applies to all elements of class myClass that are inside div elements, whereas .myClass div {color: red;} applies to all div elements that are in elements of class myClass.

CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium.

The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. This process is called cascading.

One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

CSS was first proposed by [Håkon Wium Lie](https://en.wikipedia.org/wiki/H%C3%A5kon_Wium_Lie) on October 10, 1994.[[16]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chss-proposal-16) At the time, Lie was working with [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee) at [CERN](https://en.wikipedia.org/wiki/CERN).[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17) Several other style sheet languages for the web were proposed around the same time, and discussions on public mailing lists and inside [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) resulted in the first W3C CSS Recommendation (CSS1)[[18]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-w3c-css1-18) being released in 1996. In particular, [Bert Bos](https://en.wikipedia.org/wiki/Bert_Bos)' proposal was influential; he became co-author of CSS1 and is regarded as co-creator of CSS.[[19]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-WWW3-19)

Style sheets have existed in one form or another since the beginnings of Standard Generalized Markup Language ([SGML](https://en.wikipedia.org/wiki/SGML)) in the 1980s, and CSS was developed to provide style sheets for the web.[[20]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-css-phd-20) One requirement for a web style sheet language was for style sheets to come from different sources on the web. Therefore, existing style sheet languages like [DSSSL](https://en.wikipedia.org/wiki/Document_Style_Semantics_and_Specification_Language) and [FOSI](https://en.wikipedia.org/wiki/Formatting_Output_Specification_Instance) were not suitable. CSS, on the other hand, let a document's style be influenced by multiple style sheets by way of "cascading" styles.[[20]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-css-phd-20)

As [HTML](https://en.wikipedia.org/wiki/HTML) grew, it came to encompass a wider variety of stylistic capabilities to meet the demands of [web developers](https://en.wikipedia.org/wiki/Web_development). This evolution gave the designer more control over site appearance, at the cost of more complex HTML. Variations in [web browser](https://en.wikipedia.org/wiki/Web_browser) implementations, such as [ViolaWWW](https://en.wikipedia.org/wiki/ViolaWWW) and [WorldWideWeb](https://en.wikipedia.org/wiki/WorldWideWeb),[[21]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-IEEE-21) made consistent site appearance difficult, and users had less control over how web content was displayed. The browser/editor developed by Tim Berners-Lee had style sheets that were hard-coded into the program. The style sheets could therefore not be linked to documents on the web.[[22]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-cssdftw-22) [Robert Cailliau](https://en.wikipedia.org/wiki/Robert_Cailliau), also of CERN, wanted to separate the structure from the presentation so that different style sheets could describe different presentation for printing, screen-based presentations, and editors.[[21]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-IEEE-21)

Improving web presentation capabilities was a topic of interest to many in the web community and nine different style sheet languages were proposed on the www-style mailing list.[[20]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-css-phd-20) Of these nine proposals, two were especially influential on what became CSS: Cascading HTML Style Sheets[[16]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chss-proposal-16) and Stream-based Style Sheet Proposal (SSP).[[19]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-WWW3-19)[[23]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-ssp-23) Two browsers served as testbeds for the initial proposals; Lie worked with [Yves Lafon](https://en.wikipedia.org/w/index.php?title=Yves_Lafon&action=edit&redlink=1) to implement CSS in [Dave Raggett](https://en.wikipedia.org/wiki/Dave_Raggett)'s [Arena](https://en.wikipedia.org/wiki/Arena_%28web_browser%29) browser.[[24]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-24)[[25]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-25)[[26]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-26) [Bert Bos](https://en.wikipedia.org/wiki/Bert_Bos) implemented his own SSP proposal in the [Argo](https://en.wikipedia.org/wiki/Argo_%28web_browser%29) browser.[[19]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-WWW3-19) Thereafter, Lie and Bos worked together to develop the CSS standard (the 'H' was removed from the name because these style sheets could also be applied to other markup languages besides HTML).[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17)

Lie's proposal was presented at the "[Mosaic and the Web](https://en.wikipedia.org/w/index.php?title=Mosaic_and_the_Web&action=edit&redlink=1)" conference (later called WWW2) in [Chicago, Illinois](https://en.wikipedia.org/wiki/Chicago,_Illinois) in 1994, and again with Bert Bos in 1995.[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17) Around this time the W3C was already being established, and took an interest in the development of CSS. It organized a workshop toward that end chaired by [Steven Pemberton](https://en.wikipedia.org/wiki/Steven_Pemberton). This resulted in W3C adding work on CSS to the deliverables of the HTML editorial review board (ERB). Lie and Bos were the primary technical staff on this aspect of the project, with additional members, including [Thomas Reardon](https://en.wikipedia.org/wiki/Thomas_Reardon) of [Microsoft](https://en.wikipedia.org/wiki/Microsoft), participating as well. In August 1996 [Netscape Communication Corporation](https://en.wikipedia.org/wiki/Netscape) presented an alternative style sheet language called [JavaScript Style Sheets](https://en.wikipedia.org/wiki/JavaScript_Style_Sheets) (JSSS).[[17]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-chapter20-17) The spec was never finished and is deprecated.[[27]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-27) By the end of 1996, CSS was ready to become official, and the CSS level 1 Recommendation was published in December.

Development of HTML, CSS, and the [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) had all been taking place in one group, the HTML Editorial Review Board (ERB). Early in 1997, the ERB was split into three [working groups](https://en.wikipedia.org/wiki/Working_group): [HTML Working group](https://en.wikipedia.org/wiki/HTML_Working_group), chaired by [Dan Connolly](https://en.wikipedia.org/wiki/Dan_Connolly_%28computer_scientist%29) of W3C; DOM Working group, chaired by Lauren Wood of [SoftQuad](https://en.wikipedia.org/wiki/SoftQuad); and [CSS Working group](https://en.wikipedia.org/wiki/CSS_Working_group), chaired by [Chris Lilley](https://en.wikipedia.org/wiki/Chris_Lilley_%28W3C%29) of W3C.

The CSS Working Group began tackling issues that had not been addressed with CSS level 1, resulting in the creation of CSS level 2 on November 4, 1997. It was published as a W3C Recommendation on May 12, 1998. CSS level 3, which was started in 1998, is still under development as of 2014.

In 2005 the CSS Working Groups decided to enforce the requirements for standards more strictly. This meant that already published standards like CSS 2.1, CSS 3 Selectors and CSS 3 Text were pulled back from Candidate Recommendation to Working Draft level.

## 6.3 MYSQL Server

MySQL  is an [open-source](https://en.wikipedia.org/wiki/Open-source) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS);[[6]](https://en.wikipedia.org/wiki/MySQL#cite_note-6) in July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source [client–server model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) RDBMS. It is named after co-founder [Michael Widenius](https://en.wikipedia.org/wiki/Michael_Widenius)'s daughter, My. The [SQL](https://en.wikipedia.org/wiki/SQL) acronym stands for [Structured Query Language](https://en.wikipedia.org/wiki/Structured_Query_Language). The MySQL development project has made its [source code](https://en.wikipedia.org/wiki/Source_code) available under the terms of the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), as well as under a variety of [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) agreements. MySQL was owned and sponsored by a single [for-profit](https://en.wikipedia.org/wiki/Business) firm, the [Swedish](https://en.wikipedia.org/wiki/Sweden)company [MySQL AB](https://en.wikipedia.org/wiki/MySQL_AB), now owned by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation). For proprietary use, several paid editions are available, and offer additional functionality.

SQL Server Management Studio (SSMS) is a software application first launched with [Microsoft](https://en.wikipedia.org/wiki/Microsoft) [SQL Server 2005](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) that is used for configuring, managing, and administering all components within [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server). The tool includes both script editors and graphical tools which work with objects and features of the server.[[1]](https://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-1)

A central feature of SSMS is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server.[[2]](https://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-2) It also shipped a separate Express edition that could be freely downloaded, however recent versions of SSMS are fully capable of connecting to and manage any SQL Server Express instance. Microsoft also incorporated backwards compatibility for older versions of SQL Server thus allowing a newer version of SSMS to connect to older versions of SQL Server instances.

Starting from version 11, the application was based on the [Visual Studio 2010](https://en.wikipedia.org/wiki/Visual_Studio_2010) shell, using [WPF](https://en.wikipedia.org/wiki/Windows_Presentation_Foundation) for the user interface.

In June 2015, Microsoft announced their intention to release future versions of SSMS independently of SQL Server database engine releases.[[3]](https://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-3).

## 6.4PYTHON

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see [The Python Standard Library](https://docs.python.org/3/library/index.html#library-index). [The Python Language Reference](https://docs.python.org/3/reference/index.html#reference-index) gives a more formal definition of the language. To write extensions in C or C++, read [Extending and Embedding the Python Interpreter](https://docs.python.org/3/extending/index.html#extending-index) and [Python/C API Reference Manual](https://docs.python.org/3/c-api/index.html#c-api-index). There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python’s most noteworthy features, and will give you a good idea of the language’s flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in [The Python Standard Library](https://docs.python.org/3/library/index.html#library-index).

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to [Perl](https://en.wikipedia.org/wiki/Perl)'s "[there is more than one way to do it](https://en.wikipedia.org/wiki/There_is_more_than_one_way_to_do_it)" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy.[[64]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-PEP20-64) [Alex Martelli](https://en.wikipedia.org/wiki/Alex_Martelli), a [Fellow](https://en.wikipedia.org/wiki/Fellow) at the [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation) and Python book author, wrote: "To describe something as 'clever' is not considered a compliment in the Python culture."[[65]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-19-65)

Python's developers strive to avoid [premature optimization](https://en.wikipedia.org/wiki/Premature_optimization), and reject patches to non-critical parts of the [CPython](https://en.wikipedia.org/wiki/CPython) reference implementation that would offer marginal increases in speed at the cost of clarity.[[66]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-20-66) When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use [PyPy](https://en.wikipedia.org/wiki/PyPy), a [just-in-time compiler](https://en.wikipedia.org/wiki/Just-in-time_compilation). [Cython](https://en.wikipedia.org/wiki/Cython) is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim for it to be fun to use. This is reflected in its name—a tribute to the British comedy group [Monty Python](https://en.wikipedia.org/wiki/Monty_Python)[[67]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-24-67)—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a [Monty Python sketch](https://en.wikipedia.org/wiki/Spam_(Monty_Python))) instead of the standard [foo and bar](https://en.wikipedia.org/wiki/Foobar).[[68]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-68)[[69]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-69)

## 6.5 DJANGO

With Django, you can take web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django ([/ˈdʒæŋɡoʊ/](https://en.wikipedia.org/wiki/Help:IPA/English) [JANG-goh](https://en.wikipedia.org/wiki/Help:Pronunciation_respelling_key); sometimes stylized as django)[[6]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-6) is a [Python](https://en.wikipedia.org/wiki/Python_(programming_language))-based [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [web framework](https://en.wikipedia.org/wiki/Web_framework) that follows the model–template–views (MTV) [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern_(computer_science)).[[7]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-faq-mvc-7)[[8]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-djangobook-mvc-8) It is maintained by the [Django Software Foundation](https://en.wikipedia.org/wiki/Django_Software_Foundation) (DSF), an independent organization established in the US as a [501(c)(3)](https://en.wikipedia.org/wiki/501(c)(3)) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability) and "pluggability" of components, less code, low coupling, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself).[[9]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-9) Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Type_introspection) and configured via admin models.

Some well-known sites that use Django include [Instagram](https://en.wikipedia.org/wiki/Instagram),[[10]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-10) [Mozilla](https://en.wikipedia.org/wiki/Mozilla_Foundation),[[11]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-11) [Disqus](https://en.wikipedia.org/wiki/Disqus),[[12]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-12) [Bitbucket](https://en.wikipedia.org/wiki/Bitbucket),[[13]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-13) [Nextdoor](https://en.wikipedia.org/wiki/Nextdoor)[[14]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-14) and [Clubhouse](https://en.wikipedia.org/wiki/Clubhouse_(app)).[[15]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-15)

The Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

Django comes with a lightweight web server for developing and testing applications. This server is pre-configured to work with Django, and more importantly, it restarts whenever you modify the code.

**CHAPTER 7- SYSTEM STUDY**

**7.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

# CHAPTER 8-TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**8.1Non Functional Requirements**

Non-functional requirements are the quality requirements that stipulate how well software does what it has to do. These are Quality attributes of any system; these can be seen at the execution of the system and they can also be the part of the system architecture.

**8.2 Accuracy:**

The system will be accurate and reliable based on the design architecture. If there is any problem in the accuracy then the system will provide alternative ways to solve the problem.

**8.3 Usability:**

The proposed system will be simple and easy to use by the users. The users will comfort in order to communicate with the system. The user will be provided with an easy interface of the system.

**8.4 Accessibility:**

The system will be accessible through internet and there should be no any known problem.

* 1. **Performance:**

The system performance will be at its best when performing the functionality of the system.

* 1. **Reliability:**

The proposed system will be reliable in all circumstances and if there is any problem that will be affectively handle in the design.

* 1. **Security:**

The proposed system will be highly secured; every user will be required registration and username/password to use the system. The system will do the proper authorization and authentication of the users based on their types and their requirements. The proposed system will be designed persistently to avoid any misuse of the application.

# CHAPTER 9-SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TESTCASE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **SCENARIO** | **INPUT** | **EXPECTED**  **OUTPUT** | **ACTUAL OUTPUT** |
| 1 | Admin login Details | Admin will enter email and password | Login successfully or if incorrect login details  “Login unsuccessfully” | Login successfully or  Login unsuccessfully |
| 2 | Add Member | Admin will enter all Member details | If all the Member details  “Created Successfully” | Created successfully or created unsuccessfully |
| 3 | Edit and delete Member | Admin can edit and delete the Member details | If any changes edit updated successfullly or delete successfullly | Updated successfully or unsuccessfully  Delete successfully or unsuccessfully |
| 4 | Create Bills | Bills created by all details like sno, product name ,price | All the details Create successfully | Created successfully or created unsuccessfully |
| 5 | Assign Fee package | Admin add the Fee package like 5000,10000 | all the Fee package details “updated  successfully” | updated successfully or  unsuccessfully |
| 6 | Assign Notification For Monthly | Admin add the Notification For Monthly like march, may | all the Notification For Monthly details “updated  successfully” | updated successfully or  unsuccessfully |
| 8 | Supplement Store | Admin will check the Supplement Store | Admin view Supplement Store details | View all Supplement Store details |
| 9 | Add Diet Details | Admin will enter all diet details | If all the diet  “Created Successfully” | Created successfully or created unsuccessfully |
| 10 | Update Attendance | Admin add the Attendance details like present,absent | all the Attendance login details “updated  successfully” | updated successfully or  unsuccessfully |
| 11 | Schedule Details | Admin add the Schedule details | all the Schedule details “updated  successfully” | updated successfully or  unsuccessfully |
| 12 | Member login | Member will enter email and password | Login successfully or if incorrect login details  “Login unsuccessfully” | Login successfully or  Login unsuccessfully |
| 13 | View Bill Receipt | Member will check the Bill Receipt details | Member will update Bill Receipt | Updated successfully or unsuccessfully |
| 14 | View Diet | Member will check the Diet details | Member will update Diet details | Updated successfully or unsuccessfully |
| 15 | View Exercise Schedule | Member will check the Exercise Schedule details | Member will update Exercise Schedule details | Updated successfully or unsuccessfully |
| 16 | View Supplementary Products | Member will check the Supplementary Products details | Member will update Supplementary Products details | Updated successfully or unsuccessfully |
| 17 | View Attendance | Member will check the Attendance details | Member will update Attendance details | Updated successfully or unsuccessfully |

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**9.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page

**9.2 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**9.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# 

The “GYM MANAGEMENT SYSTEM” is successfully designed and developed to fulfilling the necessary requirements, as identified in the requirements analysis phase, such as the system is very much user friendly, form level validation and field level validation are performing very efficiently. The new computerized system was found to be much faster and reliable and user friendly then the existing system, the system has been designed and developed step by step and tested successfully. It eliminates the human error that are likely to creep in the kind of working in which a bulk quantity of data and calculations as to be processed. The system results in quick retrieval of information that is very vital for the progress any organization. Cost is minimized in case of stationary. Burden of manual work is reduced as whenever transaction takes place, there is a no need to record it in many places manually.

# CHAPTER 11- REFERENCES

* Brown, A., & White, B. (2019). "Web Security in Django." *Proceedings of the International Conference on Web Engineering.*
* Chen, S., et al. (2017). "A Secure Authentication Scheme for Django Web Applications." *Journal of Computer Science and Technology.*
* Gupta, R., et al. (2018). "Responsive Web Design Using Django Framework." *International Journal of Computer Applications.*
* Jones, M., & Smith, P. (2020). "Enhancing User Engagement in Gym Management Systems." *Journal of Sports Science & Medicine.*
* Kim, J., & Lee, S. (2021). "Financial Tracking in Gym Management Systems." *International Journal of Sports Science & Coaching.*
* Li, Q., et al. (2020). "Mobile Responsive Design in Fitness Applications." *Journal of Mobile Technology in Medicine.*
* Marshall, T., et al. (2019). "Integrated Systems for Gym Management." *International Journal of Sports Management and Marketing.*
* Nguyen, H., & Tran, M. (2018). "Responsive Web Design with Django." *Proceedings of the International Conference on Advanced Engineering Theory and Applications.*
* Robinson, D., & Turner, K. (2018). "Billing Systems in Health and Fitness Clubs." *International Journal of Sports Economics and Management.*
* Wang, Y., et al. (2019). "Effective Communication in Fitness Apps." *Proceedings of the International Conference on Human-Computer Interaction.*

# CHAPTER 12- CODING

View.py

from django.shortcuts import get\_object\_or\_404, redirect, render

from django.http import HttpResponse

from django.forms import inlineformset\_factory

from django.contrib.auth import authenticate, login, logout

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

# Create your views here.

from django.contrib.auth.hashers import make\_password

from django.contrib.auth.models import User

# from .forms import

# from .filters import

from datetime import date,datetime

import os

from django.contrib import auth

from .models import \*

def home(request):

    return render(request, 'gym/home.html')

def index(request):

    return render(request, 'gym/index.html')

def adminlogin(request):

    return redirect('admin')

def member\_login(request):

    if request.method == 'POST':

        try:

            user=Member.objects.get(Email = request.POST['Email'],Password= request.POST['Password'])

            return redirect('home')

        except Member.DoesNotExist:

            return render (request,'gym/member\_login.html', {'error':'Username or password is incorrect!'})

    else:

        return render(request, 'gym/member\_login.html')

def view\_bill(request):

    Email=request.COOKIES['Email']

    getdata = Bill.objects.filter(Customer\_id=Email)

    #if not getdata and getdata is None:

    if getdata.exists():

        get\_amount=Bill.objects.values\_list('Amount', flat=True).get(Customer\_id=Email)

        get\_payed=Bill.objects.values\_list('Payed', flat=True).get(Customer\_id=Email)

        Balance=int(get\_amount)-int(get\_payed)

        return render(request, 'gym/view\_bill.html', {'getdata': getdata,'Balance': Balance})

    else :

       return render (request,'gym/view\_bill.html', {'error':'there is no Bill'})

def view\_diet(request):

    getdata = Diet.objects.all()

    if not getdata:

       return render (request,'gym/view\_diet.html', {'error':'there is no diet'})

    else :

        return render(request, 'gym/view\_diet.html', {'getdata': getdata})

def view\_schedule(request):

    getdata = Schedule.objects.all()

    if not getdata:

       return render (request,'gym/view\_schedule.html', {'error':'there is no schedule'})

    else :

        return render(request, 'gym/view\_schedule.html', {'getdata': getdata})

def view\_store(request):

    getdata = Store.objects.all()

    if not getdata:

       return render (request,'gym/view\_store.html', {'error':'there is no supplements'})

    else :

        return render(request, 'gym/view\_store.html', {'getdata': getdata})

def logout(request):

    if request.method == 'POST':

        auth.logout(request)

    return redirect('index')