Detailed Workflow: Adding Subscriptions

- **User Authentication**:
    - Users must be signed in to access subscription features. Utilize the existing signup/sign-in mechanism to identify the userId.
- **Razorpay Account**:
    - Create a Razorpay account.
    - Set up subscription plans (e.g., monthly, yearly).
    - Generate key_id and key_secret for API integration.
- **Database Setup**:
    - Create a subscriptions table to store subscription details:
        - **Table Structure**:
            - id: Unique identifier for each subscription (Primary Key).
            - subscription_id: Unique ID from Razorpay.
            - user_id: ID of the user (Foreign Key).
            - plan_id: ID of the selected plan.
            - status: Current status of the subscription (e.g., pending, active, cancelled).
            - start_date: Date when the subscription starts.
            - end_date: Date when the subscription ends.

# Workflow Steps

- Frontend: Fetch and Display Plans
- **Route**: /api/subscription/plans
- Fetch available subscription plans from Razorpay using the API.
- Render these plans on the frontend UI, including pricing and descriptions.
- User Selection
- When the user selects a plan, collect the planId.
- Backend: Create Subscription
- **Route**: /api/subscription/create
- **Input**:
    - planId (from Razorpay)
    - userId (from authentication token/session)

- **Process**:
    - Call Razorpay’s subscriptions.create API to create a new subscription.
    - Save the following details to the subscriptions table:
        - subscription_id (from Razorpay)
        - user_id
        - plan_id
        - Set initial status as 'pending'.
- **Frontend**: Razorpay Checkout for Payment

- Once the subscription is created, launch Razorpay Checkout.
- Pass the subscription_id to Razorpay Checkout to handle payment.

- **Post Payment**

- Razorpay returns payment_id, subscription_id, and a signature.
- Send these details to your backend for validation.

- **Backend**: Validate Payment

- **Route**: /api/subscription/validate
- **Validation Process**:
    - Validate the Razorpay signature using the received payment_id and subscription_id.
- **If Valid**:
    - Update the subscription status to 'active' in the database.
    - Save start_date and end_date based on the selected plan.

- **Webhook**: Razorpay Subscription Events

- Razorpay sends events (e.g., subscription.charged) to your webhook.
- **Route**: /api/subscription/webhook
- **Process**:
    - Validate the webhook signature.
    - Update the subscriptions table based on events received (e.g., update status to 'active' or 'cancelled').

- **Frontend**: Display Subscription Status

- Call /api/subscription/status to check the subscription status for the logged-in user.
- **Display Messages Based on Status**:
    - If status is 'active': "You have an active subscription."

    - If status is 'pending': "Payment pending."
    - If status is 'cancelled': "Subscription cancelled."

- **Protect Premium Features**

- Implement middleware to verify user subscription status before accessing premium routes.