VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore • Chennai

CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Certificate of Merit

This is to certify that the following team of students of B.Tech (CSE) studying in the School of Computer Science and Engineering, VIT, Chennai Campus have successfully completed the project work entitled

GYM MANAGEMENT SYSTEM

between 25th April 2023 and 5th July 2023.

The work was found commendable and good.

Team Members:

1. 21BCE1568 Adit Deshpande
2. 21BCE1832 Aviral Bansal

[illegible]

Duly Examined and Certified by

Dr. PRAVEEN JOE I.R
Assistant Professor Grade II
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai Campus, Chennai - 600 127

Software engineering Project

GYM MANAGEMENT SYSTEM

(SRS REPORT)

Team members:

- Name: Adit Deshpande
  Roll no: 21BCE1568
- Name: Aviral Bansal
  Roll no: 21BCE1832

Team Guide:

Dr. Praveen Joe

College Name: VIT Chennai **Department:** School of Computer Science

Product’s Purpose:

Introduction:

With the sudden trend of fitness and health, the need and importance of gym has become all more necessary. Hence, it also requires a better management system. The Gym Management System focuses on improving the management of gym in colleges.

You would be able to check when your plan is ending, you can choose to renew it, check you work out info, reserve your slots and more.

It also makes the system easy for the trainers and admins as they can easily check whether the student is a member or not, check their payment and keep track of who is doing what in the gym.

Problem Statement:

To digitize the gym management system to solve the problems faced by gym goers and gym faculty.

Problems faced:

1. Gym overcrowding due to non-members using gym facility
2. Difficulty in acquiring Gym membership
3. Difficulty in acquiring gym Id and verifying it on each entry

Purpose:

The projects aim is to create a product which will solve the above problems for Gym goers, Gym trainer and Gym faculty.

The project will be a website which will digitize the gym going process and make accessing the gym easier by digitize and storing the membership information on a database.

Scope:

The scope of a gym management system typically includes various features and functionalities that help streamline and automate the operations of a gym or fitness centre. Here are some key areas that a gym management system may cover:

Functional Requirements:

1. **User Authentication:** The website should have a user authentication system that proves whether a person is a paid member or not which can be checked by the admin.

2. **Membership Management:** The website should have a system that allows gym faculty to manage memberships, including registration, renewals, cancellations, and payments.

3. **Billing and Payment Processing:** The website should redirect to the correct billing and payment processing system that can receive and process membership fees and payments for membership such as vtop.

4. **Reporting and Analytics:** The website should provide reporting and analytics capabilities that allow gym faculty to monitor usage, membership, and financial data, and track overall gym performance over time.

Definition, Acronym, Abbreviation:

- **ER –** Entity Relationship
- **DFD –** Data flow diagram

- **SRS –** Software Requirement Specification
- **CFD –** Context Flow diagram
- **SQL –** Structured Query Language

Hardware/System Requirements:

- 2GB ram(minimum)
- Windows 7/8/10/11
- Web Browser like Chrome, Mozilla, Edge.
- Processor: Intel Pentium 4 or later
- Screen resolution: 1280x1024 or larger
- Application window size: 1024x680 or larger
- Internet connection: Required

Software Tools and Technologies Used:

- <u>Software Tools:</u>
    1. Star Uml
    2. Team Gantt
    3. VS Code
    4. Selenium
    5. XAMPP
- <u>Frontend:</u>
    1. HTML
    2. CSS
    3. JAVA SCRIPT
- <u>Backend:</u>
    1. Python
    2. Django

- <u>Database:</u>

1. SQL

MODULES:

1. **HTML Structure**: The HTML structure defines the layout of the website. It consists of different sections, such as header, main content, footer, and several sections within the main content representing different parts of the website.

2. **CSS Styling**: The CSS styling is applied to the HTML elements to control the appearance of the website. The CSS code is linked to the HTML file using the `<link>` tag, which references an external CSS file (`style.css`) and a font file (`stylesheet.css`).

3. **Header**: The header section contains the navigation menu, logo, and a responsive navigation icon. The navigation menu includes links to different sections of the website, such as Home, About, Courses, Trainers, Contact, and an Admin link.

4. **Main Content**: The main content section includes different sections of the website, such as the hero section, about section, courses section, offer section, trainers section, review section, and contact section. Each section is defined with a specific class and ID to apply styling and scrolling functionality.

5. **Hero Section**: The hero section represents the introductory part of the website, featuring a heading and description.

6. **About Section**: The about section provides information about the gym, its mission, and the services it offers. It includes a heading, description, and images.

7. **Courses Section**: The courses section displays different courses offered by the gym. Each course is presented as a card with an image, overlay, and description.

8. **Offer Section**: The offer section highlights a special offer or promotion provided by the gym. It includes a heading, description, and an image.

9. **Plans Section**: The Plans section showcases different plans offered by the gym. Each plan is presented as a card with an image, price, and details.

10. **Review Section**: The review section includes testimonials from gym members. Each testimonial is displayed in a card format with an image, text, and the name of the person giving the review.

11. **Contact Section**: The contact section provides a form for users to get in touch with the gym. It includes fields for name, email, phone, and a message.

12. **Footer**: The footer section contains the contact details of the gym, a navigation menu, a newsletter section, and the legal information. The contact details include the gym's name, location, phone number, email, and social media icons.

13. **JavaScript**: The JavaScript code is linked to the HTML file using the `<script>` tag. The JavaScript file (`index.js`) contains the logic and functionality for the website, such as handling form submissions or implementing interactive features.

Overall, this code represents a basic structure and layout for a gym website, with different sections providing information about the gym, its services, plans, and contact details.

DATA DICTIONARY:

1. user Table:

- user_id: unique identifier for each member (primary key)
- first_name: first name of the member
- last_name: last name of the member
- email: email address of the member
- phone_number: contact phone number of the member
- address: address of the member
- date_of_birth: date of birth of the member
- gender: gender of the member
- membership_type: type of membership (e.g., monthly, yearly)
- start_date: start date of the membership
- end_date: end date of the membership
- created_at: timestamp indicating when the member was added

- updated_at: timestamp indicating the last update to the member's details

2. Membership Payment Table:

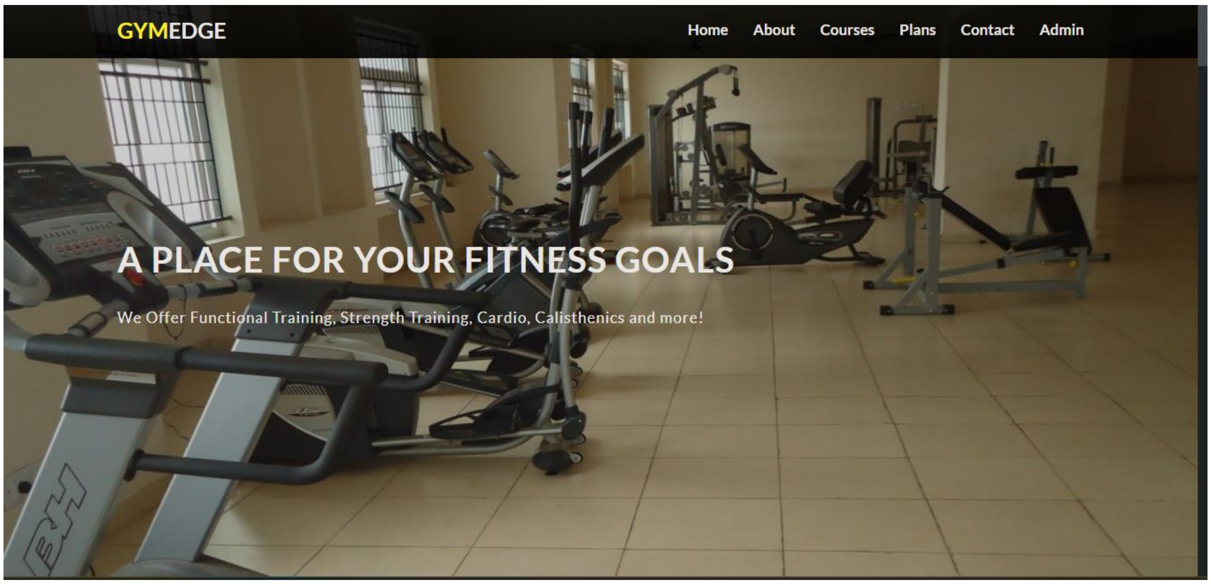
- payment_id: unique identifier for each payment (primary key)
- member_id: foreign key referencing the member the payment belongs to
- payment_date: date of the payment
- amount: amount paid

3. Attendance Table:

- attendance_id: unique identifier for each attendance record (primary key)
- member_id: foreign key referencing the member attending the class
- class_id: foreign key referencing the class attended

UI DESIGN:

Our website’s UI design:


GYMEDGE
Home
About
Courses
Plans
Contact
Admin
A PLACE FOR YOUR FITNESS GOALS
We Offer Functional Training, Strength Training, Cardio, Calisthenics and more!

The code starts with the document type declaration (<!DOCTYPE html>) and the opening HTML tag.

The head section contains metadata such as character encoding, viewport settings, page title, and references to CSS stylesheets and favicon.

The body section contains the main content of the website.

a. The header (header tag) contains the navigation menu, including a logo, menu items, and an admin link.

b. The main content is wrapped in the main tag and divided into several sections, each represented by a section tag.

The first section is the hero section (section-hero), featuring a heading and a description.

The second section is the "About" section (section-about), providing information about the gym's mission and a link to read more.

The third section is the "Courses" section (section-courses), displaying different courses offered by the gym with corresponding descriptions and images.

The fourth section is the "Offer" section (section-offer), showcasing a special offer for the current semester.

The fifth section is the "Trainers" section (section-coaches), presenting various gym plans with their prices, features, and images.

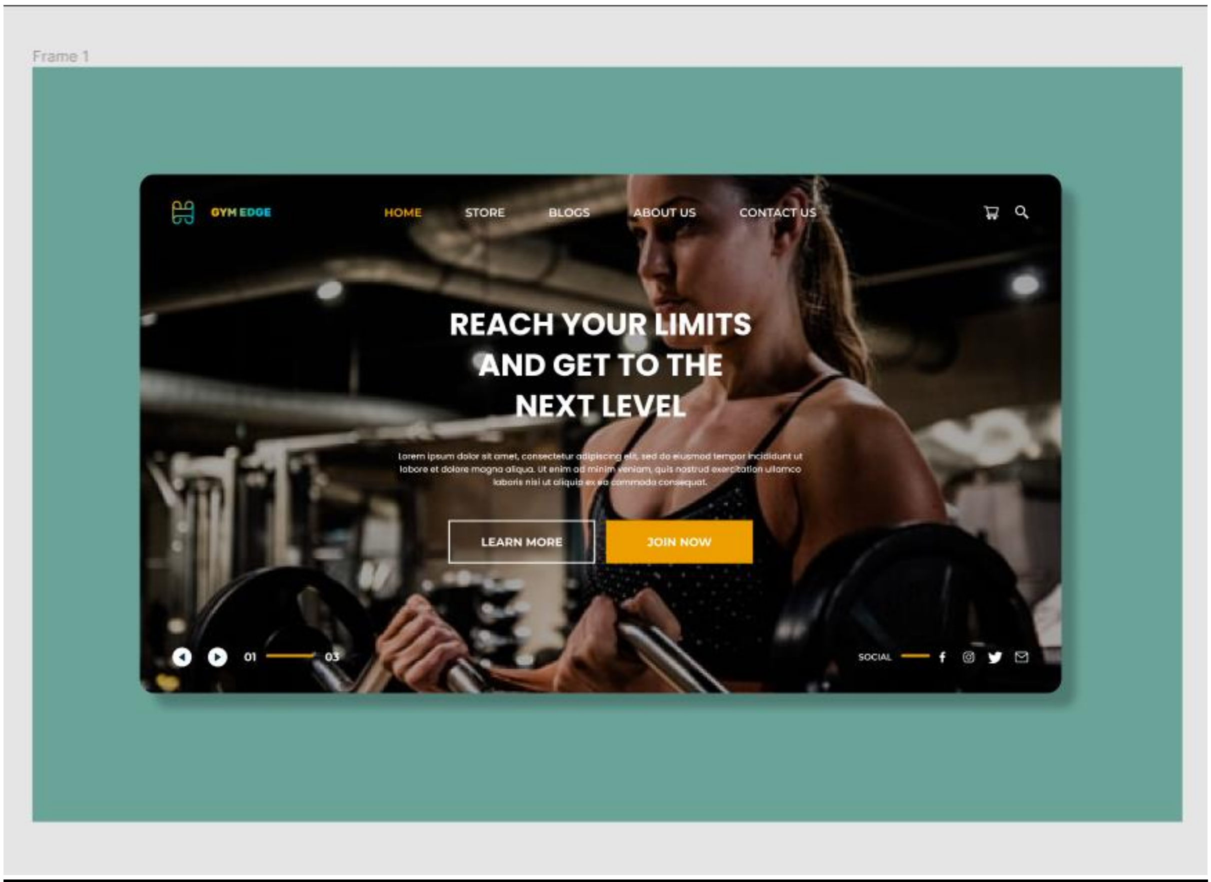
The sixth section is the "Review" section (section-review), featuring testimonials from gym members.

The seventh section is the "Contact" section (section-contact), providing a form for users to get in touch and a map location.

c. The footer (footer tag) contains contact details of the gym, a navigation menu, a newsletter section, and legal information.

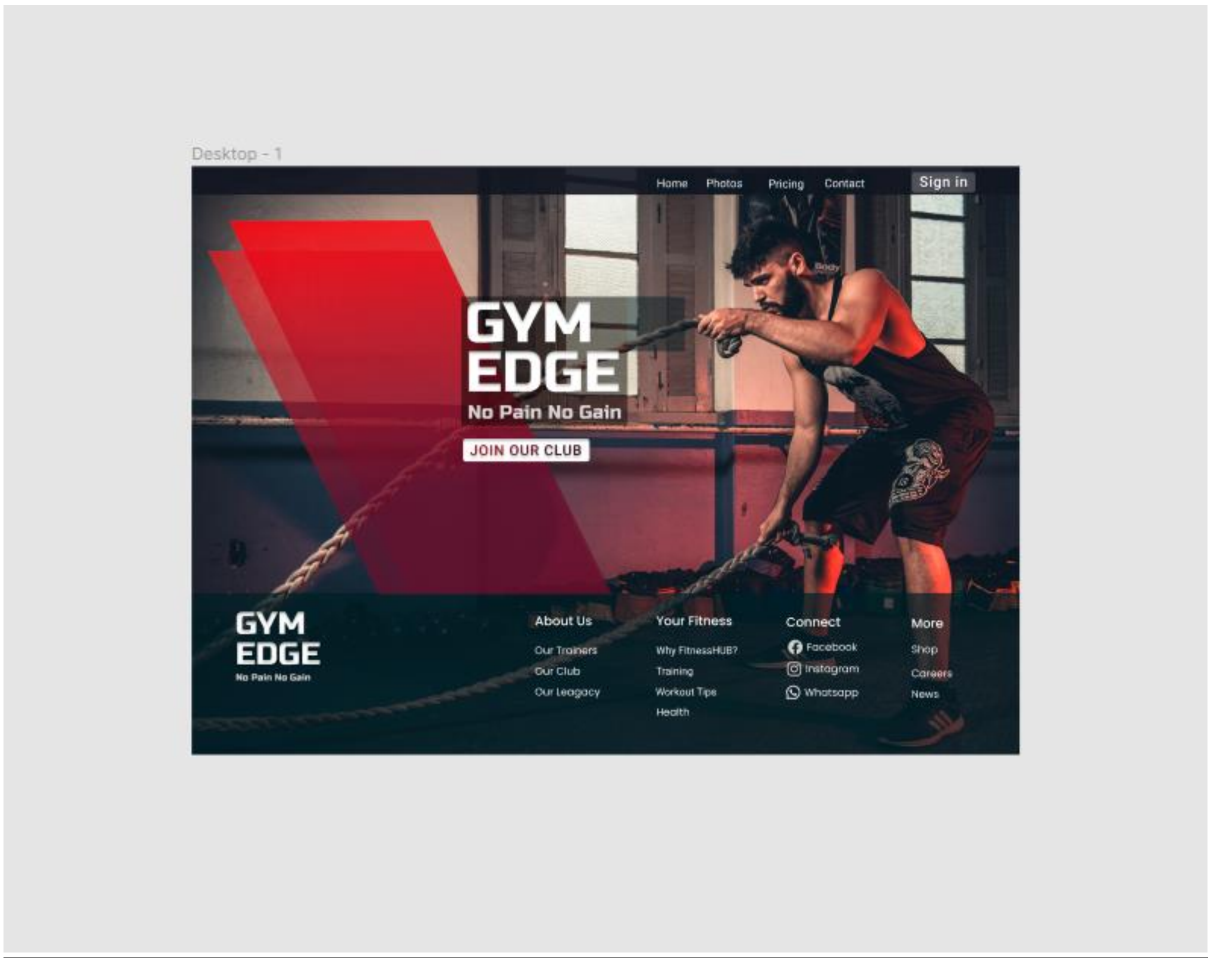
The code includes a reference to an external JavaScript file (index.js) at the end, which is used for additional functionality.

<u>Some additional Figma UI FILES which showcase different designs:</u>



REACH YOUR LIMITS AND GET TO THE NEXT LEVEL

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



OVERALL ARCHITECTURE:

The gym management website follows a client-server architecture, where the system is divided into two main components: the client-side and the server-side. The client-side refers to the user interface and functionality presented to the website users, while the server-side handles the processing, storage, and management of data.

Client-Side Architecture:

The client-side architecture of the gym management website is built using a combination of HTML, CSS, and JavaScript to provide an interactive and responsive user interface. The website is designed to be accessed through standard web browsers, ensuring compatibility across different devices and platforms. The client-side architecture incorporates the following components:

1. **Web Browser**: The website is accessed and rendered using popular web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge.

2. **HTML/CSS**: Hypertext Markup Language (HTML) is used to structure the content and layout of web pages, while Cascading Style Sheets (CSS) is used for presentation and styling.

3. **JavaScript**: JavaScript is used to enhance the user experience by providing dynamic and interactive features, such as form validation, data manipulation, and asynchronous communication with the server.

Server-Side Architecture:

The server-side architecture of the gym management website handles the processing, storage, and retrieval of data, as well as the execution of business logic. It consists of the following components:

1. **Web Server**: A web server, such as Apache HTTP Server or Nginx, receives and processes client requests, and delivers the appropriate responses.

2. **Application Server**: The application server hosts the gym management application and provides the necessary runtime environment to execute server-side code. It handles business logic, data processing, and interaction with the database.

3. **Database**: A relational database management system (RDBMS), such as MySQL or PostgreSQL, is used to store and manage the gym management website's data. It stores information about users, members, classes, payments, and other relevant entities.

4. **Server-Side Programming**: The server-side code is typically written in a server-side programming language like PHP, Python, or Java. It handles user authentication, data validation, database interactions, and implements the business logic of the system.

3.3 Communication Protocol

The client-side and server-side components communicate with each other using the HTTP (Hypertext Transfer Protocol) protocol. HTTP is a widely adopted protocol for communication between web browsers and servers. It ensures the secure and reliable transfer of data, including requests and responses, between the client and the server.

The client-side initiates HTTP requests to interact with the server-side, such as retrieving member information, submitting forms, or making payments. The server-side processes these requests, executes the necessary operations, and returns the appropriate HTTP responses, including data, status codes, and error messages.

The client-server architecture provides a scalable and modular structure for the gym management website, allowing for efficient handling of user requests, separation of concerns between the user interface and server-side logic, and centralized data management through the database server.

Non-Functional Requirements:

Performance: The system should be able to handle a high volume of users and transactions without significant delays or performance issues.

Scalability: The system should be scalable to accommodate future growth and increased user demands.

Reliability: The system should be reliable, with minimal downtime and the ability to recover quickly in case of failures.

Security: The system should have robust security measures in place to protect sensitive member information and prevent unauthorized access.

Usability: The system should have a user-friendly interface and intuitive navigation to ensure ease of use for both staff and members.

Accessibility: The system should be accessible to users with disabilities, complying with accessibility standards and guidelines.

Compatibility: The system should be compatible with various devices, browsers, and operating systems to ensure widespread usability.

Data Integrity: The system should ensure the accuracy and integrity of data, preventing data loss or corruption.

Maintenance and Support: The system should be easy to maintain and should have reliable technical support available to address any issues or provide assistance when needed.

User requirements :

1. User Roles:

a. Users: Regular gym members who can access their membership details, view available training slots.

b. Gym Instructors: Trainers who can view their schedules, view member details and training record

c. Admins: System administrators who have overall control and access to manage memberships, transactions, training slots, and user accounts.

2. Authentication and Authorization:

a. gym instructors, and admins should have unique login credentials to access the system.

b. Access levels and permissions should be defined based on user roles. For example, admins will have full access, while gym instructors can only manage their assigned tasks, and users can only view the website and can pay for membership and contact the admins

3. Membership Management:

a. Users should be able to register for gym memberships, providing personal details and selecting appropriate membership slot by paying on the vtop gym payment system.

b. Membership details should include membership type, duration, start and end dates, and payment status.

c. The system should allow admins to manage memberships, including creating, modifying, and canceling memberships.

4. Transactions:

a. The system should track and record financial transactions related to membership fees, additional services, and product purchases.

b. Users should be able to view their transaction history, including payment dates, amounts, and itemized details.

c. Admins should have access to manage and generate financial reports, including transaction summaries and invoices.

5. Training Slot Management:

a. Gym instructors should be able to manage their schedules.

b. The system should prevent overbooking by enforcing limits on the number of participants per training slot.

6. Reporting and Analytics:

The system should provide various reports and analytics for admins, such as membership statistics, financial summaries, and attendance records.

7. Data Security:

a. The system should ensure the confidentiality and integrity of user data by implementing appropriate security measures, such as encryption and access controls.

b. Regular data backups should be performed to prevent data loss.

Constraints:

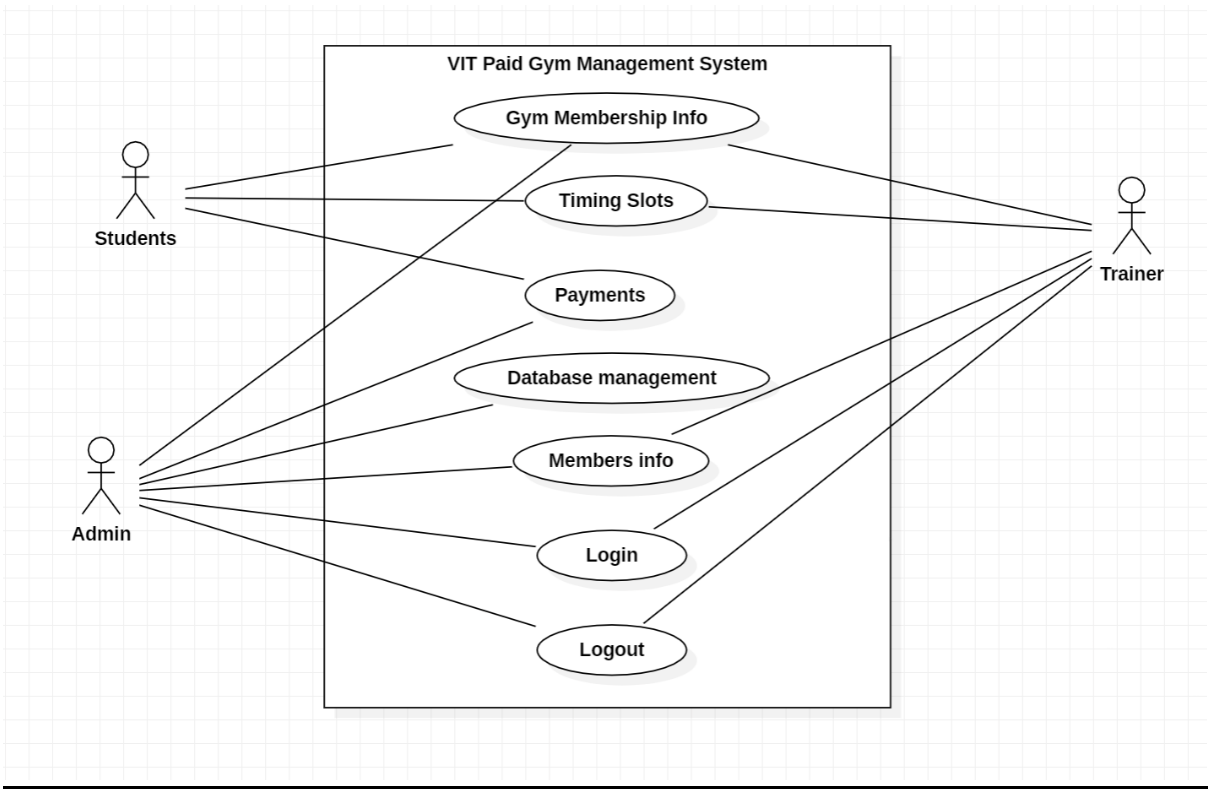
Budget constraints: The system development and implementation should be within the allocated budget.

Time constraints: The system should be developed and deployed within a specified timeline.

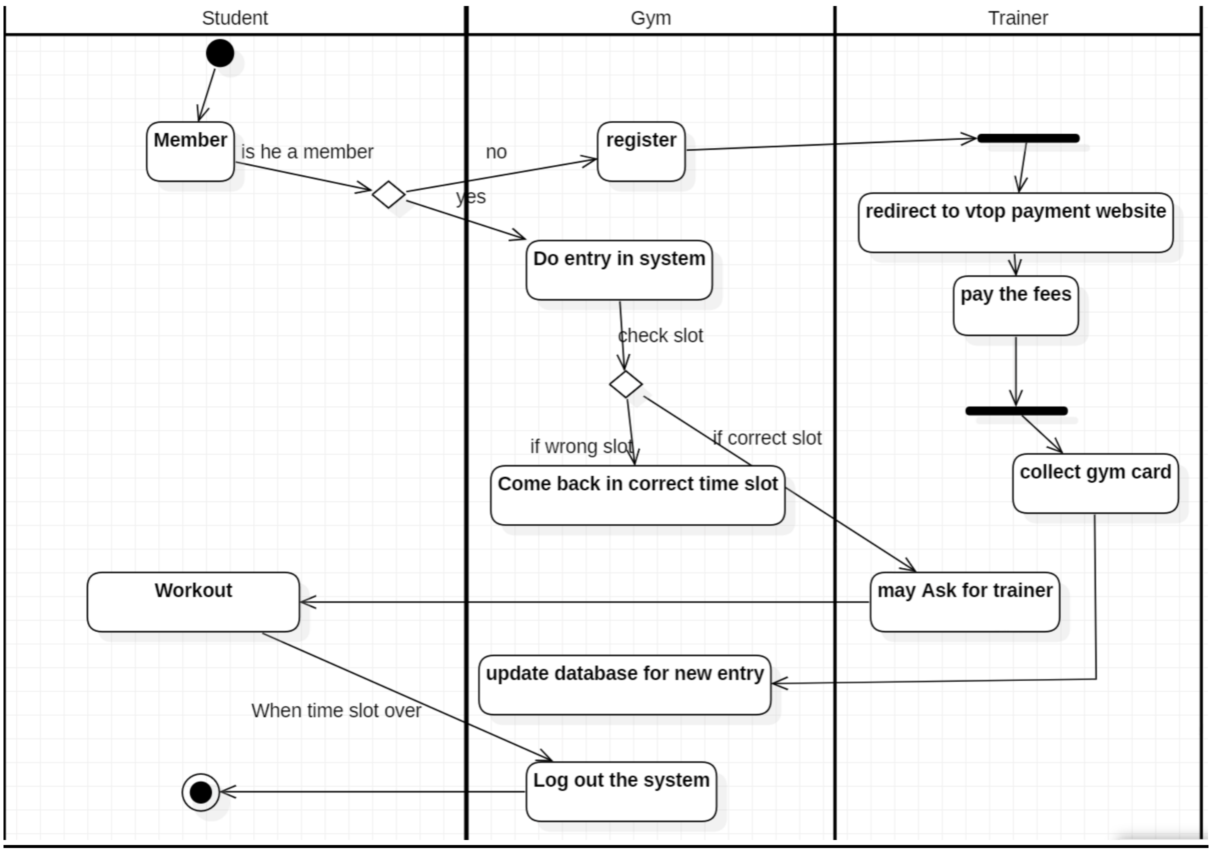
Technical constraints: The system should be compatible with existing hardware, software, and network infrastructure.

Resource constraints: Adequate resources such as skilled developers, hardware, and software should be available for system development.

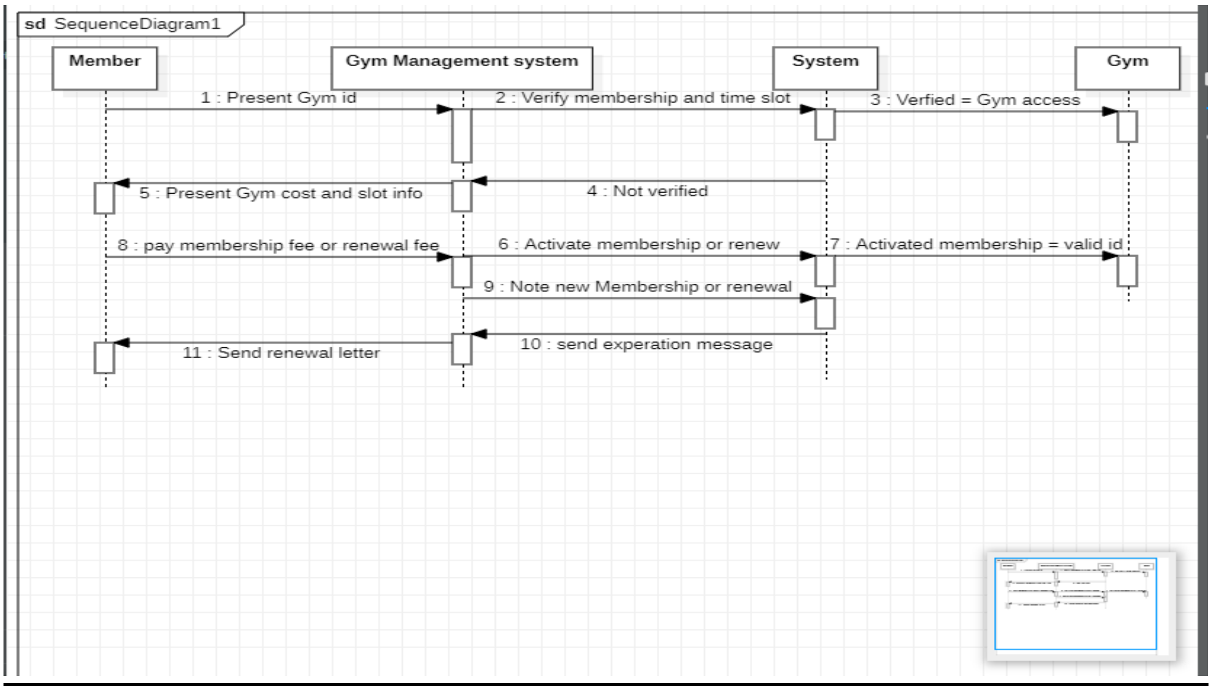
USE CASE DAIGRAM:



ACTIVITY DIAGRAM:

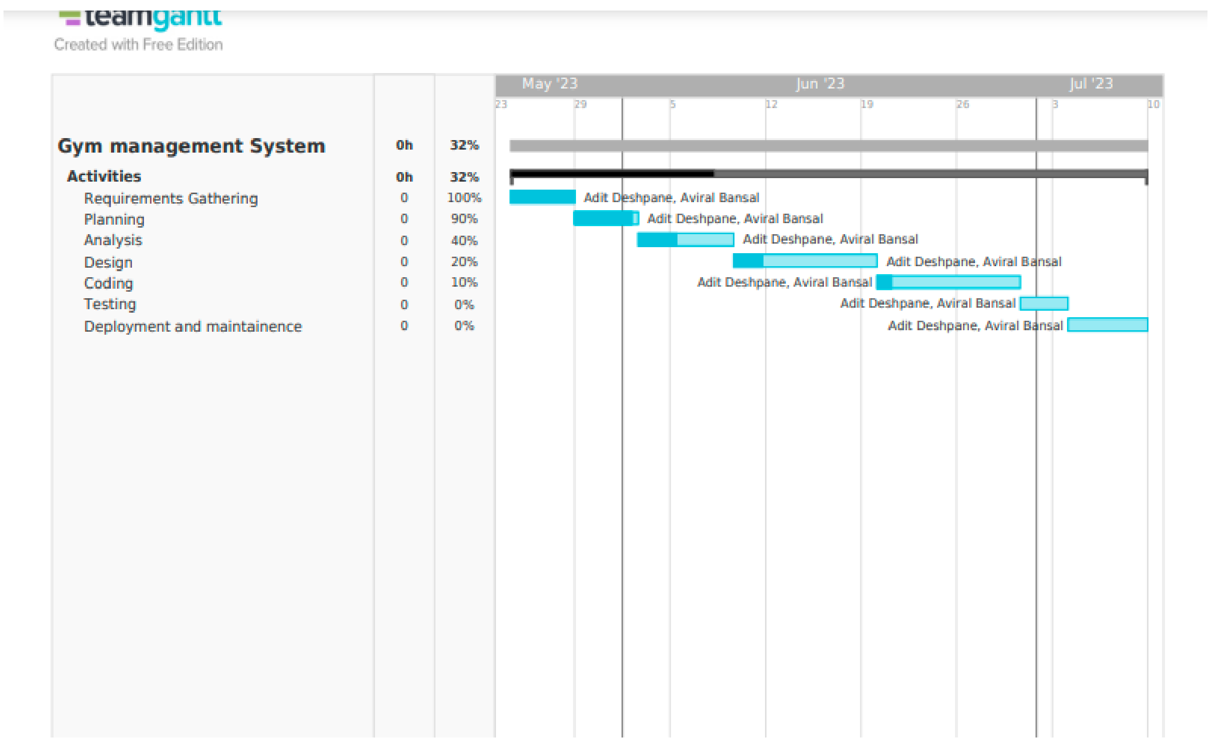


SEQUENCE DIAGRAM:



sd SequenceDiagram1
Member
Gym Management system
System
Gym
1 : Present Gym id
2 : Verify membership and time slot
3 : Verfied = Gym access
4 : Not verified
5 : Present Gym cost and slot info
8 : pay membership fee or renewal fee
6 : Activate membership or renew
7 : Activated membership = valid id
9 : Note new Membership or renewal
10 : send experation message
11 : Send renewal letter

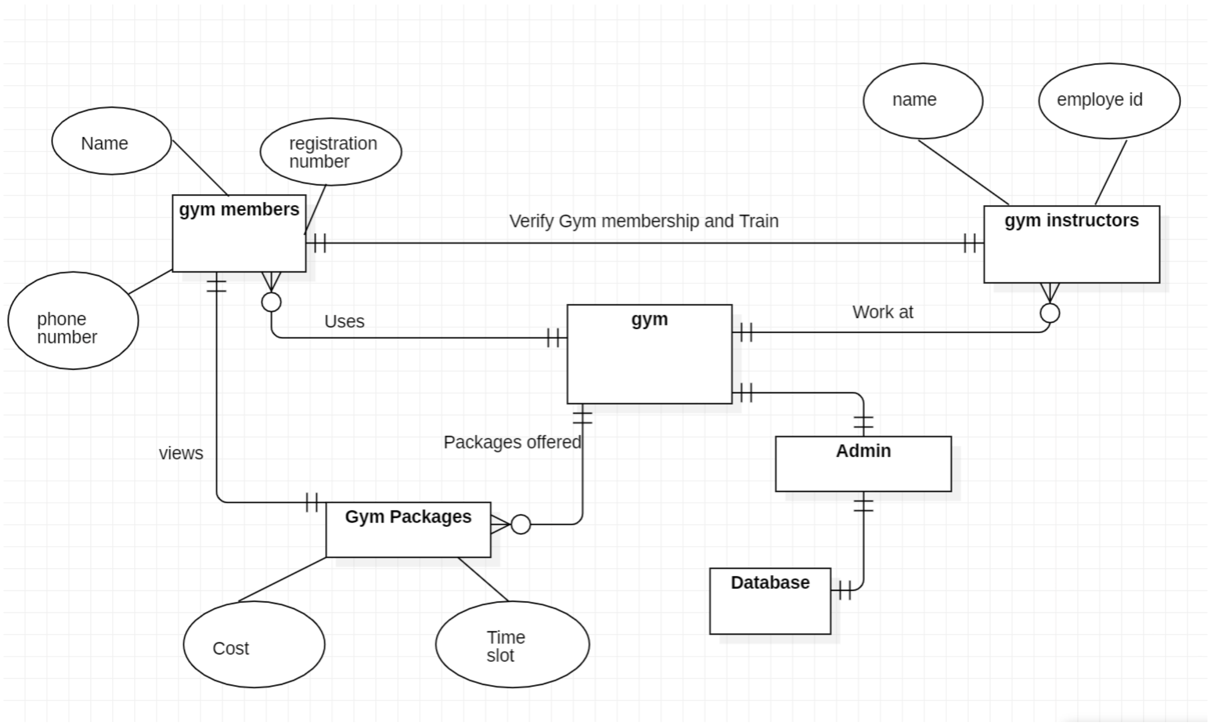
GANTT CHART:



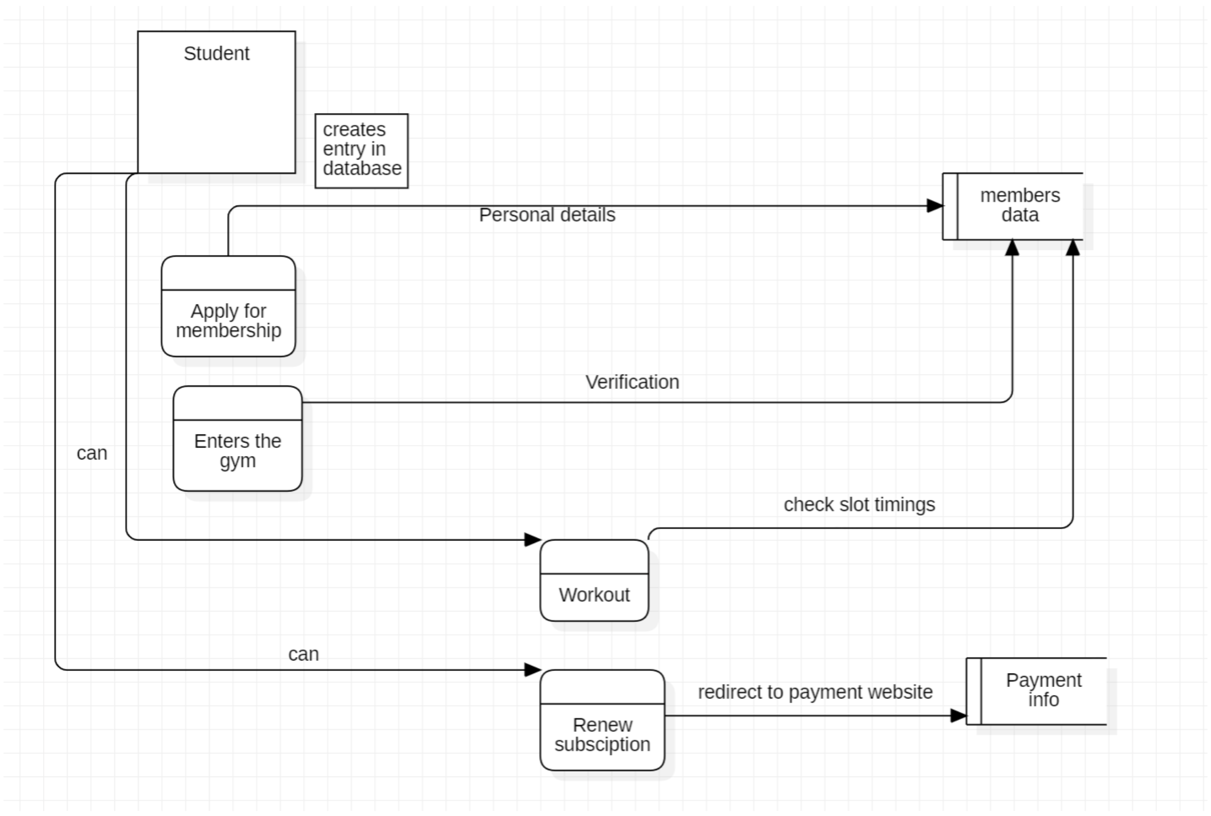
teamgantt
Created with Free Edition

Gym management System	**0h**	**32%**	
Activities	**0h**	**32%**	
Requirements Gathering	0	100%	Adit Deshpane, Aviral Bansal
Planning	0	90%	Adit Deshpane, Aviral Bansal
Analysis	0	40%	Adit Deshpane, Aviral Bansal
Design	0	20%	Adit Deshpane, Aviral Bansal
Coding	0	10%	Adit Deshpane, Aviral Bansal
Testing	0	0%	Adit Deshpane, Aviral Bansal
Deployment and maintainence	0	0%	Adit Deshpane, Aviral Bansal

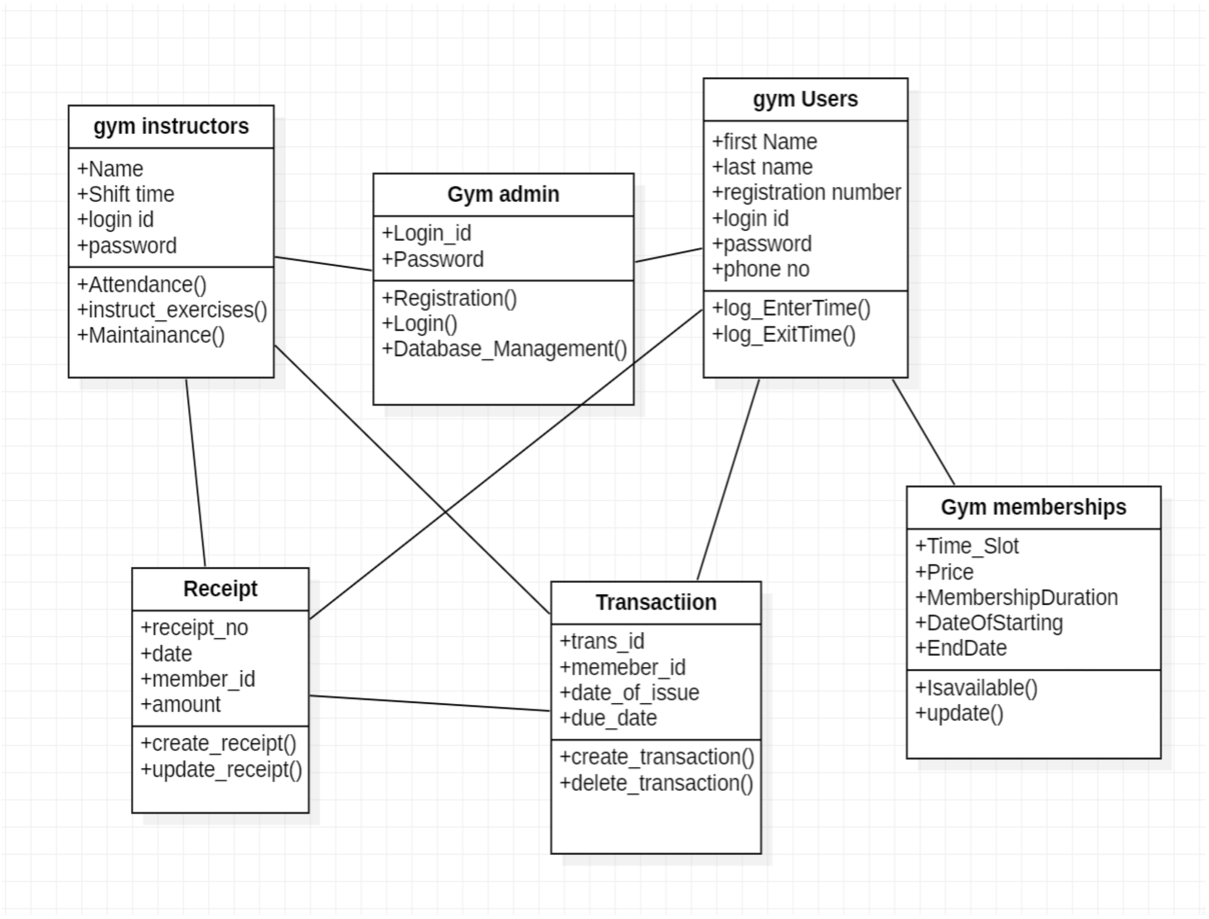
ER DIAGRAM:



DATA FLOW DIAGRAM:



Class Diagram:



SAMPLE SOURCE CODE:

1. HTML

```html
<!DOCTYPE html>
<html lang="zxx">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta
      name="description"
      content="MFitness reponsive webite build using HTML5, CSS3 and
Javascript The complete layout of the page is build using grid layout and
flexbox with some cool animations."
    />
    <title>GYM EDGE- A VIT Chennai Gym</title>
    <link rel="icon" type="image/png" sizes="32x32" href="/favicon.png" />
    <link rel="stylesheet" href="fonts/stylesheet.css" media="screen" />
    <link rel="stylesheet" href="css/style.css" media="screen" />
  </head>
  <body>
    <!-- HEADER -->
    <header class="header" id="home">
      <div class="container">
        <nav class="header-navigation" aria-label="navigation">
          <div class="logo"><span class="yellow">GYM</span>EDGE</div>

          <img
            src="img/nav-icon.svg"
            alt="navigation icon"
            class="nav-icon"
            width="30"
            height="30"
          />

          <ul>
            <li>
              <a href="#">Home</a>
            </li>
            <li>
              <a href="#about">About</a>
            </li>
            <li>
              <a href="#courses">Courses</a>
            </li>
            <li>
              <a href="#trainers">Plans</a>
            </li>
            <li>
              <a href="#contact">Contact</a>
            </li>
            <li>
              <a href="http://127.0.0.1:8000/accounts/login/">Admin</a>
            </li>
          </ul>
        </nav>
      </div>
    </header>

    <main>
      <!-- HERO SECTION -->
      <section class="section-hero">
        <div class="container hero-box">
          <div class="hero-content margin-bottom">
            <h1 class="heading heading--1 margin-bottom">
              A place for your fitness goals
            </h1>
            <p class="description">
              We Offer Functional Training, Strength Training, Cardio,
Calisthenics and more!
            </p>
          </div>

        </div>
      </section>

      <!-- SECTION ABOUT -->
      <section class="section-about" id="about">
        <div class="about-box container">
          <div class="about-box__content" data-aos="fade-up">
            <h2 class="heading heading--2">
              About <span class="yellow">GYM</span>EDGE
            </h2>
            <p>
              Welcome to our gym! We are dedicated to providing a safe and
welcoming space for individuals of all fitness levels. Our mission is to help
our members achieve their fitness goals by offering state-of-the-art equipment
and expert guidance from our certified trainers. Whether you're a seasoned
athlete or just starting your fitness journey, we have everything you need to
succeed. Join our community and start reaching your fitness goals today!
            </p>
            <a href="#" class="link-button">Read More &rarr;</a>
          </div>
          <figure class="about-box__image" data-aos="fade-up">
            <img
```

```html
              src="img/about-img-1.jpg"
              alt="about-image-one"
              width="350"
              height="233"
            />
            <img
              src="img/about-img-2.jpg"
              alt="about-image-twwo"
              width="350"
              height="233"
            />
            <img
              src="img/about-img-3.jpg"
              alt="about-image-three"
              width="420"
              height="280"
            />
          </figure>
        </div>
      </section>

      <!-- SECTION COURSES -->
      <section class="section-courses" id="courses">
        <div class="container courses-box" data-aos="fade-up">
          <header class="courses-heading">
            <h2 class="heading heading--2 underline">Our Courses</h2>
            <p>
              At our gym, we offer a wide variety of classes to suit every
fitness level and interest. From high-energy cardio to strength training and
yoga, our certified instructors are here to guide you towards your fitness
goals. Our group classes are designed to motivate and challenge you while
providing a fun and supportive atmosphere. Whether you're looking to improve
your endurance, gain strength, or simply relieve stress, we have a class for
you. So come on in and try one of our classes today!
            </p>
          </header>

          <article class="courses-content">
            <div class="class-card">
              <img
                class="class-card__img"
                src="./img/courses-bodybuilding.jpg"
                alt="body building course"
                width="550"
                height="550"
              />

              <div class="class-card__overlay">
```

```html
                <div class="text">
                  <p>
                    Wide range of equipments and machines to help
                    you to achieve your bodybuilding goals!
                  </p>


                </div>
              </div>

              <div class="class-card__title">Body Building</div>
            </div>

            <div class="class-card">
              <img
                class="class-card__img"
                src="./img/courses-crossfit.jpg"
                alt="cross fit course"
                width="550"
                height="550"
              />
              <div class="class-card__overlay">
                <div class="text">
                  <p>
                    A dedicated Cross Fit section helps to break away the
sweat and put your body to a good workout.

                  </p>


                </div>
              </div>
              <div class="class-card__title">Cross Fit</div>
            </div>

            <div class="class-card">
              <img
                class="class-card__img"
                src="./img/courses-gymnastic.jpg"
                alt="gymnastic course"
                width="550"
                height="550"
              />

              <div class="class-card__overlay">
                <div class="text">
                  <p>
```

```html
                    Get yourself competion ready by practicing and improving
your gymnastic skills!
                  </p>


                </div>
              </div>

              <div class="class-card__title">Gymnastic</div>
            </div>

            <div class="class-card">
              <img
                class="class-card__img"
                src="./img/courses-fitness.jpg"
                alt="fitness course"
                width="550"
                height="550"
              />
              <div class="class-card__overlay">
                <div class="text">
                  <p>
                    Shed those Pounds and kep yourself in good shape with
Cardio&Fitness.
                  </p>


                </div>
              </div>
              <div class="class-card__title">Fitness</div>
            </div>

            <div class="class-card">
              <img
                class="class-card__img"
                src="./img/courses-TRX.jpg"
                alt="fitness course"
                width="550"
                height="550"
              />
              <div class="class-card__overlay">
                <div class="text">
                  <p>
                    Your Calisthenics journey starts here! Star those pullups
right now!
                  </p>
                </div>
              </div>
              <div class="class-card__title">Calisthenics</div>
            </div>

            <div class="class-card">
              <img
                class="class-card__img"
                src="./img/courses-boxing.jpg"
                alt="fitness course"
                width="550"
                height="550"
              />
              <div class="class-card__overlay">
                <div class="text">
                  <p>
                    Get that pent up anger out and unleash your inner CREED!
                  </p>


                </div>
              </div>
              <div class="class-card__title">Boxing</div>
            </div>
          </article>
        </div>
      </section>

      <!-- SECTION OFFER -->
      <section class="section-offer" id="offer">
        <div class="offer-box">
          <div class="offer" data-aos="fade-up">
            <h2 class="heading heading--2 margin-bottom">
              Special offer this semester, get full Benefits along with access
to the Table Tennis and Pool Table.
            </h2>
            <p class="margin-bottom">
              Join our Full AC Gym with In-house music system and shower rooms
for both men and women.
              Choose from the plenty of equipments and machines and get
healthy!
            </p>

          </div>

          <div class="offer-img"></div>
        </div>
      </section>
```

```html
      <!-- SECTION COACHES -->
      <section class="section-coaches" id="trainers">
        <div class="container coaches-box" data-aos="fade-up">
          <header class="coache-heading">
            <h2 class="heading heading--2 underline margin-bottom">
              OUR PLANS
            </h2>
            <p>
              Here are the plans offered by our Gym. Flexible plans for all
kinds of users.
              Half the money will be refunded if the customer wnats to leave
the gym.
              Full cashback guarantee for first 30 days to come try the gym.
            </p>
          </header>

          <article class="coache-content">
            <div class="c-card">
              <img
                src="img/coache-1.jpg"
                alt="coache one"
                class="c-card__img"
                width="550"
                height="550"
              />



              <div class="c-card__content">
                <p class="c-name">PRICE: 2000 INR</p>
                <p class="c-title">No shower room <br></p>
                <p class = "c-title">Rigid with Timing</p>
                <p class="c-title">No Trainer help<br></p>
              </div>
            </div>

            <div class="c-card">
              <img
                src="img/coache-2.jpg"
                alt="coache one"
                class="c-card__img"
                width="550"
                height="550"
              />
              <div class="c-card__content">
                <p class="c-name">PRICE : 3500 INR</p>
                <p class="c-title">Shower room allowed <br></p>
                <p class="c-title">Rigid with timing</p>
                <p class="c-title"> Trainer help given<br></p>
              </div>
            </div>

            <div class="c-card">
              <img
                src="img/coache-3.jpg"
                alt="coache three"
                class="c-card__img"
                width="550"
                height="550"
              />


              <div class="c-card__content">
                <p class="c-name">PRICE: 6000 INR</p>
                <p class="c-title">Benefits of previous plans</p>
                <p class="c-title">Flexible with timing</p>

              </div>
            </div>
          </article>
          <p class="paying">For payment and reciept:</p>
          <button class="btn btn__primary vtop"><a
href="https://vtopcc.vit.ac.in/vtop/open/page">GO</a></button>
        </div>
      </section>

      <!-- SECTION REVIEW -->
      <section class="section-review" id="review">
        <div class="review-box container" data-aos="fade-up">
          <h2 class="heading heading--2 underline margin-bottom grid-center">
            what people say
          </h2>

          <div class="review-card">
            <img
              src="img/review-img-1.jpg"
              alt="user"
              class="review-card__image"
              width="100"
              height="100"
            />
            <blockquote class="review-card__content">
```

```html
              <p class="text">
                A very good experience. The gym is great with so many
facilities. Helped me get down my weight from 90kgs
                to 60kgs and build my muscles. Would recommened everyone to
join!
              </p>

              <div class="name">Shrestha Nagpal</div>
            </blockquote>
          </div>

          <div class="review-card">
            <img
              src="img/review-img-2.jpg"
              alt="user"
              class="review-card__image"
              width="100"
              height="100"
            />
            <blockquote class="review-card__content">
              <p class="text">
                I have been a member of this gym for 2 years now. I have been
able to maintain my weight and stay fit.
                The trainers are very helpful and the gym is very clean.

              </p>
              <div class="name">Somya Kapoor</div>
            </blockquote>
          </div>
        </div>
      </section>

      <!-- SECTION CONTACT -->
      <section class="section-contact" id="contact">
        <div class="contact-box">
          <form  class="form-grp" data-aos="fade-up">
            <h2 class="heading heading--2 margin-bottom">
              Get in <span class="yellow">Touch</span>
            </h2>

            <input type="text" class="form form__input" placeholder="Name" />
            <input type="text" class="form form__input" placeholder="Email" />
            <input type="text" class="form form__input" placeholder="Phone" />
            <textarea
              name="message"
              cols="1"
              rows="1"
              class="form form__input"
              placeholder="Block"
            ></textarea>
            <button type="button" class="btn btn__primary form-flex Aviral"
id="mack">Send</button>
          </form>
          <div id="mapid" data-aos="fade-up"></div>
        </div>
      </section>
    </main>

    <!-- FOOTER -->
    <footer class="section-footer" id="footer">
      <div class="footer-box container">
        <div class="contact-details">
          <h2 class="margin-bottom"><span class="yellow">GYM</span>EDGE</h2>

          <ul class="contact-data">
            <li class="location">VIT CHENNAI, KELAMBAKKAM, CHENNAI</li>
            <li class="phone">+91 9704567841</li>
            <li class="email">gymedge@gmai.com</li>
            <li class="social">
              <img
                src="./img/logo-whatsapp.svg"
                alt="whatsapp icon"
                width="35"
                height="35"
              />

              <img
                src="./img/logo-facebook.svg"
                alt="facebook icon"
                width="35"
                height="35"
              />

              <img
                src="./img/logo-instagram.svg"
                alt="instagram icon"
                width="35"
                height="35"
              />

              <img
                src="./img/logo-twitter.svg"
                alt="twitter icon"
                width="35"
                height="35"
              />
            </li>
          </ul>
        </div>

        <nav class="footer-nav" aria-label="navigation">
          <div class="nav-name">Quick Links</div>
          <ul>
            <li><a href="#home">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#courses">Courses</a></li>
            <li><a href="#trainers">Plans</a></li>
            <li><a href="#contact">Contact</a></li>
          </ul>
        </nav>

        <div class="newsletter">
          <div class="newsletter__title">About College</div>
          <div class="newsletter__text">
            Interested in knowing more about the college? Click here!
          </div>

          <button class="btn btn__primary"><a
href="https://chennai.vit.ac.in/">Take a Tour</a></button>
        </div>

        <div class="legel">
          <p class="text">Copyright &copy; by Aviral&nbsp;</p>
          <a href="https://github.com/Aviral1611">
            <img
              src="img/github.svg"
              alt="github"
              class="github"
              width="35"
              height="35"
            />
          </a>
        </div>
      </div>
    </footer>

    <script src="./index.js"></script>
  </body>
</html>
```

2. CSS

```css
:root {
  --color-yellow: hsl(56, 99%, 52%);
  --color-yellow-dark: hsl(56, 99%, 32%);
  --color-yellow-light: hsl(56, 99%, 82%);
  --color-black: hsl(0, 0%, 15%);
  --color-black-light: hsl(0, 0%, 22%);
  --color-red: hsl(359, 77%, 56%);
  --grid-column-gap: clamp(2rem, 6vw, 8rem);
  --grid-row-gap: 4rem;
  --two-col-layout: 2;
  --three-col-layout: 3;
}
@media only screen and (max-width: 56.25em) {
  :root {
    --two-col-layout: 1;
  }
}
@media only screen and (max-width: 59em) {
  :root {
    --three-col-layout: 2;
  }
}
@media only screen and (max-width: 37.5em) {
  :root {
    --three-col-layout: 1;
  }
}

*,
*::before,
*::after {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

html {
  box-sizing: border-box;
  font-size: 62.5%;
  scroll-behavior: smooth;
}

body {
  font-family: 'Lato', sans-serif;
  font-weight: 400;
  font-size: 1.6rem;
  line-height: 1.6;
  color: #fff;
  overflow-y: scroll;
```

```css
}

.container {
  max-width: clamp(50rem, 85vw, 114rem);
  padding: 0 2.4rem;
  margin: 0 auto;
}

section:nth-child(odd) {
  background-color: var(--color-black-light);
}

section:nth-child(even) {
  background-color: var(--color-black);
}

.header {
  position: absolute;
  top: 0;
  right: 0;
  color: #fff;
  width: 100%;
  z-index: 100;
  background: linear-gradient(rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 0.9));
}

.header-navigation {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1.2rem 0;
}
.header-navigation ul {
  display: flex;
  text-transform: capitalize;
  font-size: 1.7rem;
  list-style: none;
  font-weight: 700;
}
@media only screen and (max-width: 50em) {
  .header-navigation ul {
    display: none;
  }
}
.header-navigation ul li {
  padding-left: 2.8rem;
}
.header-navigation ul li a {
  text-decoration: none;
  color: #fff;
  padding: 1.8rem 0;
  border-bottom: 2px solid transparent;
  transition: border 0.5s;
}
.header-navigation ul li a:hover {
  border-bottom: 2px solid var(--color-yellow);
}
.header-navigation .logo {
  font-size: 2.5rem;
  font-weight: 700;
}
.header-navigation .nav-icon {
  display: none;
  border: 1px solid #fff;
}
@media only screen and (max-width: 50em) {
  .header-navigation .nav-icon {
    display: block;
  }
}

.heading {
  font-weight: 700;
  text-transform: uppercase;
}
.heading--1 {
  font-size: clamp(2.5rem, 4vw, 4rem);
}
.heading--2 {
  font-size: clamp(2rem, 4vw, 3rem);
}

.btn {
  text-transform: uppercase;
  padding: 1rem clamp(1.2rem, 3vw, 3rem);
  border-radius: 3px;
  font-family: inherit;
  font-size: 1.5rem;
  font-weight: 700;
  text-transform: uppercase;
  border: none;
  cursor: pointer;
}
.btn__primary {
  background-color: var(--color-yellow);
  color: #000;
  border: 1px solid transparent;
  transition: background-color 0.3s;
}
.btn__primary:hover {
  background-color: var(--color-yellow-dark);
}
.btn__secondary {
  background-color: transparent;
  color: #fff;
  border: 1px solid var(--color-yellow);
  transition: all 0.3s;
}
.btn__secondary:hover {
  background-color: var(--color-yellow-light);
  border: 1px solid var(--color-yellow-light);
  color: #000;
}

.link-button {
  text-decoration: none;
  padding: 0.4rem 0;
  color: #fff;
  border-bottom: 2px solid var(--color-yellow);
}
.link-button:hover {
  font-style: italic;
}

.form {
  padding: 0.8rem clamp(1rem, 3vw, 3rem);
  border-radius: 3px;
  font-family: inherit;
  font-size: 1.6rem;
  border: none;
  width: 100%;
}
.form:focus {
  outline: none;
}
.form__input {
  border-bottom: 2px solid #fff;
  background-color: transparent;
  color: #fff;
  resize: none;
}
.form__input::placeholder {
  color: rgba(255, 255, 255, 0.7);
}
```

```css
.section-about {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.about-box {
  display: grid;
  grid-template-columns: repeat(
    var(--two-col-layout),
    minmax(min-content, 1fr)
  );
  align-items: flex-start;
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
}
.about-box__content {
  display: grid;
  gap: 2rem;
  justify-items: flex-start;
  max-height: 100%;
}
.about-box__image {
  align-self: center;
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  grid-template-rows: repeat(2, 1fr);
  gap: 2rem;
  justify-items: center;
  align-items: center;
}
.about-box__image img {
  background-size: cover;
  background-position: center;
  max-width: 100%;
  height: auto;
  outline: 0.2rem solid;
}
.about-box__image img:last-child {
  grid-column: 1/-1;
  max-width: 60%;
  margin-top: -5rem;
  outline: 0.5rem solid var(--color-yellow);
  outline-offset: 0.5rem;
}

.section-courses {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}
```

```
.courses-box .courses-heading {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  gap: 2rem;
  text-align: center;
  margin-bottom: 4rem;
  max-height: 100%;
}

.courses-box .courses-content {
  display: grid;
  grid-template-columns: repeat(
    var(--three-col-layout),
    minmax(min-content, 1fr)
  );
  gap: 3rem;
}
.courses-box .courses-content .class-card {
  position: relative;
}
.courses-box .courses-content .class-card__img {
  border-left: 2px solid var(--color-yellow);
  background-position: center;
  background-size: cover;
  max-width: 100%;
  height: 100%;
}
.courses-box .courses-content .class-card__overlay {
  position: absolute;
  top: 0%;
  left: 0%;
  height: 0%;
  overflow: hidden;
  background-color: rgba(0, 0, 0, 0.7);
  transition: height 0.5s;
}
.courses-box .courses-content .class-card__overlay .text {
  color: #fff;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: 100%;
  padding: 2.5rem;
  text-align: center;
```

```css
}
.courses-box .courses-content .class-card__title {
  position: absolute;
  top: 0;
  left: 0;
  width: 70%;
  padding: 1rem;
  clip-path: polygon(0 0, 100% 0, 70% 100%, 0 100%);
  background: linear-gradient(
    to right,
    var(--color-yellow),
    var(--color-yellow-dark)
  );
  text-transform: capitalize;
  font-weight: 700;
  color: #000;
}
.courses-box .courses-content .class-card:hover .class-card__overlay {
  height: 100%;
}

.offer-box {
  display: grid;
  grid-template-columns: repeat(
    var(--two-col-layout),
    minmax(min-content, 1fr)
  );
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
  align-items: center;
}

.offer {
  max-height: 100%;
  padding: 4rem 2.4rem;
}

.offer-img {
  background-image: linear-gradient(
      to right,
      rgba(254, 239, 27, 0.5),
      rgba(254, 239, 27, 0.3),
      rgba(254, 239, 27, 0.1)
    ),
    url('../img/offer-img.jpg');
  filter: brightness(1.1);
  background-position: center;
  background-size: cover;
  max-width: 100%;
  height: 50rem;
}

.section-coaches {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.coaches-box .coache-heading {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  gap: 2rem;
  text-align: center;
  margin-bottom: 4rem;
  max-height: 100%;
}

.coaches-box .coache-content {
  display: grid;
  grid-template-columns: repeat(
    var(--three-col-layout),
    minmax(min-content, 1fr)
  );
  gap: 3rem;
}
.coaches-box .coache-content .c-card {
  position: relative;
}
.coaches-box .coache-content .c-card__img {
  background-position: center;
  background-size: cover;
  max-width: 100%;
  height: 100%;
  border-left: 2px solid var(--color-yellow);
}
.coaches-box .coache-content .c-card .overlay {
  position: absolute;
  top: 0%;
  right: 0;
  color: #fff;
  width: 0%;
  height: 100%;
  overflow: hidden;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  background-color: rgba(43, 43, 43, 0.9);
  transition: width 0.4s;
}
.coaches-box .coache-content .c-card .overlay * {
  flex: 1;
}
.coaches-box .coache-content .c-card__content {
  position: absolute;
  bottom: 0;
  background: linear-gradient(
    to right,
    var(--color-yellow),
    var(--color-yellow-dark)
  );
  color: #000;
  clip-path: polygon(0 0, 100% 0, 70% 100%, 0 100%);
  width: 70%;
  padding: 0.5rem 1rem;
}
.coaches-box .coache-content .c-card__content .c-name {
  color: blue;
  text-transform: uppercase;
  font-weight: 700;
}
.coaches-box .coache-content .c-card__content .c-title {
  text-transform: capitalize;
}
.coaches-box .coache-content .c-card:hover .overlay {
  width: 16%;
}
.coaches-box .coache-content .c-card:hover .c-card__img {
  filter: brightness(0.8);
}

.section-review {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.review-box {
  display: grid;
  grid-template-columns: repeat(
    var(--two-col-layout),
    minmax(min-content, 1fr)
  );
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
}
```

```css
.review-card {
  color: #fff;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}
.review-card__image {
  max-width: 100%;
  height: auto;
  background-position: center;
  background-size: cover;
  border-radius: 50%;
}
.review-card__content {
  padding: 2rem 0rem;
  display: flex;
  flex-direction: column;
  text-align: center;
  position: relative;
}
.review-card__content .text {
  font-style: italic;
}
.review-card__content .name {
  font-style: normal;
  text-transform: uppercase;
  letter-spacing: 0.2rem;
  padding-top: 1.5rem;
}
.review-card__content::before {
  content: '\201F';
  display: block;
  font-size: 12rem;
  color: rgba(0, 0, 0, 0.8);
  position: absolute;
  top: -7rem;
  left: -1rem;
}

.contact-box {
  display: grid;
  grid-template-columns: repeat(
    var(--two-col-layout),
    minmax(min-content, 1fr)
  );
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
}

.form-grp {
  display: grid;
  row-gap: 3rem;
  padding: 4rem 2.4rem;
}

.form-flex {
  justify-self: flex-start;
}

#mapid {
  height: 55rem;
  width: 100%;
}

.section-footer {
  background-color: var(--color-black);
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.footer-box {
  display: grid;
  grid-template-columns: repeat(
    var(--three-col-layout),
    minmax(min-content, 1fr)
  );
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
  color: #fff;
}

.contact-details .contact-data {
  list-style: none;
  display: flex;
  flex-direction: column;
}
.contact-details .contact-data * {
  padding: 0.4rem 0;
}
.contact-details .contact-data .social {
  display: flex;
}
.contact-details .contact-data .social * {
  padding-right: 1rem;
}
```

```css
.footer-nav .nav-name {
  font-size: 2rem;
  font-weight: 700;
  margin-bottom: 2rem;
}

.footer-nav ul {
  display: flex;
  flex-direction: column;
}

.footer-nav ul li {
  list-style: none;
  padding: 1.5rem 0;
}

.footer-nav ul li a {
  text-decoration: none;
  color: #fff;
}

.footer-nav ul li a::before {
  content: '\2192';
  display: inline-block;
  margin-right: 10px;
  margin-top: -4px;
  color: var(--color-yellow);
}

.newsletter__title {
  font-size: 2rem;
  font-weight: 700;
  margin-bottom: 2rem;
}

.newsletter__text {
  margin-bottom: 1.6rem;
}

.newsletter__input {
  display: flex;
}

.send-icon {
  padding: 0.9rem;
  background-color: #000;
  cursor: pointer;
```

```css
}

.legel {
  border-top: 2px solid #fff;
  padding: 1.5rem 0;
  grid-column: 1/-1;
  text-align: center;
}
.legel .text {
  padding: 1rem 0;
}

.grid-center {
  grid-column: 1/-1;
  text-align: center;
}

.margin-right {
  margin-right: 2rem;
}

.margin-bottom {
  margin-bottom: 2rem;
}

.yellow {
  color: var(--color-yellow);
}

.underline::after {
  content: '';
  display: block;
  width: 80px;
  height: 2px;
  margin: auto;
  background-color: var(--color-yellow);
}

.section-hero {
  background: linear-gradient(rgba(0, 0, 0, 0.2), rgba(0, 0, 0, 0.2)),
    url('../img/header-bg-img.jpg');
  background-size: cover;
  background-position: top;
  height: max(100vh, 60rem);
}

.hero-box {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  justify-content: center;
  height: 100%;
  letter-spacing: 0.1rem;
}
.hero-box .description {
  font-size: 1.7rem;
}

.paying {
  padding-top: 30px;
  padding-bottom: 10px;
  font-size: x-large;
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-
serif;
}
```

3. SASS (used for positioning)

```
.section-about {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.about-box {
  display: grid;
  grid-template-columns: repeat(
    var(--two-col-layout),
    minmax(min-content, 1fr)
  );
  align-items: flex-start;

  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);

  &__content {
    display: grid;
    gap: 2rem;
    justify-items: flex-start;
    max-height: 100%;
  }

  &__image {
    align-self: center;
    display: grid;
    grid-template-columns: repeat(2, 1fr);
```

```css
    grid-template-rows: repeat(2, 1fr);
    gap: 2rem;
    justify-items: center;
    align-items: center;

    & img {
      background-size: cover;
      background-position: center;
      max-width: 100%;
      height: auto;
      outline: 0.2rem solid;

      &:last-child {
        grid-column: 1/-1;
        max-width: 60%;
        margin-top: -5rem;
        outline: 0.5rem solid var(--color-yellow);
        outline-offset: 0.5rem;
      }
    }
  }
}
```

```css
:root {
  //COLOR YELLOW
  --color-yellow: hsl(56, 99%, 52%);
  --color-yellow-dark: hsl(56, 99%, 32%);
  --color-yellow-light: hsl(56, 99%, 82%);

  //BLACK COLOR
  --color-black: hsl(0, 0%, 15%);
  --color-black-light: hsl(0, 0%, 22%);

  //RED COLOR
  --color-red: hsl(359, 77%, 56%);

  //GRID GAP
  --grid-column-gap: clamp(2rem, 6vw, 8rem);
  --grid-row-gap: 4rem;

  //GRID
  --two-col-layout: 2;
  --three-col-layout: 3;

  @media only screen and (max-width: 56.25em) {
    --two-col-layout: 1;
  }
  @media only screen and (max-width: 59em) {
    --three-col-layout: 2;
  }
  @media only screen and (max-width: 37.5em) {
    --three-col-layout: 1;
  }
}

*,
*::before,
*::after {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

html {
  box-sizing: border-box;
  font-size: 62.5%; //1rem = 10px
  scroll-behavior: smooth;
}

body {
  font-family: 'Lato', sans-serif;
  font-weight: 400;
  font-size: 1.6rem;
  line-height: 1.6;
  color: #fff;
  overflow-y: scroll;
}

.container {
  max-width: clamp(50rem, 85vw, 114rem);
  padding: 0 2.4rem;
  margin: 0 auto;
}

section:nth-child(odd) {
  background-color: var(--color-black-light);
}

section:nth-child(even) {
  background-color: var(--color-black);
}
```

```
.section-courses {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.courses-box {
  & .courses-heading {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    gap: 2rem;
    text-align: center;
    margin-bottom: 4rem;
    max-height: 100%;
  }

  & .courses-content {
    display: grid;
    grid-template-columns: repeat(
      var(--three-col-layout),
      minmax(min-content, 1fr)
    );
    gap: 3rem;

    & .class-card {
      position: relative;

      &__img {
        border-left: 2px solid var(--color-yellow);
        background-position: center;
        background-size: cover;
        max-width: 100%;
        height: 100%;
      }

      &__overlay {
        position: absolute;
        top: 0%;
        left: 0%;
        height: 0%;
        overflow: hidden;
        background-color: rgba(0, 0, 0, 0.7);
        transition: height 0.5s;

        & .text {
          color: #fff;
          display: flex;
          flex-direction: column;
          justify-content: center;
          align-items: center;
          height: 100%;
          padding: 2.5rem;
          text-align: center;
        }
      }

      &__title {
        position: absolute;
        top: 0;
        left: 0;
        width: 70%;
        padding: 1rem;
        clip-path: polygon(0 0, 100% 0, 70% 100%, 0 100%);
        background: linear-gradient(
          to right,
          var(--color-yellow),
          var(--color-yellow-dark)
        );
        text-transform: capitalize;
        font-weight: 700;
        color: #000;
      }

      &:hover .class-card__overlay {
        height: 100%;
      }
    }
  }
}
```

```
.section-coaches {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.coaches-box {
  & .coache-heading {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    gap: 2rem;
    text-align: center;
    margin-bottom: 4rem;
    max-height: 100%;
  }

  & .coache-content {
    display: grid;
    grid-template-columns: repeat(
      var(--three-col-layout),
      minmax(min-content, 1fr)
    );
    gap: 3rem;

    & .c-card {
      position: relative;
      &__img {
        background-position: center;
        background-size: cover;
        max-width: 100%;
        height: 100%;
        border-left: 2px solid var(--color-yellow);
      }

      & .overlay {
        position: absolute;
        top: 0%;
        right: 0;
        color: #fff;
        width: 0%;
        height: 100%;
        overflow: hidden;
        display: flex;
        justify-content: center;
        align-items: center;
        flex-direction: column;
        background-color: rgba(43, 43, 43, 0.9);
        transition: width 0.4s;

        * {
          flex: 1;
        }
      }

      &__content {
        position: absolute;
        bottom: 0;
        background: linear-gradient(
          to right,
          var(--color-yellow),
          var(--color-yellow-dark)
        );
        color: #000;
        clip-path: polygon(0 0, 100% 0, 70% 100%, 0 100%);
        width: 70%;
        padding: 0.5rem 1rem;

        & .c-name {
          color: #e5383b;
          text-transform: uppercase;
          font-weight: 700;
        }

        & .c-title {
          text-transform: capitalize;
        }
      }
      &:hover .overlay {
        width: 16%;
      }

      &:hover .c-card__img {
        filter: brightness(0.8);
      }
    }
  }
}
```

```
.btn {
  text-transform: uppercase;
  padding: 1rem clamp(1.2rem, 3vw, 3rem);
  border-radius: 3px;
  font-family: inherit;
  font-size: 1.5rem;
  font-weight: 700;
  text-transform: uppercase;
  border: none;
  cursor: pointer;

  &__primary {
    background-color: var(--color-yellow);
    color: #000;
    border: 1px solid transparent;
    transition: background-color 0.3s;

    &:hover {
      background-color: var(--color-yellow-dark);
    }
```

```scss
  }

  &__secondary {
    background-color: transparent;
    color: #fff;
    border: 1px solid var(--color-yellow);
    transition: all 0.3s;

    &:hover {
      background-color: var(--color-yellow-light);
      border: 1px solid var(--color-yellow-light);
      color: #000;
    }
  }
}

.link-button {
  text-decoration: none;
  padding: 0.4rem 0;
  color: #fff;
  border-bottom: 2px solid var(--color-yellow);

  &:hover {
    font-style: italic;
  }
}

.form {
  padding: 0.8rem clamp(1rem, 3vw, 3rem);
  border-radius: 3px;
  font-family: inherit;
  font-size: 1.6rem;
  border: none;
  width: 100%;

  &:focus {
    outline: none;
  }

  &__input {
    border-bottom: 2px solid #fff;
    background-color: transparent;
    color: #fff;
    resize: none;

    &::placeholder {
      color: rgba(255, 255, 255, 0.7);
    }
```

```css
  }
}
```

```css
.contact-box {
  display: grid;
  grid-template-columns: repeat(
    var(--two-col-layout),
    minmax(min-content, 1fr)
  );
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
}

.form-grp {
  display: grid;
  row-gap: 3rem;
  padding: 4rem 2.4rem;
}

.form-flex {
  justify-self: flex-start;
}

#mapid {
  height: 55rem;
  width: 100%;
}
```

```css
.section-footer {
  background-color: var(--color-black);
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.footer-box {
  display: grid;
  grid-template-columns: repeat(
    var(--three-col-layout),
    minmax(min-content, 1fr)
  );
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
  color: #fff;
}
```

```
.contact-details {
  & .contact-data {
    list-style: none;
    display: flex;
    flex-direction: column;

    * {
      padding: 0.4rem 0;
    }

    & .social {
      display: flex;

      * {
        padding-right: 1rem;
      }
    }
  }
}

.footer-nav {
  & .nav-name {
    font-size: 2rem;
    font-weight: 700;
    margin-bottom: 2rem;
  }

  & ul {
    display: flex;
    flex-direction: column;
  }

  & ul li {
    list-style: none;
    padding: 1.5rem 0;
  }

  & ul li a {
    text-decoration: none;
    color: #fff;
  }

  & ul li a::before {
    content: '\2192';
    display: inline-block;
    margin-right: 10px;
    margin-top: -4px;
    color: var(--color-yellow);
```

```scss
  }
}

.newsletter {
  &__title {
    font-size: 2rem;
    font-weight: 700;
    margin-bottom: 2rem;
  }
  &__text {
    margin-bottom: 1.6rem;
  }

  &__input {
    display: flex;
  }
}

.send-icon {
  padding: 0.9rem;
  background-color: #000;
  cursor: pointer;
}

.legel {
  border-top: 2px solid #fff;
  padding: 1.5rem 0;
  grid-column: 1/-1;
  text-align: center;

  & .text {
    padding: 1rem 0;
  }
}
```

```scss
.header {
  position: absolute;
  top: 0;
  right: 0;
  color: #fff;
  width: 100%;
  z-index: 100;
  background: linear-gradient(rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 0.9));
}

.header-navigation {
```

```css
  display: flex;
  justify-content: space-between;
  align-items: center;

  padding: 1.2rem 0;

  & ul {
    display: flex;
    text-transform: capitalize;
    font-size: 1.7rem;
    list-style: none;
    font-weight: 700;

    @media only screen and (max-width: 50em) {
      display: none;
    }

    & li {
      padding-left: 2.8rem;
      & a {
        text-decoration: none;
        color: #fff;
        padding: 1.8rem 0;
        border-bottom: 2px solid transparent;
        transition: border 0.5s;

        &:hover {
          border-bottom: 2px solid var(--color-yellow);
        }
      }
    }
  }

  & .logo {
    font-size: 2.5rem;
    font-weight: 700;
  }

  & .nav-icon {
    display: none;
    border: 1px solid #fff;

    @media only screen and (max-width: 50em) {
      display: block;
    }
  }
}
```

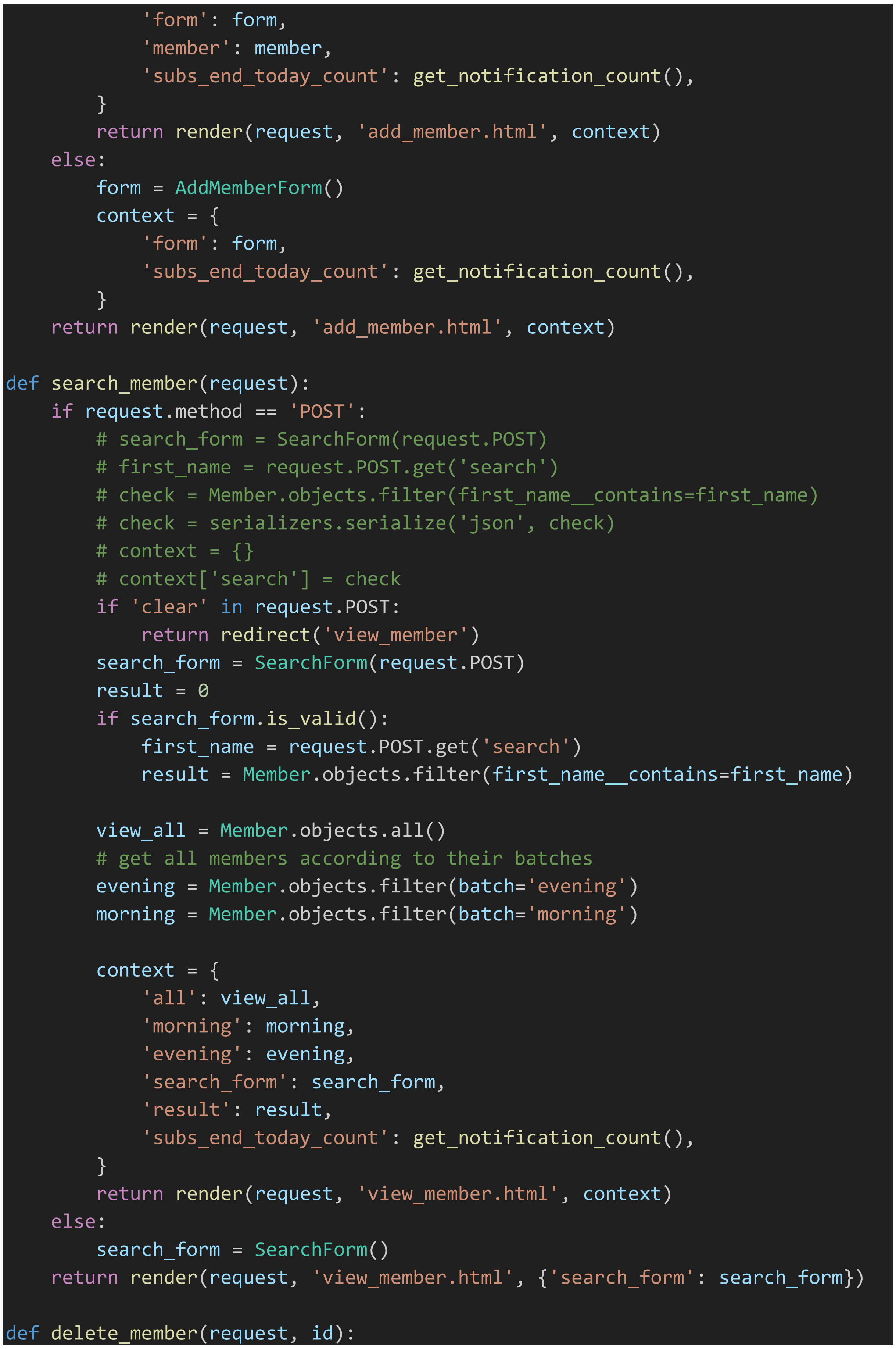
```css
.section-review {
  padding: clamp(4rem, 10vw, 12rem) 0rem;
}

.review-box {
  display: grid;
  grid-template-columns: repeat(
    var(--two-col-layout),
    minmax(min-content, 1fr)
  );
  column-gap: var(--grid-column-gap);
  row-gap: var(--grid-row-gap);
}

.review-card {
  color: #fff;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;

  &__image {
    max-width: 100%;
    height: auto;
    background-position: center;
    background-size: cover;
    border-radius: 50%;
  }

  &__content {
    padding: 2rem 0rem;
    display: flex;
    flex-direction: column;
    text-align: center;
    position: relative;

    & .text {
      font-style: italic;
    }

    & .name {
      font-style: normal;
      text-transform: uppercase;
      letter-spacing: 0.2rem;
      padding-top: 1.5rem;
    }
  }
  &__content::before {
    content: '\201F';
    display: block;
    font-size: 12rem;
    color: rgba(0, 0, 0, 0.8);

    position: absolute;
    top: -7rem;
    left: -1rem;
  }
}
```

```
@import './base';
@import './header';
@import './typography';
@import './components';
@import './about';
@import './classes';
@import './offer';
@import './coaches';
@import './review';
@import './contact';
@import './footer';
@import './utilities';
@import './hero';
```

Sample Source code for member's backend:

```
from django.shortcuts import render, redirect, reverse
from django.http import JsonResponse
from django.core import serializers
from .models import AddMemberForm, Member, SearchForm, UpdateMemberGymForm,
UpdateMemberInfoForm
import datetime, csv
from django.http import HttpResponse
import dateutil.relativedelta as delta
import dateutil.parser as parser
from django.core.files.storage import FileSystemStorage
from payments.models import Payments
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from notifications.config import get_notification_count
from django.db.models.signals import post_save
from notifications.config import my_handler
from django.contrib import messages
```

```python
def model_save(model):
    post_save.disconnect(my_handler, sender=Member)
    model.save()
    post_save.connect(my_handler, sender=Member)

def check_status(request, object):
    object.stop = 1 if request.POST.get('stop') == '1' else 0
    return object

# Export user information.
def export_all(user_obj):
    response = HttpResponse(content_type='text/csv')
    writer = csv.writer(response)
    writer.writerow(['First name', 'Last name', 'Mobile', 'Admission Date',
'Subscription Type', 'Batch'])
    members = user_obj.values_list('first_name', 'last_name', 'mobile_number',
'admitted_on', 'subscription_type', 'batch')
    for user in members:
        first_name = user[0]
        last_name = user[1]
        writer.writerow(user)

    response['Content-Disposition'] = 'attachment; filename="' + first_name +
' ' + last_name + '.csv"'
    return response

def members(request):
    form = AddMemberForm()
    context = {
        'form': form,
        'subs_end_today_count': get_notification_count(),
    }
    return render(request, 'add_member.html', context)

def view_member(request):
    view_all = Member.objects.filter(stop=0).order_by('first_name')
    paginator = Paginator(view_all, 100)
    try:
        page = request.GET.get('page', 1)
        view_all = paginator.page(page)
    except PageNotAnInteger:
        view_all = paginator.page(1)
    except EmptyPage:
        view_all = paginator.page(paginator.num_pages)
    search_form = SearchForm()
    # get all members according to their batches
    evening = Member.objects.filter(batch='evening',
stop=0).order_by('first_name')
    morning = Member.objects.filter(batch='morning',
stop=0).order_by('first_name')
    stopped = Member.objects.filter(stop=1).order_by('first_name')
    context = {
        'all': view_all,
        'morning': morning,
        'evening': evening,
        'stopped': stopped,
        'search_form': search_form,
        'subs_end_today_count': get_notification_count(),
    }
    return render(request, 'view_member.html', context)

def add_member(request):
    view_all = Member.objects.all()
    success = 0
    member = None
    if request.method == 'POST':
        form = AddMemberForm(request.POST, request.FILES)
        if form.is_valid():
            temp = form.save(commit=False)
            temp.first_name = request.POST.get('first_name').capitalize()
            temp.last_name = request.POST.get('last_name').capitalize()
            temp.registration_upto =
parser.parse(request.POST.get('registration_date')) +
delta.relativedelta(months=int(request.POST.get('subscription_period')))
            if request.POST.get('fee_status') == 'pending':
                temp.notification = 1

            model_save(temp)
            success = 'Successfully Added Member'

            # Add payments if payment is 'paid'
            if temp.fee_status == 'paid':
                payments = Payments(
                                    user=temp,
                                    payment_date=temp.registration_date,
                                    payment_period=temp.subscription_period,
                                    payment_amount=temp.amount)
                payments.save()

            form = AddMemberForm()
            member = Member.objects.last()

        context = {
            'add_success': success,
```



```python
            'form': form,
            'member': member,
            'subs_end_today_count': get_notification_count(),
        }
        return render(request, 'add_member.html', context)
    else:
        form = AddMemberForm()
        context = {
            'form': form,
            'subs_end_today_count': get_notification_count(),
        }
    return render(request, 'add_member.html', context)

def search_member(request):
    if request.method == 'POST':
        # search_form = SearchForm(request.POST)
        # first_name = request.POST.get('search')
        # check = Member.objects.filter(first_name__contains=first_name)
        # check = serializers.serialize('json', check)
        # context = {}
        # context['search'] = check
        if 'clear' in request.POST:
            return redirect('view_member')
        search_form = SearchForm(request.POST)
        result = 0
        if search_form.is_valid():
            first_name = request.POST.get('search')
            result = Member.objects.filter(first_name__contains=first_name)

        view_all = Member.objects.all()
        # get all members according to their batches
        evening = Member.objects.filter(batch='evening')
        morning = Member.objects.filter(batch='morning')

        context = {
            'all': view_all,
            'morning': morning,
            'evening': evening,
            'search_form': search_form,
            'result': result,
            'subs_end_today_count': get_notification_count(),
        }
        return render(request, 'view_member.html', context)
    else:
        search_form = SearchForm()
    return render(request, 'view_member.html', {'search_form': search_form})

def delete_member(request, id):
    print(id)
    Member.objects.filter(pk=id).delete()
    return redirect('view_member')

def update_member(request, id):
    if request.method == 'POST' and request.POST.get('export'):
        return export_all(Member.objects.filter(pk=id))
    if request.method == 'POST' and request.POST.get('no'):
        return redirect('/')
    if request.method == 'POST' and request.POST.get('gym_membership'):
            gym_form = UpdateMemberGymForm(request.POST)
            if gym_form.is_valid():
                object = Member.objects.get(pk=id)
                amount = request.POST.get('amount')
                day = (parser.parse(request.POST.get('registration_upto')) -
delta.relativedelta(months=int(request.POST.get('subscription_period')))).day
                last_day = parser.parse(str(object.registration_upto)).day

                month =
parser.parse(request.POST.get('registration_upto')).month
                last_month = parser.parse(str(object.registration_upto)).month
                # if status is stopped then do not update anything
                if object.stop == 1 and not request.POST.get('stop') == '0'
and request.POST.get('gym_membership'):
                    messages.error(request, 'Please start the status of user
to update the record')
                    return redirect('update_member', id=object.pk)
                # to change only the batch
                elif (object.batch != request.POST.get('batch')):
                    object.batch = request.POST.get('batch')
                    object = check_status(request, object)
                    model_save(object)
                # check if user has modified only the date
                elif
(datetime.datetime.strptime(str(object.registration_date), "%Y-%m-%d") !=
datetime.datetime.strptime(request.POST.get('registration_date'), "%Y-%m-
%d")):
                        object.registration_date
=  parser.parse(request.POST.get('registration_date'))
                        object.registration_upto
=  parser.parse(request.POST.get('registration_date')) +
delta.relativedelta(months=int(request.POST.get('subscription_period')))
                        object.fee_status = request.POST.get('fee_status')
                        object = check_status(request, object)
                        model_save(object)
                # if amount and period are changed
                elif (object.amount != amount) and (object.subscription_period
!= request.POST.get('subscription_period')):
```

```
                    object.subscription_type
=  request.POST.get('subscription_type')
                    object.subscription_period
=  request.POST.get('subscription_period')
                    object.registration_date
=  parser.parse(request.POST.get('registration_upto'))
                    object.registration_upto
=  parser.parse(request.POST.get('registration_upto')) +
delta.relativedelta(months=int(request.POST.get('subscription_period')))
                    object.fee_status = request.POST.get('fee_status')
                    object.amount =  request.POST.get('amount')
                    object = check_status(request, object)
                    model_save(object)
                # if only subscription_period is Changed
                elif (object.subscription_period !=
request.POST.get('subscription_period')):
                    object.subscription_period
=  request.POST.get('subscription_period')
                    object = check_status(request, object)
                    model_save(object)
                # if amount and type are changed
                elif (object.amount != amount) and (object.subscription_type
!= request.POST.get('subscription_type')):
                    object.subscription_type
=  request.POST.get('subscription_type')
                    object.subscription_period
=  request.POST.get('subscription_period')
                    object.registration_date
=  parser.parse(request.POST.get('registration_upto'))
                    object.registration_upto
=  parser.parse(request.POST.get('registration_upto')) +
delta.relativedelta(months=int(request.POST.get('subscription_period')))
                    object.fee_status = request.POST.get('fee_status')
                    object.amount =  request.POST.get('amount')
                    object = check_status(request, object)
                    model_save(object)
                # if amount ad fee status are changed
                elif (object.amount != amount) and
((request.POST.get('fee_status') == 'paid') or (request.POST.get('fee_status')
== 'pending')):
                        object.amount = amount
                        object.fee_status = request.POST.get('fee_status')
                        object = check_status(request, object)
                        model_save(object)
                # if only amount is channged
                elif (object.amount != amount):
                    object.registration_date
=  parser.parse(request.POST.get('registration_upto'))
```

```
                    object.registration_upto
=  parser.parse(request.POST.get('registration_upto')) +
delta.relativedelta(months=int(request.POST.get('subscription_period')))
                    object.fee_status = request.POST.get('fee_status')
                    object.amount =  request.POST.get('amount')
                    if request.POST.get('fee_status') == 'pending':
                        object.notification =  1
                    elif request.POST.get('fee_status') == 'paid':
                        object.notification = 2
                    object = check_status(request, object)
                    model_save(object)
                # nothing is changed
                else:
                    if not request.POST.get('stop') == '1':
                        object.registration_date
=  parser.parse(request.POST.get('registration_upto'))
                        object.registration_upto
=  parser.parse(request.POST.get('registration_upto')) +
delta.relativedelta(months=int(request.POST.get('subscription_period')))
                        object.amount =  request.POST.get('amount')
                        if request.POST.get('fee_status') == 'pending':
                            object.notification =  1
                        elif request.POST.get('fee_status') == 'paid':
                            object.notification = 2
                    object.fee_status = request.POST.get('fee_status')
                    object = check_status(request, object)
                    model_save(object)

                # Add payments if payment is 'paid'
                if object.fee_status == 'paid':
                    check = Payments.objects.filter(
                        payment_date=object.registration_date,
                        user__pk=object.pk).count()
                    if check == 0:
                        payments = Payments(
                                            user=object,
                                            payment_date=object.registration_d
ate,
                                            payment_period=object.subscription
_period,
                                            payment_amount=object.amount)
                        payments.save()
                user = Member.objects.get(pk=id)
                gym_form = UpdateMemberGymForm(initial={
                                        'registration_date':
user.registration_date,
                                        'registration_upto':
user.registration_upto,
                                        'subscription_type':
user.subscription_type,
                                        'subscription_period':
user.subscription_period,
                                        'amount': user.amount,
                                        'fee_status': user.fee_status,
                                        'batch': user.batch,
                                        'stop': user.stop,
                                        })

                info_form = UpdateMemberInfoForm(initial={
                                        'first_name': user.first_name,
                                        'last_name': user.last_name,
                                        'dob': user.dob,
                                        })

                try:
                    payments = Payments.objects.filter(user=user)
                except Payments.DoesNotExist:
                    payments = 'No Records'
                messages.success(request, 'Record updated successfully!')
                return redirect('update_member', id=user.pk)
            else:
                user = Member.objects.get(pk=id)
                info_form = UpdateMemberInfoForm(initial={
                                        'first_name': user.first_name,
                                        'last_name': user.last_name,
                                        'dob': user.dob,
                                        })

                try:
                    payments = Payments.objects.filter(user=user)
                except Payments.DoesNotExist:
                    payments = 'No Records'
                return render(request,
                    'update.html',
                    {
                        'payments': payments,
                        'gym_form': gym_form,
                        'info_form': info_form,
                        'user': user,
                        'subs_end_today_count': get_notification_count(),
                    })
    elif request.method == 'POST' and request.POST.get('info'):
        object = Member.objects.get(pk=id)
        object.first_name = request.POST.get('first_name')
        object.last_name = request.POST.get('last_name')
        object.dob = request.POST.get('dob')
        # for updating photo
        if 'photo' in request.FILES:
            myfile = request.FILES['photo']
            fs = FileSystemStorage(base_url="")
            photo = fs.save(myfile.name, myfile)
            object.photo = fs.url(photo)
        model_save(object)

        user = Member.objects.get(pk=id)
        gym_form = UpdateMemberGymForm(initial={
                                'registration_date': user.registration_date,
                                'registration_upto': user.registration_upto,
                                'subscription_type': user.subscription_type,
                                'subscription_period':
user.subscription_period,
                                'amount': user.amount,
                                'fee_status': user.fee_status,
                                'batch': user.batch,
                                'stop': user.stop,
                                })

        info_form = UpdateMemberInfoForm(initial={
                                'first_name': user.first_name,
                                'last_name': user.last_name,
                                'dob': user.dob,
                                })

        try:
            payments = Payments.objects.filter(user=user)
        except Payments.DoesNotExist:
            payments = 'No Records'

        return render(request,
            'update.html',
            {
                'payments': payments,
                'gym_form': gym_form,
                'info_form': info_form,
                'user': user,
                'updated': 'Record Updated Successfully',
                'subs_end_today_count': get_notification_count(),
            })
    else:
        user = Member.objects.get(pk=id)

        if len(Payments.objects.filter(user=user)) > 0:
            payments = Payments.objects.filter(user=user)
        else:
            payments = 'No Records'
        gym_form = UpdateMemberGymForm(initial={
                                'registration_date': user.registration_date,
                                'registration_upto': user.registration_upto,
                                'subscription_type': user.subscription_type,
                                'subscription_period':
user.subscription_period,
                                'amount': user.amount,
                                'fee_status': user.fee_status,
                                'batch': user.batch,
                                'stop': user.stop,
                                })

        info_form = UpdateMemberInfoForm(initial={
                                'first_name': user.first_name,
                                'last_name': user.last_name,
                                'dob': user.dob,
                                })
        return render(request,
                        'update.html',
                        {
                            'payments': payments,
                            'gym_form': gym_form,
                            'info_form': info_form,
                            'user': user,
                            'subs_end_today_count': get_notification_count(),
                        }
                    )
```

Database(SQL) code:

```
-- phpMyAdmin SQL Dump
-- version 4.9.2
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Apr 02, 2020 at 10:39 PM
-- Server version: 10.4.10-MariaDB
-- PHP Version: 7.3.12

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
```

```sql
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `gymnasium`
--

-- --------------------------------------------------------

--
-- Table structure for table `accounts_wallpaper`
--

CREATE TABLE `accounts_wallpaper` (
  `id` int(11) NOT NULL,
  `photo` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `accounts_wallpaper`
--

INSERT INTO `accounts_wallpaper` (`id`, `photo`) VALUES
(1, '\\media\\wallpaper/test.jpg');

-- --------------------------------------------------------

--
-- Table structure for table `auth_group`
--

CREATE TABLE `auth_group` (
  `id` int(11) NOT NULL,
  `name` varchar(150) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `auth_group_permissions`
--

CREATE TABLE `auth_group_permissions` (
  `id` int(11) NOT NULL,
  `group_id` int(11) NOT NULL,
  `permission_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `auth_permission`
--

CREATE TABLE `auth_permission` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `content_type_id` int(11) NOT NULL,
  `codename` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `auth_permission`
--

INSERT INTO `auth_permission` (`id`, `name`, `content_type_id`, `codename`)
VALUES
(1, 'Can add log entry', 1, 'add_logentry'),
(2, 'Can change log entry', 1, 'change_logentry'),
(3, 'Can delete log entry', 1, 'delete_logentry'),
(4, 'Can view log entry', 1, 'view_logentry'),
(5, 'Can add permission', 2, 'add_permission'),
(6, 'Can change permission', 2, 'change_permission'),
(7, 'Can delete permission', 2, 'delete_permission'),
(8, 'Can view permission', 2, 'view_permission'),
(9, 'Can add group', 3, 'add_group'),
(10, 'Can change group', 3, 'change_group'),
(11, 'Can delete group', 3, 'delete_group'),
(12, 'Can view group', 3, 'view_group'),
(13, 'Can add user', 4, 'add_user'),
(14, 'Can change user', 4, 'change_user'),
(15, 'Can delete user', 4, 'delete_user'),
(16, 'Can view user', 4, 'view_user'),
(17, 'Can add content type', 5, 'add_contenttype'),
(18, 'Can change content type', 5, 'change_contenttype'),
(19, 'Can delete content type', 5, 'delete_contenttype'),
(20, 'Can view content type', 5, 'view_contenttype'),
(21, 'Can add session', 6, 'add_session'),
(22, 'Can change session', 6, 'change_session'),
(23, 'Can delete session', 6, 'delete_session'),
(24, 'Can view session', 6, 'view_session'),
(25, 'Can add wallpaper', 7, 'add_wallpaper'),
(26, 'Can change wallpaper', 7, 'change_wallpaper'),
(27, 'Can delete wallpaper', 7, 'delete_wallpaper'),
```

```sql
(28, 'Can view wallpaper', 7, 'view_wallpaper'),
(29, 'Can add member', 8, 'add_member'),
(30, 'Can change member', 8, 'change_member'),
(31, 'Can delete member', 8, 'delete_member'),
(32, 'Can view member', 8, 'view_member'),
(33, 'Can add generate reports', 9, 'add_generatereports'),
(34, 'Can change generate reports', 9, 'change_generatereports'),
(35, 'Can delete generate reports', 9, 'delete_generatereports'),
(36, 'Can view generate reports', 9, 'view_generatereports'),
(37, 'Can add payments', 10, 'add_payments'),
(38, 'Can change payments', 10, 'change_payments'),
(39, 'Can delete payments', 10, 'delete_payments'),
(40, 'Can view payments', 10, 'view_payments');

-- --------------------------------------------------------

--
-- Table structure for table `auth_user`
--

CREATE TABLE `auth_user` (
  `id` int(11) NOT NULL,
  `password` varchar(128) NOT NULL,
  `last_login` datetime(6) DEFAULT NULL,
  `is_superuser` tinyint(1) NOT NULL,
  `username` varchar(150) NOT NULL,
  `first_name` varchar(30) NOT NULL,
  `last_name` varchar(150) NOT NULL,
  `email` varchar(254) NOT NULL,
  `is_staff` tinyint(1) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `date_joined` datetime(6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `auth_user`
--

INSERT INTO `auth_user` (`id`, `password`, `last_login`, `is_superuser`,
`username`, `first_name`, `last_name`, `email`, `is_staff`, `is_active`,
`date_joined`) VALUES
(1,
'pbkdf2_sha256$150000$F3gD8KGNs5ac$pSSmXq3Ff1WqcuCdlif5SRmhcDIyUQbNkONXVKWj6+Q
=', '2020-02-24 18:55:05.138714', 1, 'Admin', '', '', '', 1, 1, '2020-02-24
15:58:18.493505');

-- --------------------------------------------------------
```

```sql
--
-- Table structure for table `auth_user_groups`
--

CREATE TABLE `auth_user_groups` (
  `id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `group_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `auth_user_user_permissions`
--

CREATE TABLE `auth_user_user_permissions` (
  `id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `permission_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `django_admin_log`
--

CREATE TABLE `django_admin_log` (
  `id` int(11) NOT NULL,
  `action_time` datetime(6) NOT NULL,
  `object_id` longtext DEFAULT NULL,
  `object_repr` varchar(200) NOT NULL,
  `action_flag` smallint(5) UNSIGNED NOT NULL,
  `change_message` longtext NOT NULL,
  `content_type_id` int(11) DEFAULT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `django_content_type`
--

CREATE TABLE `django_content_type` (
  `id` int(11) NOT NULL,
  `app_label` varchar(100) NOT NULL,
  `model` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `django_content_type`
--

INSERT INTO `django_content_type` (`id`, `app_label`, `model`) VALUES
(7, 'accounts', 'wallpaper'),
(1, 'admin', 'logentry'),
(3, 'auth', 'group'),
(2, 'auth', 'permission'),
(4, 'auth', 'user'),
(5, 'contenttypes', 'contenttype'),
(8, 'members', 'member'),
(10, 'payments', 'payments'),
(9, 'reports', 'generatereports'),
(6, 'sessions', 'session');

-- --------------------------------------------------------

--
-- Table structure for table `django_migrations`
--

CREATE TABLE `django_migrations` (
  `id` int(11) NOT NULL,
  `app` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `applied` datetime(6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `django_migrations`
--

INSERT INTO `django_migrations` (`id`, `app`, `name`, `applied`) VALUES
(1, 'accounts', '0001_initial', '2020-02-24 15:55:38.584088'),
(2, 'contenttypes', '0001_initial', '2020-02-24 15:55:39.715067'),
(3, 'auth', '0001_initial', '2020-02-24 15:55:42.896911'),
(4, 'admin', '0001_initial', '2020-02-24 15:55:56.101400'),
(5, 'admin', '0002_logentry_remove_auto_add', '2020-02-24 15:56:00.081713'),
(6, 'admin', '0003_logentry_add_action_flag_choices', '2020-02-24
15:56:00.205076'),
(7, 'contenttypes', '0002_remove_content_type_name', '2020-02-24
15:56:01.707174'),
(8, 'auth', '0002_alter_permission_name_max_length', '2020-02-24
15:56:03.322121'),
(9, 'auth', '0003_alter_user_email_max_length', '2020-02-24 15:56:03.463451'),
(10, 'auth', '0004_alter_user_username_opts', '2020-02-24 15:56:03.698598'),
(11, 'auth', '0005_alter_user_last_login_null', '2020-02-24 15:56:10.827719'),
(12, 'auth', '0006_require_contenttypes_0002', '2020-02-24 15:56:10.911356'),
(13, 'auth', '0007_alter_validators_add_error_messages', '2020-02-24
15:56:11.017959'),
(14, 'auth', '0008_alter_user_username_max_length', '2020-02-24
15:56:11.181699'),
(15, 'auth', '0009_alter_user_last_name_max_length', '2020-02-24
15:56:11.360647'),
(16, 'auth', '0010_alter_group_name_max_length', '2020-02-24
15:56:11.542227'),
(17, 'auth', '0011_update_proxy_permissions', '2020-02-24 15:56:11.698308'),
(18, 'members', '0001_initial', '2020-02-24 15:56:12.156386'),
(19, 'payments', '0001_initial', '2020-02-24 15:56:12.563548'),
(20, 'reports', '0001_initial', '2020-02-24 15:56:15.063474'),
(21, 'sessions', '0001_initial', '2020-02-24 15:56:16.103542');

-- --------------------------------------------------------

--
-- Table structure for table `django_session`
--

CREATE TABLE `django_session` (
  `session_key` varchar(40) NOT NULL,
  `session_data` longtext NOT NULL,
  `expire_date` datetime(6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `django_session`
--

INSERT INTO `django_session` (`session_key`, `session_data`, `expire_date`)
VALUES
('1ppkeynsh73q04lheclwpa0s19jdewtd',
'ZjgzMGI0ZDYzZWI1YzkwMjM2ODMyODg5ZTAyN2MwYTBkNTczOWZlOTp7Il9hdXRoX3VzZXJfaWQiO
iIxIiwiX2F1dGhfdXNlcl9iYWNrZW5kIjoiZGphbmdvLmNvbnRyaWIuYXV0aC5iYWNrZW5kcy5Nb2R
lbEJhY2tlbmQiLCJfYXV0aF91c2VyX2hhc2giOiJhYWI3OTU4YzY5NDhkZDQzNGMwNDc4NjM1MzVmZ
mYzNWIzM2ZmY2FkIn0=', '2020-03-09 15:58:40.840413');

-- --------------------------------------------------------

--
-- Table structure for table `members_member`
--
```

```sql
CREATE TABLE `members_member` (
  `member_id` int(11) NOT NULL,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `mobile_number` varchar(10) NOT NULL,
  `email` varchar(254) DEFAULT NULL,
  `address` varchar(300) NOT NULL,
  `medical_history` varchar(300) NOT NULL,
  `admitted_on` date NOT NULL,
  `registration_date` date NOT NULL,
  `registration_upto` date NOT NULL,
  `dob` date NOT NULL,
  `subscription_type` varchar(30) NOT NULL,
  `subscription_period` varchar(30) NOT NULL,
  `amount` varchar(30) NOT NULL,
  `fee_status` varchar(30) NOT NULL,
  `batch` varchar(30) NOT NULL,
  `photo` varchar(100) NOT NULL,
  `notification` int(11) NOT NULL,
  `stop` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `members_member`
--

INSERT INTO `members_member` (`member_id`, `first_name`, `last_name`,
`mobile_number`, `email`, `address`, `medical_history`, `admitted_on`,
`registration_date`, `registration_upto`, `dob`, `subscription_type`,
`subscription_period`, `amount`, `fee_status`, `batch`, `photo`,
`notification`, `stop`) VALUES
(1, 'Keval', 'Senghani', '8656423958', 'keval@gmail.com', 'B-105, Ganesh
Apartment, Gandhi Road', 'None', '2020-02-24', '2020-04-24', '2020-05-24',
'1999-05-11', 'gym', '1', '500', 'pending', 'morning', '', 0, 1),
(2, 'Mitesh', 'Masurkar', '9023568745', 'mitesh@gmail.com', 'A-3, Sai
Apartment, Ganesh Nagar', 'None', '2020-02-24', '2020-02-24', '2020-03-24',
'1999-06-05', 'gym_and_cross_fit', '1', '750', 'paid', 'evening', '', 2, 0),
(3, 'Keval', 'Patel', '9056487562', 'keval@gmail.com', 'A-104, Ganesh
Apartment, Gandhi Road', 'None', '2020-02-24', '2020-02-24', '2020-03-24',
'1999-11-08', 'gym', '1', '500', 'paid', 'morning', '', 2, 0),
(4, 'Suyash', 'Gupte', '8945623578', 'suyash@gmail.com', '105, Sai Sadan,
Sector 5', 'None', '2020-02-24', '2020-04-25', '2020-05-25', '1998-12-11',
'pt', '1', '1500', 'paid', 'morning', '', 2, 0),
(5, 'Rajan', 'Yadav', '9056458725', 'rajan@gmail.com', 'C-2, Apollo Apartment,
Sector 20', 'None', '2020-02-24', '2020-04-25', '2020-05-25', '1999-07-05',
'cross_fit', '1', '500', 'paid', 'morning', '', 2, 0),
(8, 'Rajat', 'Singh', '9856235487', 'rajat@gmail.com', 'B-103, Ganesh
Apartment, Sector 10', 'None', '2020-02-25', '2020-01-29', '2020-03-29',
'1998-09-02', 'pt', '2', '2000', 'paid', 'morning', '', 2, 0),
(9, 'Mahesh', 'Sawant', '6545892536', 'mahesh@gmail.com', 'B-102, Sai Sadan,
Gandhi Road', 'None', '2020-02-25', '2020-02-25', '2020-03-25', '1999-10-29',
'gym_and_cross_fit', '1', '1000', 'paid', 'morning', '/profile_cKCVlHE.jpg',
2, 0);

-- --------------------------------------------------------

--
-- Table structure for table `payments_payments`
--

CREATE TABLE `payments_payments` (
  `id` int(11) NOT NULL,
  `payment_date` date NOT NULL,
  `payment_period` int(11) NOT NULL,
  `payment_amount` int(11) NOT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `payments_payments`
--

INSERT INTO `payments_payments` (`id`, `payment_date`, `payment_period`,
`payment_amount`, `user_id`) VALUES
(1, '2020-02-24', 1, 500, 1),
(2, '2020-03-25', 1, 500, 1),
(3, '2020-03-24', 1, 500, 1),
(4, '2020-02-24', 1, 750, 2),
(5, '2020-02-24', 1, 500, 3),
(6, '2020-02-24', 1, 1500, 4),
(7, '2020-02-24', 1, 500, 5),
(8, '2020-02-25', 1, 500, 5),
(9, '2020-02-25', 1, 1500, 4),
(10, '2020-04-25', 1, 1500, 4),
(14, '2020-01-29', 2, 2000, 8),
(15, '2020-02-25', 1, 1000, 9),
(16, '2020-04-25', 1, 500, 5);

-- --------------------------------------------------------

--
-- Table structure for table `reports_generatereports`
--
```

```sql
CREATE TABLE `reports_generatereports` (
  `id` int(11) NOT NULL,
  `month` int(11) NOT NULL,
  `year` int(11) NOT NULL,
  `batch` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Indexes for dumped tables
--

--
-- Indexes for table `accounts_wallpaper`
--
ALTER TABLE `accounts_wallpaper`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `auth_group`
--
ALTER TABLE `auth_group`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `name` (`name`);

--
-- Indexes for table `auth_group_permissions`
--
ALTER TABLE `auth_group_permissions`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `auth_group_permissions_group_id_permission_id_0cd325b0_uniq`
(`group_id`,`permission_id`),
  ADD KEY `auth_group_permissio_permission_id_84c5c92e_fk_auth_perm`
(`permission_id`);

--
-- Indexes for table `auth_permission`
--
ALTER TABLE `auth_permission`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `auth_permission_content_type_id_codename_01ab375a_uniq`
(`content_type_id`,`codename`);

--
-- Indexes for table `auth_user`
--
ALTER TABLE `auth_user`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `username` (`username`);
```

```sql
--
-- Indexes for table `auth_user_groups`
--
ALTER TABLE `auth_user_groups`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `auth_user_groups_user_id_group_id_94350c0c_uniq`
(`user_id`,`group_id`),
  ADD KEY `auth_user_groups_group_id_97559544_fk_auth_group_id` (`group_id`);

--
-- Indexes for table `auth_user_user_permissions`
--
ALTER TABLE `auth_user_user_permissions`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY
`auth_user_user_permissions_user_id_permission_id_14a6b632_uniq`
(`user_id`,`permission_id`),
  ADD KEY `auth_user_user_permi_permission_id_1fbb5f2c_fk_auth_perm`
(`permission_id`);

--
-- Indexes for table `django_admin_log`
--
ALTER TABLE `django_admin_log`
  ADD PRIMARY KEY (`id`),
  ADD KEY `django_admin_log_content_type_id_c4bce8eb_fk_django_co`
(`content_type_id`),
  ADD KEY `django_admin_log_user_id_c564eba6_fk_auth_user_id` (`user_id`);

--
-- Indexes for table `django_content_type`
--
ALTER TABLE `django_content_type`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `django_content_type_app_label_model_76bd3d3b_uniq`
(`app_label`,`model`);

--
-- Indexes for table `django_migrations`
--
ALTER TABLE `django_migrations`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `django_session`
--
ALTER TABLE `django_session`
```

```sql
  ADD PRIMARY KEY (`session_key`),
  ADD KEY `django_session_expire_date_a5c62663` (`expire_date`);

--
-- Indexes for table `members_member`
--
ALTER TABLE `members_member`
  ADD PRIMARY KEY (`member_id`),
  ADD UNIQUE KEY `mobile_number` (`mobile_number`);

--
-- Indexes for table `payments_payments`
--
ALTER TABLE `payments_payments`
  ADD PRIMARY KEY (`id`),
  ADD KEY `payments_payments_user_id_4ec24b26_fk_members_member_member_id`
(`user_id`);

--
-- Indexes for table `reports_generatereports`
--
ALTER TABLE `reports_generatereports`
  ADD PRIMARY KEY (`id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `accounts_wallpaper`
--
ALTER TABLE `accounts_wallpaper`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `auth_group`
--
ALTER TABLE `auth_group`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `auth_group_permissions`
--
ALTER TABLE `auth_group_permissions`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `auth_permission`
```

```sql
--
ALTER TABLE `auth_permission`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=41;

--
-- AUTO_INCREMENT for table `auth_user`
--
ALTER TABLE `auth_user`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `auth_user_groups`
--
ALTER TABLE `auth_user_groups`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `auth_user_user_permissions`
--
ALTER TABLE `auth_user_user_permissions`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `django_admin_log`
--
ALTER TABLE `django_admin_log`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `django_content_type`
--
ALTER TABLE `django_content_type`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `django_migrations`
--
ALTER TABLE `django_migrations`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=22;

--
-- AUTO_INCREMENT for table `members_member`
--
ALTER TABLE `members_member`
  MODIFY `member_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;

--
-- AUTO_INCREMENT for table `payments_payments`
```

```
--
ALTER TABLE `payments_payments`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=17;

--
-- AUTO_INCREMENT for table `reports_generatereports`
--
ALTER TABLE `reports_generatereports`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `auth_group_permissions`
--
ALTER TABLE `auth_group_permissions`
  ADD CONSTRAINT `auth_group_permissio_permission_id_84c5c92e_fk_auth_perm`
FOREIGN KEY (`permission_id`) REFERENCES `auth_permission` (`id`),
  ADD CONSTRAINT `auth_group_permissions_group_id_b120cbf9_fk_auth_group_id`
FOREIGN KEY (`group_id`) REFERENCES `auth_group` (`id`);

--
-- Constraints for table `auth_permission`
--
ALTER TABLE `auth_permission`
  ADD CONSTRAINT `auth_permission_content_type_id_2f476e4b_fk_django_co`
FOREIGN KEY (`content_type_id`) REFERENCES `django_content_type` (`id`);

--
-- Constraints for table `auth_user_groups`
--
ALTER TABLE `auth_user_groups`
  ADD CONSTRAINT `auth_user_groups_group_id_97559544_fk_auth_group_id` FOREIGN
KEY (`group_id`) REFERENCES `auth_group` (`id`),
  ADD CONSTRAINT `auth_user_groups_user_id_6a12ed8b_fk_auth_user_id` FOREIGN
KEY (`user_id`) REFERENCES `auth_user` (`id`);

--
-- Constraints for table `auth_user_user_permissions`
--
ALTER TABLE `auth_user_user_permissions`
  ADD CONSTRAINT `auth_user_user_permi_permission_id_1fbb5f2c_fk_auth_perm`
FOREIGN KEY (`permission_id`) REFERENCES `auth_permission` (`id`),
  ADD CONSTRAINT `auth_user_user_permissions_user_id_a95ead1b_fk_auth_user_id`
FOREIGN KEY (`user_id`) REFERENCES `auth_user` (`id`);
```

```
--
-- Constraints for table `django_admin_log`
--
ALTER TABLE `django_admin_log`
  ADD CONSTRAINT `django_admin_log_content_type_id_c4bce8eb_fk_django_co`
FOREIGN KEY (`content_type_id`) REFERENCES `django_content_type` (`id`),
  ADD CONSTRAINT `django_admin_log_user_id_c564eba6_fk_auth_user_id` FOREIGN
KEY (`user_id`) REFERENCES `auth_user` (`id`);

--
-- Constraints for table `payments_payments`
--
ALTER TABLE `payments_payments`
  ADD CONSTRAINT
`payments_payments_user_id_4ec24b26_fk_members_member_member_id` FOREIGN KEY
(`user_id`) REFERENCES `members_member` (`member_id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

TEST PLAN/METHOD:

1. BASIC FUNCTIONALITY TESTING

Begin by making sure that every button on every screen works. You also need to ensure that you can enter simple text into each field without crashing the software. You don't have to try out all the different combinations of clicks and characters, or edge conditions, because that's what your testers do—and they're really good at that.

2. CODE REVIEW

Another pair of eyes looking at the source code can uncover a lot of problems. If your coding methodology requires peer review, perform this step before you hand the code over for testing. Remember to do your basic functionality testing before the code review, though.

3. STATIC CODE ANALYSIS

There are tools that can perform analysis on source code or bytecode without executing it. These static code analysis tools can look for many weaknesses in the source code, such as security vulnerabilities and potential concurrency issues. Use static code analysis tools to enforce coding standards, and configure those tools to run automatically as part of the build.

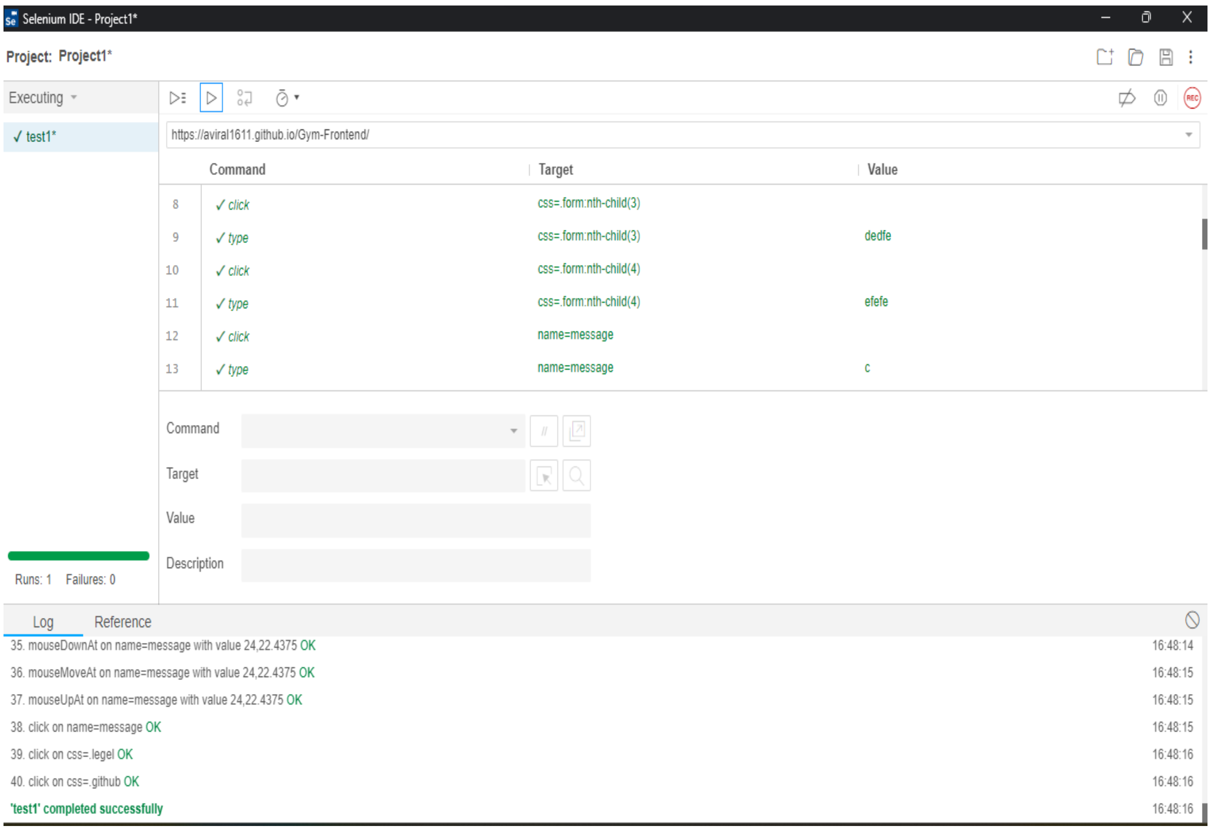
4. SINGLE-USER PERFORMANCE TESTING

Some teams have load and performance testing baked into their continuous integration process and run load tests as soon as code is checked in. This is particularly true for back-end code. But developers should also be looking at single-user performance on the front end and making sure the software is responsive when only they are using the system.

TESTING LIVE EXAMPLE:

So for this project testing I have made used of an Open source testing tool called Selenium. It checks the user checking all the functionalities and records everything; every click, every button pressed, every object interacted with.

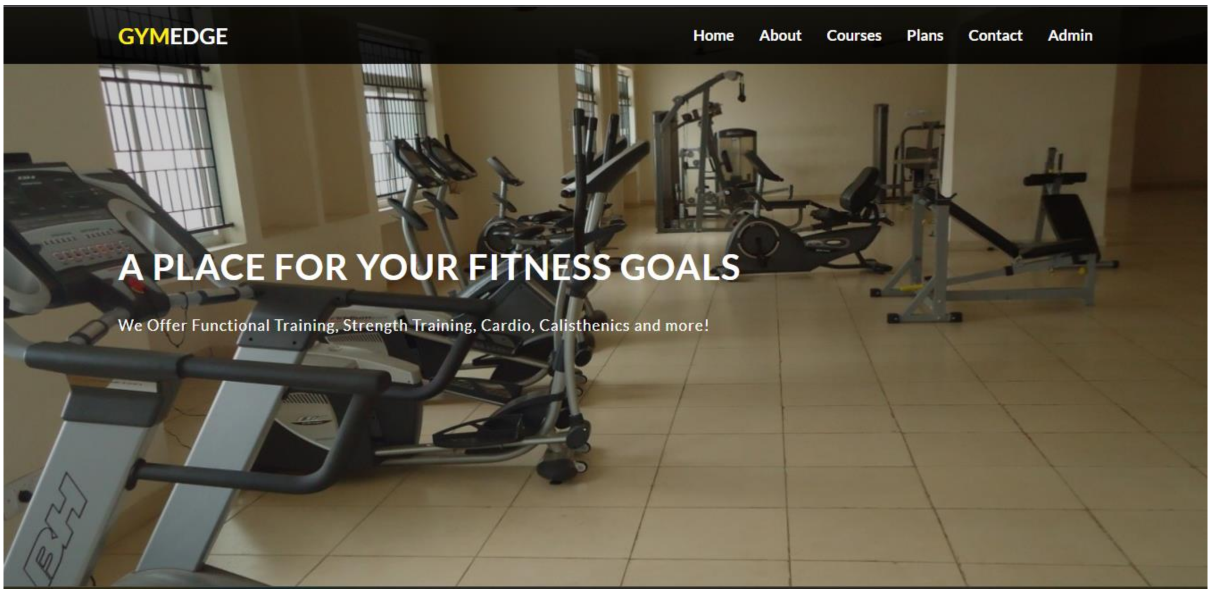
Then we save this inside a test case. Then we run the test case where Selenium follows everything the user did in an automated way. If everything works, then it shows that all the cases have passed successfully.



	Command	Target	Value
8	✓ click	css=.form:nth-child(3)	
9	✓ type	css=.form:nth-child(3)	dedfe
10	✓ click	css=.form:nth-child(4)	
11	✓ type	css=.form:nth-child(4)	efefe
12	✓ click	name=message	
13	✓ type	name=message	c

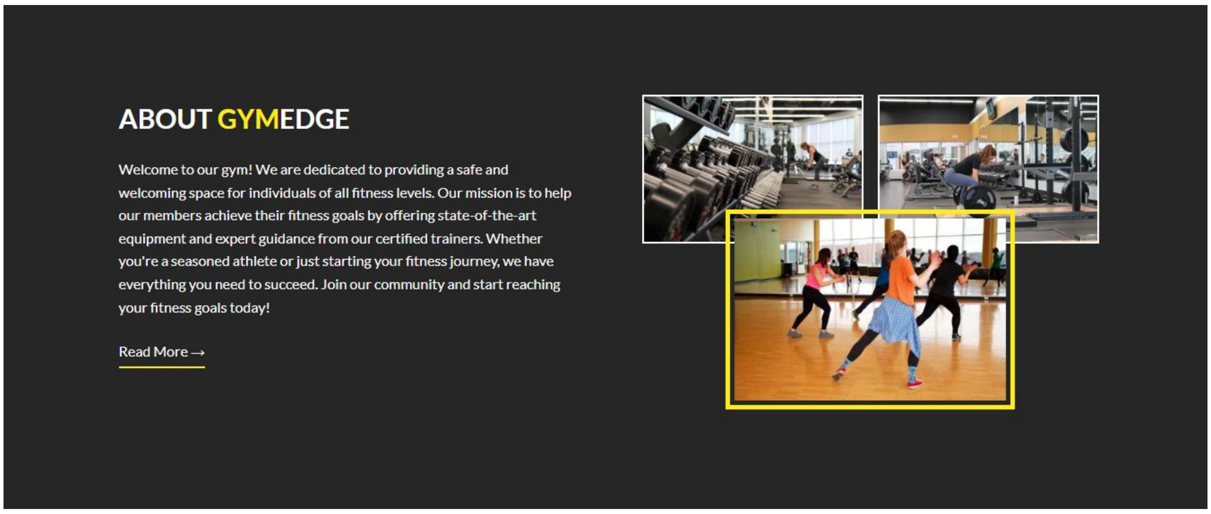
```
35. mouseDownAt on name=message with value 24,22.4375 OK     16:48:14
36. mouseMoveAt on name=message with value 24,22.4375 OK     16:48:15
37. mouseUpAt on name=message with value 24,22.4375 OK       16:48:15
38. click on name=message OK                                 16:48:15
39. click on css=.legel OK                                   16:48:16
40. click on css=.github OK                                  16:48:16
'test1' completed successfully                               16:48:16
```

SCREENSHOTS

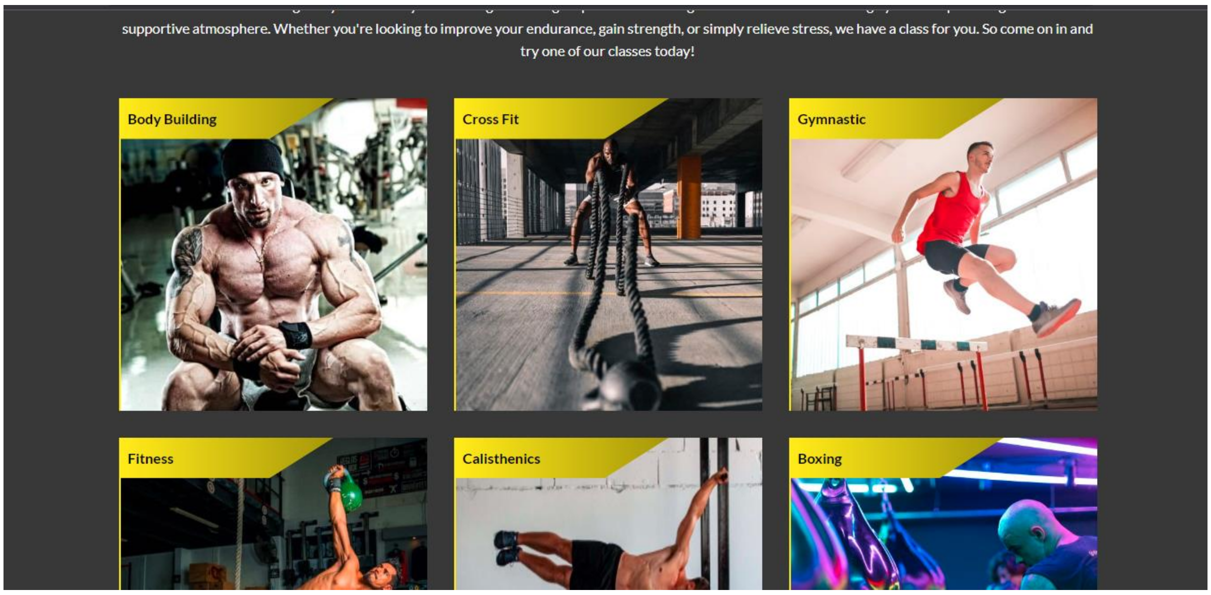


A PLACE FOR YOUR FITNESS GOALS

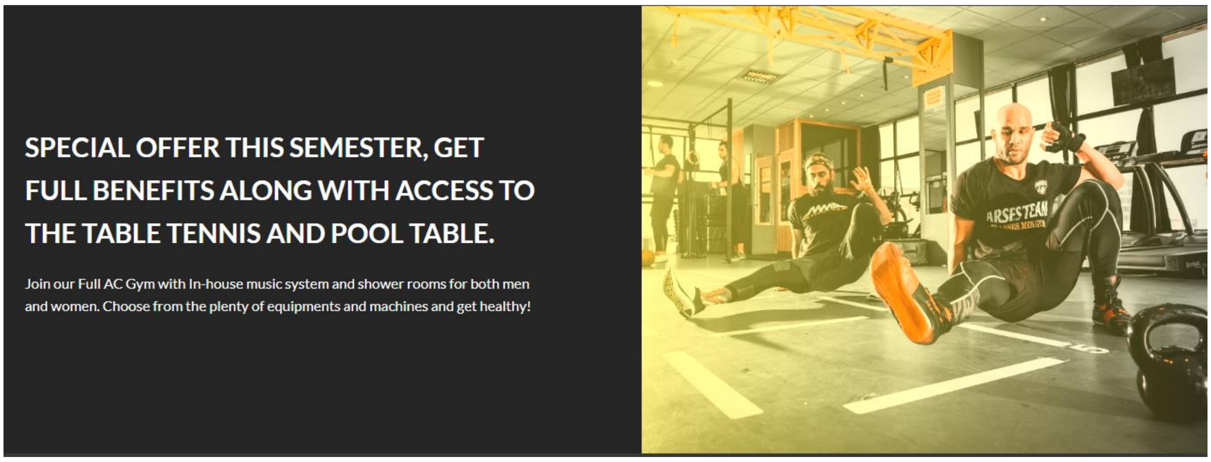
We Offer Functional Training, Strength Training, Cardio, Calisthenics and more!


ABOUT GYMEDGE

Welcome to our gym! We are dedicated to providing a safe and welcoming space for individuals of all fitness levels. Our mission is to help our members achieve their fitness goals by offering state-of-the-art equipment and expert guidance from our certified trainers. Whether you're a seasoned athlete or just starting your fitness journey, we have everything you need to succeed. Join our community and start reaching your fitness goals today!

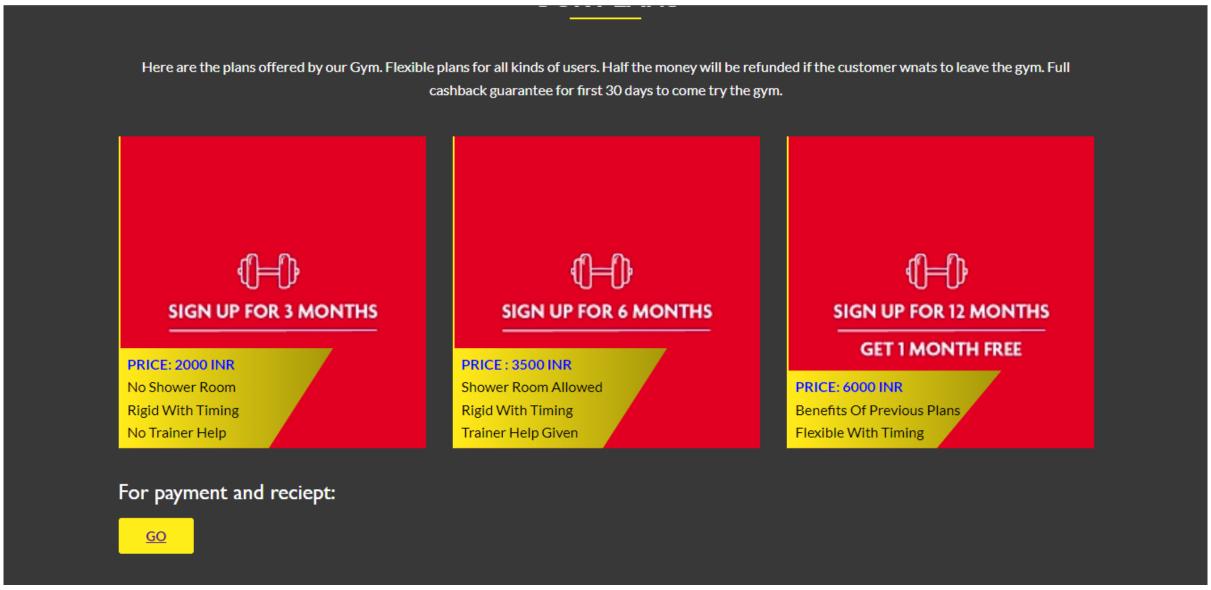


supportive atmosphere. Whether you're looking to improve your endurance, gain strength, or simply relieve stress, we have a class for you. So come on in and try one of our classes today!



SPECIAL OFFER THIS SEMESTER, GET FULL BENEFITS ALONG WITH ACCESS TO THE TABLE TENNIS AND POOL TABLE.

Join our Full AC Gym with In-house music system and shower rooms for both men and women. Choose from the plenty of equipments and machines and get healthy!



Here are the plans offered by our Gym. Flexible plans for all kinds of users. Half the money will be refunded if the customer wnats to leave the gym. Full cashback guarantee for first 30 days to come try the gym.

SIGN UP FOR 3 MONTHS

PRICE: 2000 INR
No Shower Room
Rigid With Timing
No Trainer Help

SIGN UP FOR 6 MONTHS

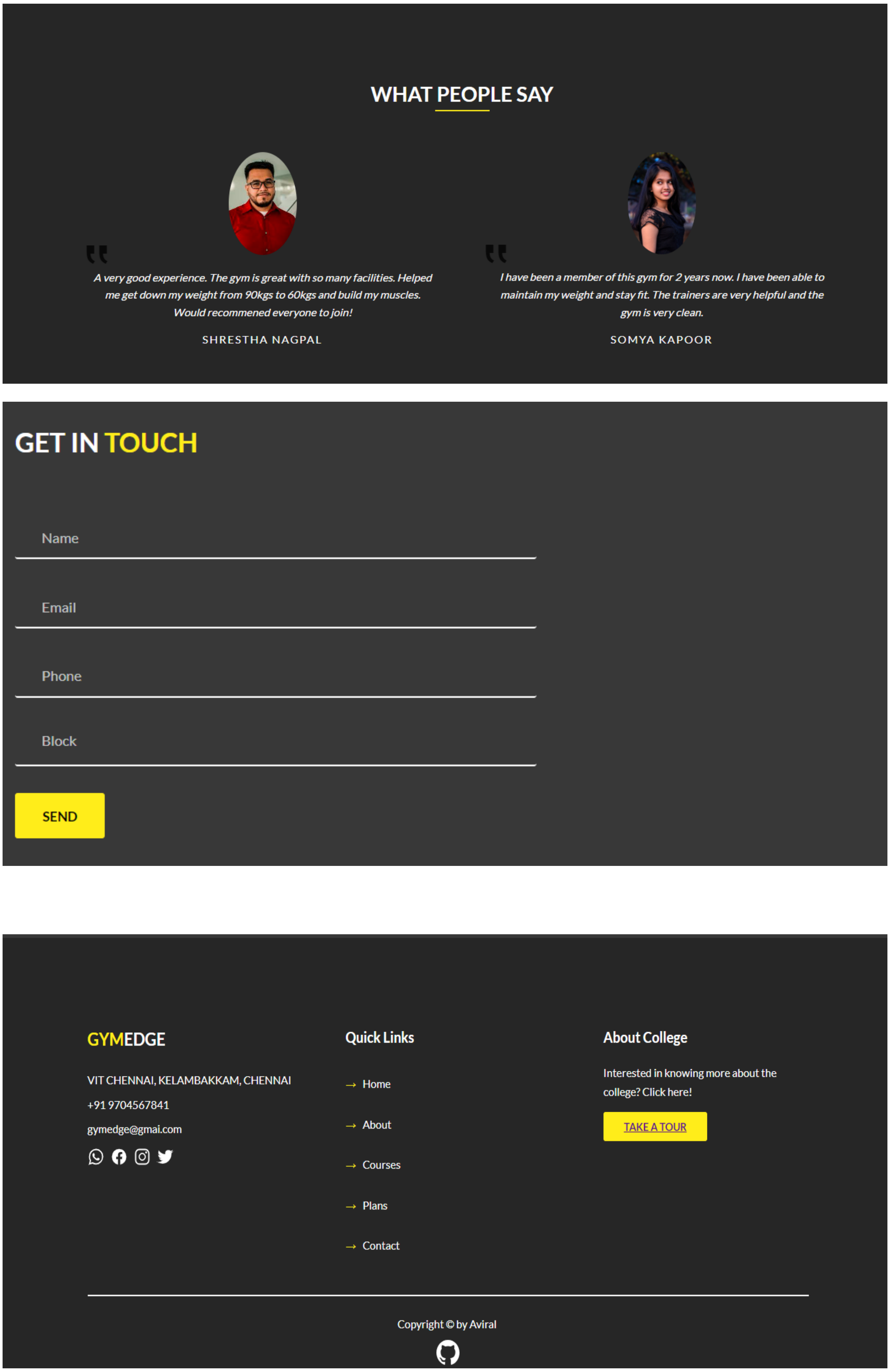
PRICE : 3500 INR
Shower Room Allowed
Rigid With Timing
Trainer Help Given

SIGN UP FOR 12 MONTHS

GET 1 MONTH FREE

PRICE: 6000 INR
Benefits Of Previous Plans
Flexible With Timing

For payment and reciept:



WHAT PEOPLE SAY

A very good experience. The gym is great with so many facilities. Helped me get down my weight from 90kgs to 60kgs and build my muscles. Would recommened everyone to join!

SHRESTHA NAGPAL

I have been a member of this gym for 2 years now. I have been able to maintain my weight and stay fit. The trainers are very helpful and the gym is very clean.

SOMYA KAPOOR

GET IN TOUCH

GYMEDGE

VIT CHENNAI, KELAMBAKKAM, CHENNAI

+91 9704567841

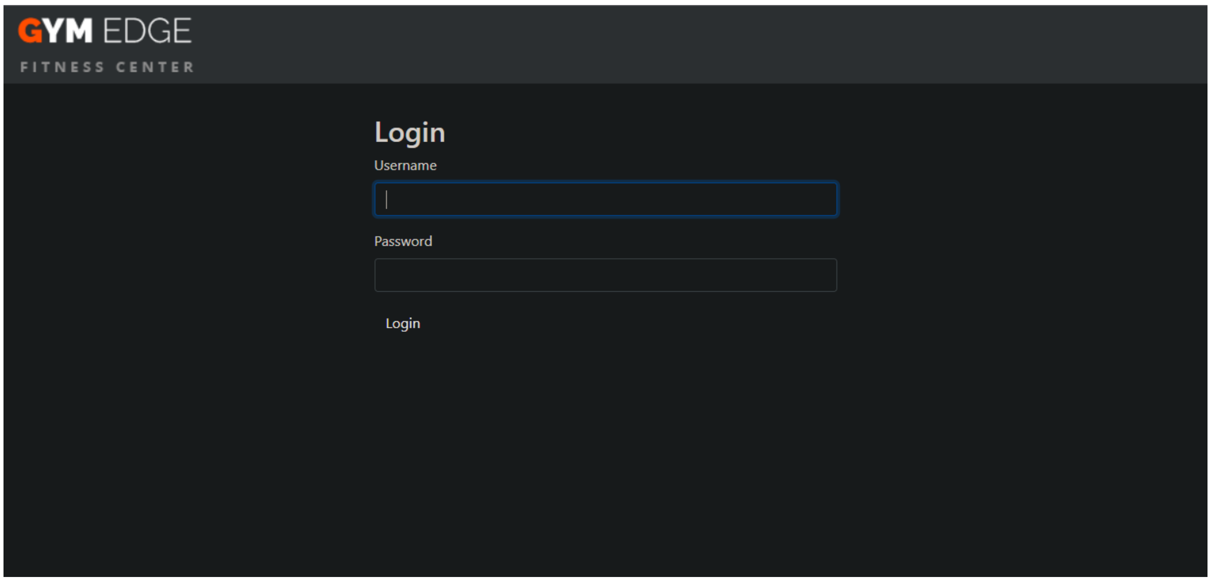
gymedge@gmai.com

Quick Links

About College

Interested in knowing more about the college? Click here!

Copyright © by Aviral

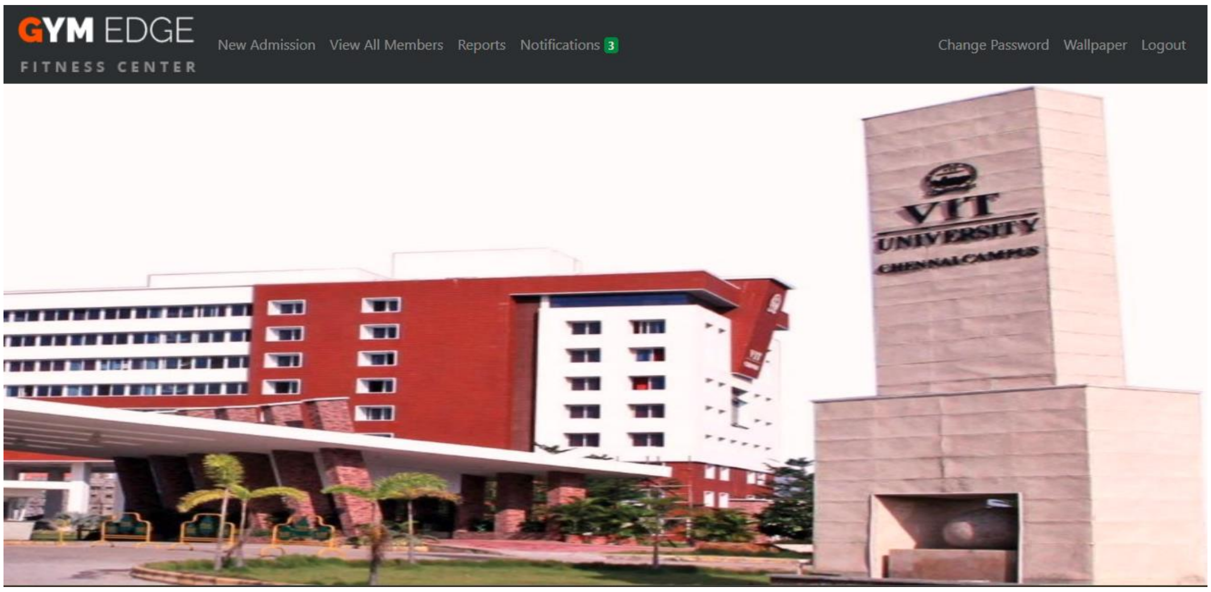


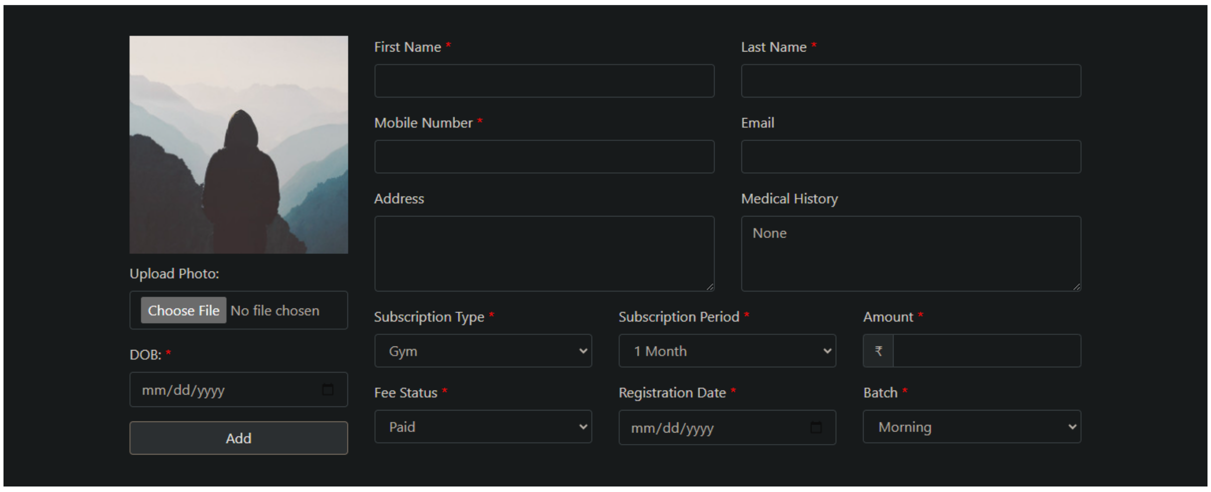
Login

Username

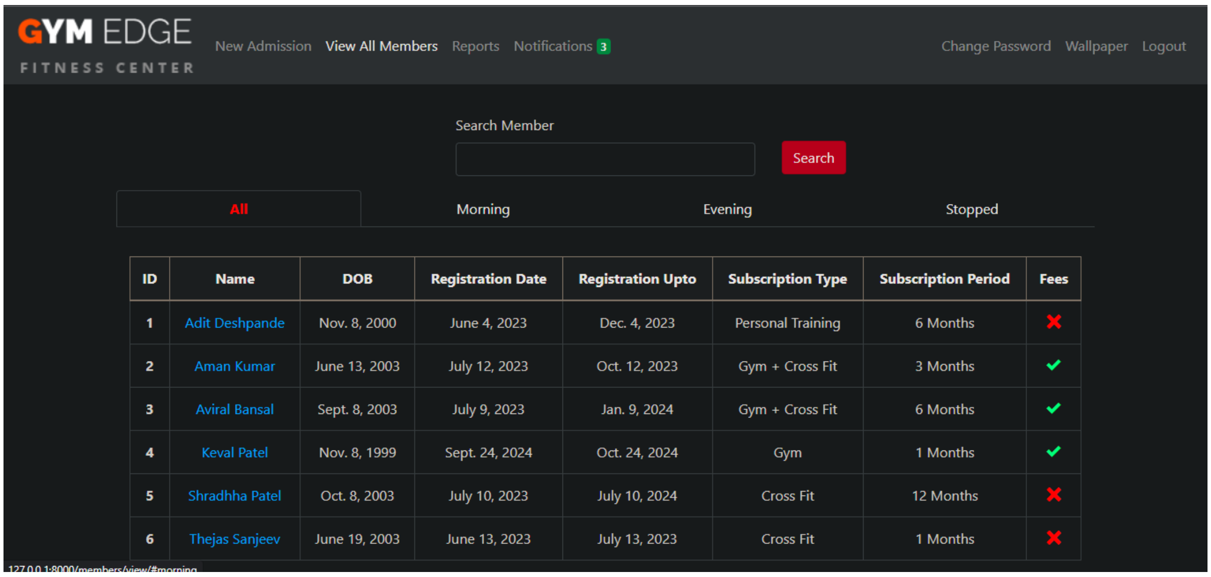
Password

Login



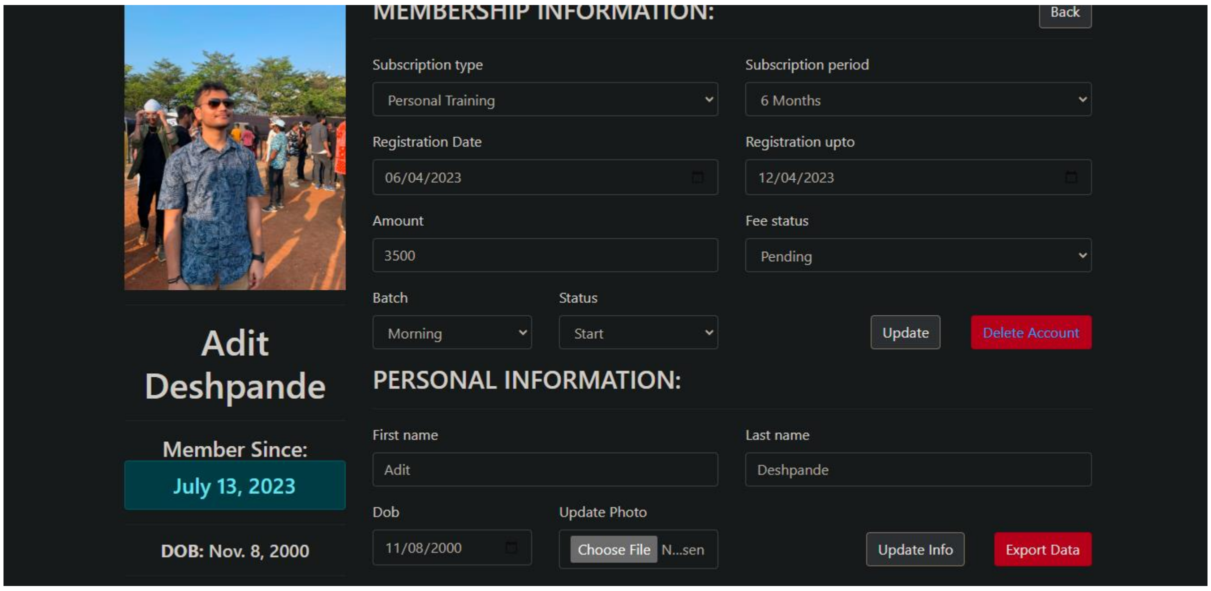


First Name *	Last Name *	
Mobile Number *	Email	
Address	Medical History: None	
DOB: * mm/dd/yyyy		
Subscription Type *: Gym	Subscription Period *: 1 Month	Amount *: ₹
Fee Status *: Paid	Registration Date *: mm/dd/yyyy	Batch *: Morning



ID	Name	DOB	Registration Date	Registration Upto	Subscription Type	Subscription Period	Fees
1	Adit Deshpande	Nov. 8, 2000	June 4, 2023	Dec. 4, 2023	Personal Training	6 Months	✗
2	Aman Kumar	June 13, 2003	July 12, 2023	Oct. 12, 2023	Gym + Cross Fit	3 Months	✓
3	Aviral Bansal	Sept. 8, 2003	July 9, 2023	Jan. 9, 2024	Gym + Cross Fit	6 Months	✓
4	Keval Patel	Nov. 8, 1999	Sept. 24, 2024	Oct. 24, 2024	Gym	1 Months	✓
5	Shradhha Patel	Oct. 8, 2003	July 10, 2023	July 10, 2024	Cross Fit	12 Months	✗
6	Thejas Sanjeev	June 19, 2003	June 13, 2023	July 13, 2023	Cross Fit	1 Months	✗

127.0.0.1:8000/members/view/#morning



MEMBERSHIP INFORMATION:

Subscription type: Personal Training	Subscription period: 6 Months
Registration Date: 06/04/2023	Registration upto: 12/04/2023
Amount: 3500	Fee status: Pending
Batch: Morning	Status: Start

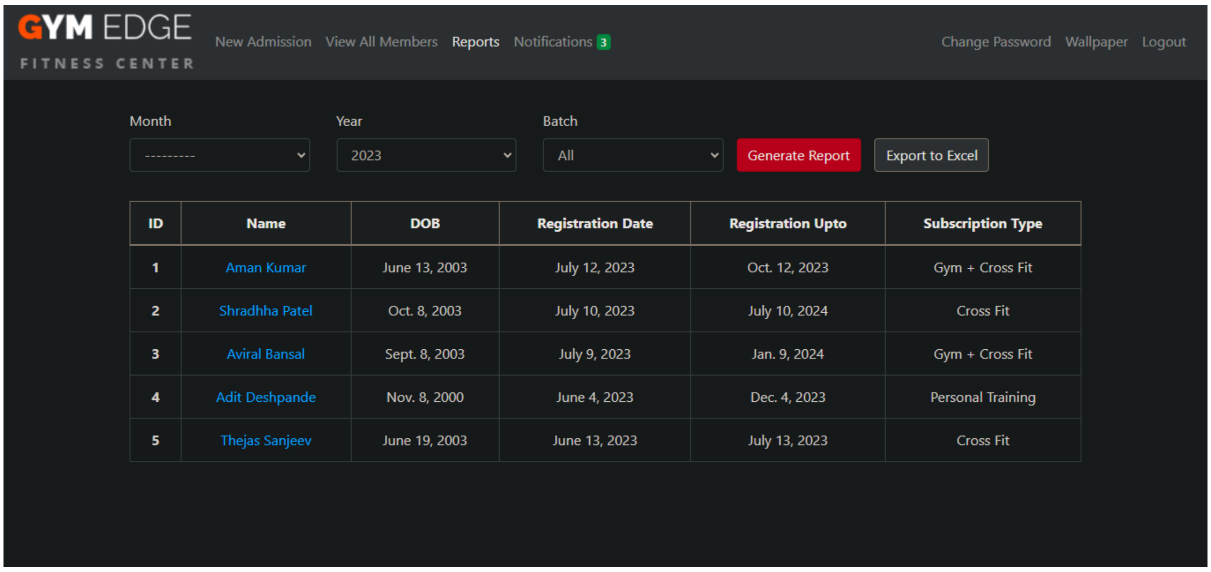
PERSONAL INFORMATION:

First name: Adit	Last name: Deshpande
Dob: 11/08/2000	Update Photo: N...sen

Adit Deshpande

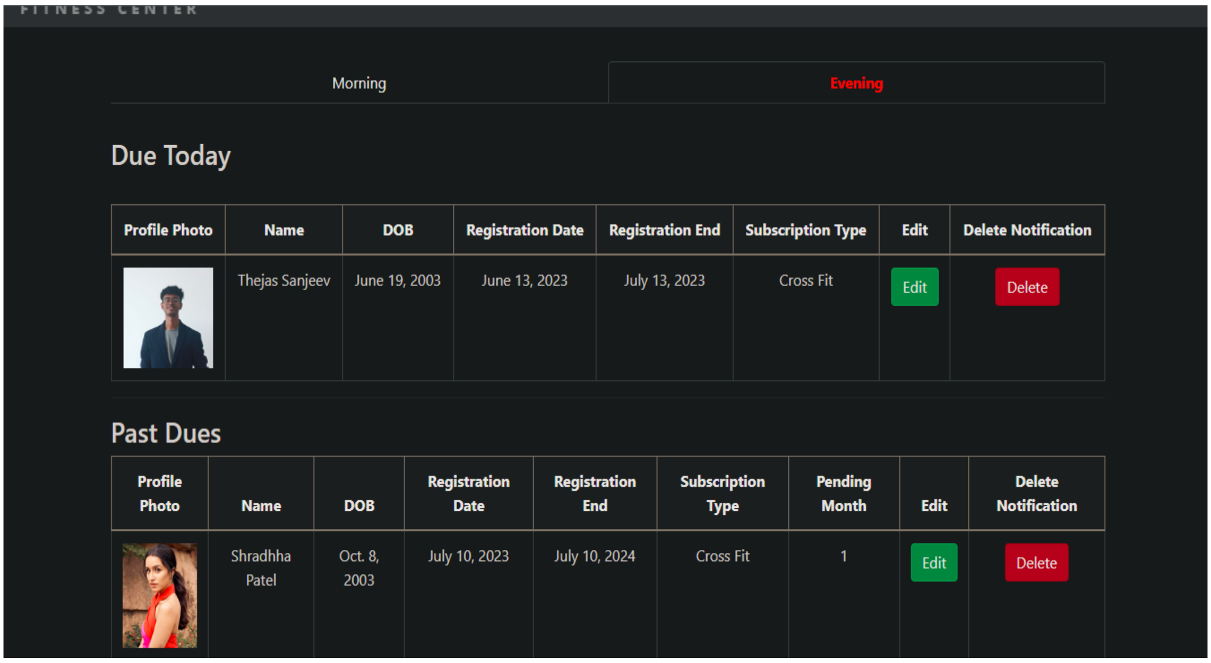
Member Since:
July 13, 2023

DOB: Nov. 8, 2000



Month: --------- Year: 2023 Batch: All

ID	Name	DOB	Registration Date	Registration Upto	Subscription Type
1	Aman Kumar	June 13, 2003	July 12, 2023	Oct. 12, 2023	Gym + Cross Fit
2	Shradhha Patel	Oct. 8, 2003	July 10, 2023	July 10, 2024	Cross Fit
3	Aviral Bansal	Sept. 8, 2003	July 9, 2023	Jan. 9, 2024	Gym + Cross Fit
4	Adit Deshpande	Nov. 8, 2000	June 4, 2023	Dec. 4, 2023	Personal Training
5	Thejas Sanjeev	June 19, 2003	June 13, 2023	July 13, 2023	Cross Fit


FITNESS CENTER

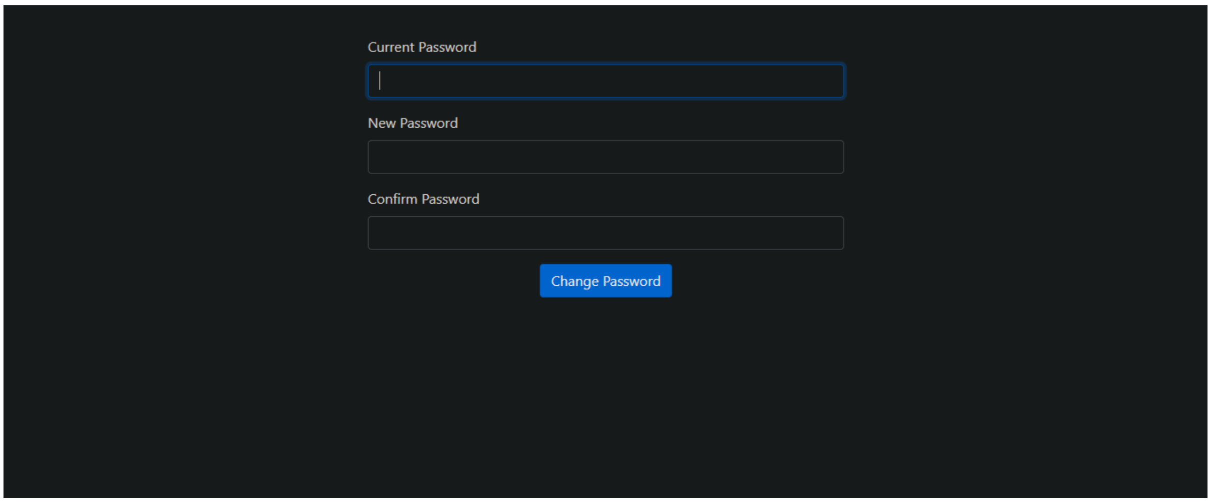
Morning | Evening

Due Today

Profile Photo	Name	DOB	Registration Date	Registration End	Subscription Type	Edit	Delete Notification
	Thejas Sanjeev	June 19, 2003	June 13, 2023	July 13, 2023	Cross Fit	Edit	Delete

Past Dues

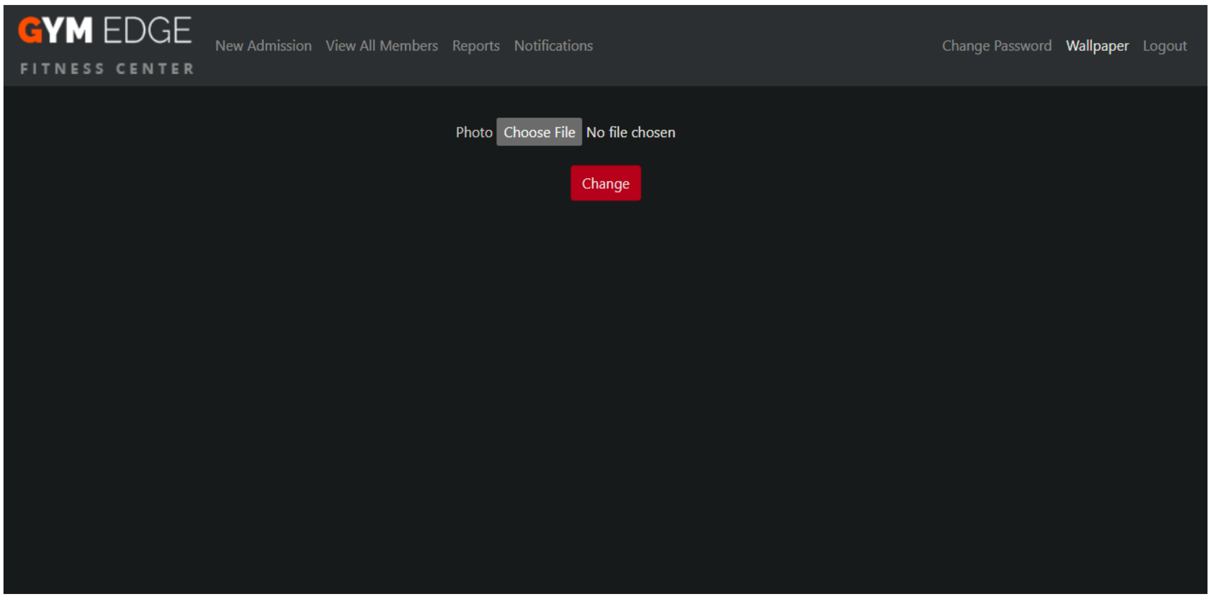
Profile Photo	Name	DOB	Registration Date	Registration End	Subscription Type	Pending Month	Edit	Delete Notification
	Shradhha Patel	Oct. 8, 2003	July 10, 2023	July 10, 2024	Cross Fit	1	Edit	Delete


Current Password

New Password

Confirm Password

Change Password


GYM EDGE
FITNESS CENTER

New Admission View All Members Reports Notifications Change Password Wallpaper Logout

Photo Choose File No file chosen

Change

CONCLUSION:

We have solved our problems for which we designed the Gym Management System. It gives a database to the admin so that he can manage all the members of the gym digitally and does not have to carry around a notebook with him. He can verify the entry of each member, check which batch they belong to, whether they have paid the fees or not, which type of plan do they have and more. They can generate reports and save them in excel file. The program notifies them whose fees are due and by how long.

For the users we made a marketing page for the gym where they can check how the gym looks like, what it is about, what courses it offers, what are the plans and if he wishes to join he can pay as well.

It gives the functionality of being able to connect with the gym and check other users review as well. He can check out the social media handles and check out the college website.

FUTURE WORK:

- The ability to do the payment and generate the receipt inside the website itself.
- A whole database login for the user which helps them to check their stats, their plan, duration left, whether they have any dues left and more.