Flex forge - Your One-Stop Solution for Fitness and Wellness.

Welcome to Flex forge! This platform provides a variety of fitness services, including tailored workout plans, personal training, and nutrition guidance. Our goal is to make managing your fitness journey simple and accessible through a seamless, user-friendly interface.

Project Overview

Flexanza is a web application enhancing the gym experience for both members and trainers. It helps gyms showcase their services, making it easier for potential members to find the right fit and access essential information. The platform simplifies gym membership management, workout planning, and nutrition guidance for a seamless fitness journey.

Purpose

1. Attracts potential members by showcasing gym facilities and offerings.
2. Offers comprehensive features and workout plans to enhance member experience.
3. Facilitates effective communication between members and trainers.

Goals

- **Visibility -** Help gyms increase their online presence to attract more members.
- **Member Engagement -** Provide valuable features that enhance member satisfaction and retention.
- **Streamlined Operations -** Ensure smooth operation of membership management and trainer assignments.

Demo

You can view the static version of the Flexforge project here - Website

Project Structure

Here is an overview of the project structure with links to key files and directories:

- static/
  Contains static assets such as images used throughout the website.
  - images/
    This folder holds images for various services, such as workout types, feedback emojis, and home page images etc

- templates/
  Contains the HTML templates for different pages of the web application.
  - home.html

The main landing page highlights key sections such as **Contact**, **Blog**, **About ,Trainer/Member**, and **Membership Plans**, providing a seamless way to explore services and connect with the Flex forge community.

- flaskCode.py
  The main Python script is responsible for handling **membership requests**, **trainer sessions**, and the **routing logic** for managing trainers, members, and membership plans on the Flex forge gym portal.

- Code Documentation.pdf

  Documentation covering the routing and Flask Backend implementation.

- Database MySQL Queries.pdf
  A PDF containing the MySQL queries used to set up and manage the database for **members**, **trainers, progress report**, and **membership plans**.

- Libraries and Modules.pdf
  A PDF containing detailed description and purpose of libraries and modules used in the course of project.

- requirements.pdf
  A list of the Python dependencies needed to run the project locally.

- Licence
  The license under which this project is released.

- README.md
  The current project documentation.

Forms & Routing

1. Member Registration Form

The Member Registration Form collects essential details from gym members for registration. The form includes:

- Full Name

- Email (must be unique)

- Password

- Confirm Password

This information is stored securely in the `user` table of the MySQL database.

Backend Logic:

1. Password Hashing: User passwords are hashed using the `sha256` method before being stored.

2. Email Uniqueness: Duplicate email addresses are checked to ensure uniqueness.

3. Data Insertion: After validation, user details are saved in the database.

Trainer Assignment:

- The `trainer_id` in the `user` table is allocated after the user takes a membership plan and is assigned a personal trainer based on their membership tier.

User Flow:

1. User fills out the form with required details.

2. Password is hashed, and email uniqueness is validated.

3. User data is stored in the `user` table, and they are redirected to the login page.

4. The `trainer_id` will be assigned once the user completes membership registration.

2. Membership Payment Form

The Membership Payment Form is used to complete the purchase of a membership plan. It includes the following details:

- UPI ID

- Email (must match the registered email)

- The Training Type and Tier are pre-selected on the previous page.

This payment information is stored securely in the `payments` table.

Database Table: `payments`

- Columns:

- `id`: Auto-incremented unique identifier for each payment.

- `training_type`: The type of training the member selected.

- `tier`: Membership tier (Basic, Advanced, Premium).

- `amount`: The amount paid for the membership.

- `upi_id`: The UPI ID used for the transaction.

- `email`: User's email for payment verification.

- `payment_date`: Timestamp of the payment.

- `validation`: Status of the payment (default is 'no').

Backend Logic:

1. Payment Information: The form submits the UPI ID and email, while the training type and tier are pre-selected from the previous page.

2. Data Insertion: After payment, this information is stored in the `payments` table.

3. Validation: The `validation` column is set to 'no' by default. The admin reviews the payment details, and upon approval, updates the validation to 'yes'. Only then can the member log in to the member dashboard.

User Flow:

1. User selects a membership plan and training type on the previous page.

2. User fills out the UPI ID and email and clicks the "Pay Now" button.

3. After payment, the user is directed to a Thank You page, informing them that they will receive membership benefits within 24 hours.

4. The admin checks the payment details, updates the validation status to 'yes', allowing the member to access the member dashboard.

3. Member Login Page

The Member Login Page allows gym members to securely log in to their accounts. It collects the following details:

- Email

- Password

Login Process:

1. Email and Password Verification:

- On each login attempt, the system checks if the email exists in the `user` table.

- The entered password is hashed and compared with the stored hashed password in the `user` table.

2. Trainer ID Check:

- The system first checks the `trainer_id` column in the `user` table.

- If `trainer_id` is null (indicating no trainer has been assigned yet), the system then checks the `payments` table to verify if the member has paid for their membership by looking at the `validation` column.

- If `validation` is set to 'yes', a personal trainer is assigned based on the member's training type and the trainer's expertise.

- If `validation` is not set to 'yes', the member is redirected to the Membership Plans page to complete their payment.

3. Subsequent Logins:

- If a trainer has already been assigned (i.e., `trainer_id` is not null), the member can log in without reassignment of a trainer.

User Flow:

1. Member enters their email and password and submits the form.

2. The system verifies the credentials and checks for trainer assignment and payment status.

3. If valid and payment is confirmed (`validation = 'yes'`), the member is logged in, and a trainer is assigned if it’s their first login.

4. If the payment validation is not confirmed, the member is redirected to the Membership Plans page.

5. The member is then redirected to the member dashboard upon successful login.

4. Trainer Registration Page

The Trainer Registration Page allows trainers to register and provide their details for inclusion in the gym's trainer database. It collects the following information:

- Full Name

- Email

- Password

- Confirm Password

- Expertise (Select from predefined options)

- Certifications

User Flow:

1. The trainer fills out the registration form with the required details.

2. Upon submission, the system validates the inputs and checks for existing email addresses to avoid duplicates.

3. If all validations pass, the trainer's information is securely stored.

4. After successful registration, the trainer is redirected to the Trainer Login Page to access their account.

5. Trainer Login Page

The Trainer Login Page allows registered trainers to access their accounts by providing:

- Email

- Password

Login Process:

1. Trainers enter their email and password.

2. The system verifies the credentials against the `trainers` table.

3. Upon successful login, trainers are directed to their dashboard.

4. If the credentials are invalid, trainers are redirected to the Trainer Registration Page.

6. Member Progress Tracking Page

The Member Progress Tracking Page allows members to self-report their progress by providing:

- Date

- Workout Details

- Any Issues

Data Storage:

The information submitted is stored in the `progress_tracking` table with the following fields:

- `id` int AI PK

- `user_id` int

- `trainer_id` int

- `progress_date` date

- `workout_details` text

- `issues` text

- `created_at` timestamp

Submission Process:

1. Members fill out the form with their progress details.

2. Upon successful submission, they are redirected to the Member Dashboard Page.

Trainer Access:

- When trainers log into their dashboard, they can access the progress details from the `progress_tracking` table to observe and assess member progress closely.

7. Feedback Form

The Feedback Form allows members to provide their feedback through the following inputs:

- Rating (1 to 5, represented through images)

- Category (Options: Suggestion, Something is Not Quite Right, Compliment)

- Feedback Text (Please leave your feedback below)

Data Storage:

The submitted information is stored in the `feedback` table with the following fields:

- `id` int AI PK

- `user_id` int

- `rating` tinyint

- `category` varchar(50)

- `feedback` text

- `created_at` timestamp

Submission Process:

1. Members complete the form with their feedback.

2. Upon submission, the feedback is saved to the database for review.

Utilizing Feedback:

- The collected feedback can be analyzed to calculate the average rating from multiple members.

- This average feedback helps identify areas for improvement, allowing the gym to address issues and enhance member satisfaction.

Installation

To run this project locally, follow these steps:

1.Clone the Repository:

- git clone https://github.com/yourusername/Flexforge-RealTime-Project.git

2.Navigate to the Project Directory:

- cd Flexforge

3.Install the Required Dependencies:

- pip install -r requirements.txt

4.Run the Python Backend:

- python flaskCode.py

5.Open Your Browser:

- Go to http://localhost:5000 to view the application.

Technologies Used

- **Frontend**: HTML5, CSS3, JavaScript, Tailwind CSS for responsive and modern design
- **Backend**: Python, Flask for routing and form handling.
- **Database**: Managed using MySQL queries stored in Flexforge Project Database MySQL Queries.pdf.
- **Deployment**: The project is hosted using github for static files.

Code Documentation

1. **Overview**: For an in-depth understanding of the gym portal's code structure, please refer to the accompanying code documentation PDF.

2. **Module Explanations**: The document provides detailed explanations of key modules, functions, and classes, outlining their specific purposes.

3. **API Details**: It includes information on public APIs, along with their usage instructions, parameters, and return values.

4. **Comments and Best Practices**: The code is organized with clear comments and follows best practices for maintainability and ease of navigation.

5. **Access**: You can access the documentation

Documentation Validation

To ensure the quality, accuracy, and reliability of our gym portal documentation, we have implemented the following validation methods:

1. **Peer Review**:

- The documentation has undergone thorough review by multiple team members and developers. Feedback focused on clarity, completeness, and adherence to best practices. All suggestions and corrections have been incorporated into the final version to enhance understanding and usability.

2. **User Testing**:

- We conducted testing sessions with users who were unfamiliar with the gym portal. These users followed the documentation step-by-step to set up, install, and navigate the system. Their feedback was instrumental in improving the clarity of instructions and ensuring that all processes are easy to follow, thereby enhancing the overall user experience.

3. **Functional Testing**:

- Every step outlined in the documentation has been executed precisely as described to validate functionality. This includes:

- **Setting Up the Environment**: Detailed guidance on configuring the necessary development environment for the gym portal.

- **Installing Required Dependencies**: Clear instructions for installing all necessary libraries and packages to ensure the project runs smoothly.

- **Running the Project Locally**: Step-by-step instructions for launching the gym portal on a local server, including troubleshooting tips.

- **Testing Forms and Routing**: Comprehensive checks on all forms and routing functionalities to ensure they operate as expected. Each feature has been tested to confirm that the provided instructions lead to a successful setup and that the project behaves as intended.

These validation methods collectively ensure that our documentation is not only accurate but also user-friendly, providing a solid foundation for anyone looking to use or contribute to the gym portal project.

License

This project is licensed under the MIT License. See the Licence file for details.

Contact

For further information or queries, please reach out via email at [peralapranitha17@gmail.com].

Feel free to connect with me on Linkedin - LinkedIn