Here's an overview of the schema:

Entities:

1. Person: Represents individuals enrolling for the classes.
2. Payment: Records monthly fee payments.
3. Enrollment: Tracks the batch enrollment of participants for each month.

Attributes:

- Person:

1. PersonID (Primary Key)
2. FirstName
3. LastName
4. Age
5. ContactNumber

Payment:

1. PaymentID (Primary Key)
2. PersonID (Foreign Key referencing Person)
3. PaymentDate
4. Amount (Fixed at 500)

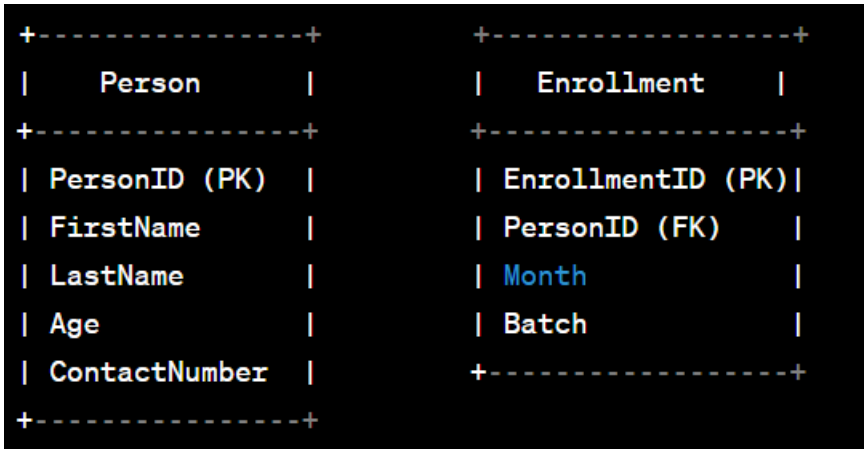
Enrollment:

1. EnrollmentID (Primary Key)
2. PersonID (Foreign Key referencing Person)
3. Month (Month of enrollment)
4. Batch (Batch chosen for the month)

Relationships:

- One person can have multiple payments (one-to-many relationship between Person and Payment).
- One person can have multiple batch enrollments (one-to-many relationship between Person and Enrollment).

ER Diagram



1. Person Table:

PersonID (Primary Key): Unique identifier for each person.

FirstName: First name of the person.

LastName: Last name of the person.

Age: Age of the person.

ContactNumber: Contact number of the person.

2. Enrollment Table:

EnrollmentID (Primary Key): Unique identifier for each enrollment.

PersonID (Foreign Key): Links to the PersonID in the Person table.

Month: The month of the enrollment.

Batch: The selected batch for the enrollment.

Explanation:

The Person table stores information about individuals interested in yoga classes.

The Enrollment table keeps track of enrollments, linking each enrollment to a specific person through the foreign key relationship with PersonID.

Each person can have multiple enrollments, but an enrollment is associated with only one person.

The Enrollment table includes information about the month of enrollment and the selected batch for the yoga class.

Additional Considerations:

In a production environment or a more complex system, you might want to consider additional factors such as:

Indexes: Adding indexes to columns that are frequently queried for performance optimization.

Data Types: Choosing appropriate data types for columns based on the nature of the data.

Normalization: Ensuring the database schema is normalized to minimize redundancy.

Concurrency Control: Implementing mechanisms to handle concurrent data access.

Explanation:

Database Initialization:

The code initializes an SQLite in-memory database using the sqlite3 package.

Table Creation:

The db.serialize block includes SQL statements to create two tables: Person and Enrollment.

Endpoint /api/enroll:

This endpoint is designed to receive enrollment data (first name, last name, age, contact number, and selected batch) from the frontend.

Server-Side Validation:

Basic validation is performed to ensure that all required fields are provided in the request body.

Inserting Data:

Person data is inserted into the Person table, and enrollment data is inserted into the Enrollment table.

Response:

A successful enrollment results in a JSON response with a success message and the assigned personID.

2. Frontend Implementation (React):

For the frontend, a simple form component (e.g., PersonForm.js) can be used to collect user details and send them to the backend.

Explanation:

Form Component:

The PersonForm component renders a simple form with input fields for user details.

State Management:

React useState is used to manage form data.

Event Handlers:

handleChange updates the form data as users type.

handleSubmit sends a POST request to the backend endpoint /api/enroll using Axios.

POST Request:

The form data is sent to the backend using an asynchronous POST request to the /api/enroll endpoint.

Logging and Error Handling:

The console logs the response from the server on successful enrollment.

Errors, if any, are logged to the console.

Notes:

Ensure that your backend server is running (node server.js) before using the frontend form.

Modify the form and backend code according to your application's needs.

Install the necessary packages (axios for the frontend) using npm install axios.

These are simplified examples for demonstration purposes. In a production setting, you'd implement more robust validation, error handling, and potentially user authentication.