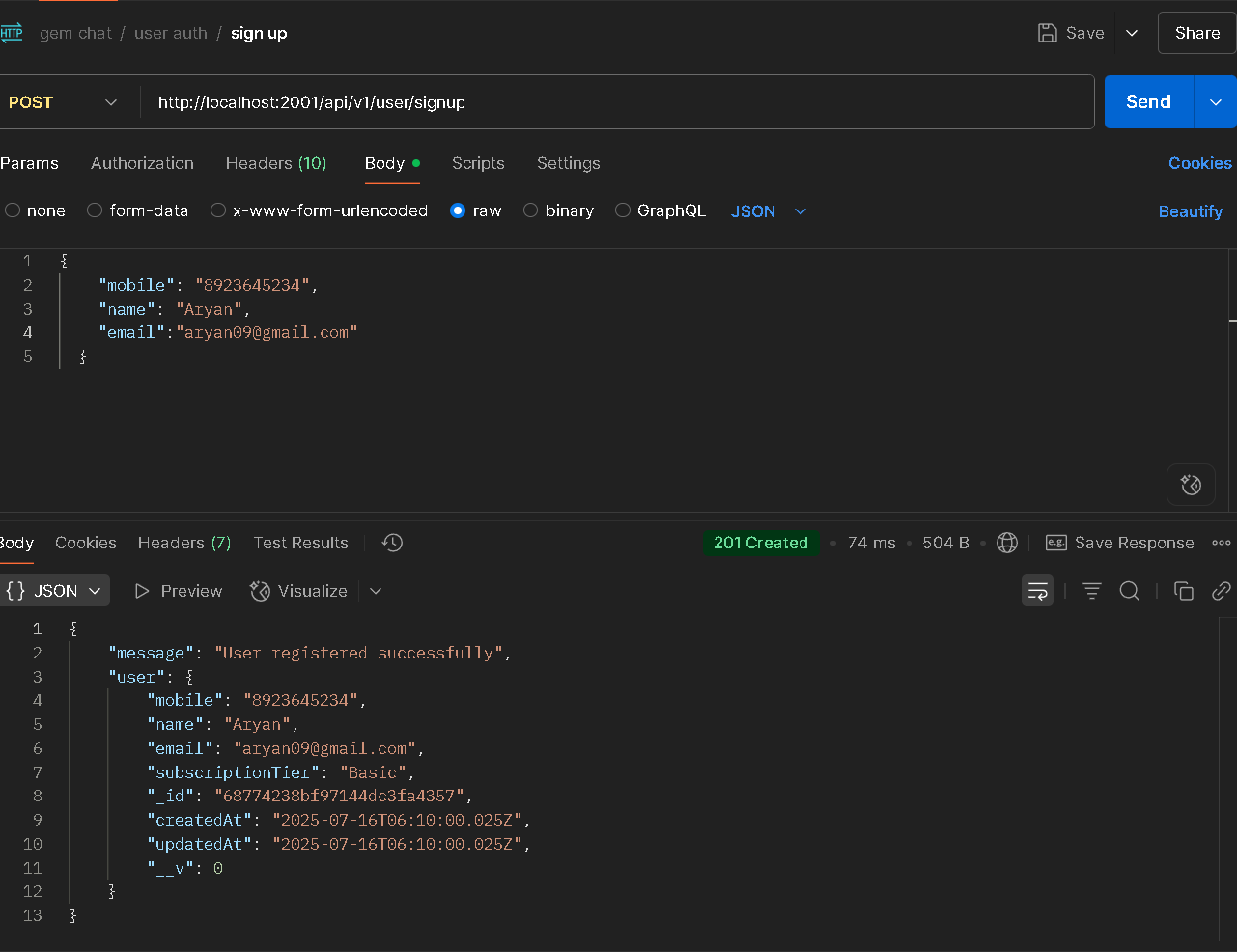
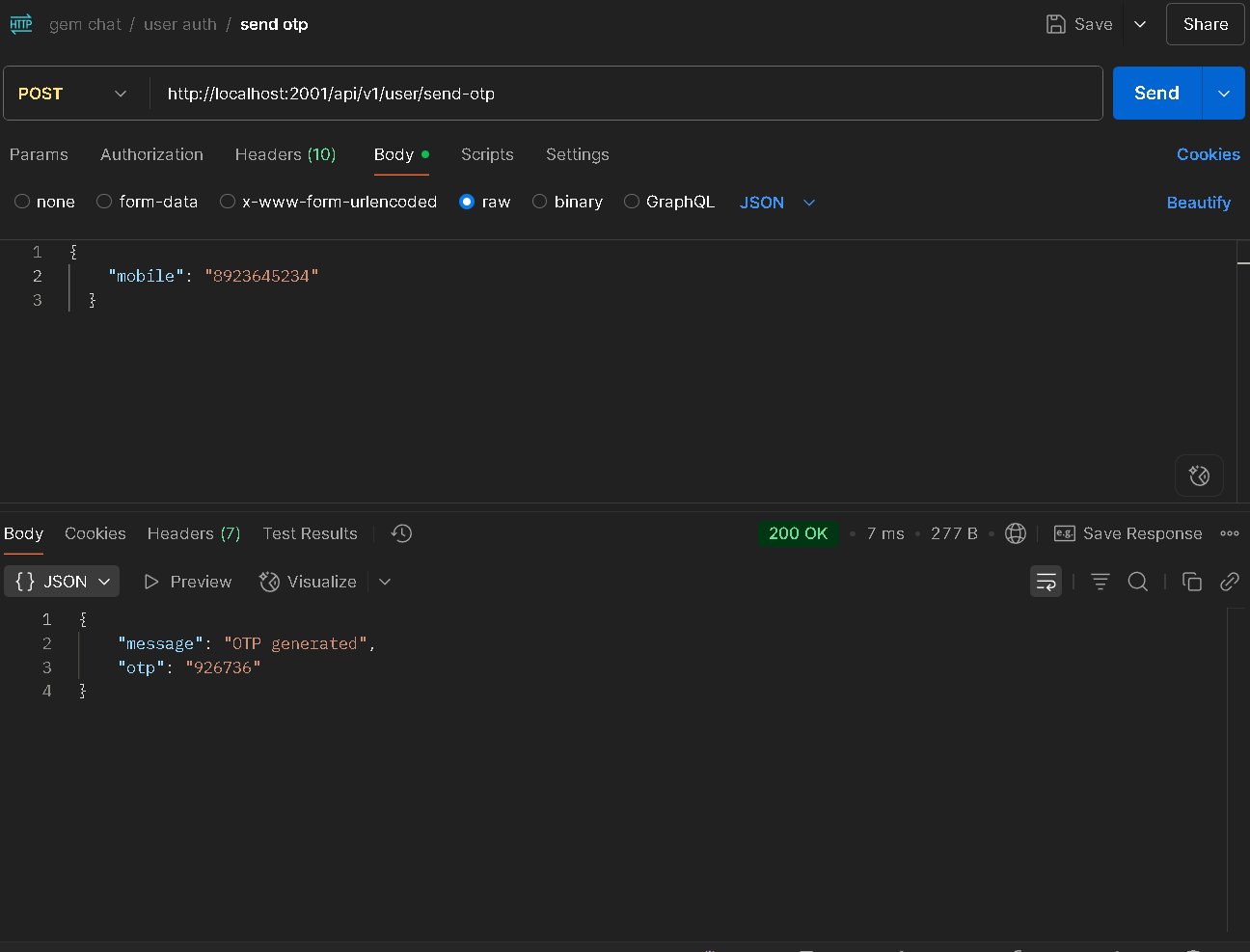
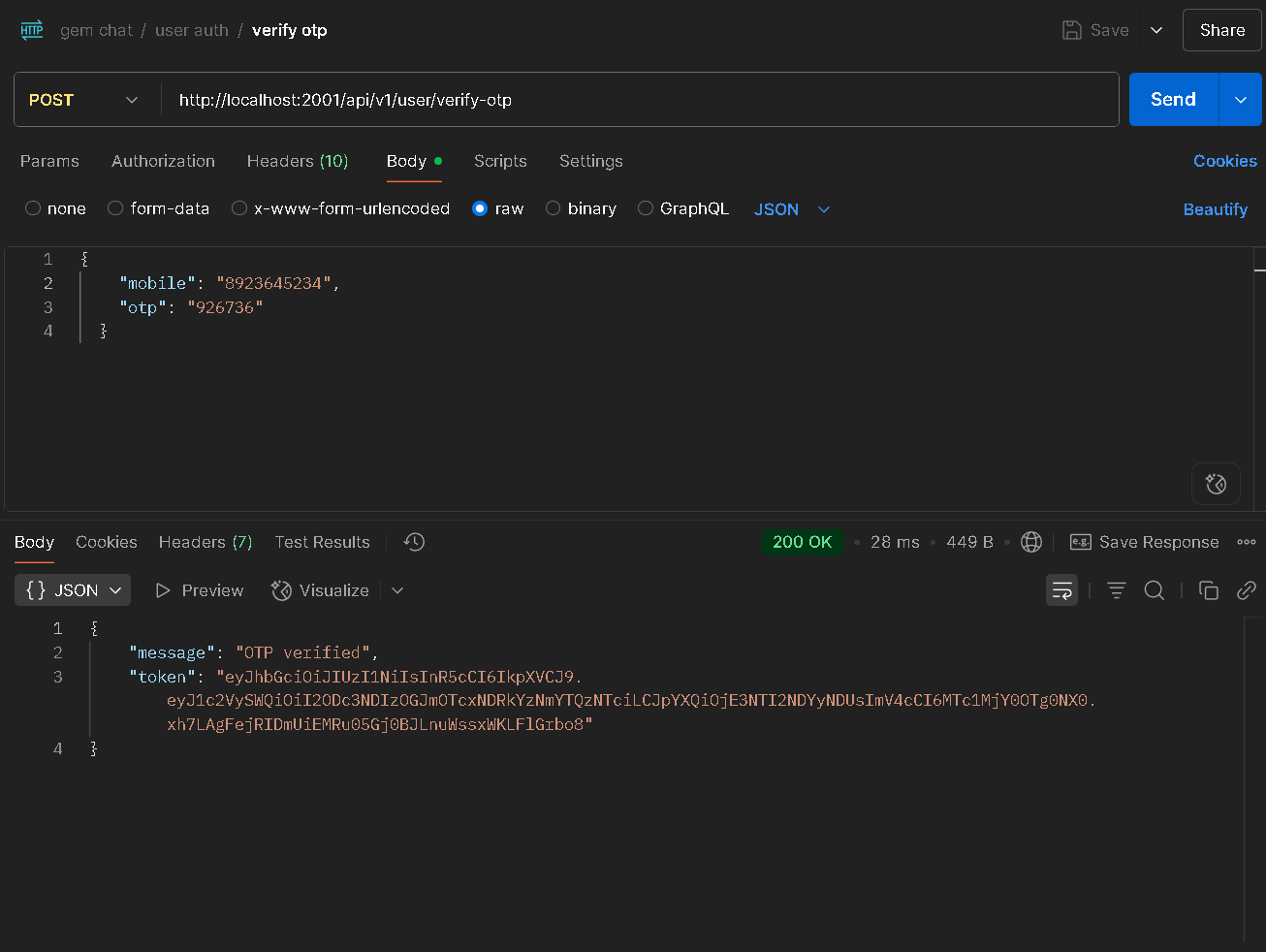
1. SignUp:  
   This signup function handles user registration by checking if a user already exists with the provided mobile number. If not found, it creates a new user with the provided mobile, name, and email, and responds accordingly.



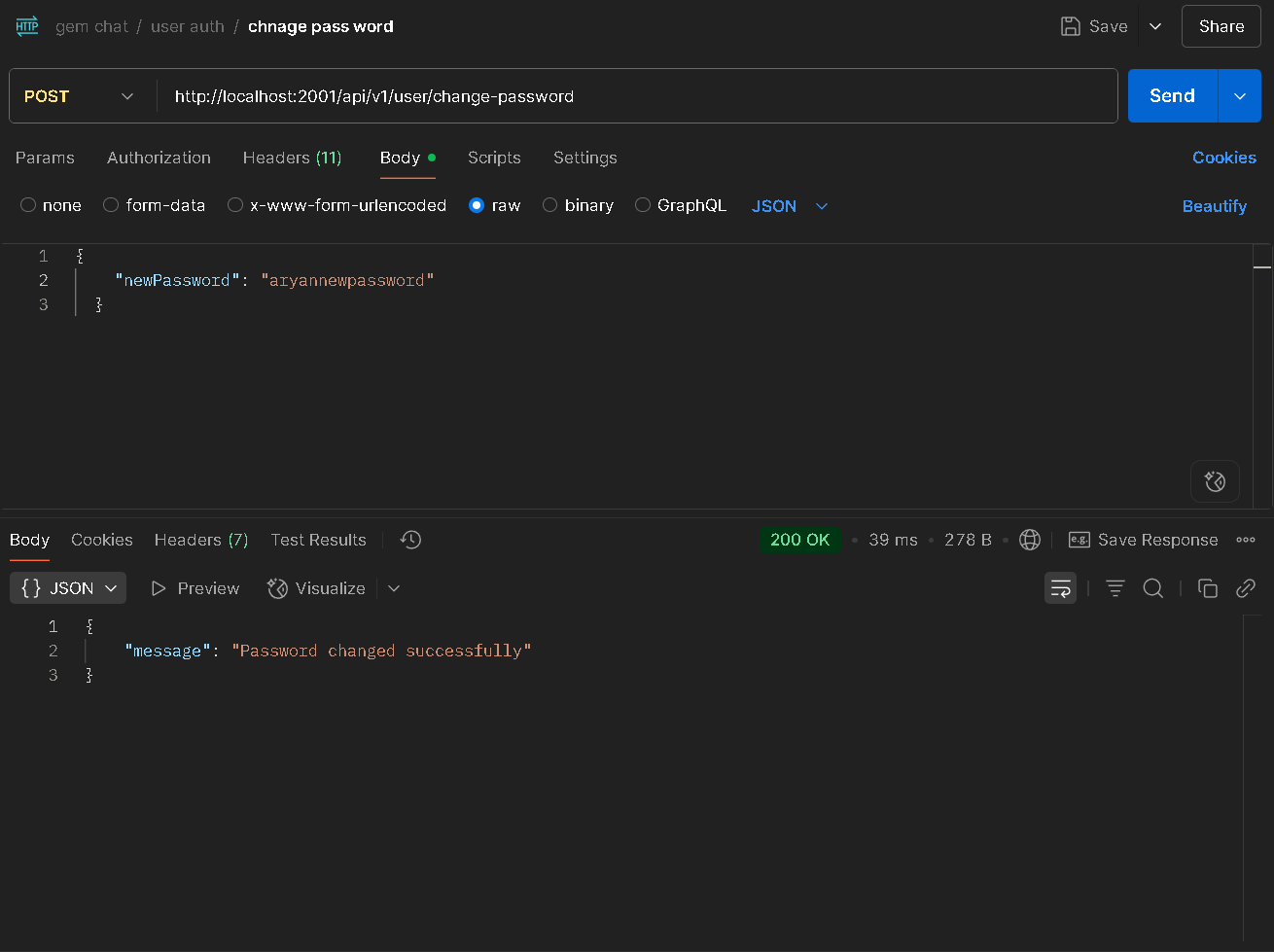
1. Send OTP:  
   The sendOTP function generates a One-Time Password (OTP) for the given mobile number and stores it with a 3-minute expiry. For demo purposes, it returns the generated OTP in the response.

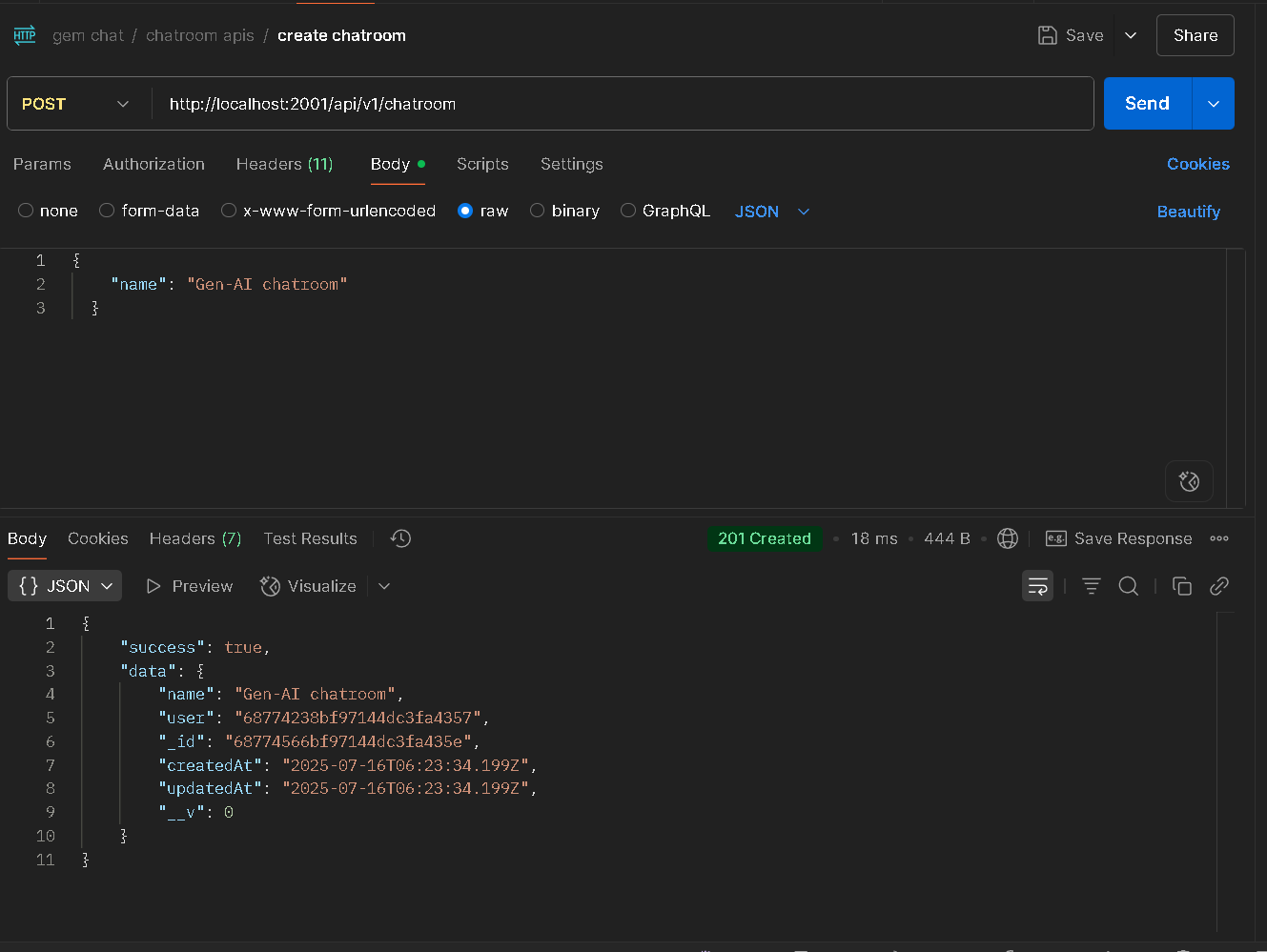
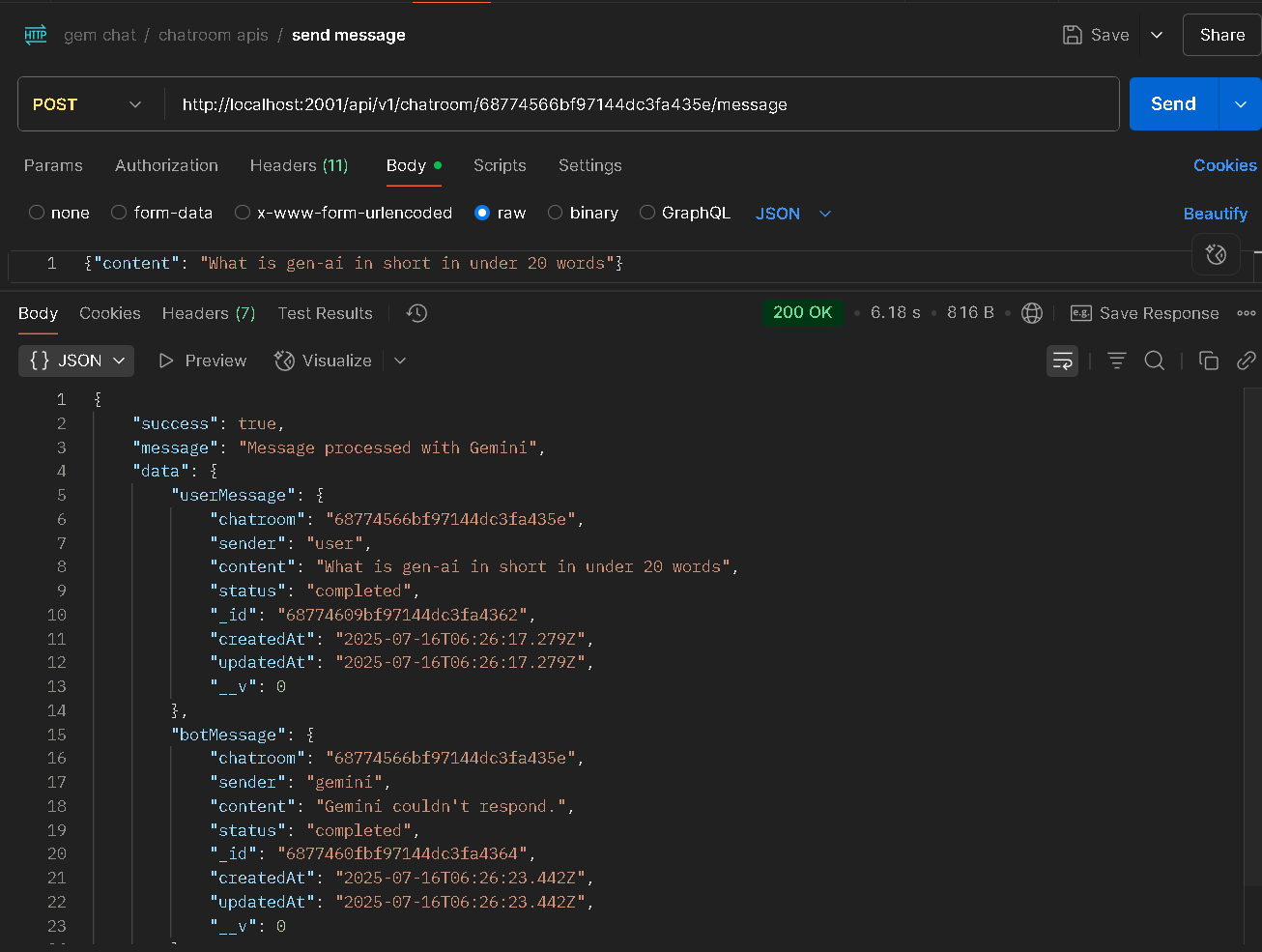
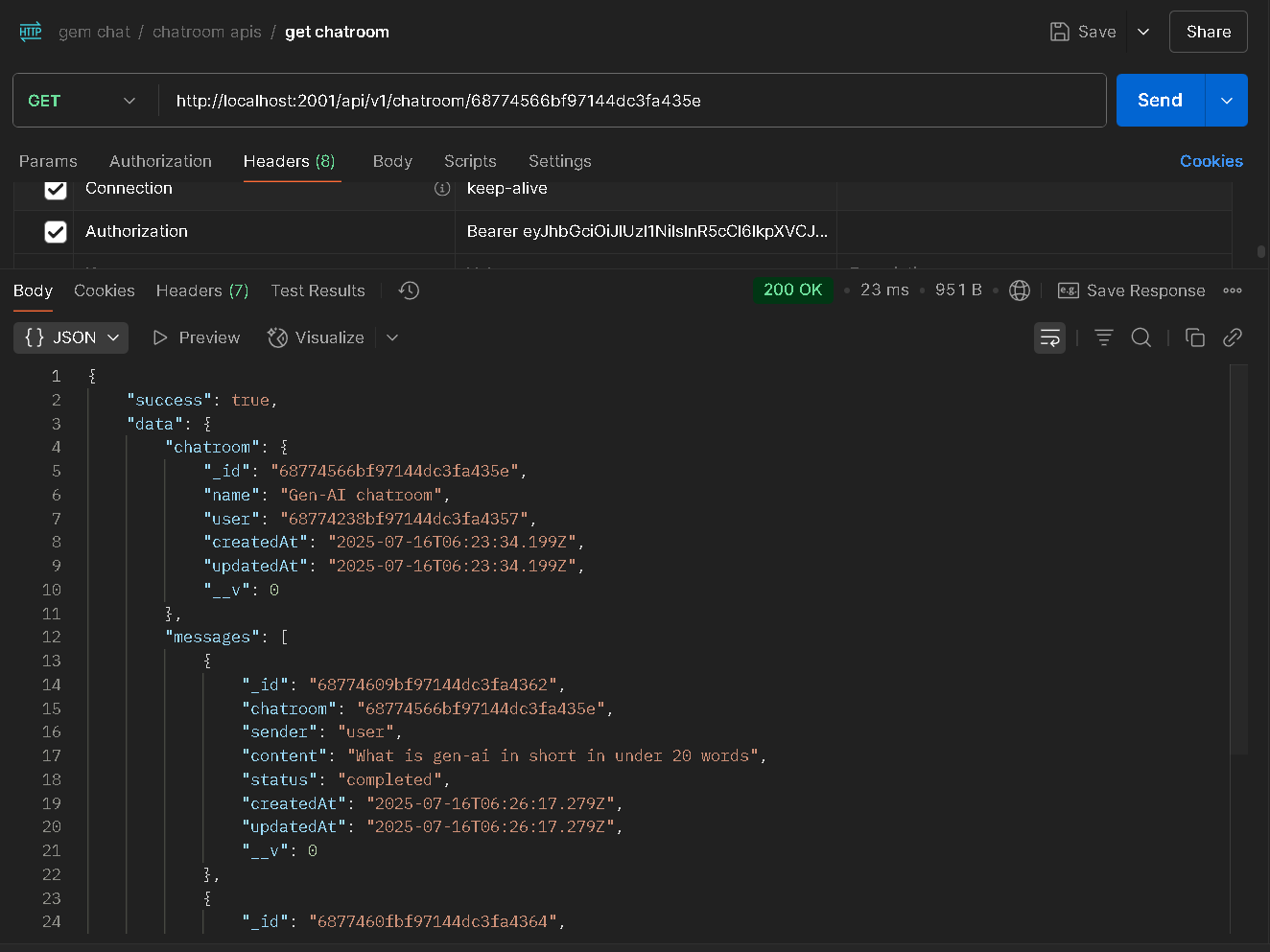
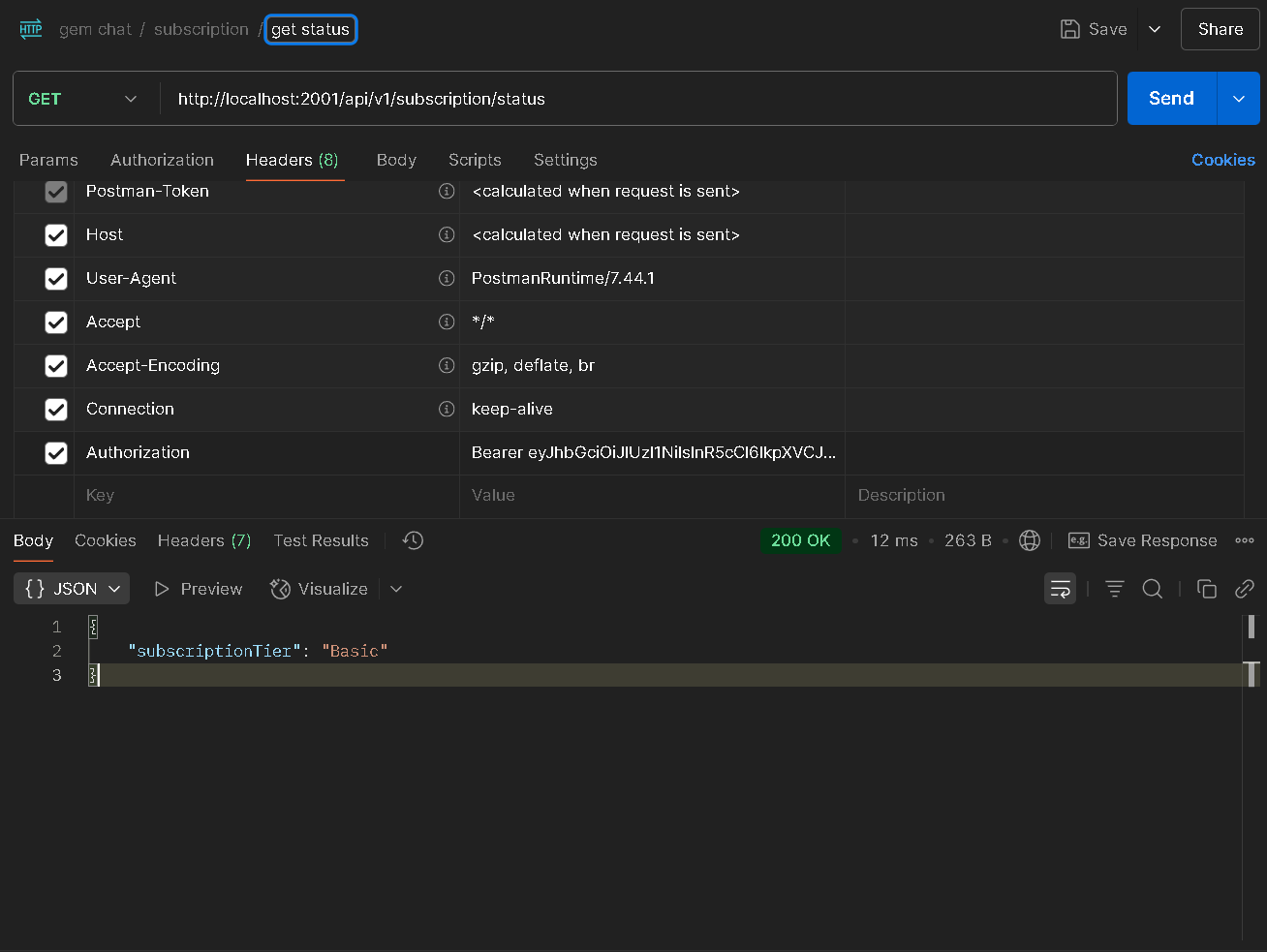
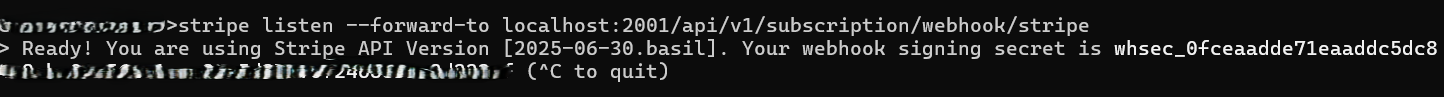
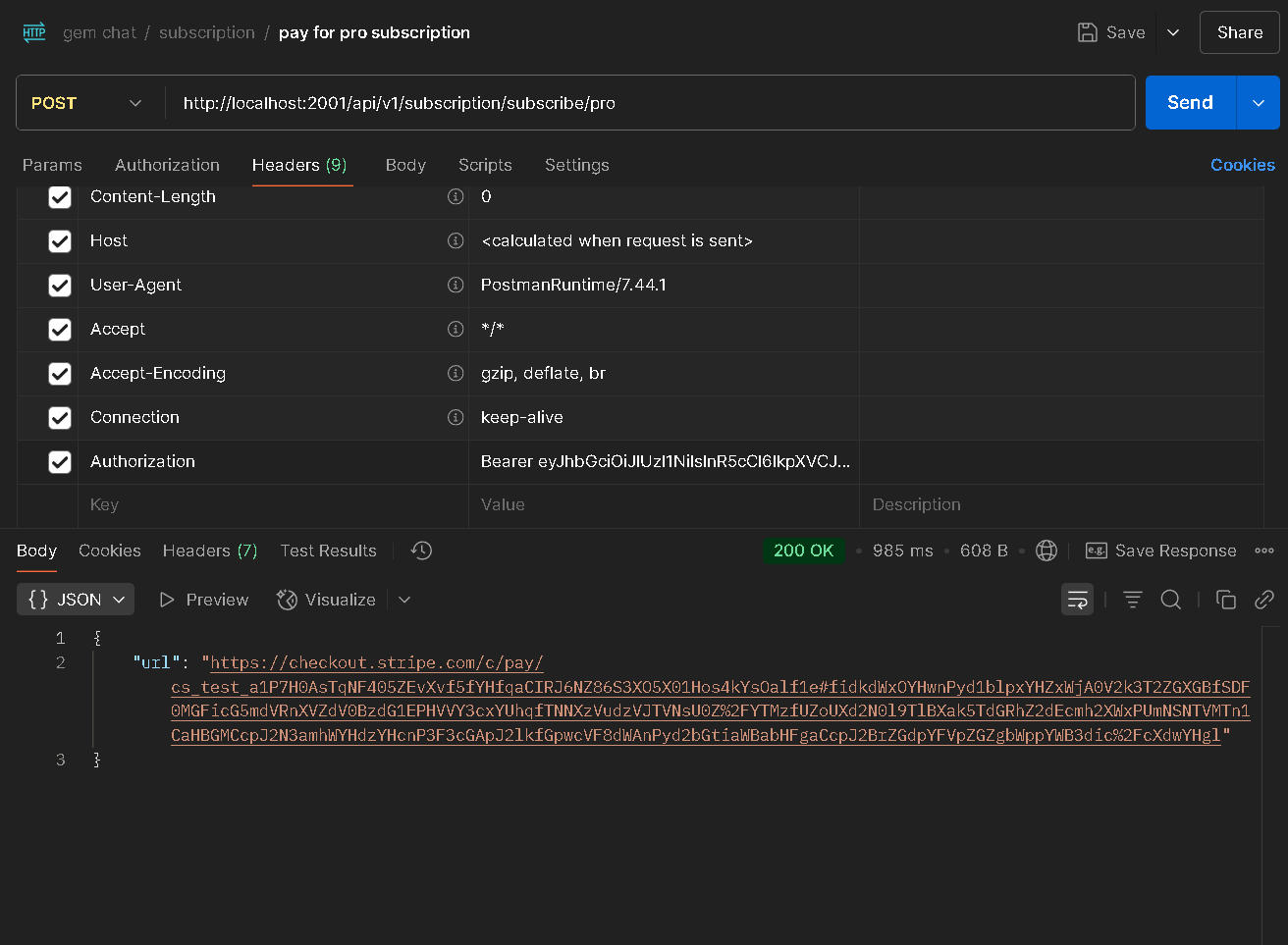


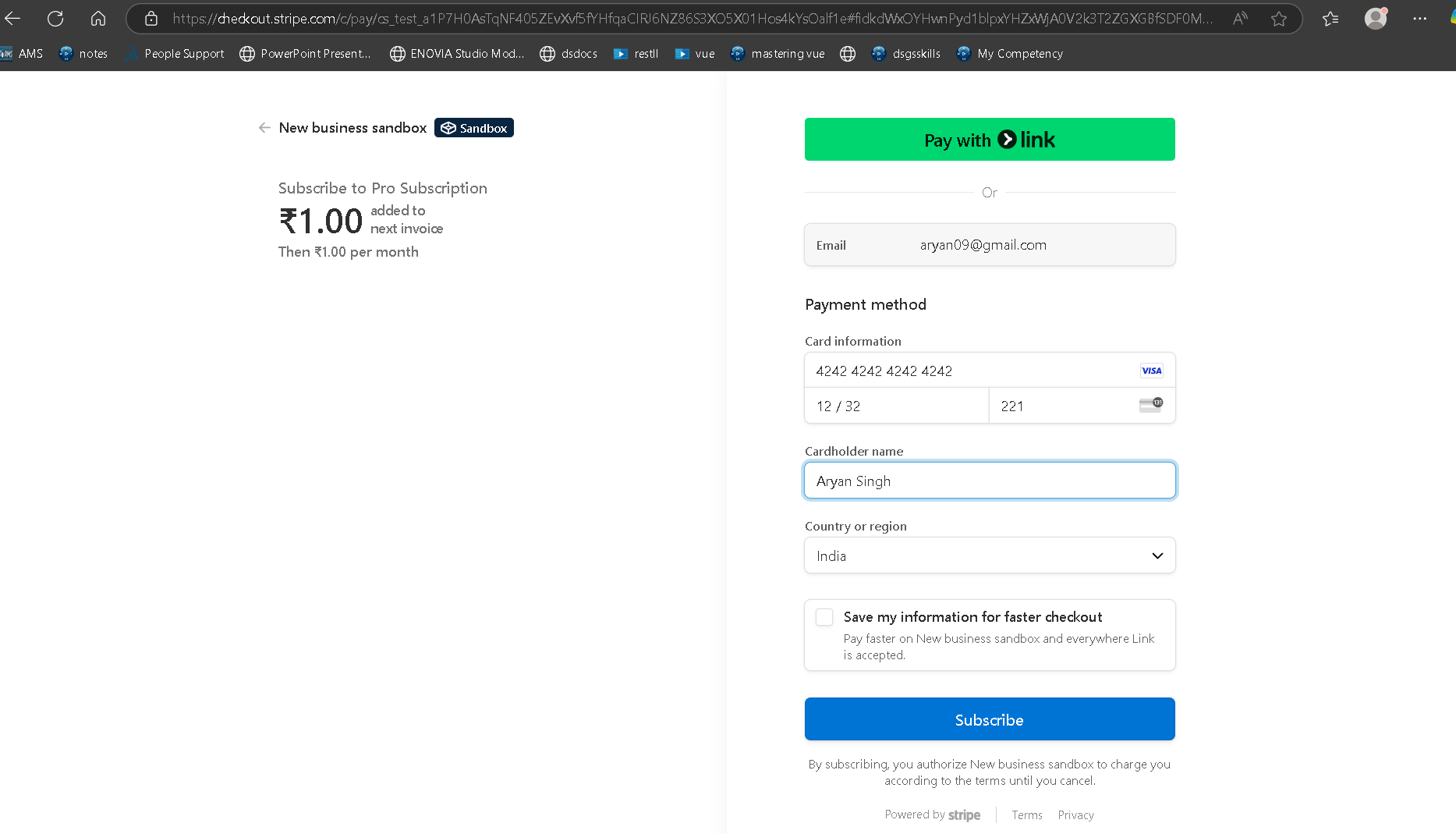
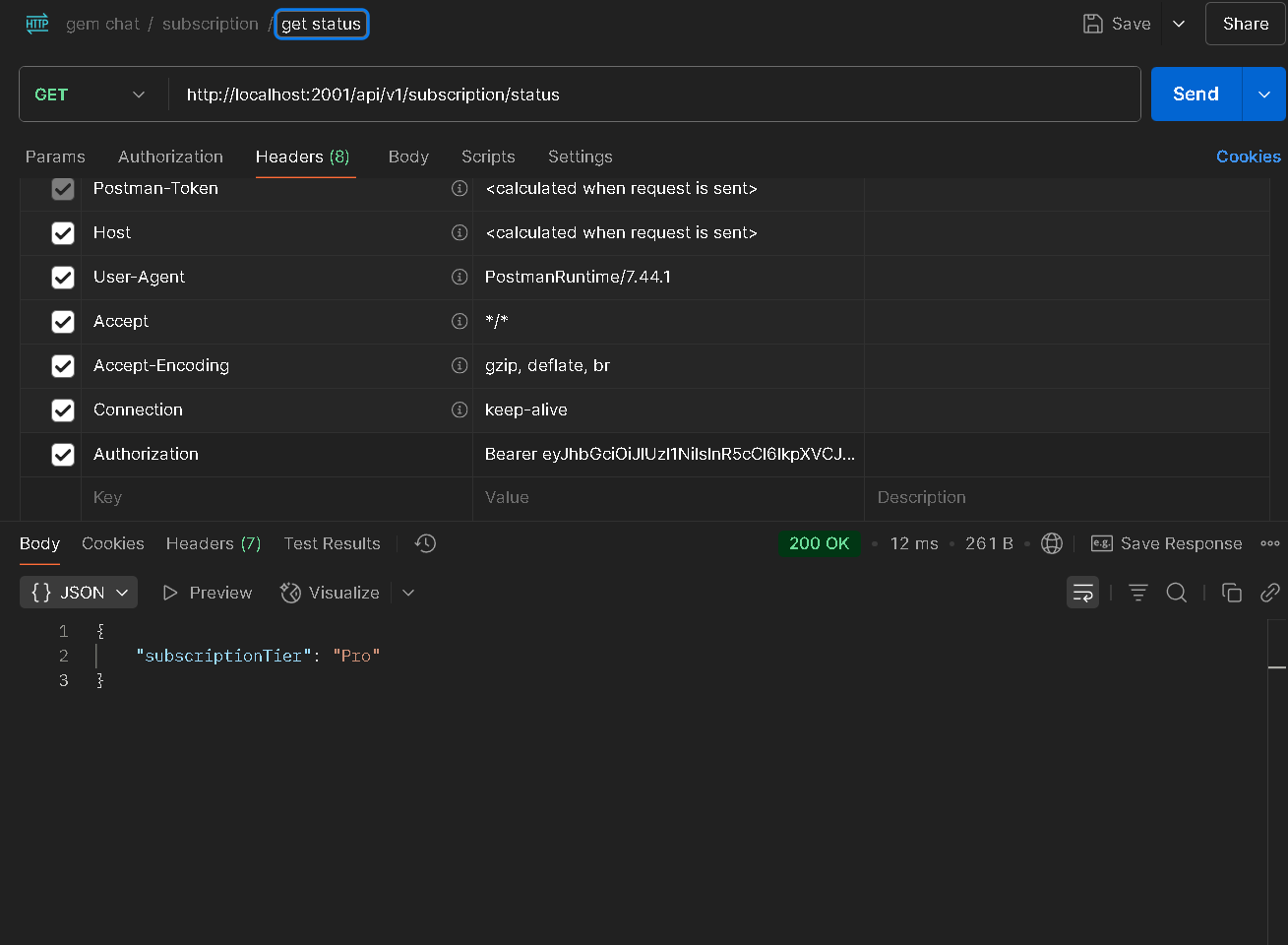
1. Verify OTP:  
   The verifyOTP function validates the OTP for a given mobile number by checking its existence, expiration, and correctness. If valid, it issues a JWT token for the corresponding user and deletes the OTP from the store.



1. Change Password:  
   This route handler allows authenticated users to change their password by providing a new one. It validates input, fetches the user from the database, updates the password, and saves the changes securely.



1. Create chatroom:  
   Creates a new chatroom for the authenticated user using the provided name. Stores the user ID as the owner and returns the created chatroom.  
     
   
2. Send message in chatroom:  
   This API sends a user's message to a specific chatroom and forwards it to Gemini AI for a response. Both the user's message and Gemini's reply are saved and returned in the response.  
     
   
3. Get chat room:  
   This API retrieves a specific chatroom (by ID) belonging to the authenticated user. It returns the chatroom details along with its complete message history.  
     
   
4. Get status of the subscription:  
   This API allows the authenticated user to check their current subscription tier. It queries the user from the database and returns either "Pro" or defaults to "Basic" if not set.  
     
   
5. Start a local Stripe webhook listener:  
   Starts a local Stripe webhook listener that forwards real Stripe events to your local webhook endpoint. Stripe then provides a **signing secret** used to verify these incoming webhook requests are authentic. You add this secret to your environment variables so your backend can securely validate events from Stripe before processing them.  
     
   
6. Subscribe for Premium subscription:  
   a. It creates a Stripe checkout session and returns a payment link for the Pro subscription.  
     
   

b. After completing the payment by entering your card details on Stripe’s hosted page, your subscription is activated.  
  
  
  
c. When you later check your subscription status, it reflects as "Pro" because your user record is updated upon successful payment via the Stripe webhook.  
  
This process ensures secure handling of payment data, as Stripe manages all sensitive information.  
Your backend listens for Stripe’s webhook events to automatically update user subscriptions without requiring manual intervention.