

Yoga Admission Form - YogaEase

Documentation



Made by Manaswi Chiwande

1. Introduction

The Yoga Classes Admission Form documentation provides a comprehensive guide for developers and stakeholders involved in building an admission form for monthly yoga classes. It outlines the requirements, implementation details, and considerations for the project. This digital form serves as a user-friendly interface through which individuals can provide necessary information and express their interest in attending yoga sessions.

The Yoga Classes Admission Form encompasses the entire enrollment process, from the initial submission of participant details to the successful completion of the payment transaction. It includes the selection of preferred batches, adhering to the specified age limit, and ensuring a smooth transition for participants who may choose different batches in subsequent months.

The Yoga Studio Management System is a web application built using the Flask web framework, designed to manage user registrations, class schedules, and payments for a yoga studio. The application uses SQLite as the database to store user information, class schedules, and payment records.

2. Requirements Analysis

In this section, we delve into the detailed requirements of the Yoga Classes Admission Form, covering aspects related to age limits, payment structures, the enrollment process, monthly fees, and the flexibility provided to participants in choosing batches.

2.1 Age Limit

Participants must fall within the age range of 18 to 65 to be eligible for enrollment in the monthly yoga classes. This age restriction ensures that participants are in a suitable demographic for engaging in yoga sessions.

2.2 Payment Structure

The payment structure is designed for month-to-month transactions. Individuals are required to pay the monthly fee for the yoga classes, and they have the flexibility to make the payment at any time during the month. This structure accommodates the diverse schedules and preferences of participants. The monthly fee for the yoga classes is fixed at 500/- Rs INR. This flat fee simplifies the payment process for participants, offering a straightforward and consistent pricing model.

2.3 Enrollment Process

Participants can enroll on any day of the month, providing flexibility and convenience. However, the enrollment requires the payment of fees for the entire month. The enrollment process includes the submission of essential details, such as personal information and preferred batches.

2.4 Flexibility for Participants

Flexibility is provided to participants regarding their batch selection. While they can choose any batch in a given month, they also have the freedom to switch to a different batch in subsequent months. This feature allows participants to adapt their schedules based on changing circumstances or preferences.

3. Implementation

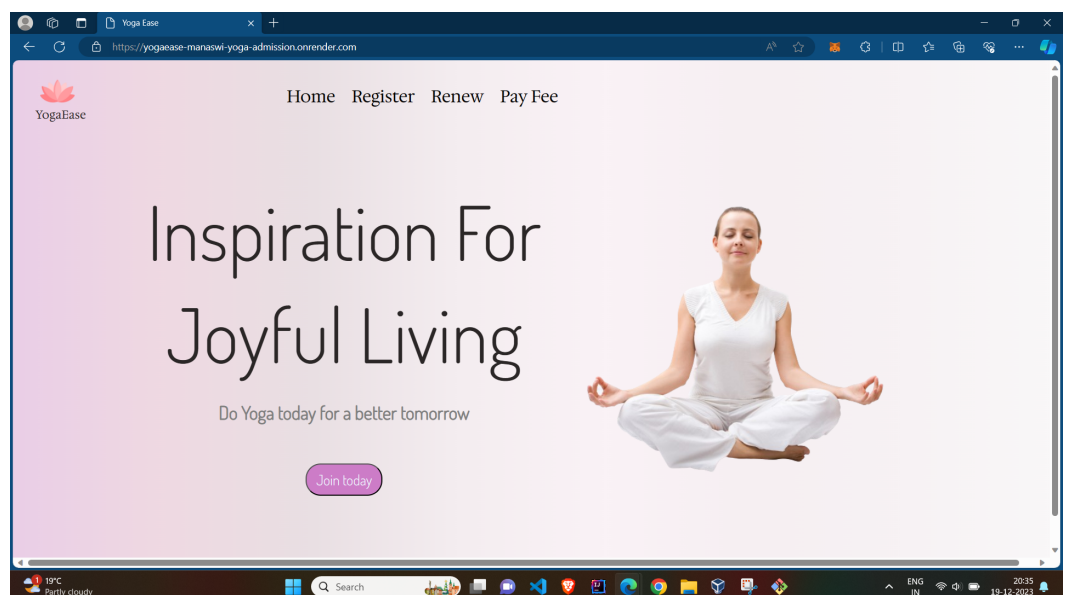
This section outlines the technical implementation details for building the Yoga Classes Admission Form. It covers frontend development, backend development, database design, a mock payment function, response handling, and hosting considerations.

3.1 Front-End Development

- As part of the Yoga Classes Admission Form project, I spearheaded the development of the frontend, crafting a seamless and intuitive user interface.
- Apart from the main pages, the website also incorporates JavaScript functionalities for dynamic user interactions, ensuring a responsive and engaging experience.
- The frontend aligns with modern web development standards, leveraging **HTML, CSS, and JavaScript** to create an aesthetically pleasing, user-centric, and fully functional website.
- To facilitate a deeper understanding of the project, users can seamlessly navigate to the dedicated "About" page, which redirects them to the comprehensive project documentation.
- Website is **Responsive**.

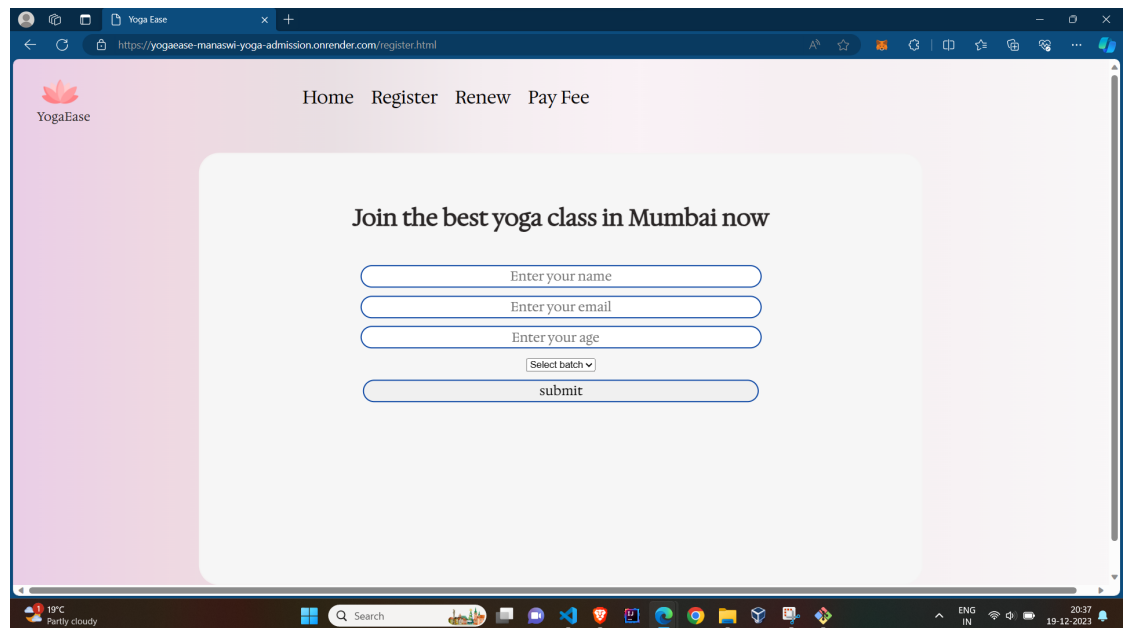
3.1.1 Home Page - Introduction

The Home page acts as the introduction to the Yoga Classes platform, providing users with essential information about the classes, their benefits, and what to expect. It serves as the welcoming gateway to the website.



3.1.2 Register Page

The Register page is where participants can initiate the enrollment process. A meticulously designed form prompts users to enter their details, including name, email, age, contact number, and select their preferred batch size. JavaScript validation is implemented to ensure that the entered email is correct, the age falls within the range of 18 to 65, and all required fields are filled before submission.

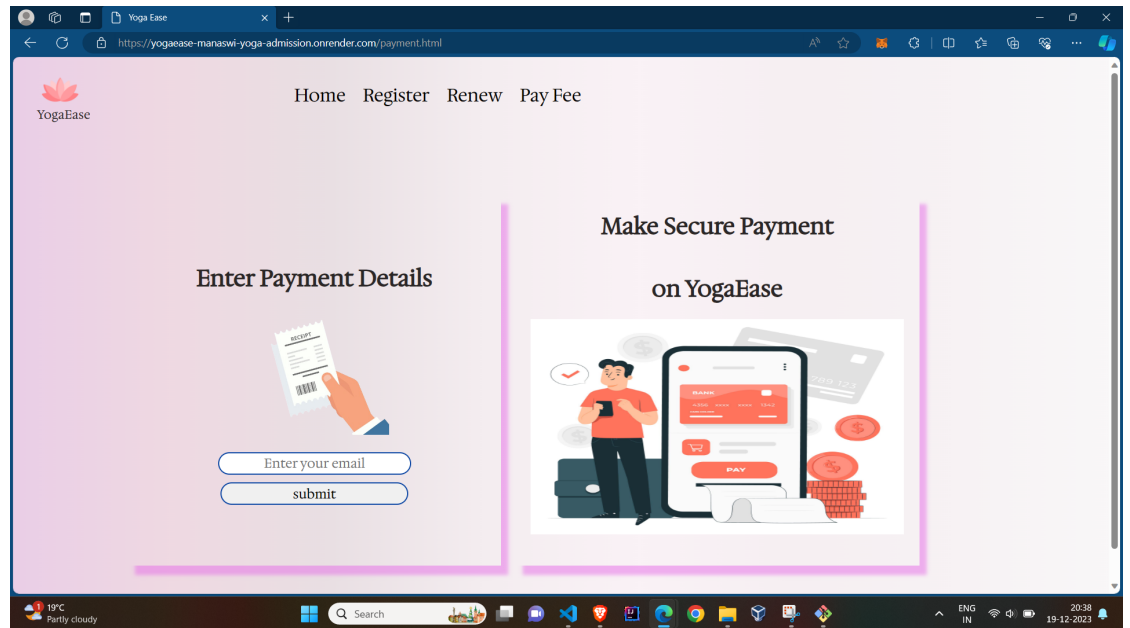
A screenshot of a web browser displaying the 'YogaEase' registration page. The browser's address bar shows the URL 'https://yogaease-manaswi-yoga-admission.onrender.com/register.html'. The page has a light pink background with a white central form area. At the top left is the 'YogaEase' logo, and at the top right are navigation links: 'Home', 'Register', 'Renew', and 'Pay Fee'. The form is titled 'Join the best yoga class in Mumbai now'. It contains five input fields: 'Enter your name', 'Enter your email', 'Enter your age', a dropdown menu labeled 'Select batch', and a 'submit' button. The Windows taskbar at the bottom shows the system clock as 20:37 on 19-12-2023, with a weather widget indicating 19°C and 'Partly cloudy'.

3.1.3 Renew Page

This page is used by already existing users to renew and register for next month. We just take email and select a batch from the user. Then information is sent to the api.

3.1.4 Payment Page

The Payment page streamlines the payment process into two steps. In the first step, users are prompted to enter email and to make a payment of 500 rupees.

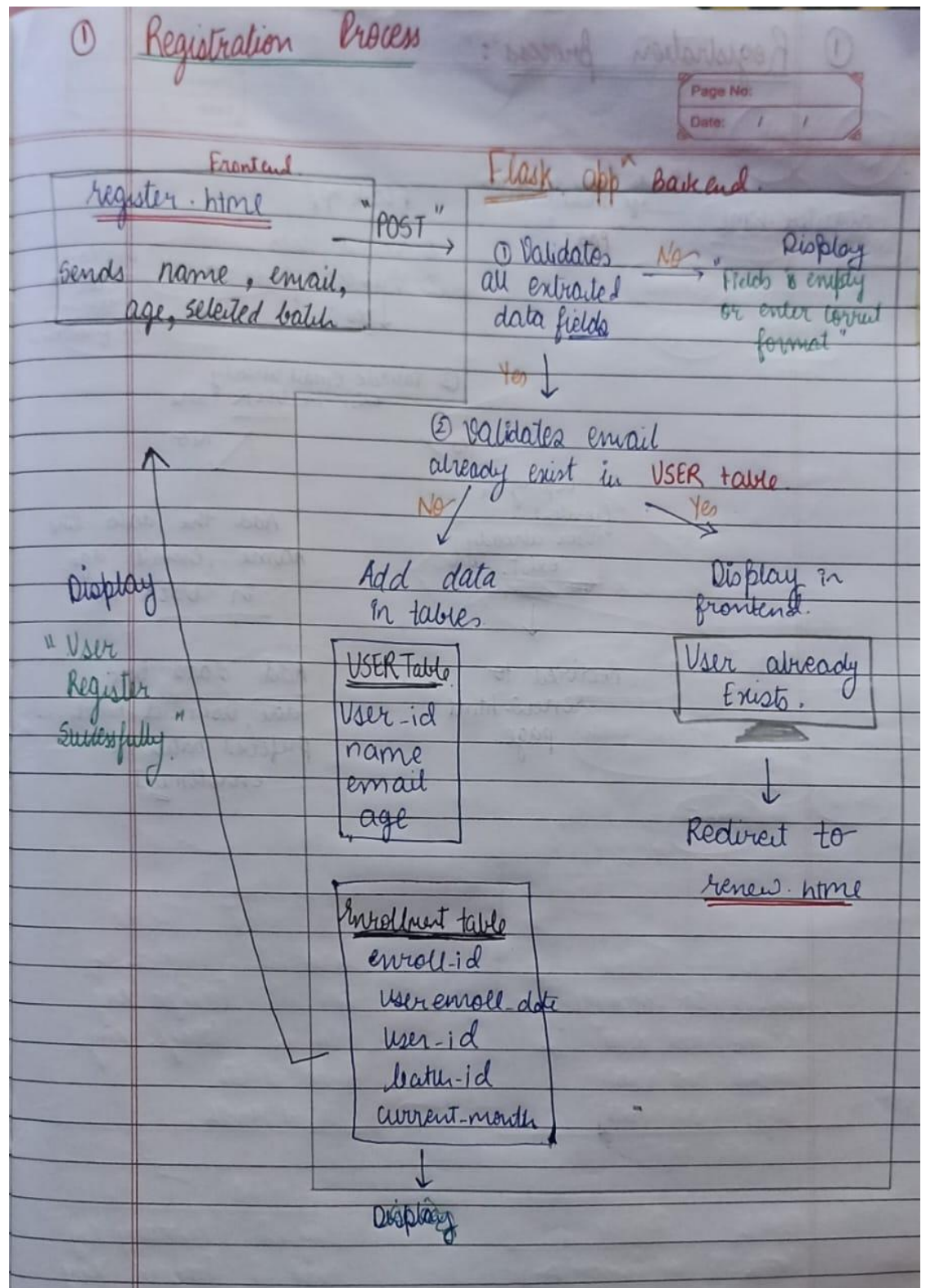


3.2 Back-End Development

Here two apis are made. One for registration and other for renewal.

3.2.1 Registration Process

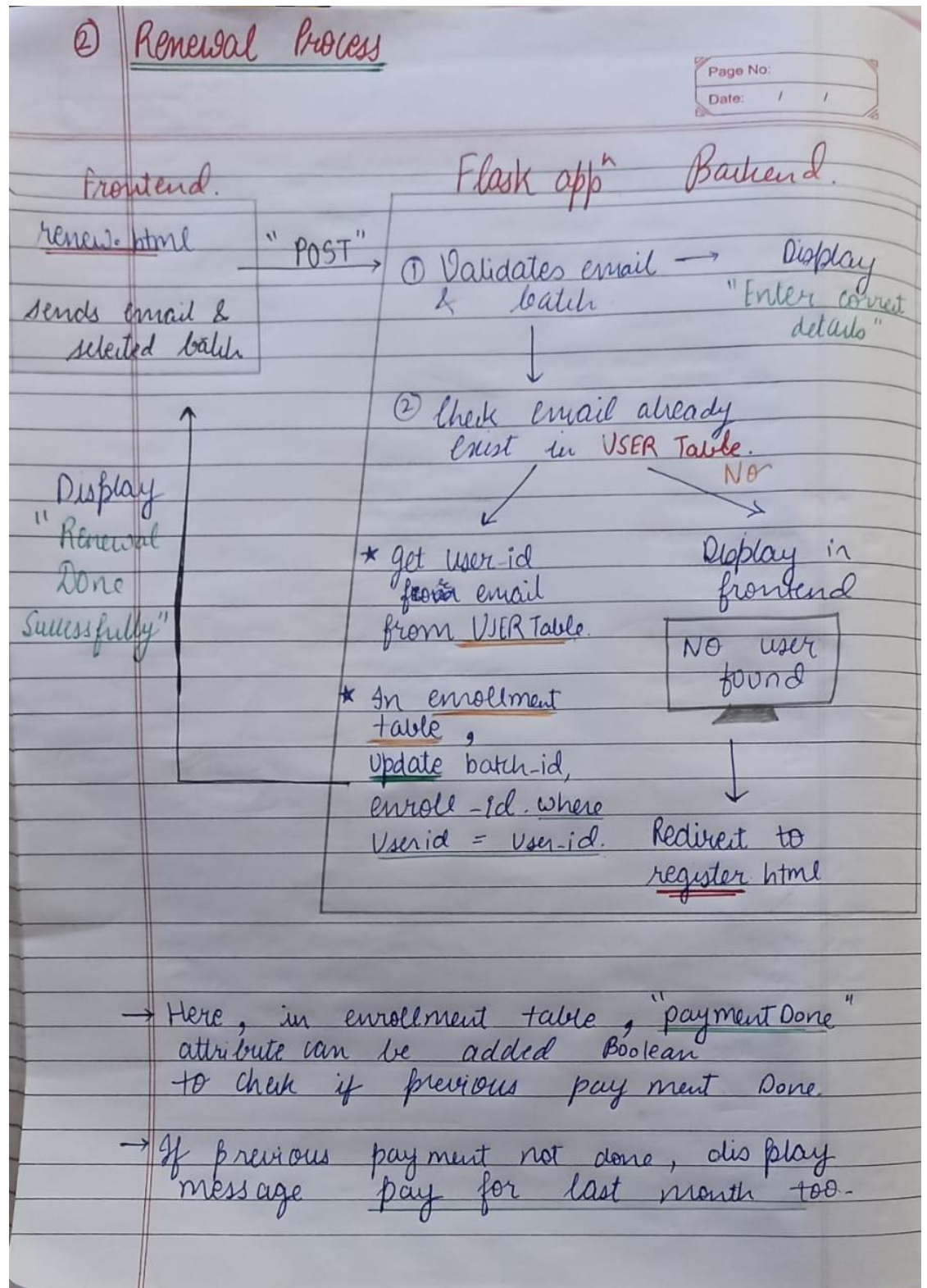
The Home page acts as the introduction to the Yoga Classes platform, providing users with essential information about the classes, their benefits, and what to expect. It serves as the welcoming gateway to the website.



3.2.2 Subscription Renewal Page

The Register page is where participants can initiate the enrollment process. A meticulously designed form prompts users to enter their details, including name, email, age, contact number, and select their preferred batch size.

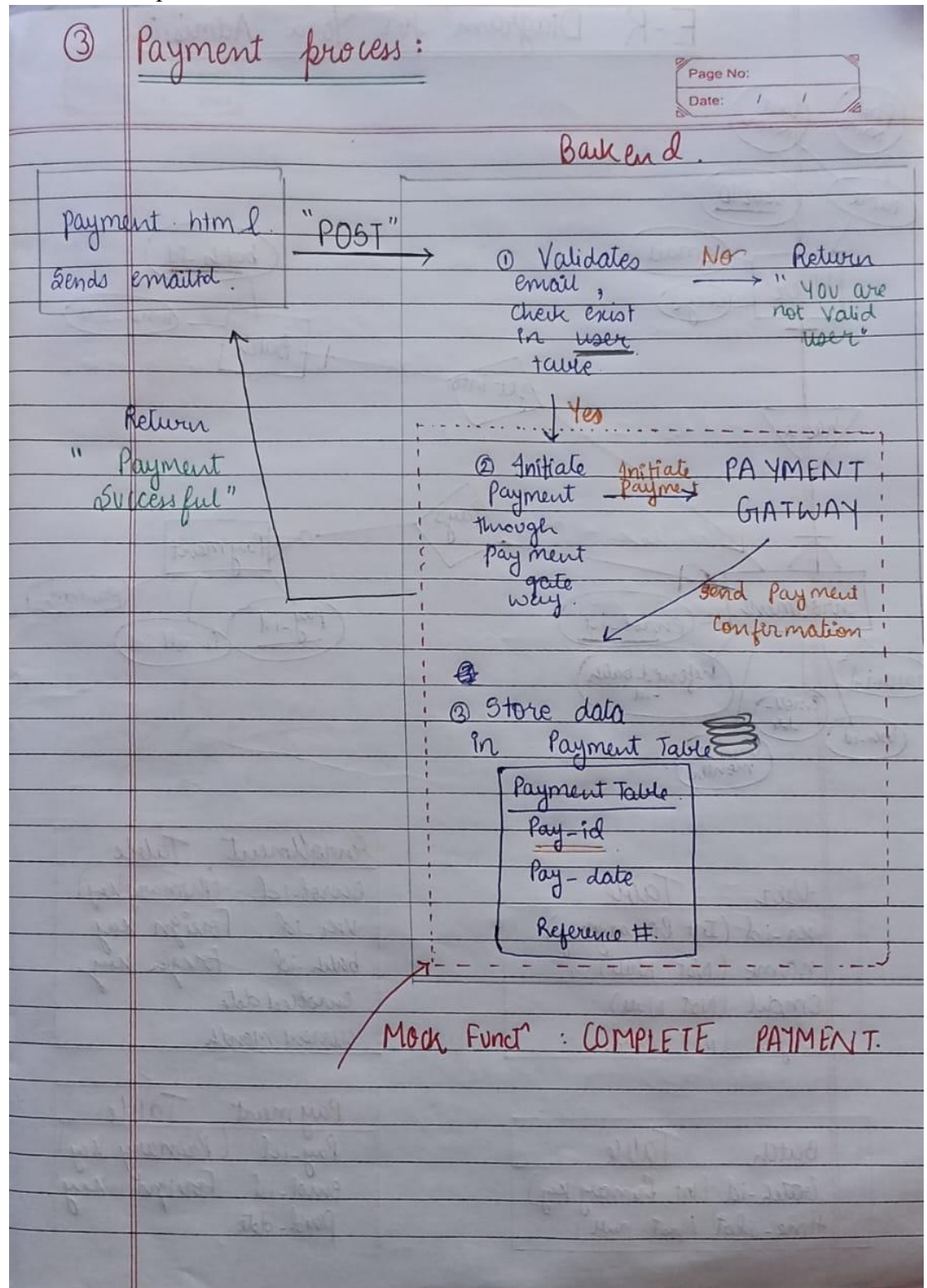
JavaScript validation is implemented to ensure that the entered email is correct, the age falls within the range of 18 to 65, and all required fields are filled before submission.



3.3.3 Payment Page

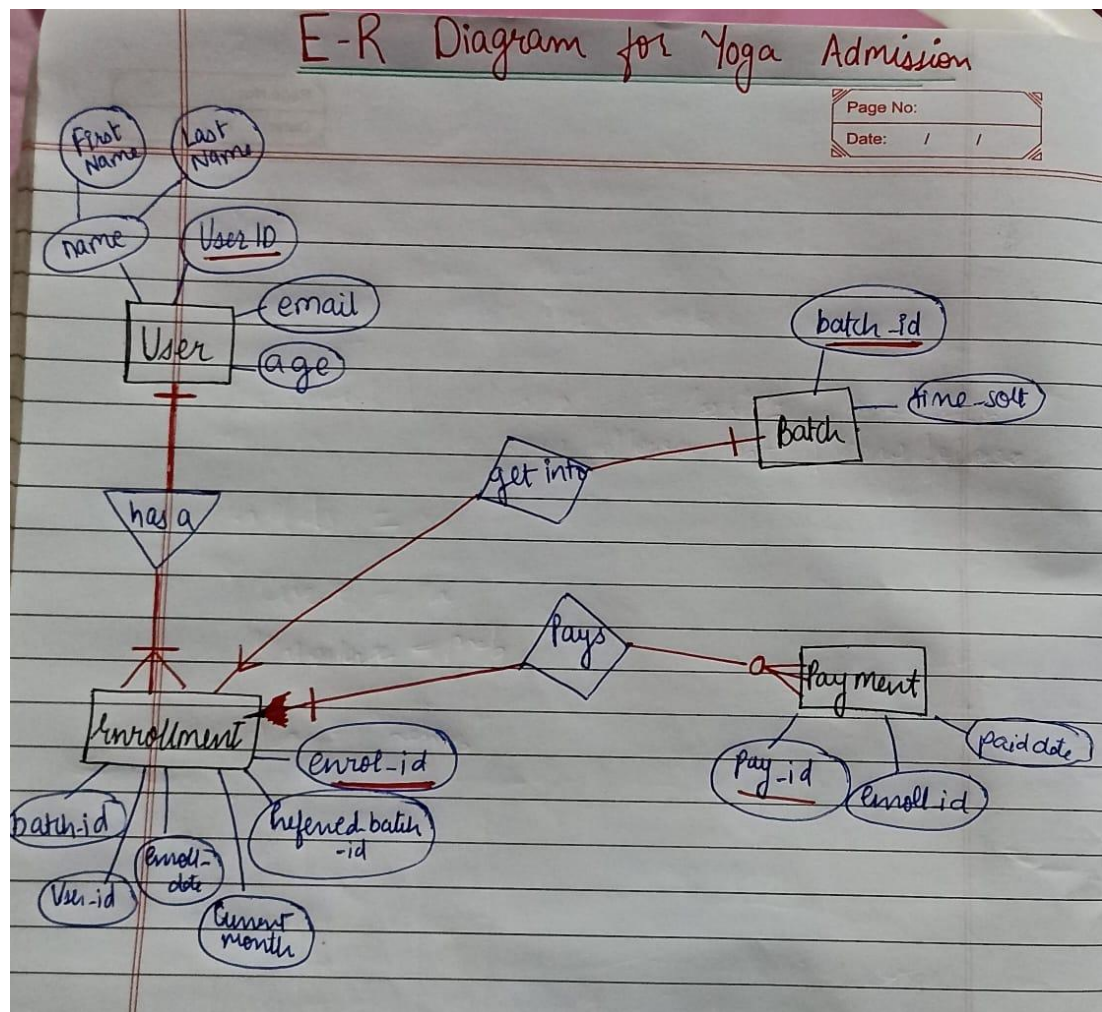
This page is used by already existing users to renew and register for next

month. We just take email and select a batch from the user. Then information is sent to the api.



3.3 Database Design

3.3.1 Entity Relationship (ER) Diagram



3.3.2 Table Definitions and Relationships

The application defines the following database models using Flask SQLAlchemy:

User: Represents a yoga studio client with attributes such as user_id, user (username), email, and age.

Batch: Represents different yoga class batches with attributes like batch_id and timeslot.

Enrollment: Records the enrollment of a user in a specific batch with attributes such as enrollment_id, user_id, batch_id, and enrolled_date.

Payment: Records payment information with attributes like payment_id, user_id, enrollment_id, and paid_date.

| User Table | Enrollment Table |
|---------------------------|------------------------|
| user-id (Int Primary key) | enrol-id (Primary key) |
| name (Not Null) | user-id Foreign key |
| email (Not Null) | batch-id Foreign key |
| age (Not null) | enrolled date |
| | current-month |

| Batch Table | Payment Table |
|----------------------------|----------------------|
| batch-id (Int Primary key) | pay-id (Primary key) |
| time-slot (not null) | enrol-id Foreign key |
| | paid-date |

Mapping Cardanility :

- Batch To Enrollment : 1: N
- Enrollment to Payment : N : (0-N)
- User to Enrollment : 1:N
- Enrollment to Batch : 1:1
- Payment to Enrollment : zero or many : 1

- Using Enrollment table , we can check previous and this timestamp month , and then update the batch timing if it is not same
- We add unique constraints on user_id & current_month to ensure only one active enrollment in a month.

3.3.3 SQL Statements for Database Creation

1. User Table :

```

21
22 #Definign tables in databases:
23 class User(db.Model):
24     user_id = db.Column(db.Integer, primary_key=True)
25     user = db.Column(db.String(255), nullable=False)
26     email = db.Column(db.String(255), unique=True, nullable=False)
27     age = db.Column(db.Integer, nullable=False)
28     enrollments = db.relationship('Enrollment', backref='user', lazy=True)
29

```

2. Enrollment Table :

```

35 class Enrollment(db.Model):
36     enrollment_id = db.Column(db.Integer, primary_key=True)
37     user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'), nullable=False)
38     batch_id = db.Column(db.Integer, db.ForeignKey('batch.batch_id'), nullable=False)
39     enrolled_date = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
40     payments = db.relationship('Payment', backref='enrollment', lazy=True)
41

```

3. Payment Table :

```

42 class Payment(db.Model):
43     payment_id = db.Column(db.Integer, primary_key=True)
44     user_id = db.Column(db.Integer, db.ForeignKey('user.user_id'), nullable=False)
45     enrollment_id = db.Column(db.Integer, db.ForeignKey('enrollment.enrollment_id'), nullable=False)
46     paid_date = db.Column(db.Date)
47

```

4. Batch Table:

```

30 class Batch(db.Model):
31     batch_id = db.Column(db.Integer, primary_key=True)
32     timeslot = db.Column(db.String(20), nullable=False, unique=True)
33     enrollments = db.relationship('Enrollment', backref='batch', lazy=True)
34

```

3.4 Extras

As part of the Yoga Classes Admission Form project, I spearheaded the development of the frontend, crafting a seamless and intuitive user interface. The website consists of five pages, each serving a specific purpose to enhance the user experience.

3.4.1 Routes and End-Points

1. Home Page

- Route: /
- Method: GET
- Description: Renders the home page of the application.

2. User Registration

- Route: /register.html
- Methods: GET, POST
- Description: Handles user registration. Validates user input for username, email, age, and selected class batch. If successful, a new user is added to the database, and the user is enrolled in the selected class batch.

3. User Profile

- Route: /<usr>
- Method: GET
- Description: Displays a confirmation page after successful

registration, indicating the user's registration status.

4. Change Class Schedule

- Route: /changeSchedule.html
- Methods: GET, POST
- Description: Allows users to change their class schedule. Validates user input for email and selected class batch. Updates the user's enrollment information in the database.

5. Payment Processing

- Route: /payment.html
- Methods: GET, POST
- Description: Handles payment processing. Validates user input for email and ensures the user is enrolled. Records the payment in the database.

3.4.2 Application Structure

The application consists of the following main components:

`app.py`: The main Flask application file that defines routes, database models, and business logic.

`templates`: Directory containing HTML templates for rendering pages.

`static` : contains css and javascript files.

`instance` has a database inside it.

3.4.3 Technologies

1. Flask: A micro web framework for Python that provides tools, libraries, and patterns to build web applications.
2. SQLite: A lightweight, file-based relational database management system.
3. Waitress: A production-ready WSGI server for Python web applications.
4. HTML: The standard markup language for creating web pages and web applications.
5. CSS: A style sheet language used for describing the look and formatting of a

document written in HTML.

6. JavaScript: A programming language that enables interactive web pages and dynamic content.

3.4.4 Running the Application

To run the application, execute the app.py file.

The application will be hosted on `http://localhost:5000` in debug mode.

Run : `python app.py`

For production deployment, the Gunicorn server is used.

The application can be hosted using the following command:`web:gunicorn app:app`

4. Conclusion and Future Scope

4.1 Conclusion:

In conclusion, the development of the Yoga Classes Admission Form has been a significant milestone, bringing together a user-friendly frontend and a robust backend. The integration of the frontend has provided an intuitive and engaging experience for users, while the backend, equipped with RESTful API endpoints, ensures seamless data processing, validation, and storage. The documentation serves as a comprehensive guide for developers, offering clear insights into the project's requirements, implementation details, and API functionalities. Through careful consideration of age limits, flexible batch selection, and a two-step payment process, the system has been designed to cater to the diverse needs of potential yoga class participants.

4.2 Future Scope:

1. Looking ahead, future considerations and enhancements could focus on scalability and performance optimization.
2. As user engagement grows, implementing caching strategies, load balancing, and auto-scaling configurations would enhance the system's responsiveness and overall performance.
3. React to be used in the frontend to have a single page application.
4. Additionally, exploring user feedback and analytics can guide further improvements, ensuring that the platform evolves to meet the dynamic expectations of its users.
5. Integration with additional features such as user accounts, personalized dashboards, or real-time notifications could further elevate the user experience, providing a more tailored and interactive journey for participants.