Project Description

QuickGPT is a MERN stack based AI chatbot web application inspired by ChatGPT. It allows users to chat with an AI model, generate images, and explore community-shared posts. The system includes a credit-based model in which users must purchase credits to use AI features. The project also provides stripe-based payment integration, JWT authentication and clean, responsive frontend UI.

Project Objective

- To develop a functional AI chatbot, using MERN stack
- To allow users to generate text and images using AI
- To implement a credit-based usage system
- To integrate secure payments using stripe
- To provide a clean and responsive UI using modern web technologies

Technology Stack

Frontend

- React.js
- Tailwind CSS

Backend

- Node.js
- Express.js
- MongoDB + mongoose

Authentication

- JSON web token

Payments

- Stripe API

AI Features

- Google Gemini API

Third-party API

- Official Joke API

Links

- **GitHub repository link**
  - https://github.com/AP-anjali/QuickGPT_CC_PROJECT
- **Deployed website link**
  - **Frontend :** https://202412063-quick-gpt.vercel.app/
  - **Backend :** https://quick-gpt-server-plum.vercel.app/

Contribution

Student ID	Name	Contribution
202412063	Anjali Patel	Entire Backend and testing
202412065	Dhara Patel	Entire Frontend
202412088	Hirwa Saraiya	Payment integration
202412121	Viraj Sharma	Third-party integration

System Architecture [as a development view]

React Client → Express Server → MongoDB
↑ ↓
Stripe Checkout ← Webhooks

➢ Flow Summary :

- User interacts with frontend
- API calls sent to backend
- Backend verifies token and then processes request if valid tocken
- AI answer generation and credits deducted
- Payments handled via stripe checkout
- MongoDB stores users, chats, messages, credits & transactions

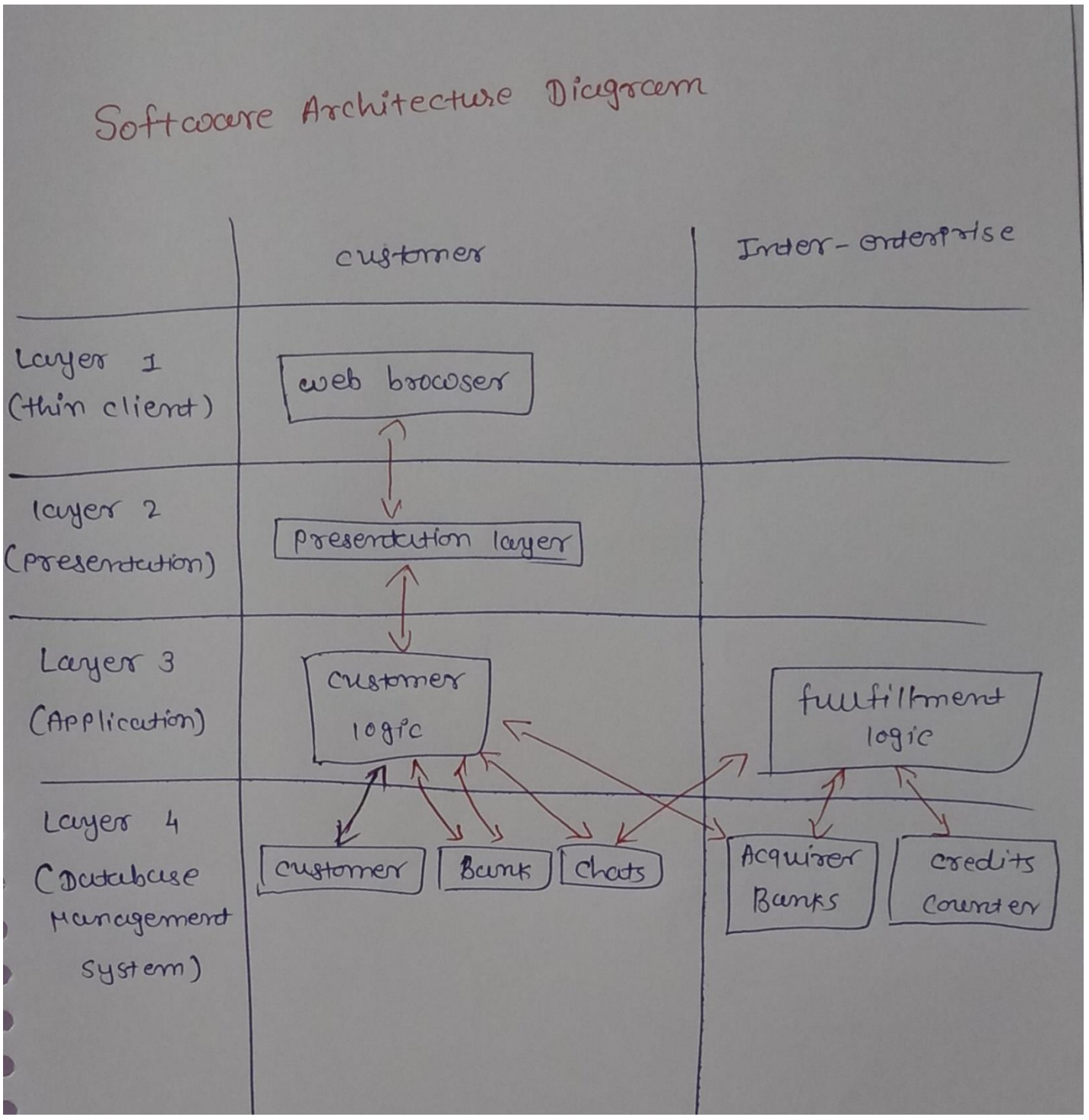
Architecture Diagram [as a deployment view]

- **Step 1 : define the business problem, that needs to be solved**
  - QuickGPT must allow end users to interact with an AI chat/image-generation service and a community posts area while enforcing a credit-based pay-to-use model, providing a secure, responsive UI and safe, scalable backend handling of AI API calls, payments(stripe), authentication(JWT), and community content.
  - Primary success criteria :
    - Low-latency chat and image generation for paying users
    - Secure payments and credit accounting (no free bypass)
    - Clear separation of real-time request handling VS background work
    - Observability and fault-tolerance

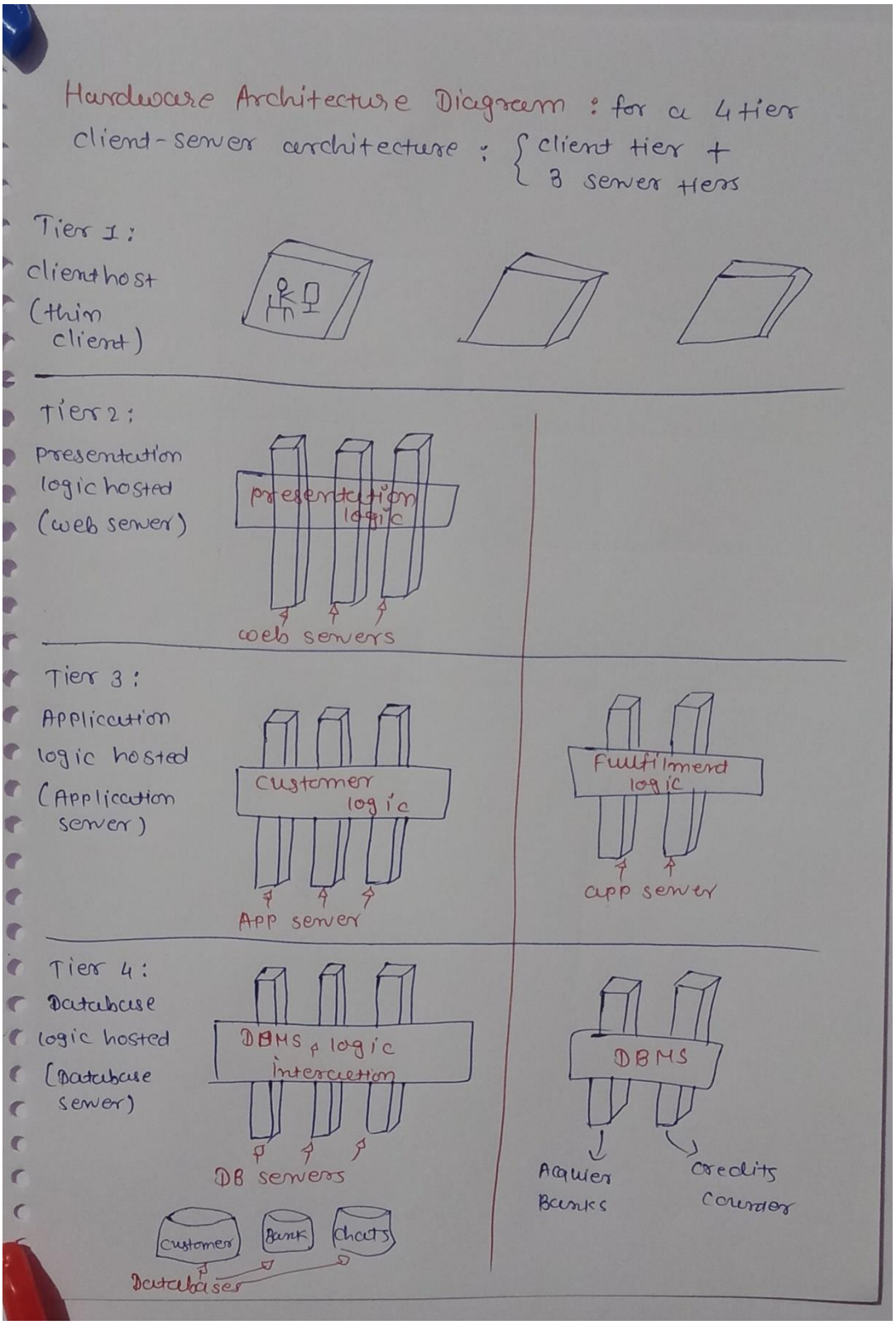
- **Step 2 : define the necessary business process, that needs to be automated by the software application**
  - Sub-function 1 : Interact with the customer (presentation) : user interface
  - Sub-function 2 : Provide application logic to automate business process
    - Customer logic
    - Fulfillment logic
  - Sub-function 3 : Share and manipulate data
    - Customer information
    - Chat data

- **Step 3 : translate the business process into the requirements (functions and features) for the software**
  - **Functional :**
    - JWT authentication
    - Stripe integration
    - Credit accounting microservice
    - DB storage
  - **Non-Functional :**
    - Horizontal scalability, auto scaling
    - High availability for critical services

- **Step 4 : define the software architecture diagram : we have the following sub-functions (from step 2)**
  - Layer 1 : **thin client** : enable the graphical user interface for the client on client's customer
  - Layer 2 : **to realize sub-function 1 :** present the necessary information (chat and published images, ...), create the GUI for the client, and enable the visualization, data entry, communication with the application and output of results
  - Layer 3 : **to realize sub-function 2 :** perform the application logic
    - Customer logic and fulfillment logic
  - Layer 4 : **to realize sub-function 3 :** manipulate and share data


Softcoare Architecture Diagram
customer
Inter-enterprise
Layer 1
(thin client)
web brocoser
layer 2
(presentation)
presentation layer
Layer 3
(Application)
customer
logic
fullfillment
logic
Layer 4
(Database
Management
System)
customer
Bank
Chats
Acquirer
Banks
credits
Counter

➢ Step 5 : define the hardware architecture diagram



Main Functionality

➢ User authentication :

- User can register with name, email, password
- Password is hashed using bcrypt
- JWT token generated during login
- Authenticated routes are protected using middleware

➢ Chat system :

- Users can create chats
- Chats include messages between user and AI
- Chat name can be edited
- Chat can be deleted anytime
- Messages are stored in mongoDB

➢ AI Text generation :

- Users can send prompts
- System calls AI API
- Returns response
- 1 credit deducted for each request

➢ AI Image generation :

- Users can generate images using AI
- 2 credits per image generation
- Generated images can be published in community gallery

- **Credit purchase system :**
  - Users can buy credits via stripe
  - Plans include predefined credit amounts
  - Stripe checkout session is created and user is redirected
  - Successful payments are confirmed via webhook
  - Corresponding credits added into user's account

- **Community Page :**
  - All users can view AI-generated community images
  - Each post shows image + user details + prompt

- **Third-Party API integration for random jokes :**
  - A random joke is fetched from Official joke API
  - Showcased in demo pages

API Documentation [overview]

- **Auth**

Method	Endpoint	Description
POST	/api/users/register	Register user
POST	/api/users/login	Login user
GET	/api/users/me	Get logged in user

- **Chat**

Method	Endpoint	Description
GET	/api/chats	List all chats
POST	/api/chats	Create chat
GET	/api/chats/:id	Get chat
PUT	/api/chats/:id	Update chat
DELETE	/api/chats/:id	Delete chat

- **AI answer generation**

Method	Endpoint	Description
POST	/api/messages/text	Text generation
POST	/api/messages/image	Image generation

- **Credits**

Method	Endpoint	Description
GET	/api/credits/plans	Plans list
POST	/api/credits/purchase	Stripe session
GET	/api/ credits/me	User credits
GET	/api/ credits/transactions	Transactions
PATCH	/api/ credits/transactions/:id	Update

- **Stripe**

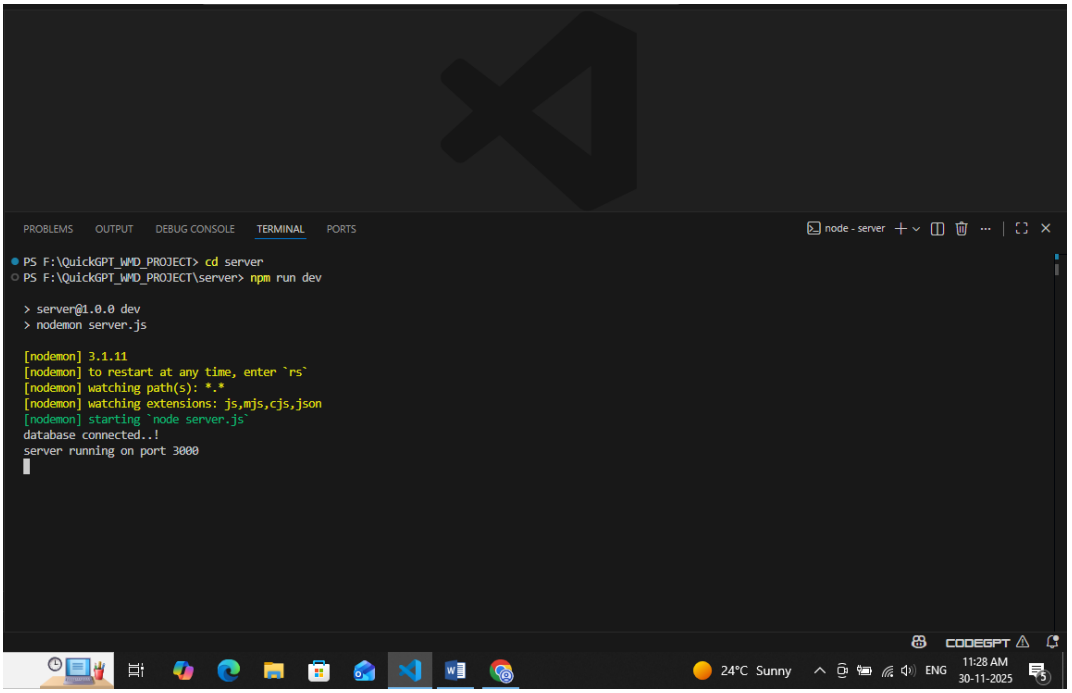
Method	Endpoint	Description
POST	/api/stripe/webhook	Raw webhook endpoint

- **Third-Party API**

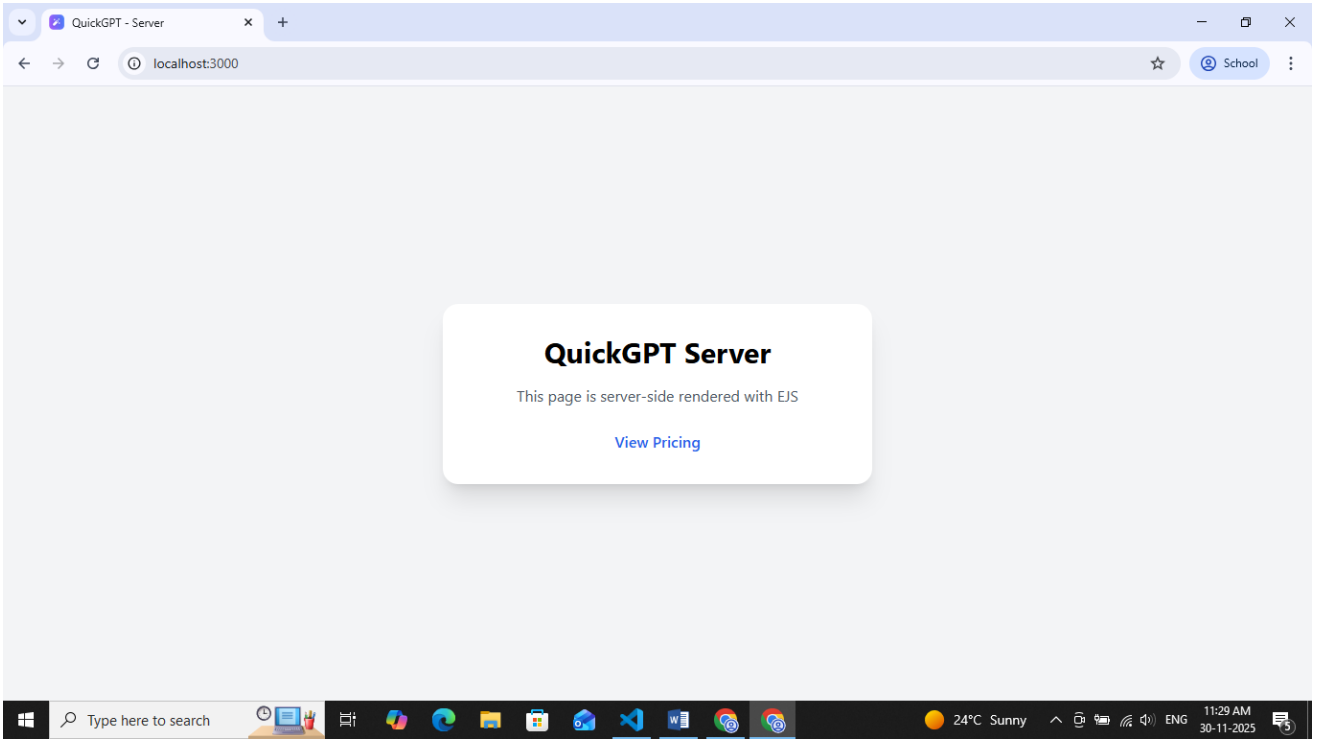
Method	Endpoint	Description
GET	/api/jokes/random	Generate random joke

System Snapshots

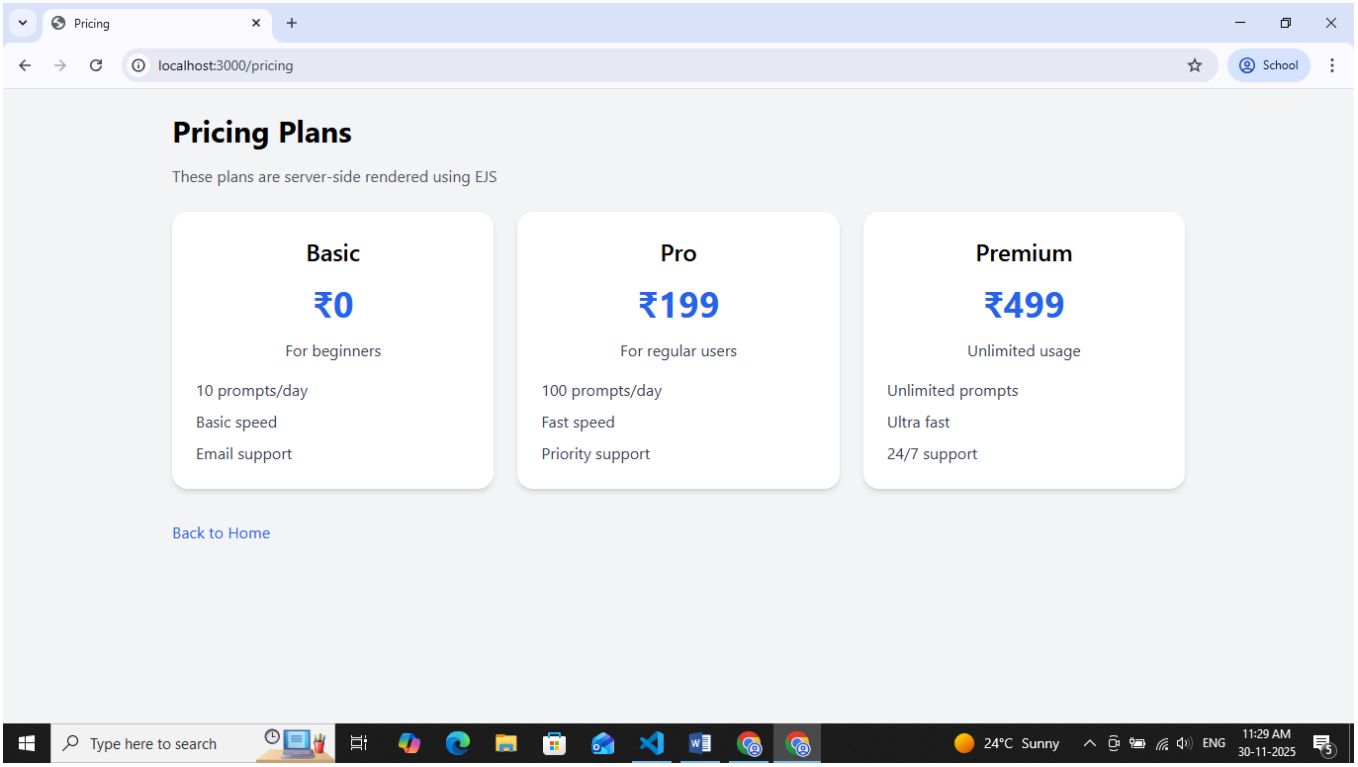
[01] running backend



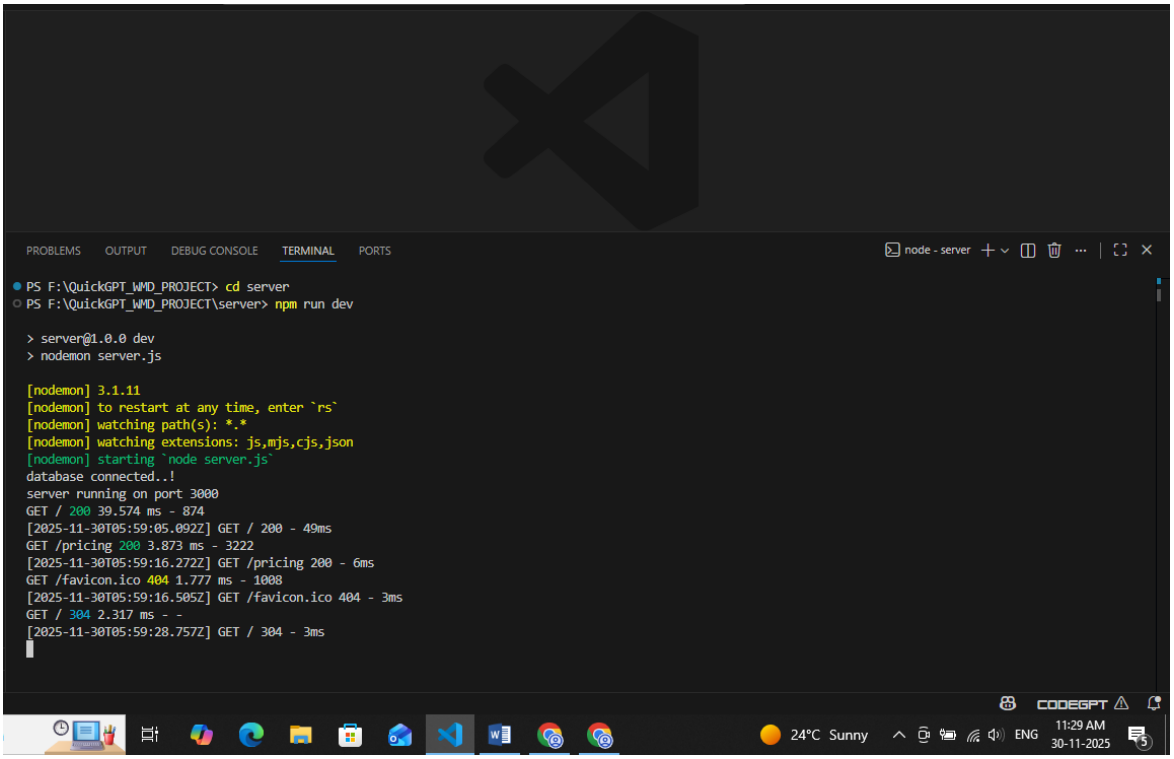
[02] server-side rendering using EJS template engine



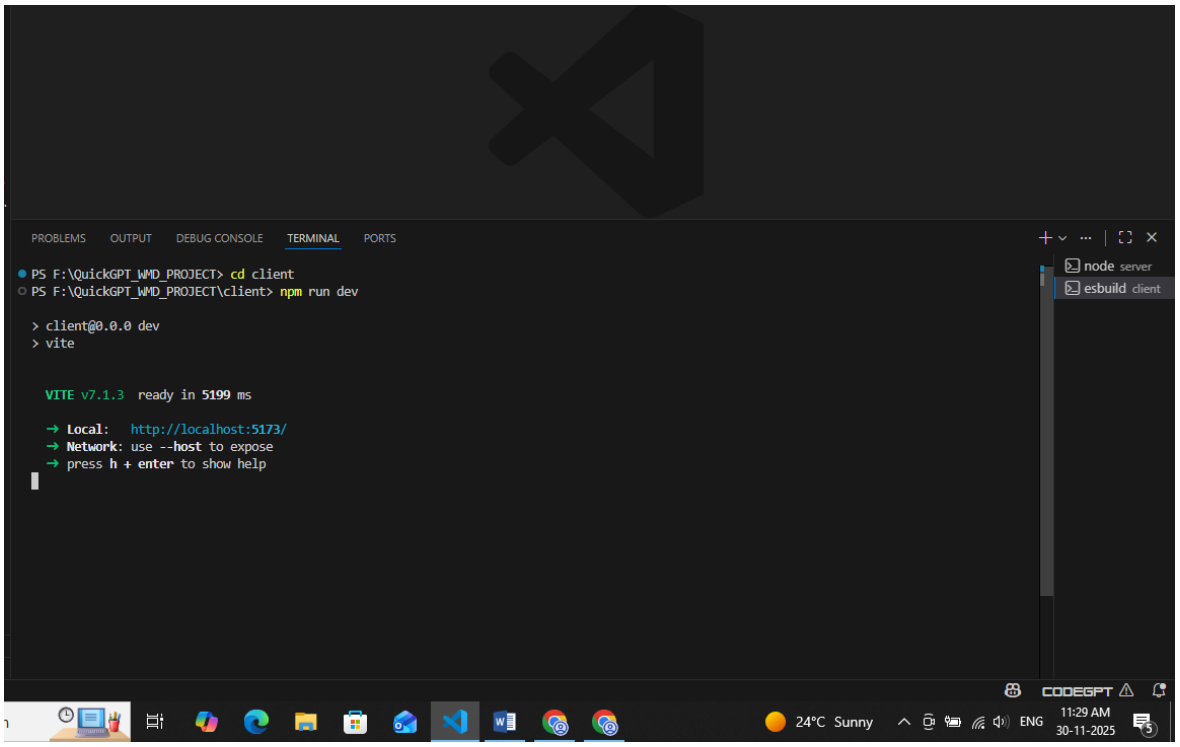
[03] server-side rendering pricing page



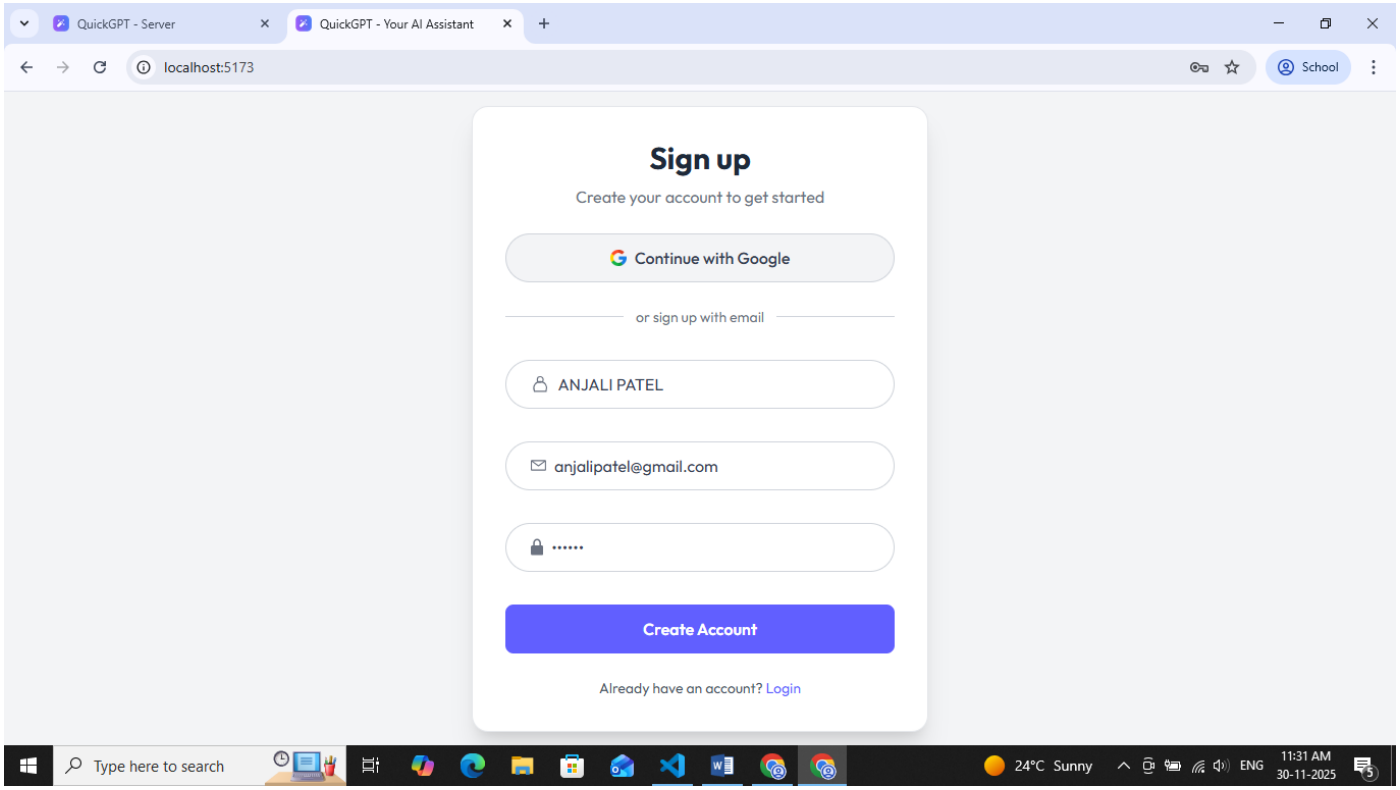
[04] logs



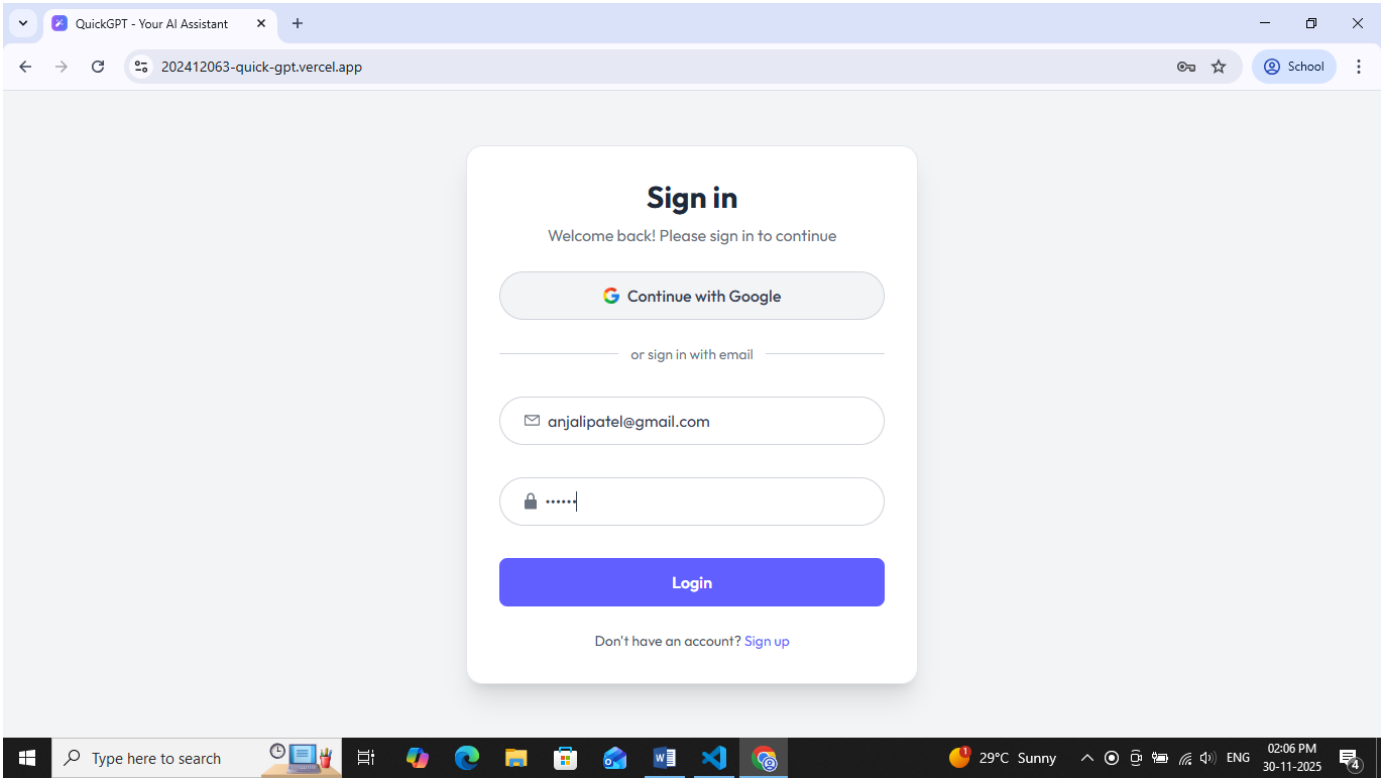
[05] running frontend



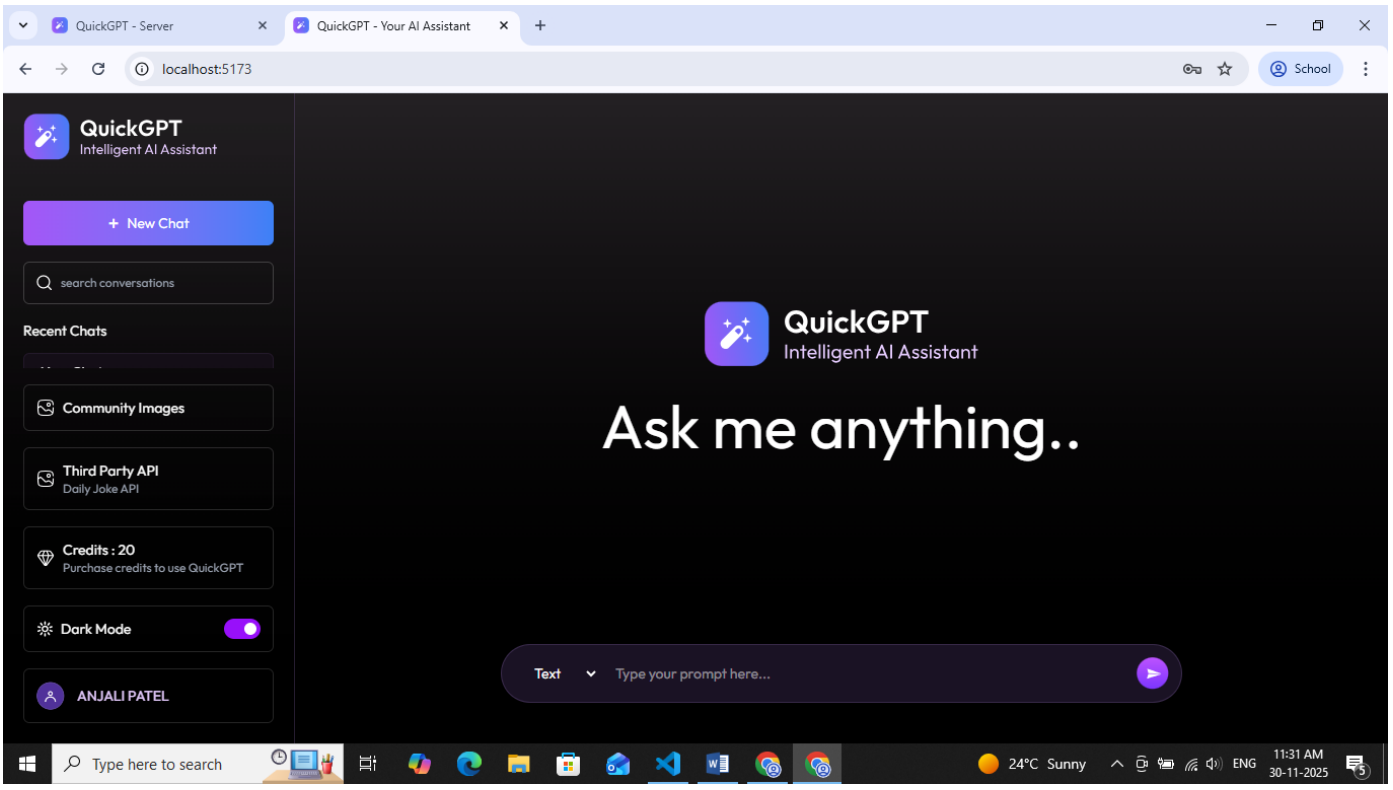
[06] registration page



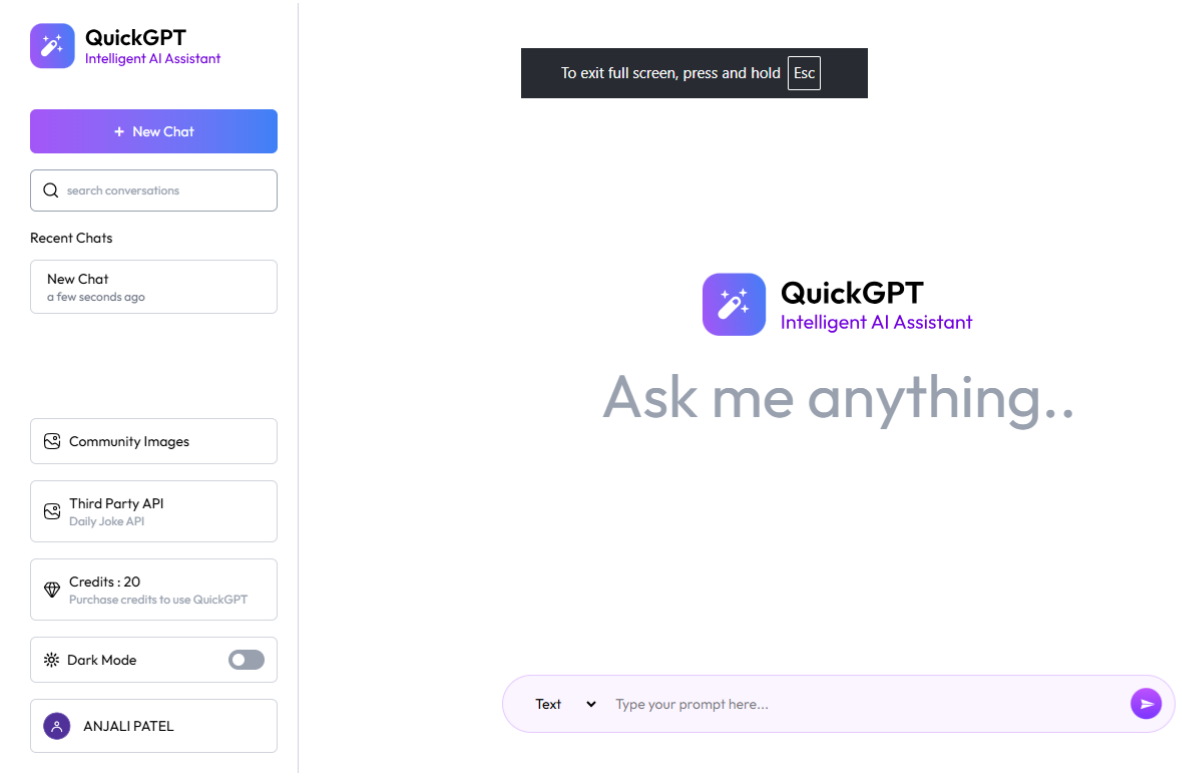
[07] login page



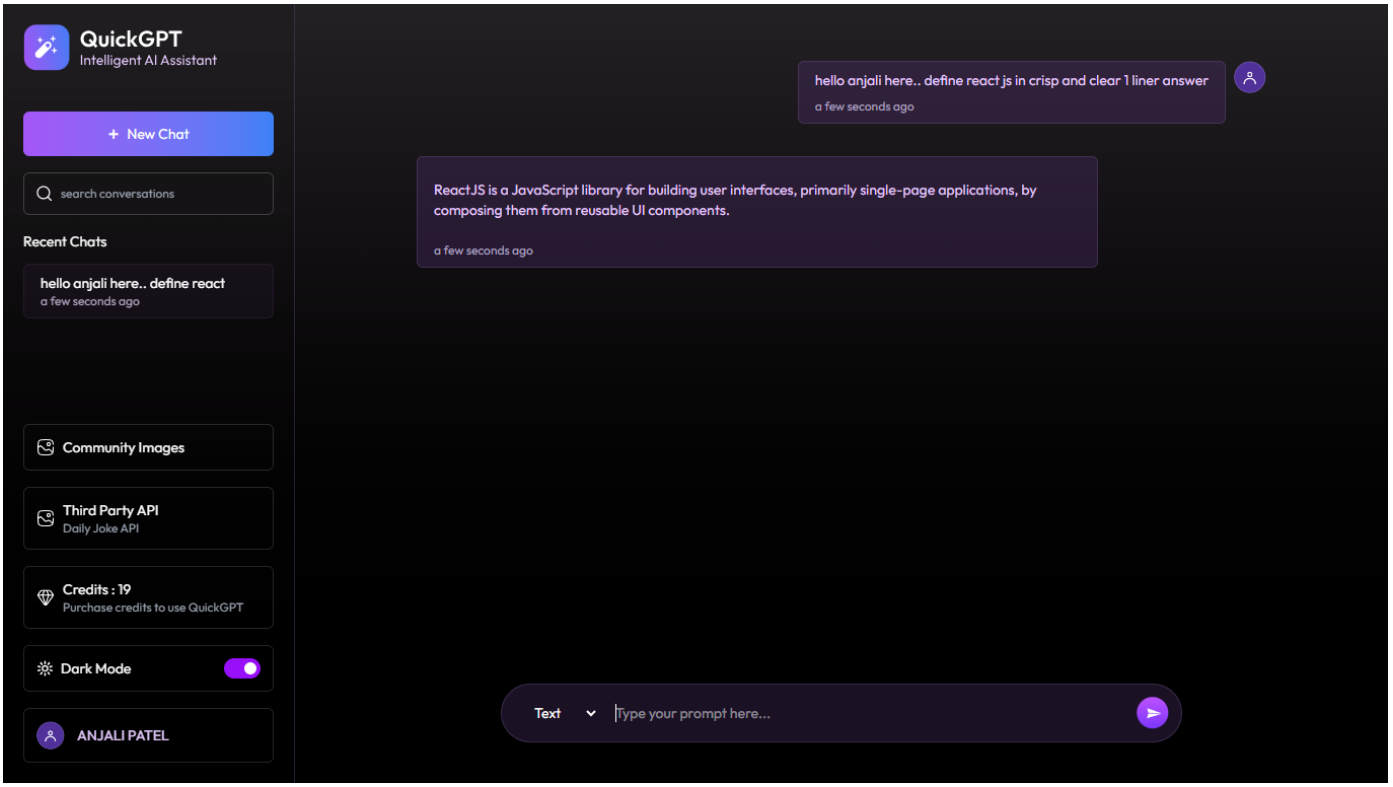
[08] home page



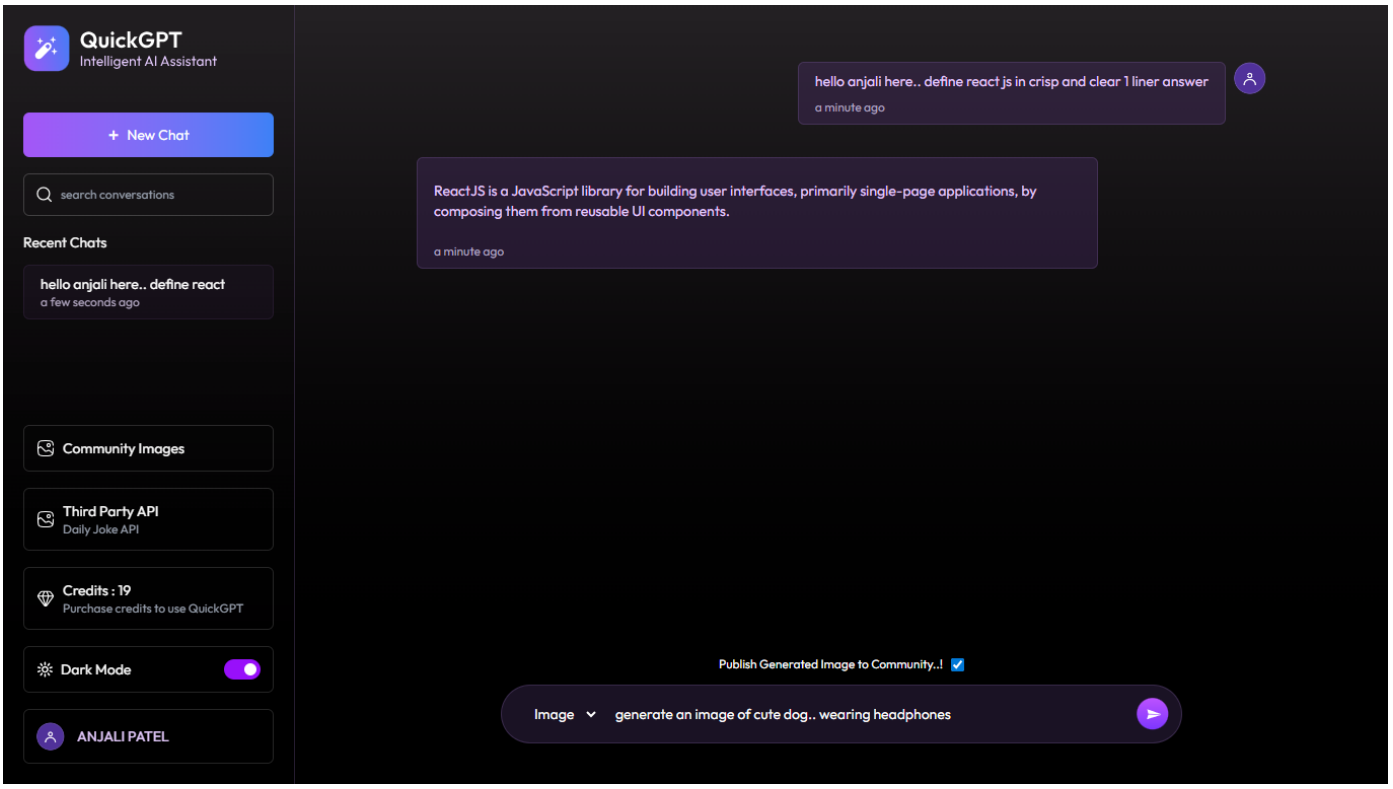
[09] home page in light mode



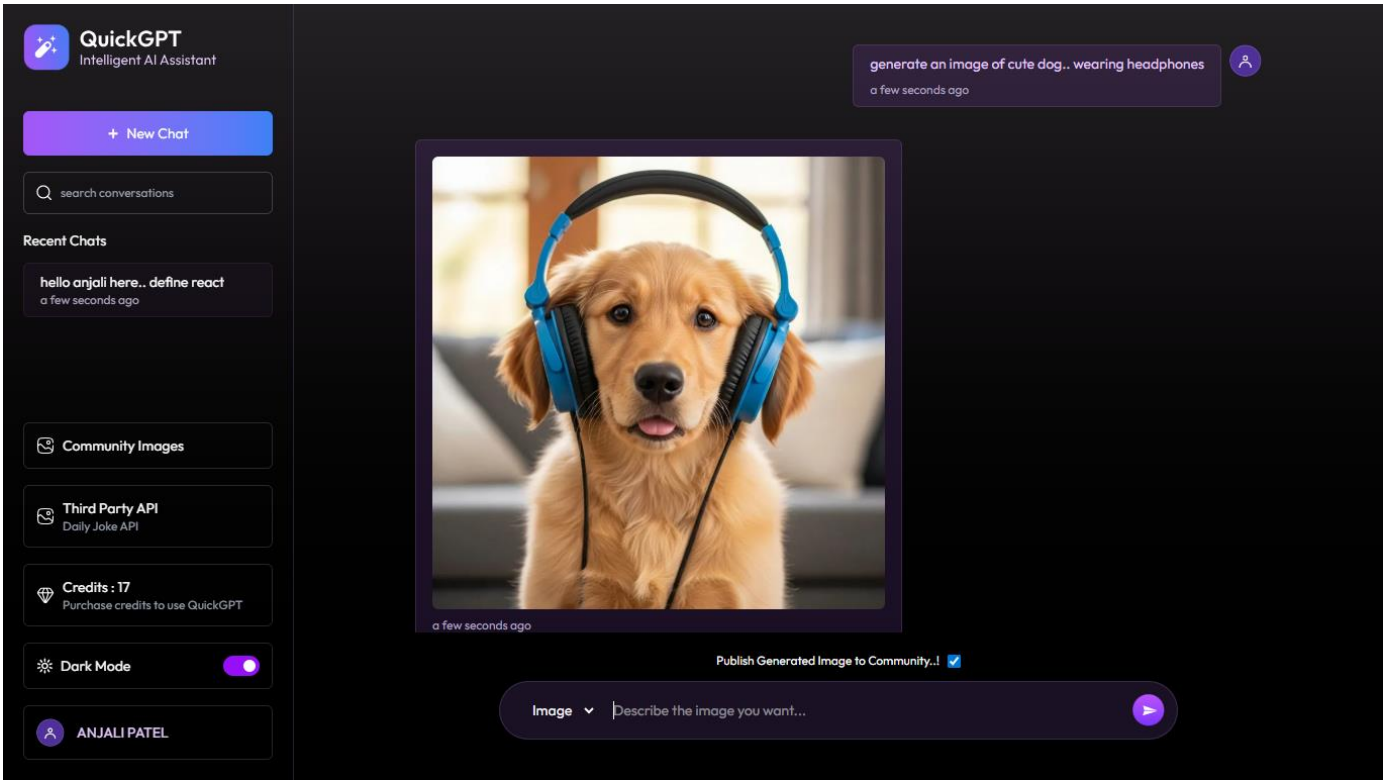
[10] text message prompt



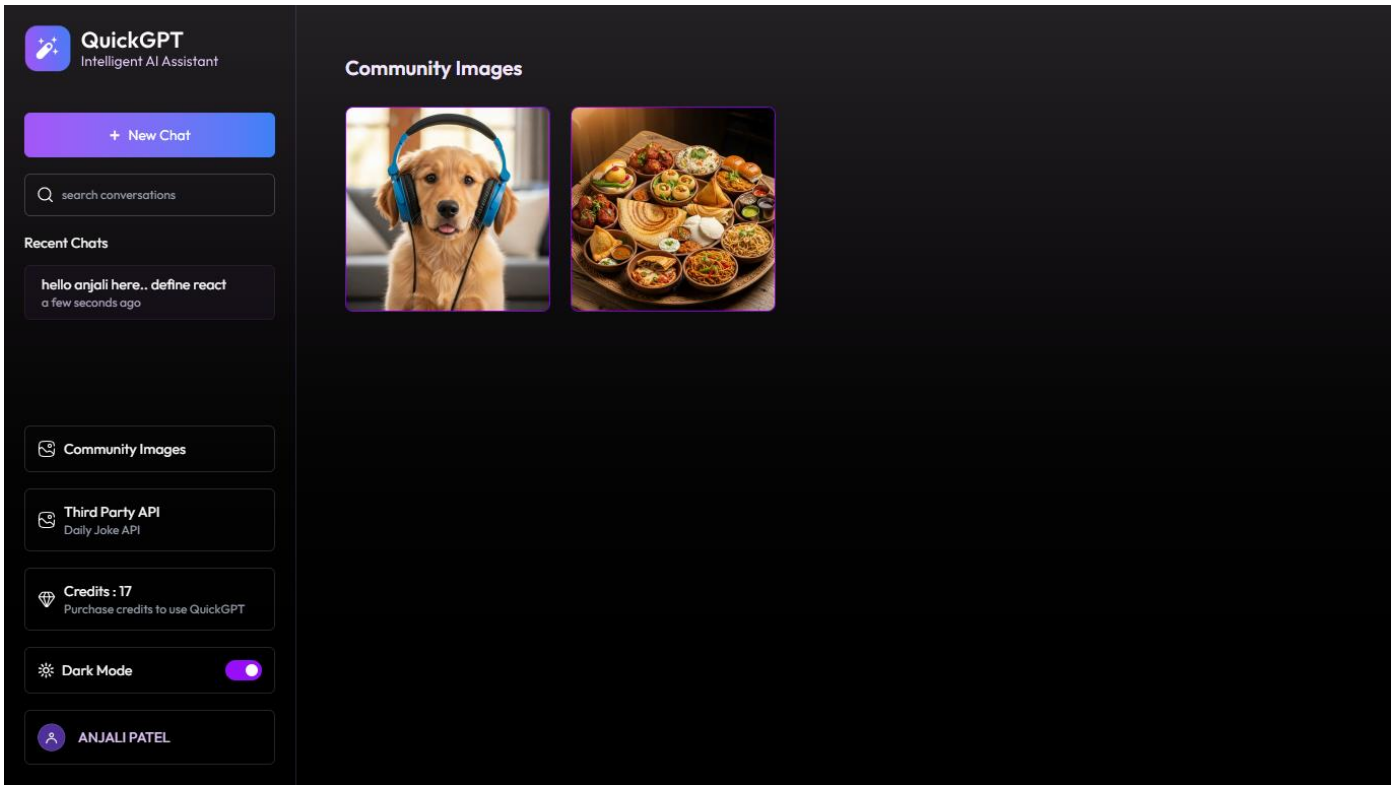
[11] image generation prompt and adding image to community



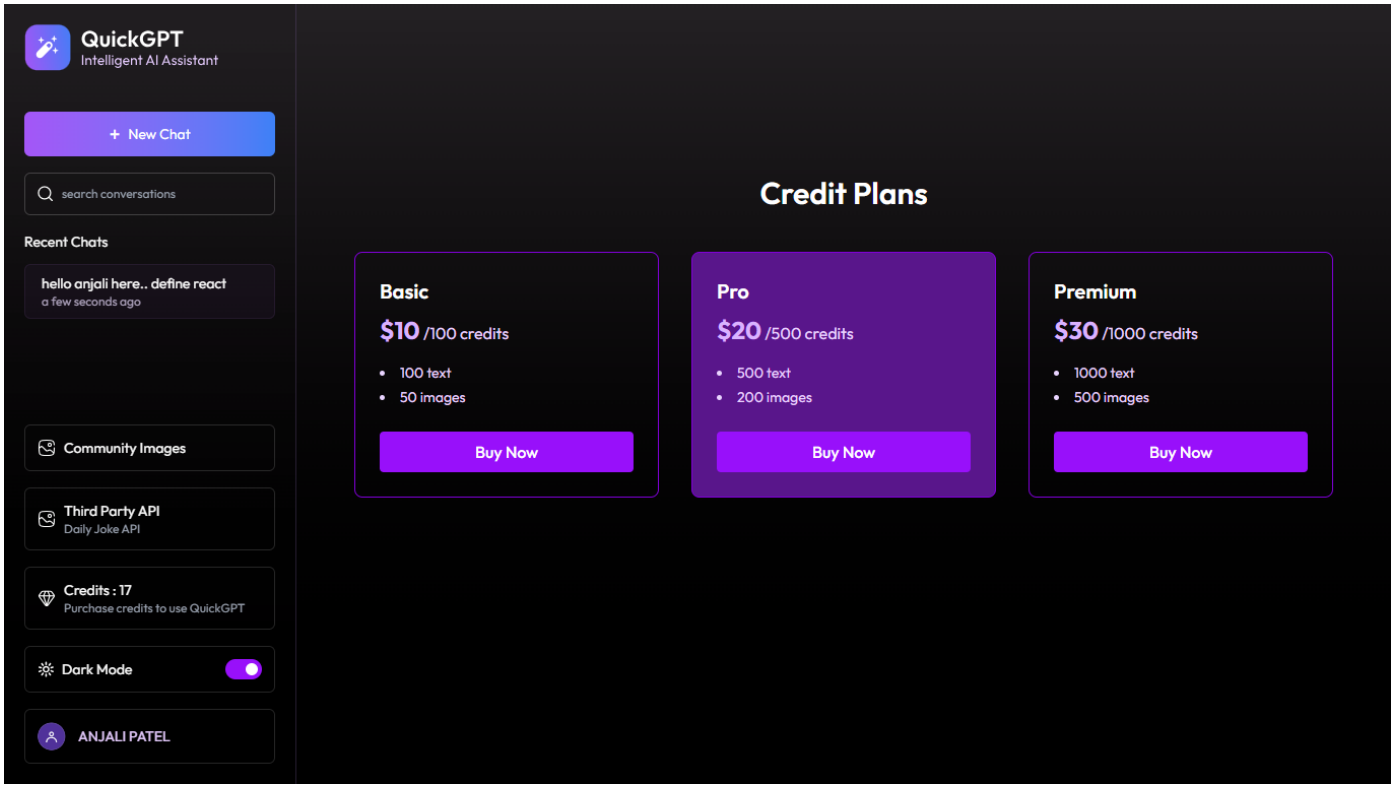
[12] image generated



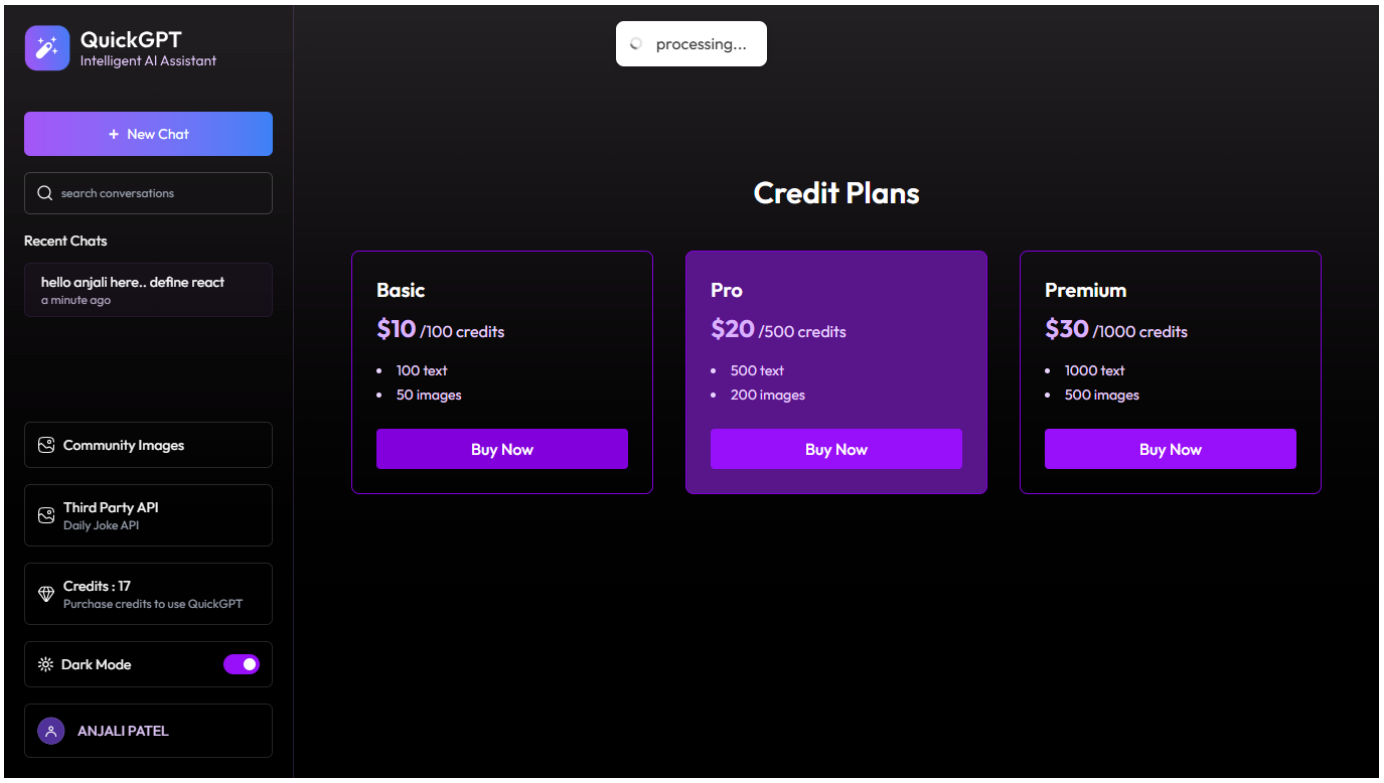
[13] community images



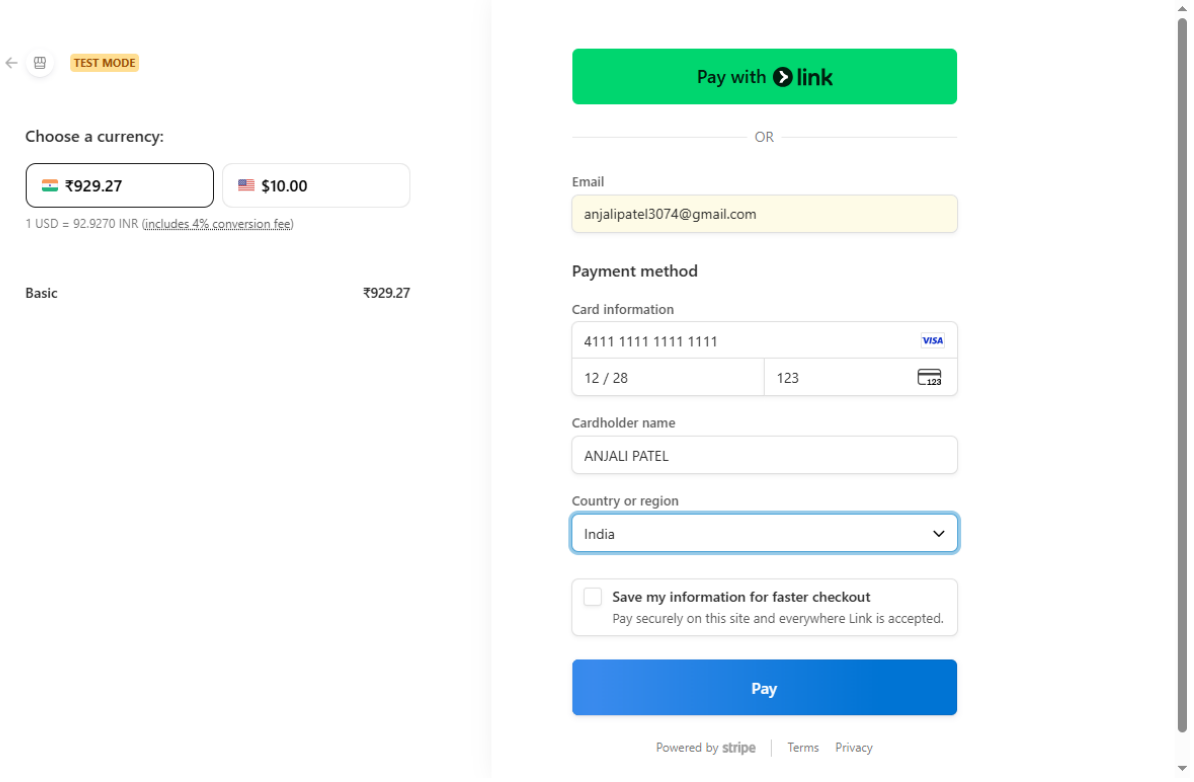
[14] pre-defined credit plans



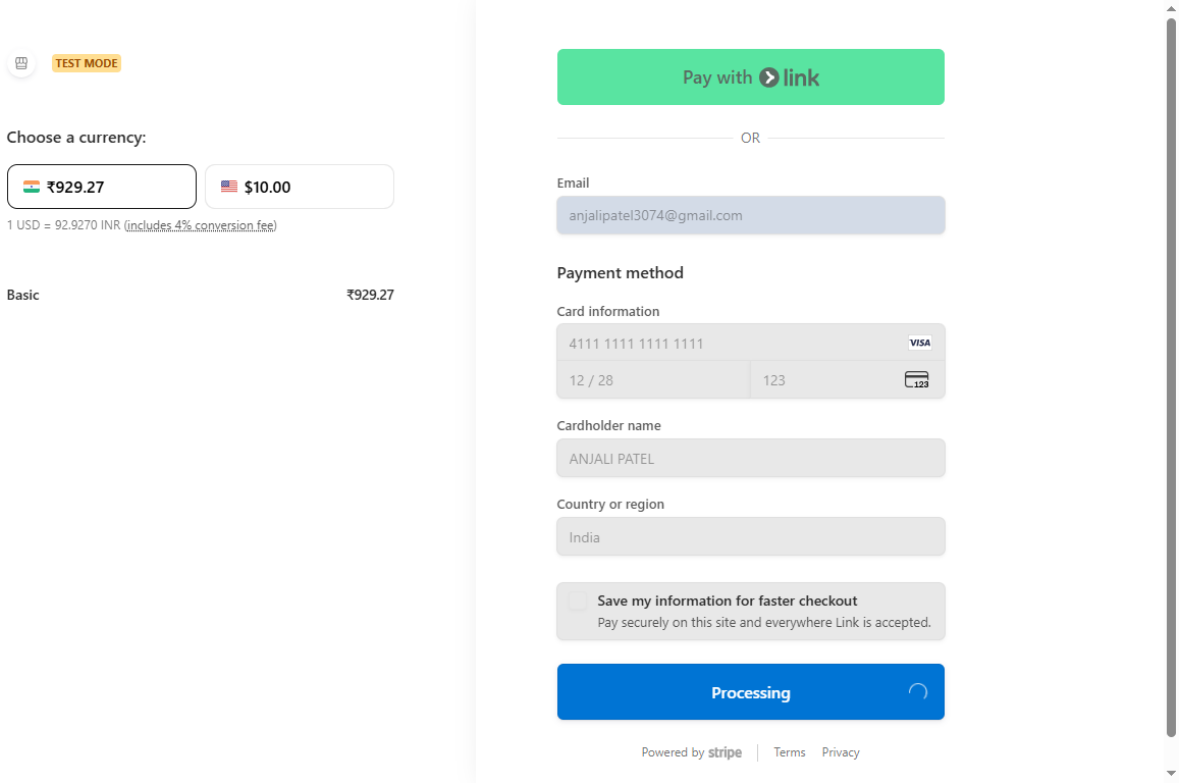
[15] initiate payment



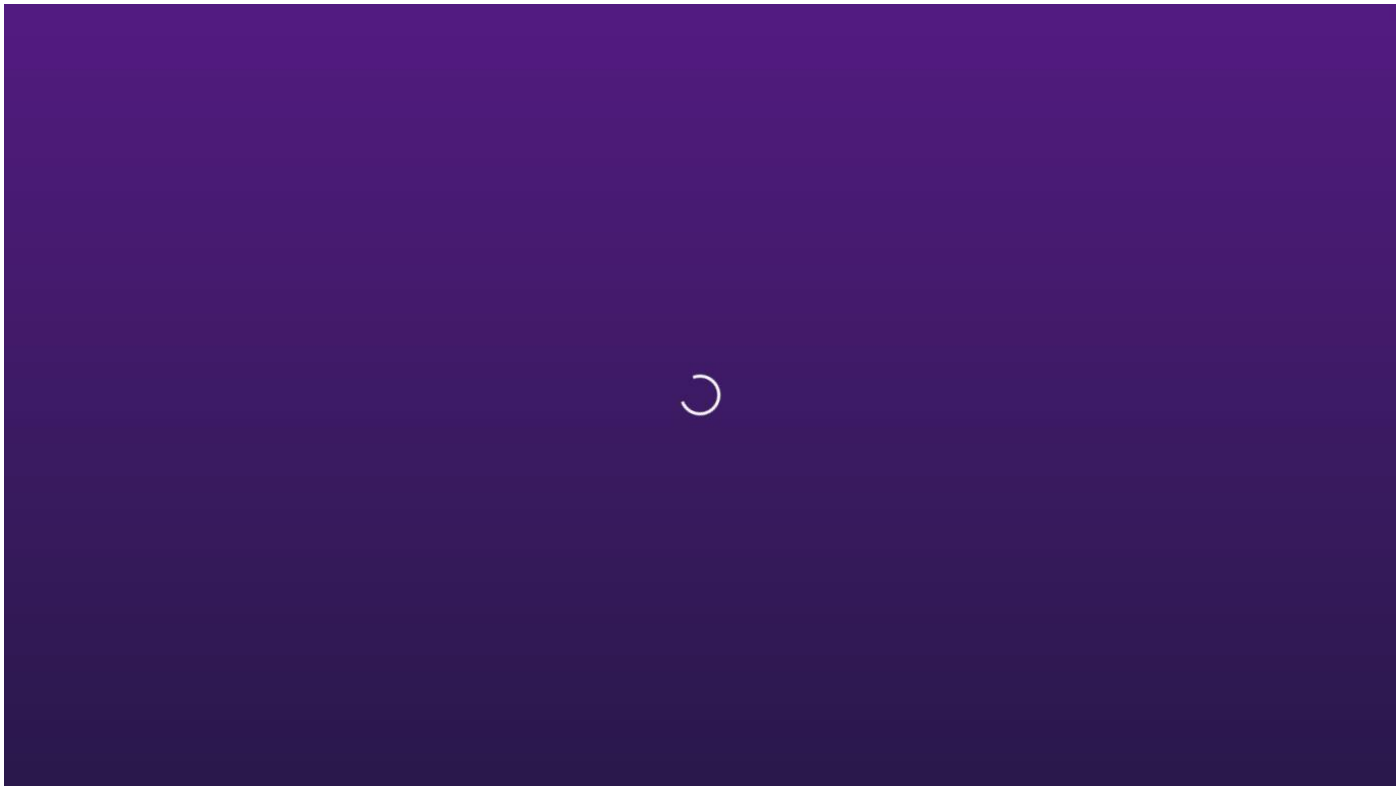
[16] stripe payment page with dummy details



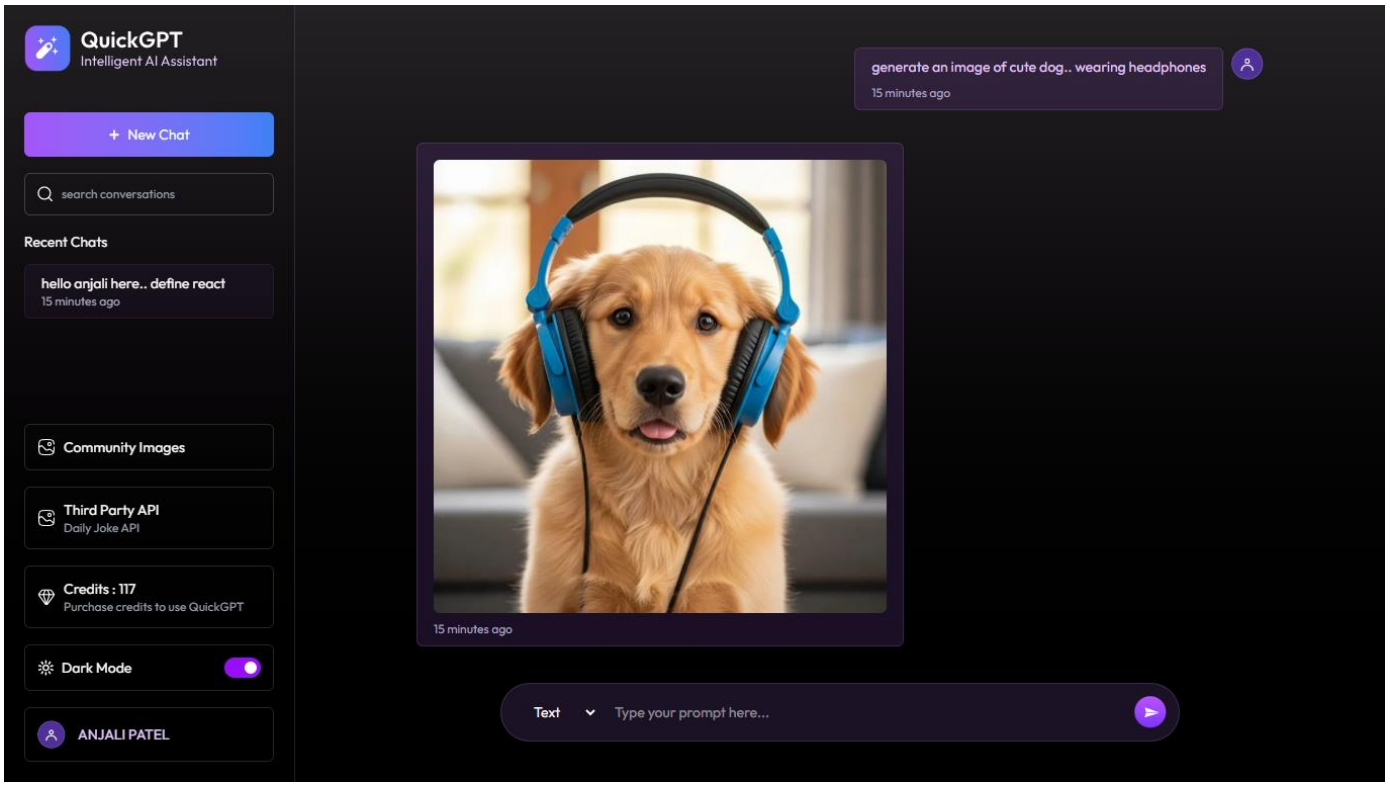
[17] payment successful



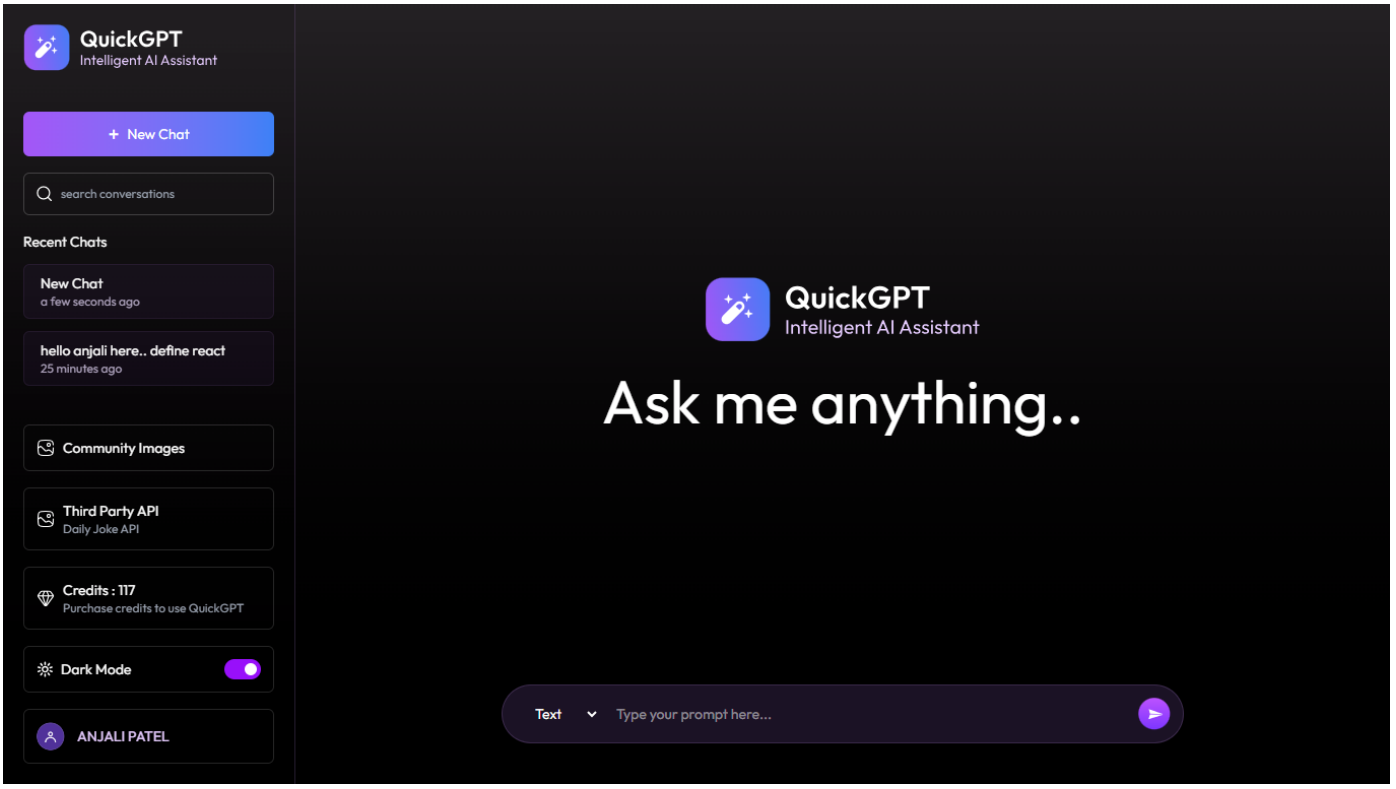
[18] redirection



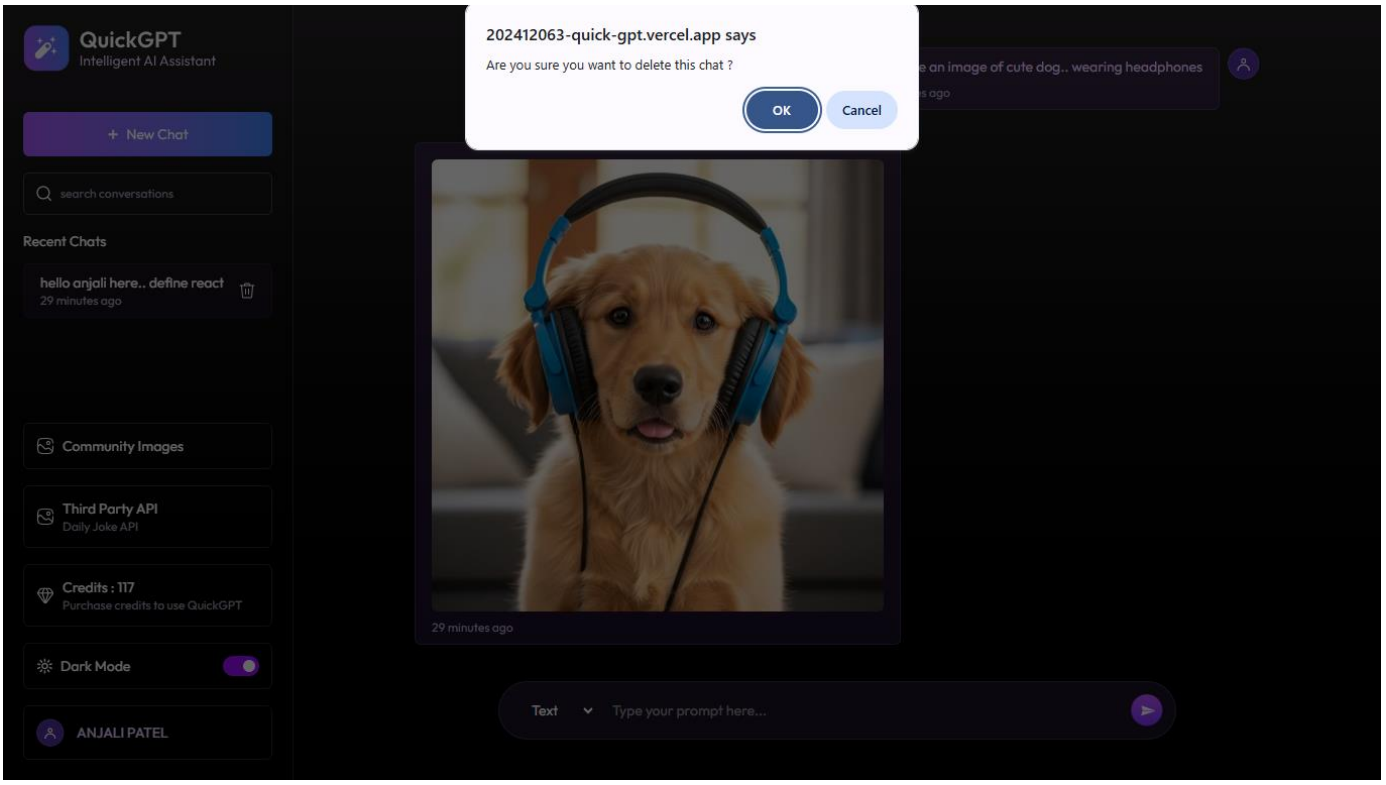
[19] updated credits



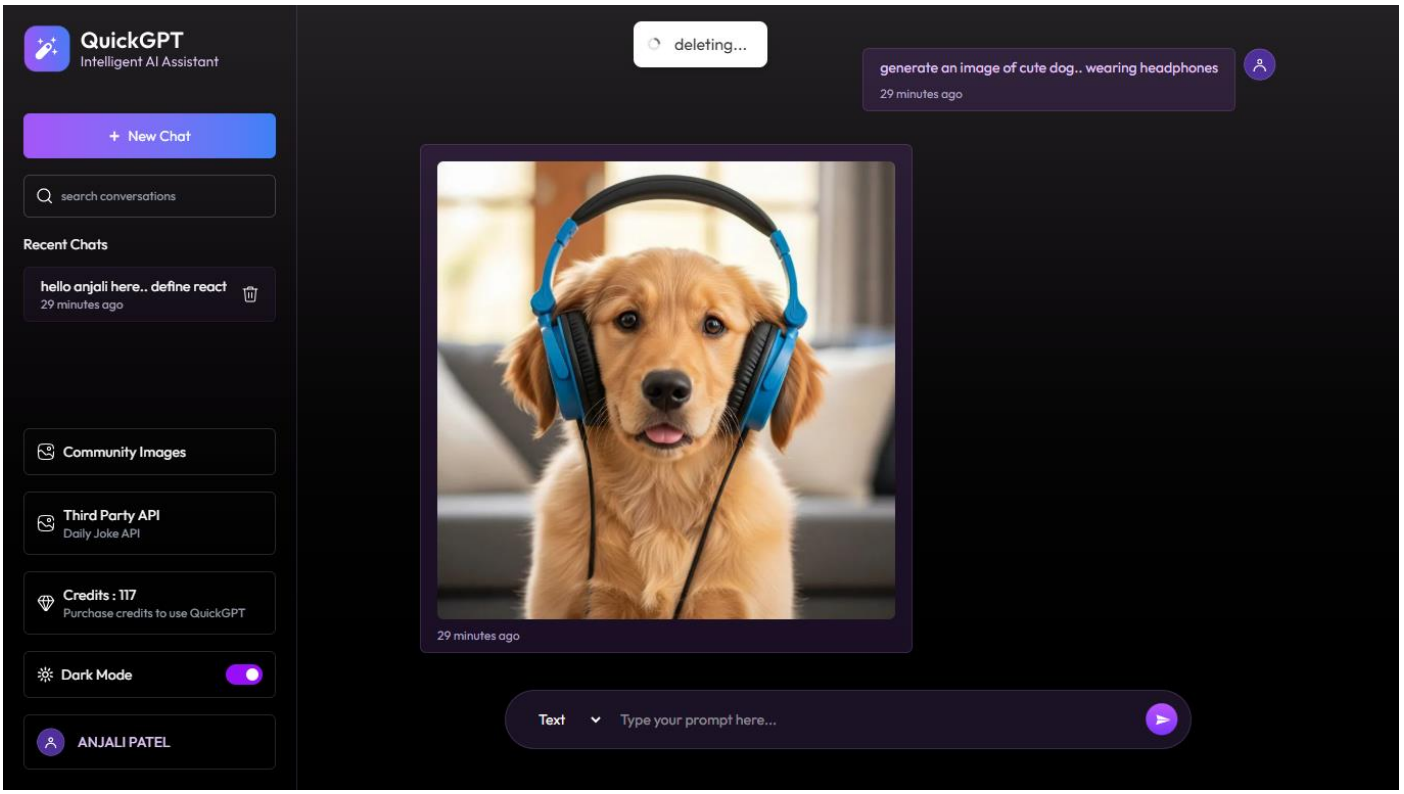
[20] new chat creation



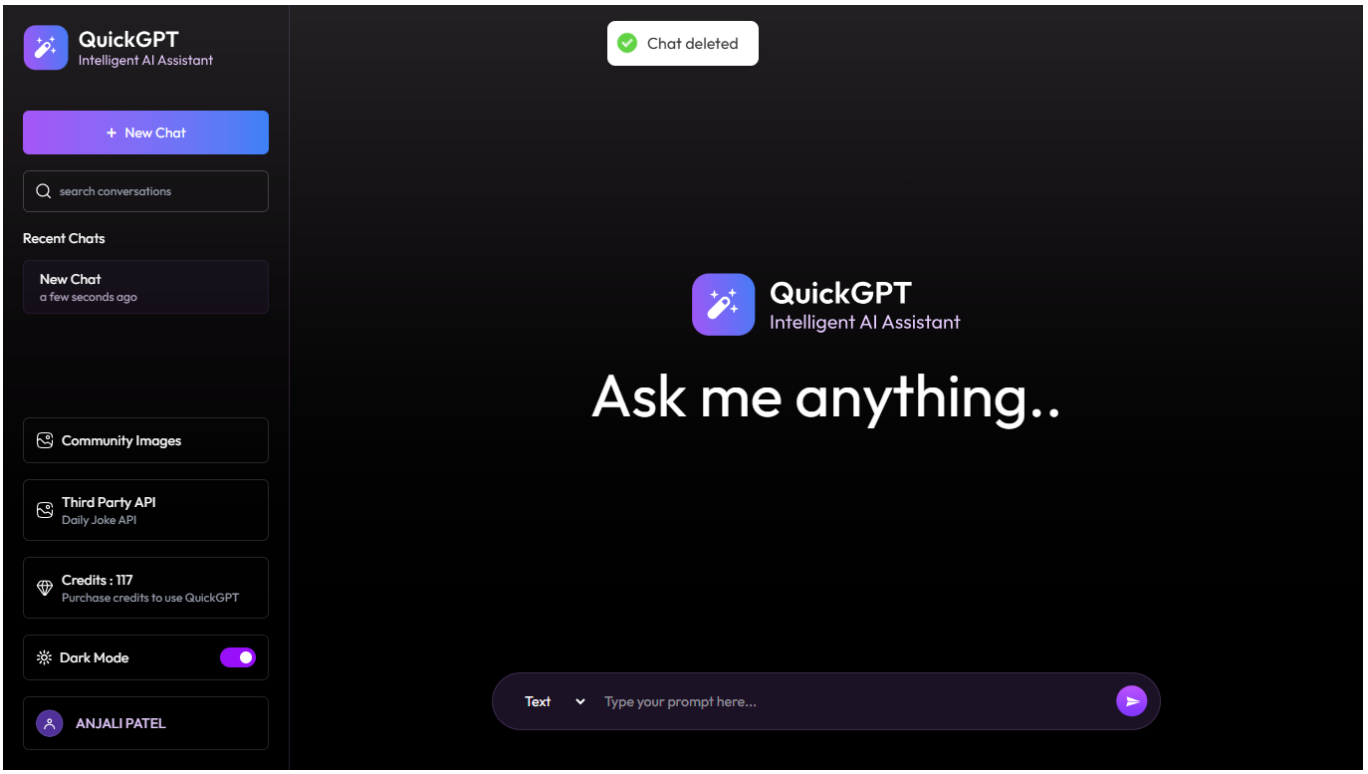
[21] delete chat



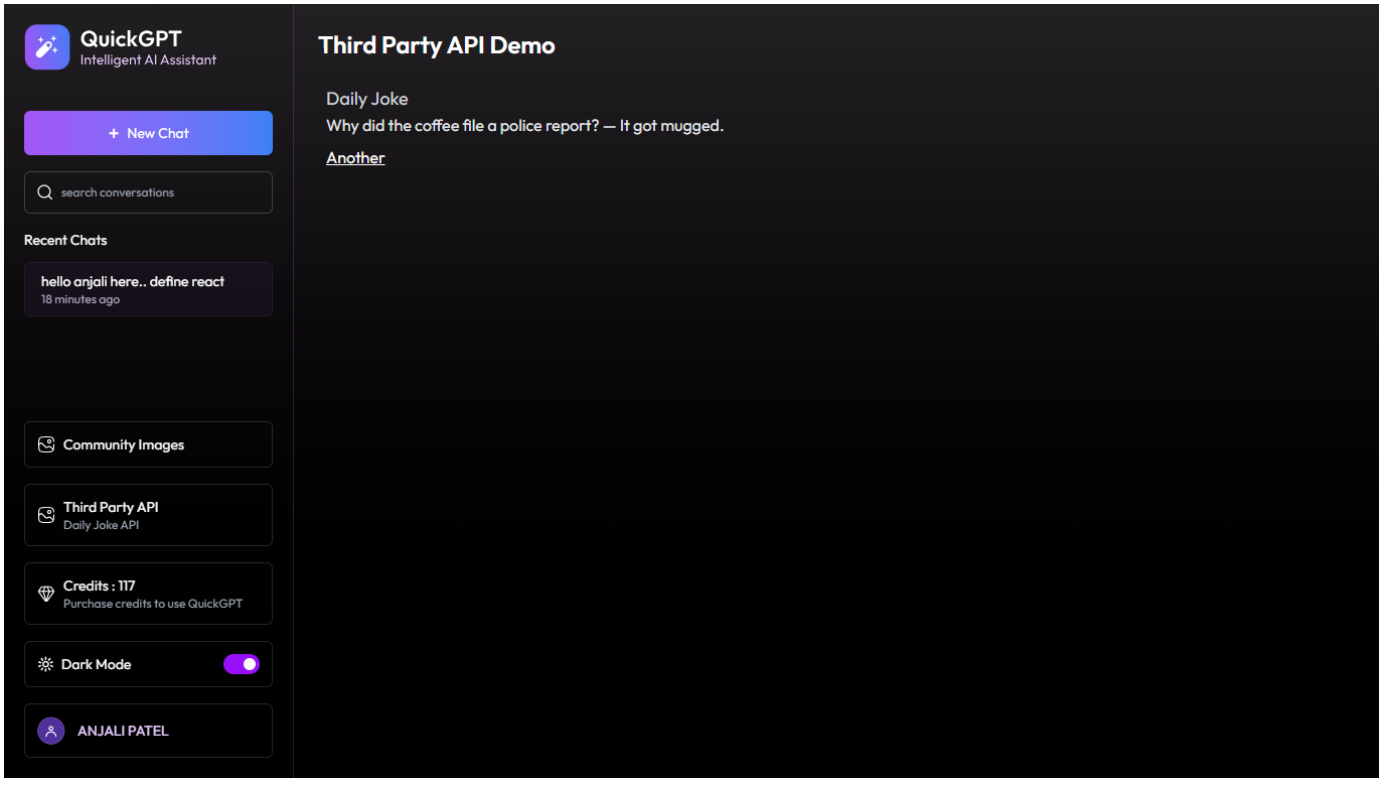
[22] deleting



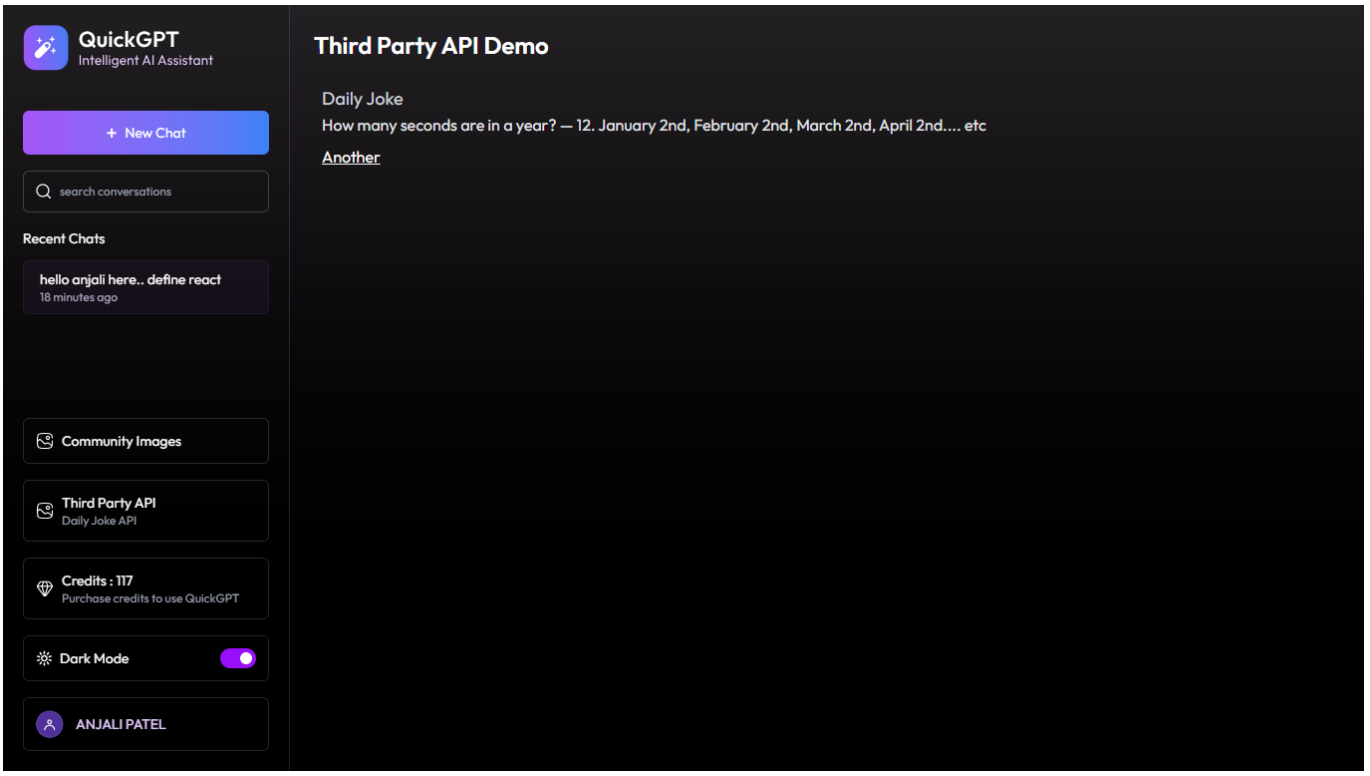
[23] chat deleted



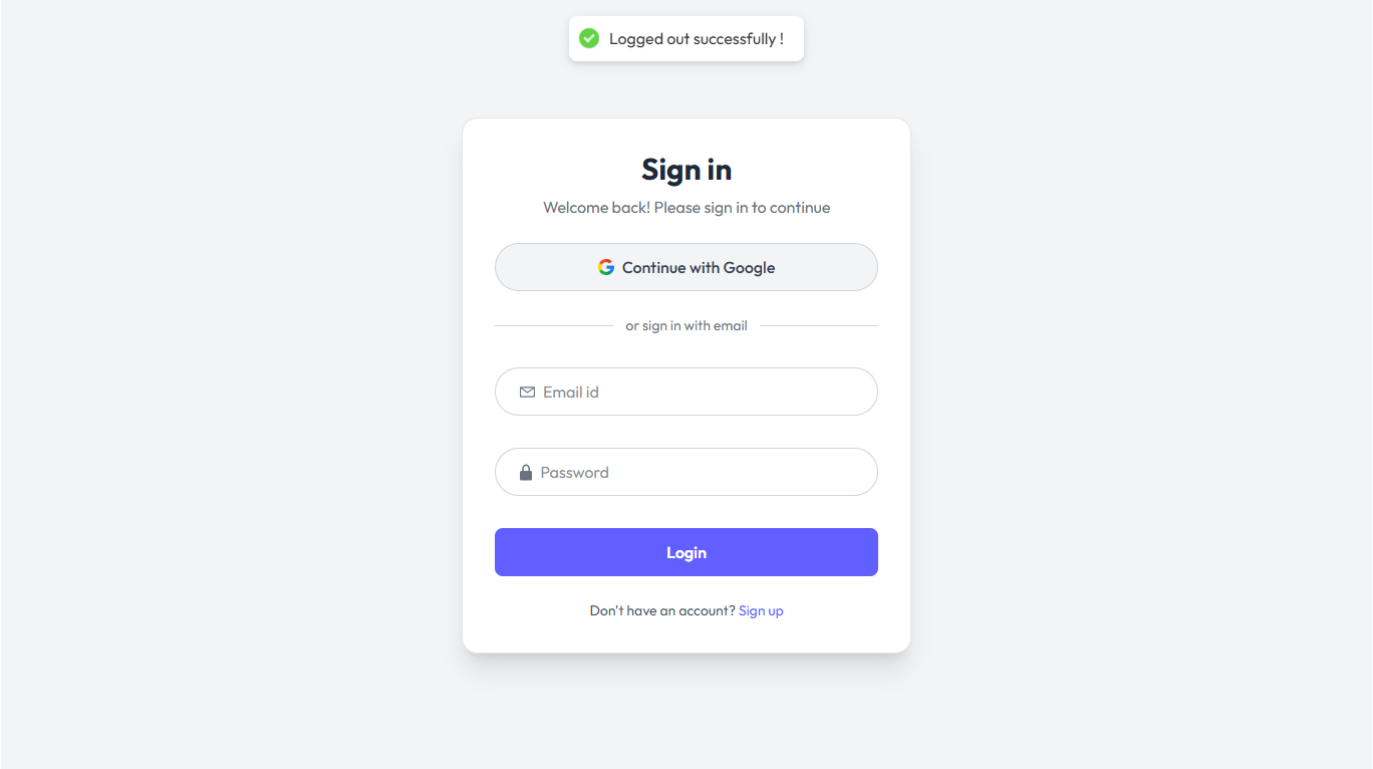
[24] third-party API



[25] random joke generation

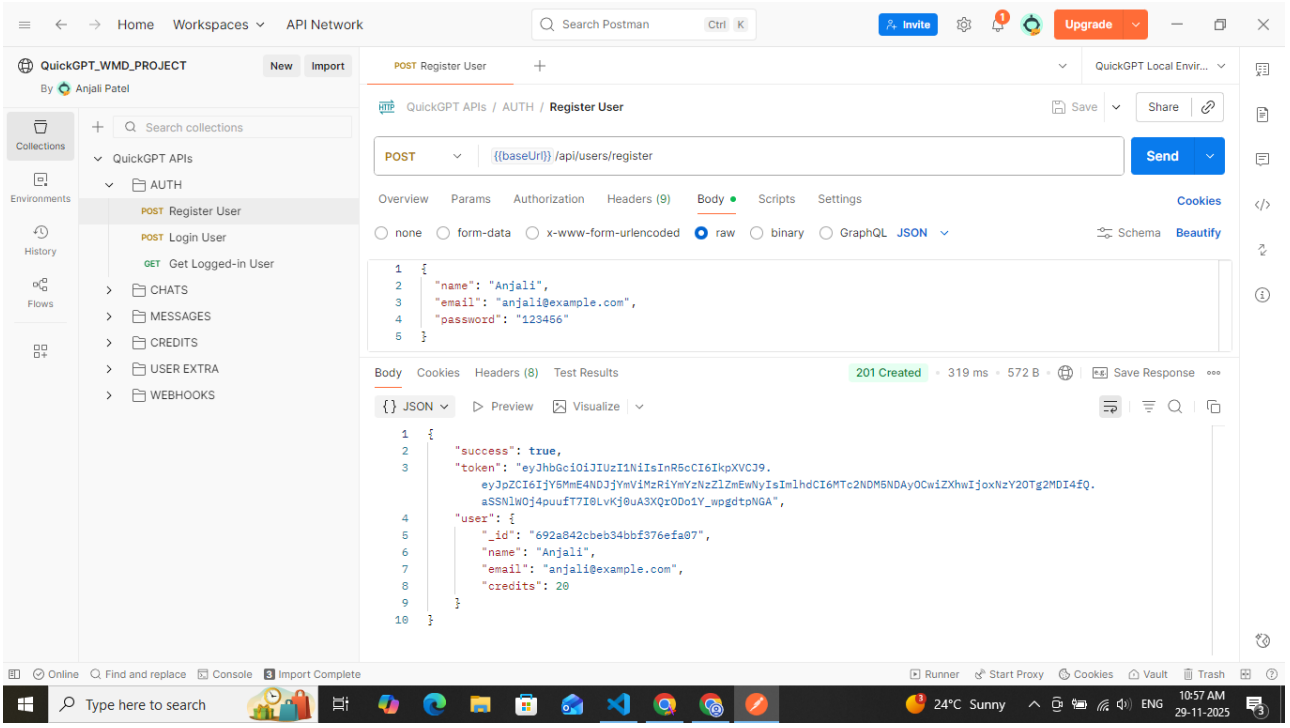


[26] logout

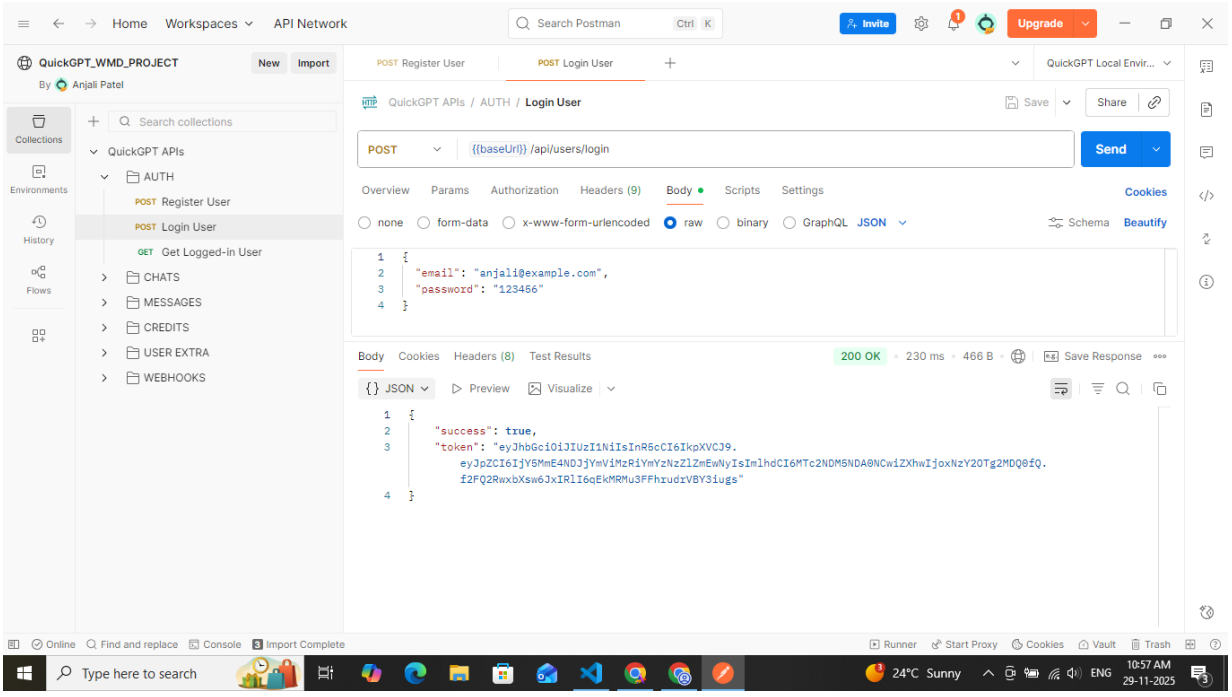


Testing Snapshots

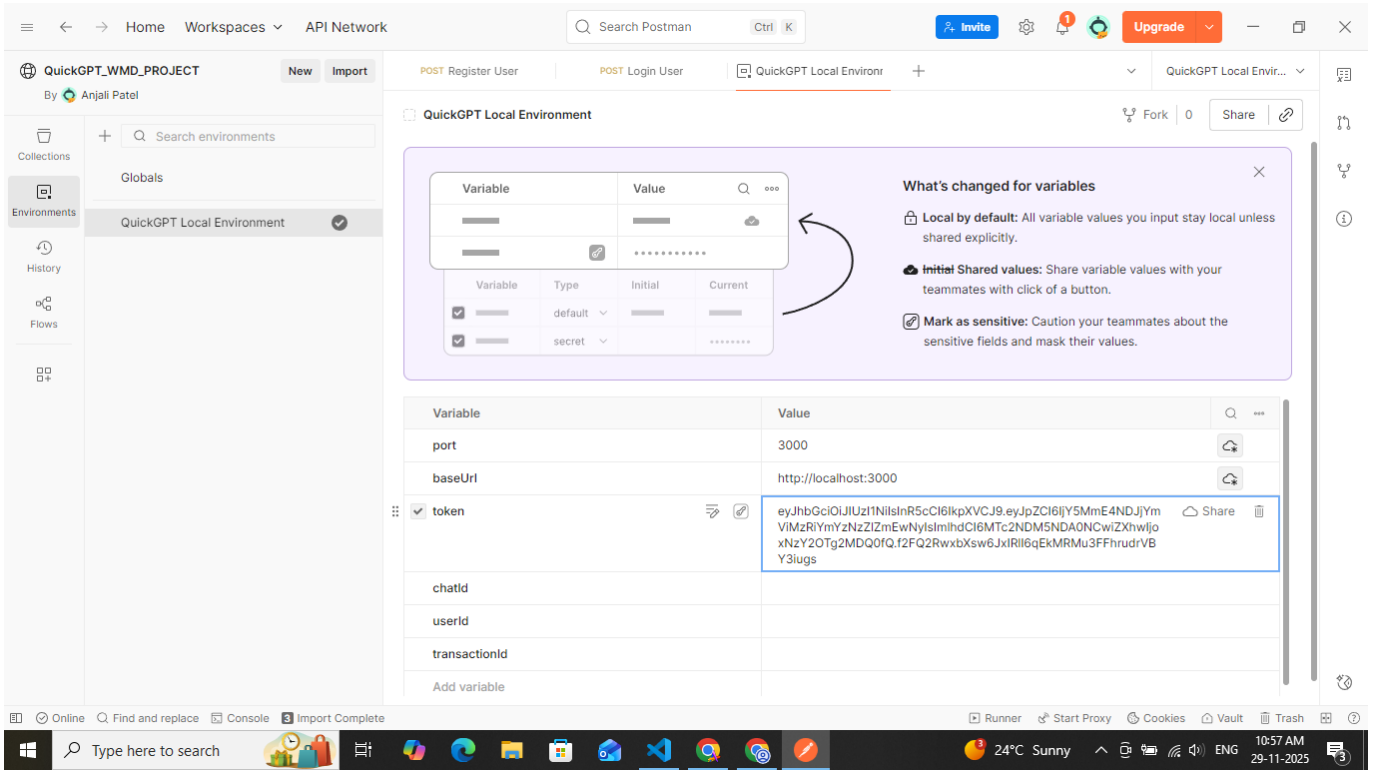
[01] register user API



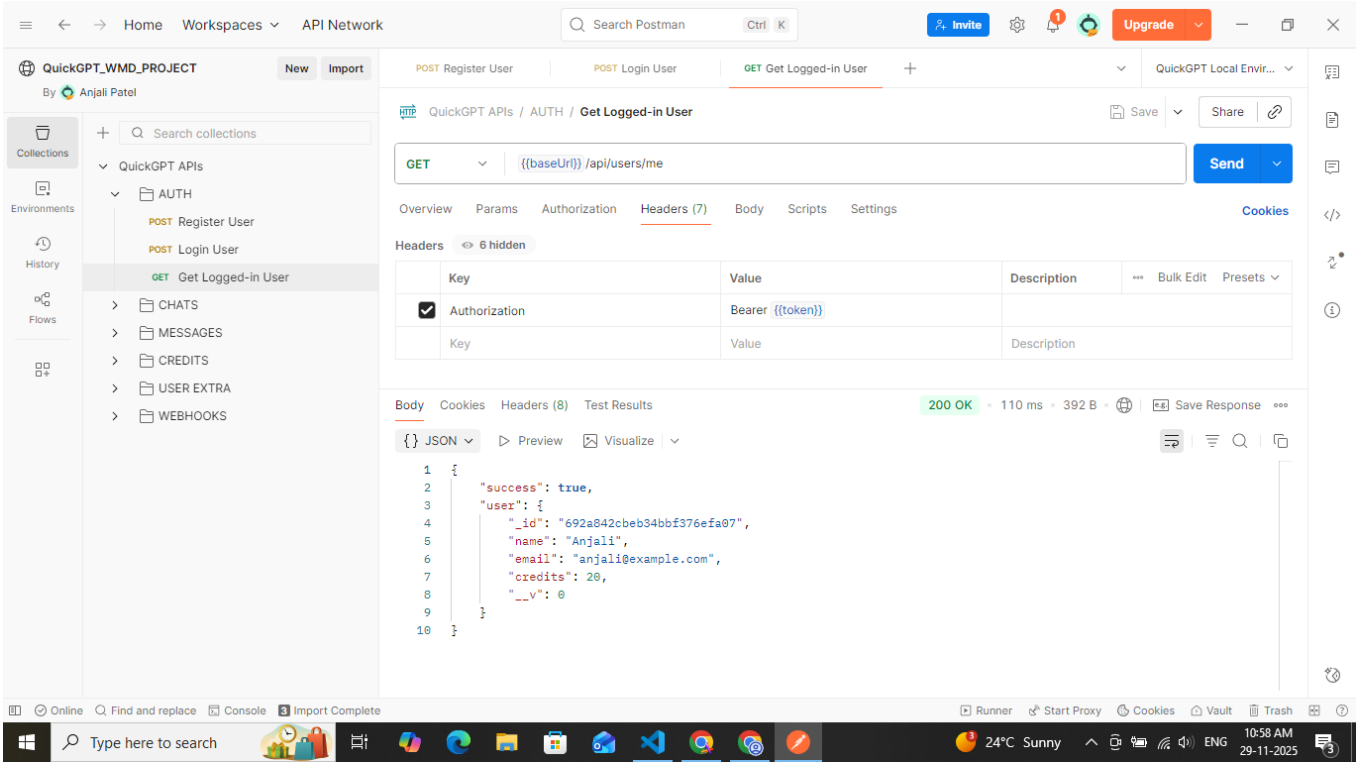
[02] login user API



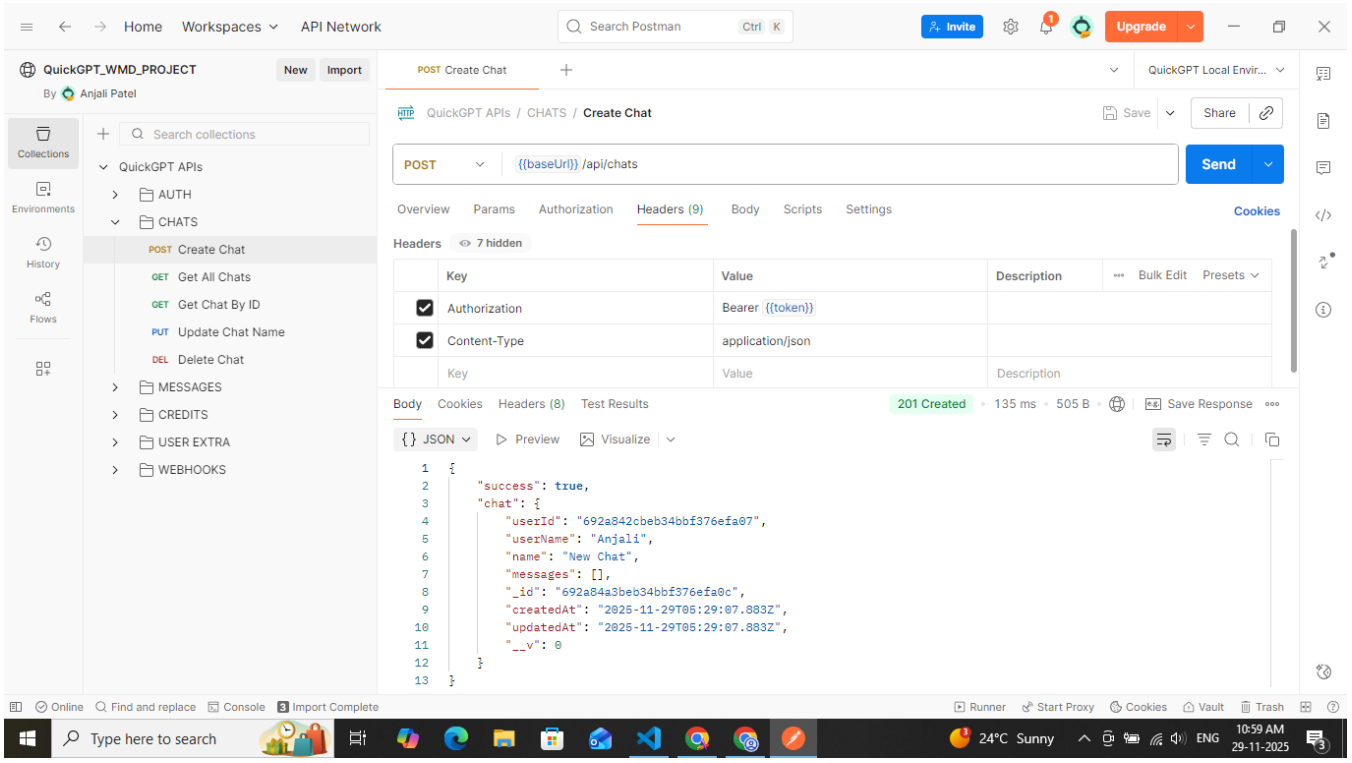
[03] adding token to environment variable



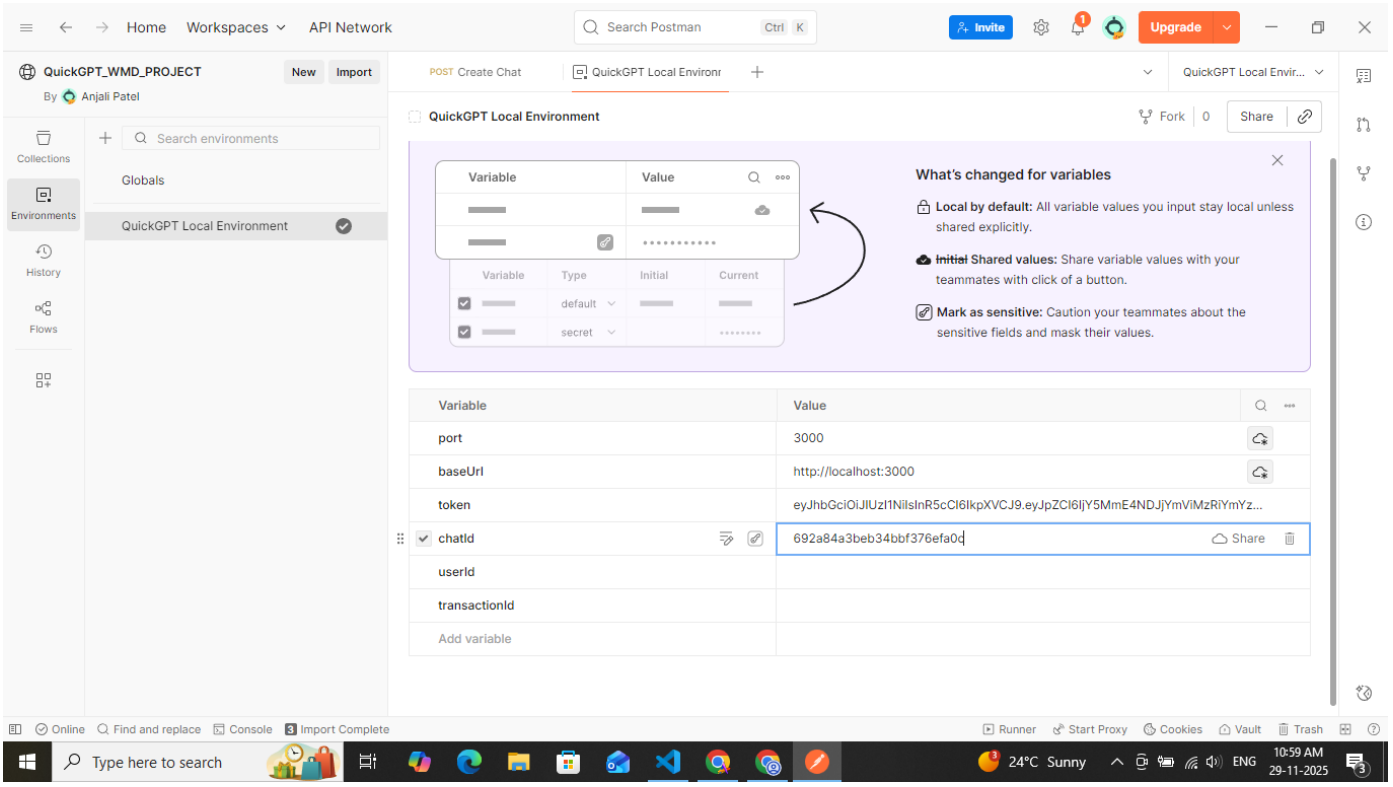
[04] get logged-in user API



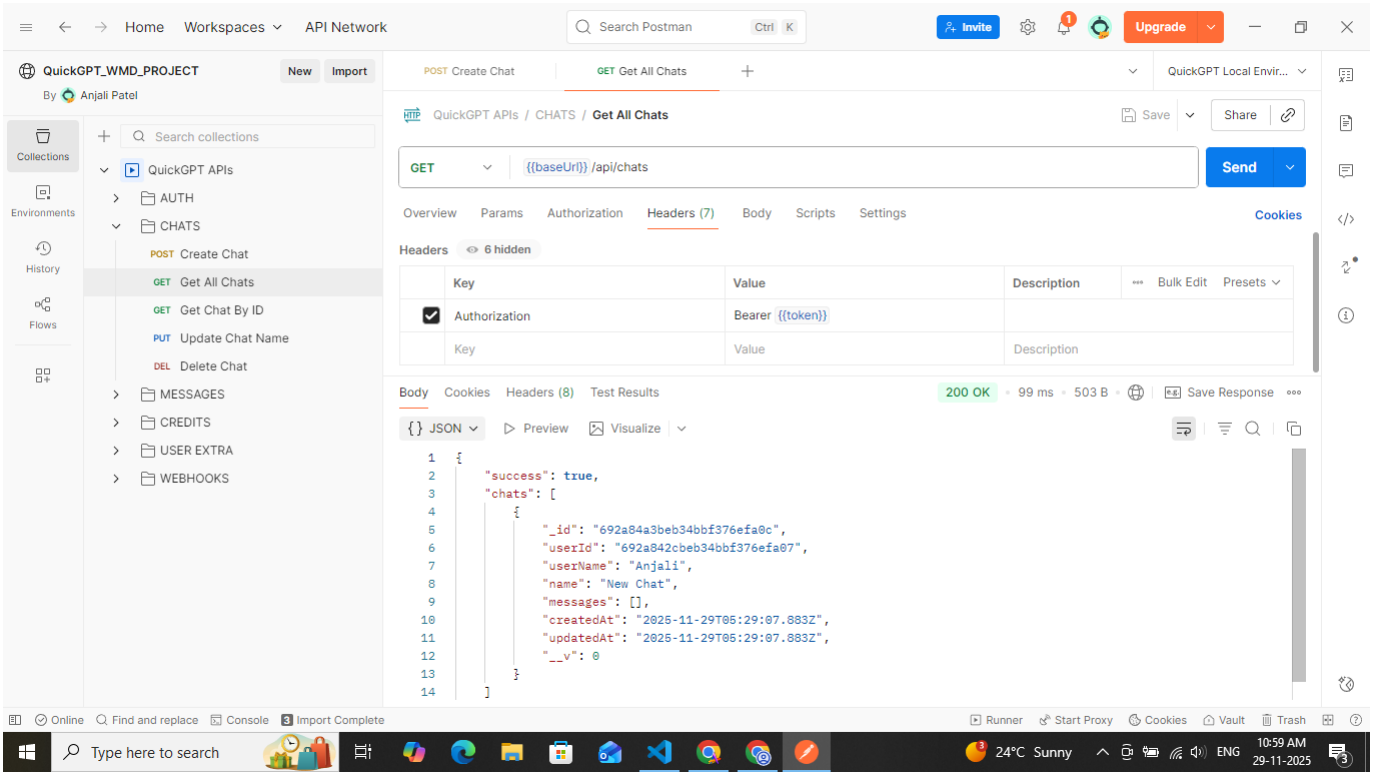
[05] create chat API



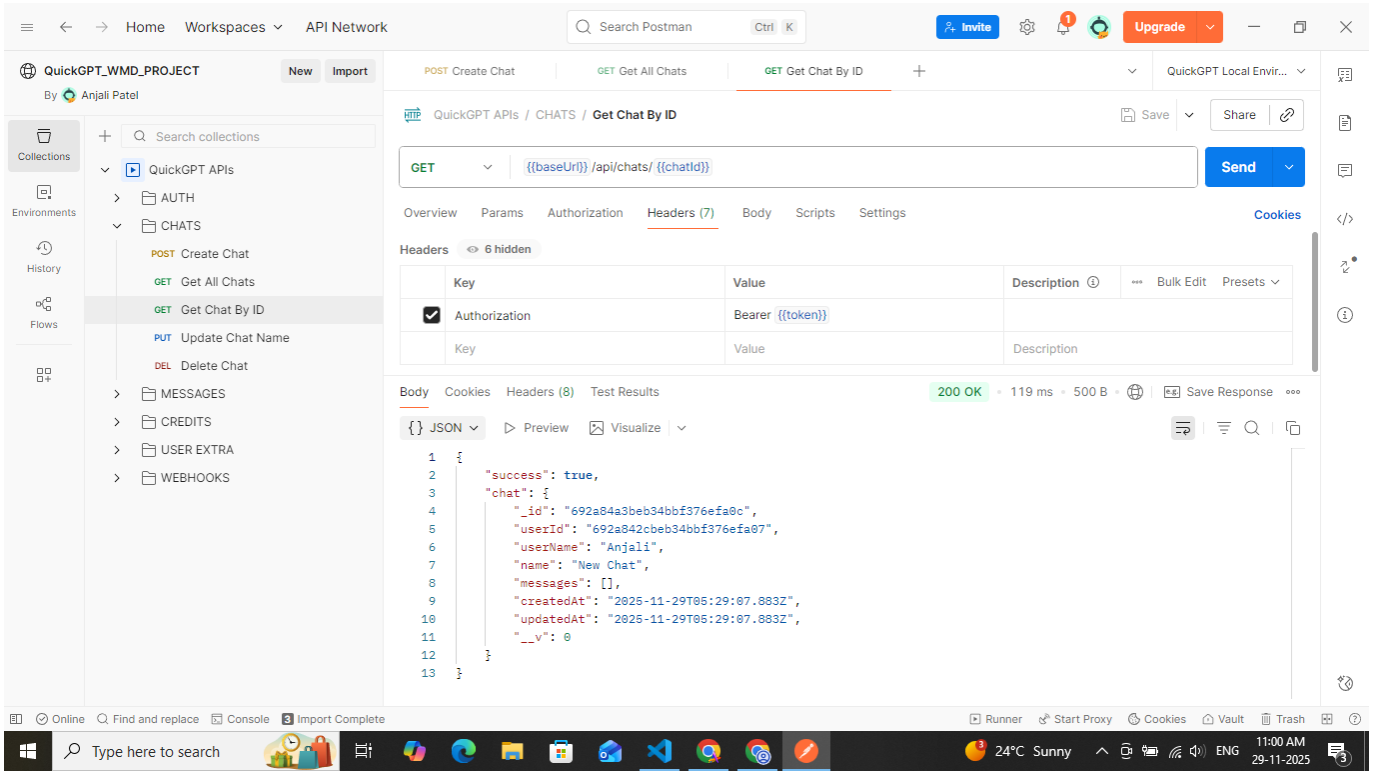
[06] adding chatID-to environment variable



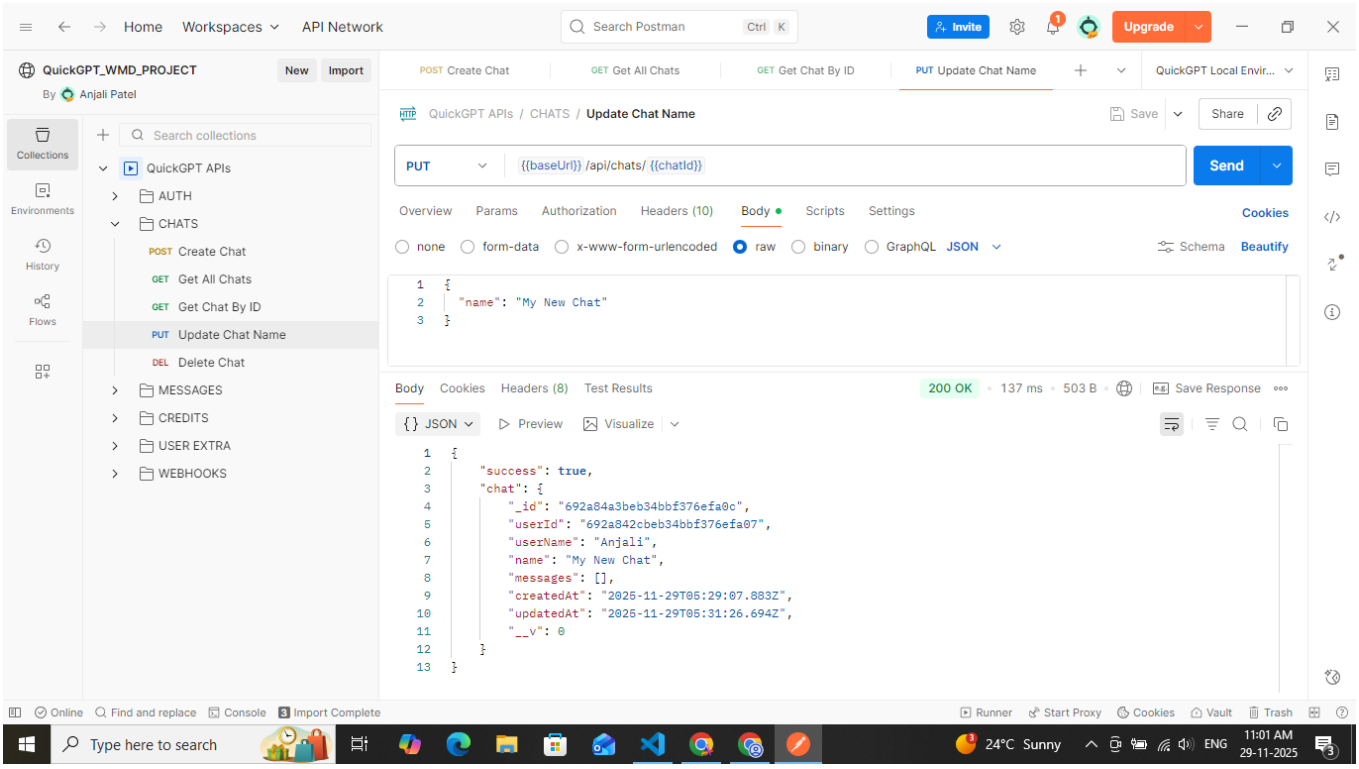
[07] get all chats API



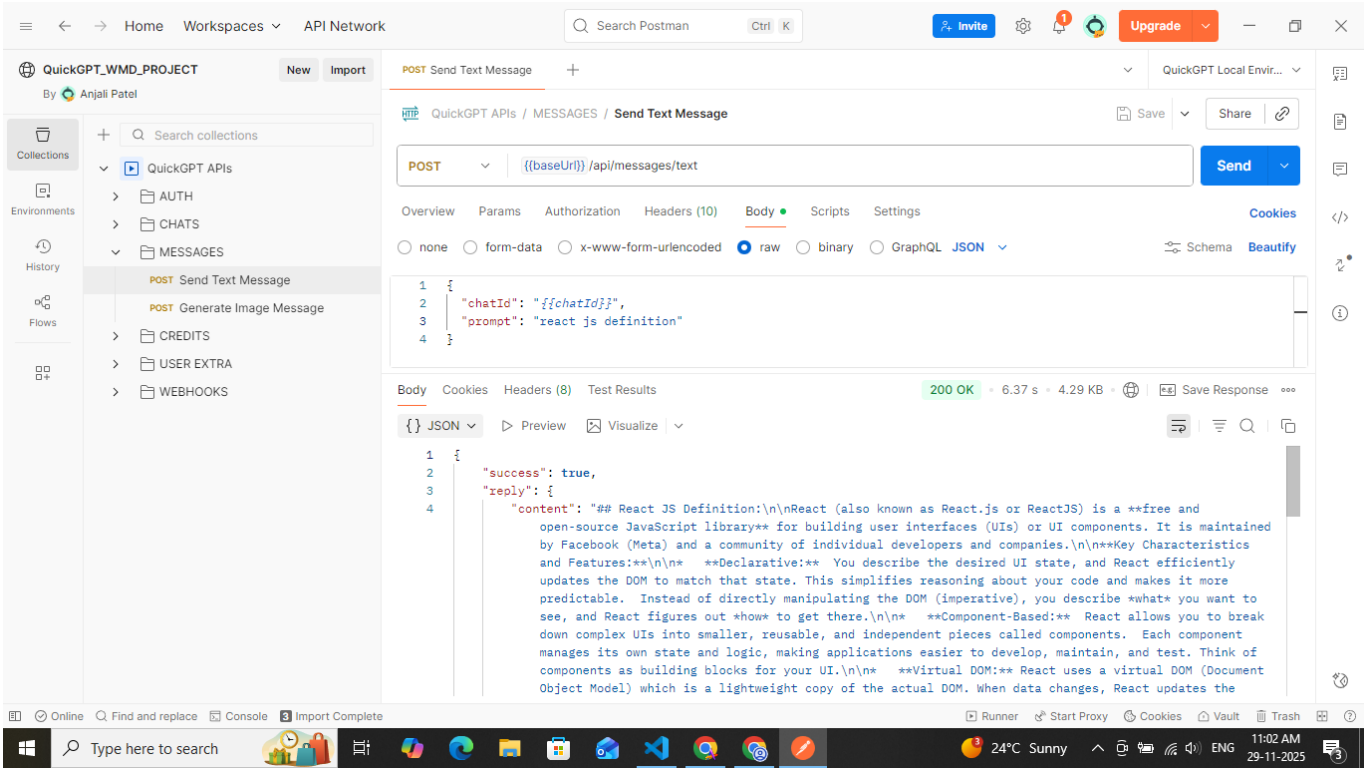
[08] get chat by ID



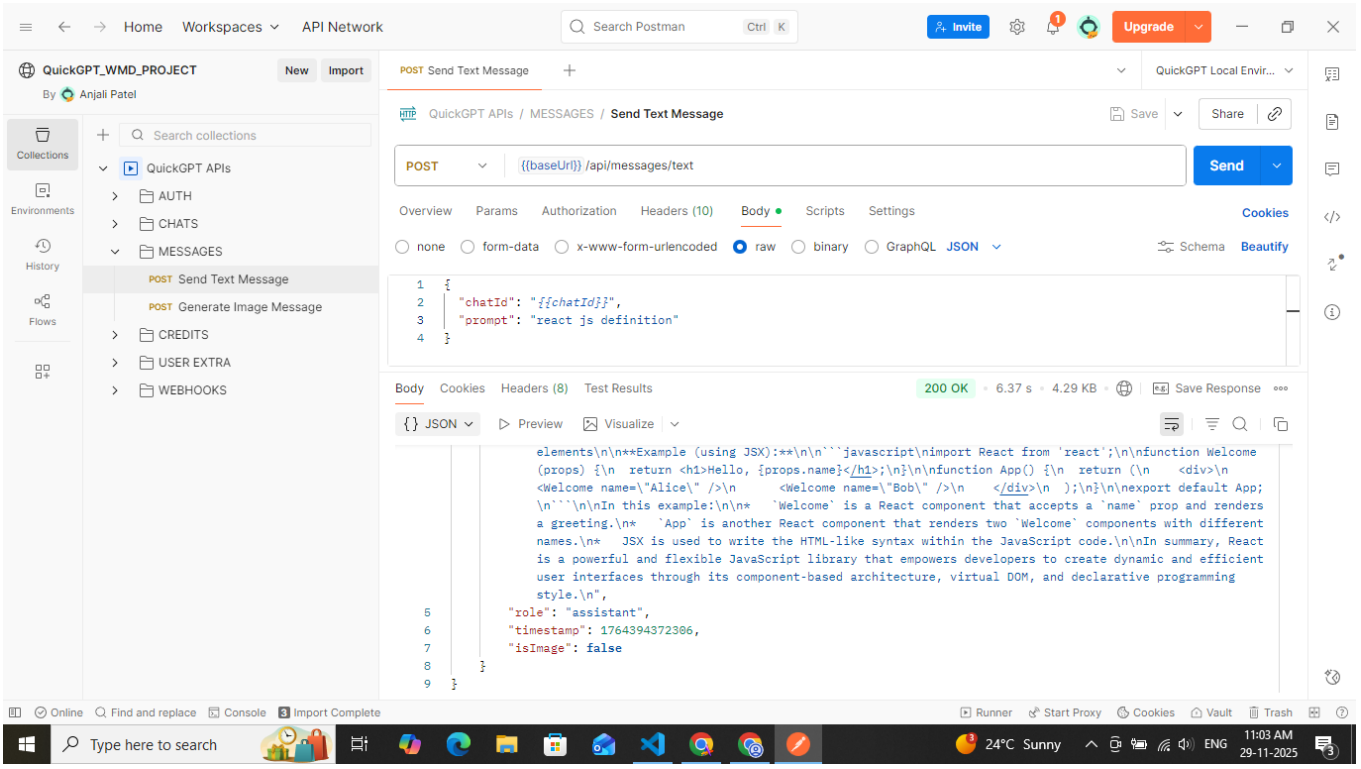
[09] update chat name



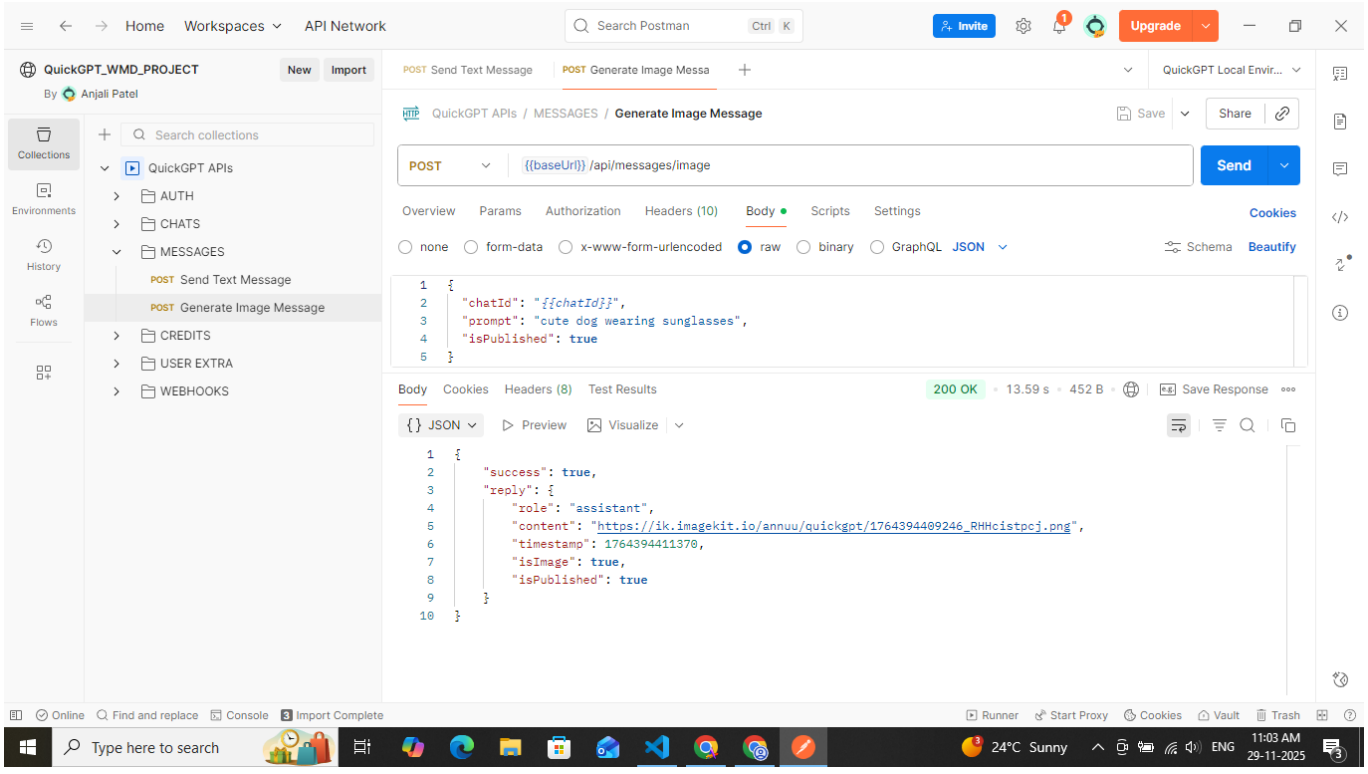
[10] send text message



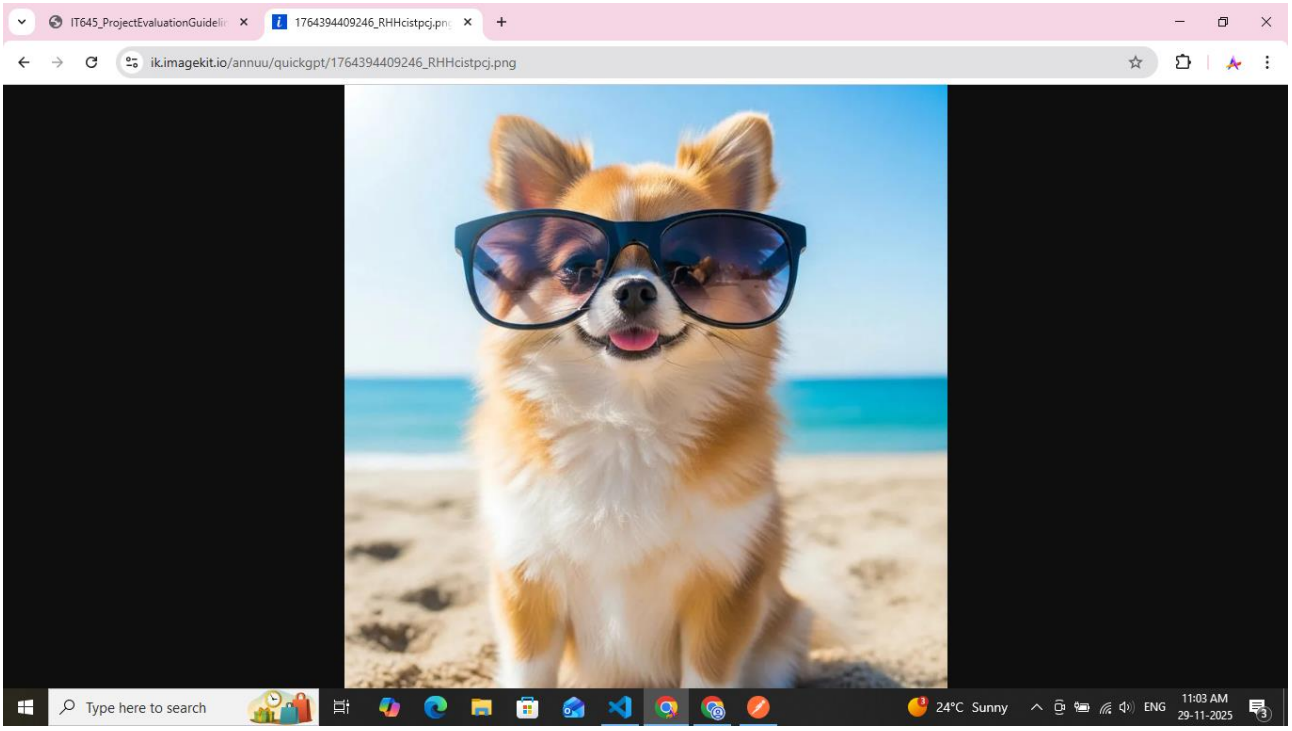
[11] send text message API 2



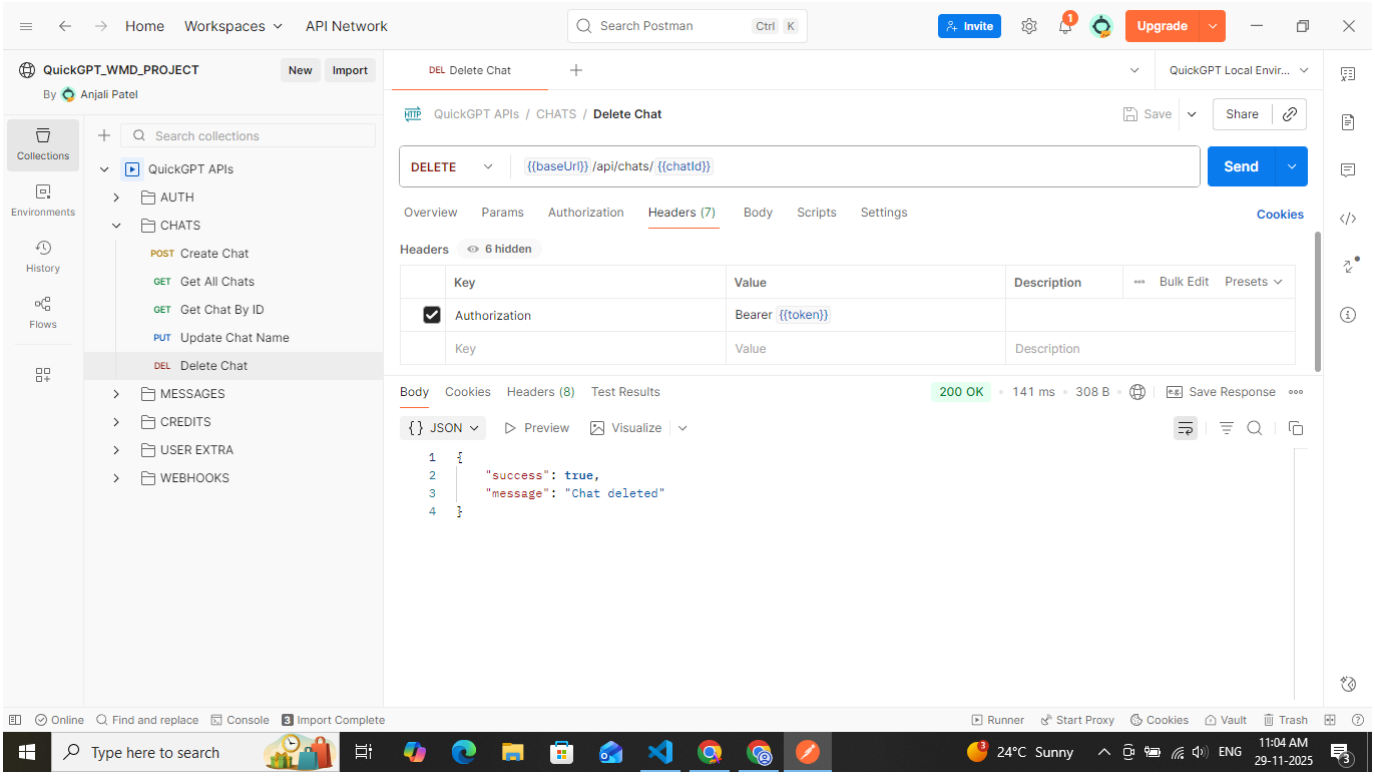
[12] generate image message



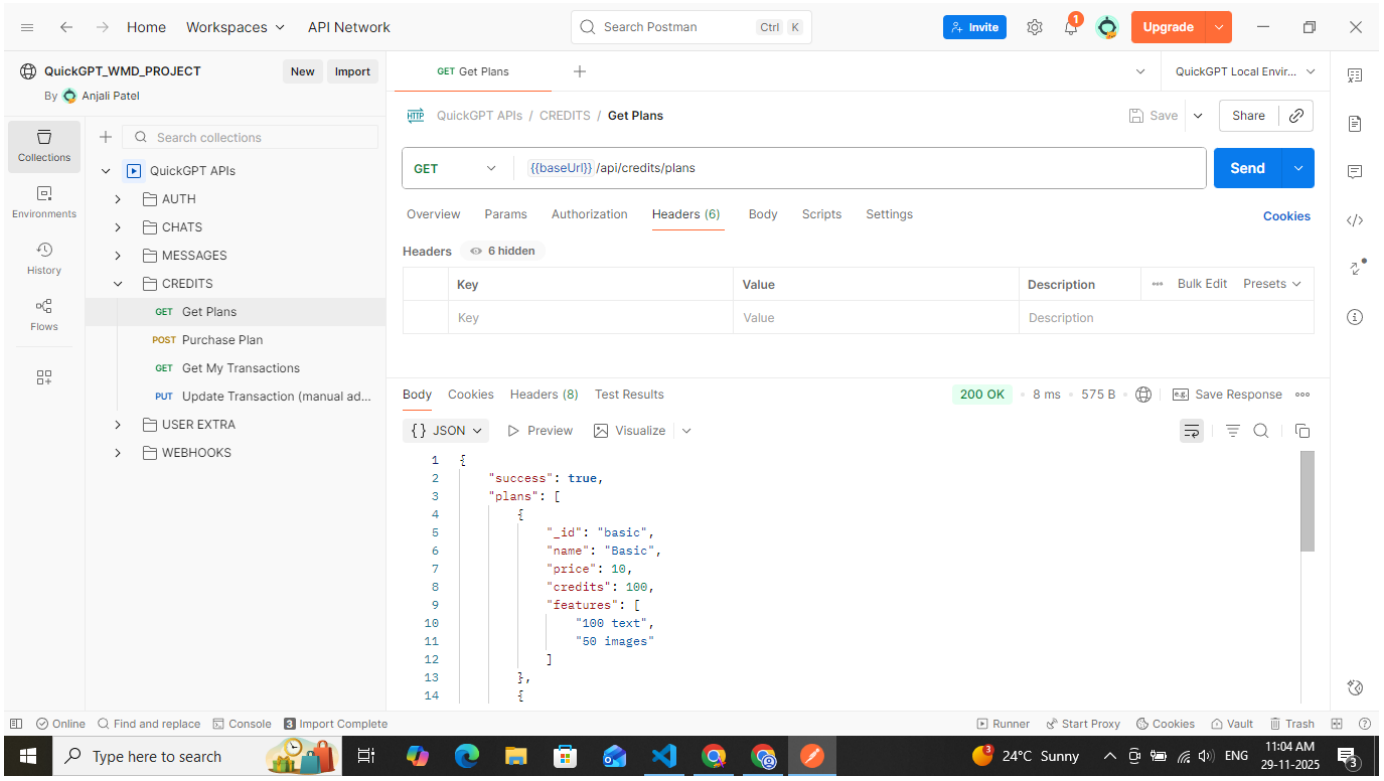
[13] getting generated image from link



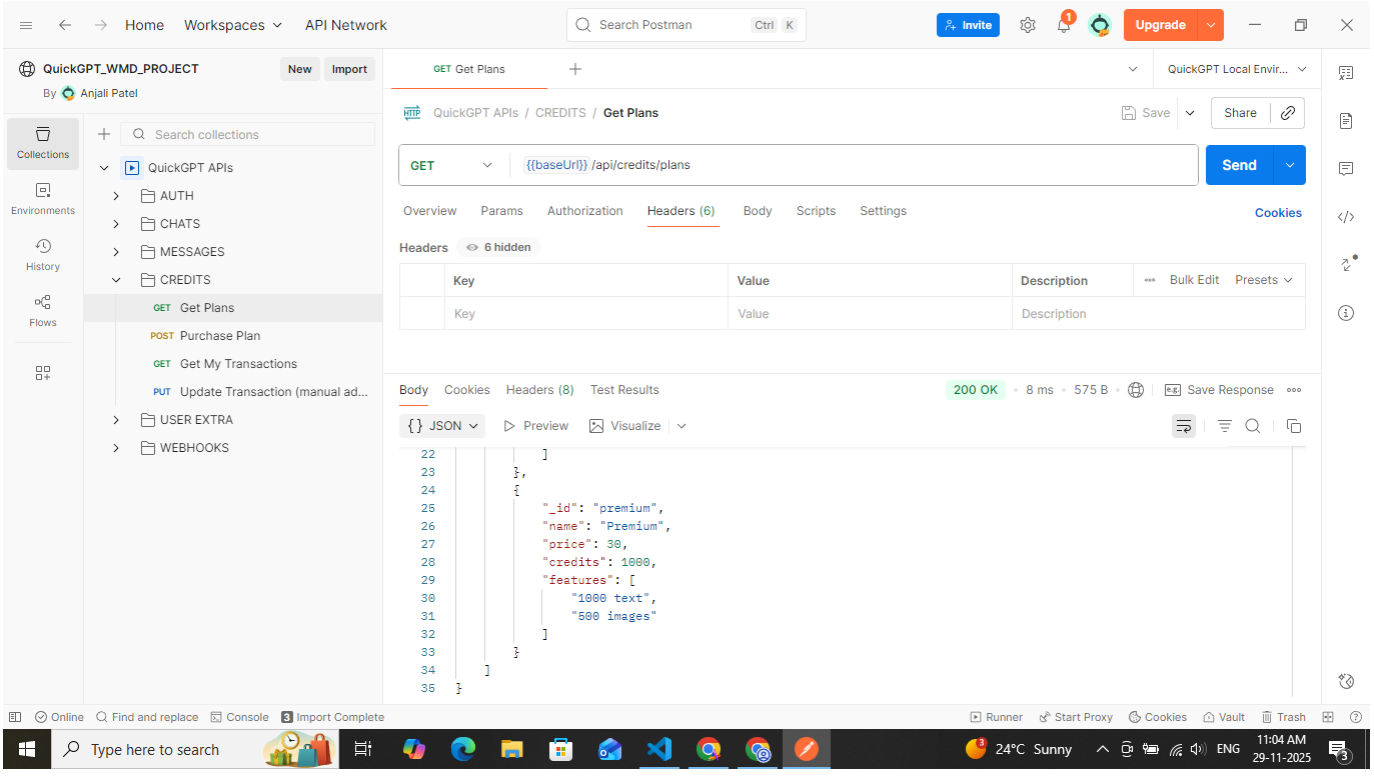
[14] delete chat API



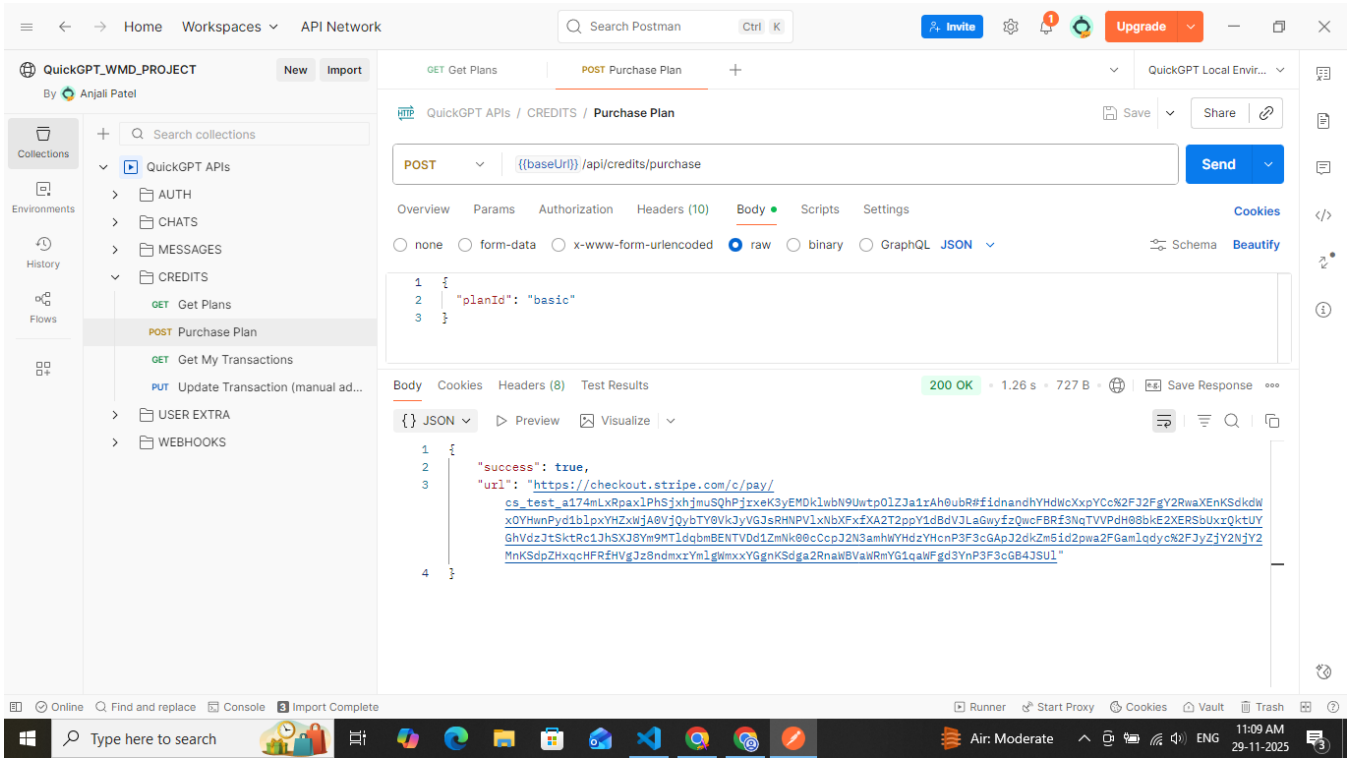
[15] get plans API



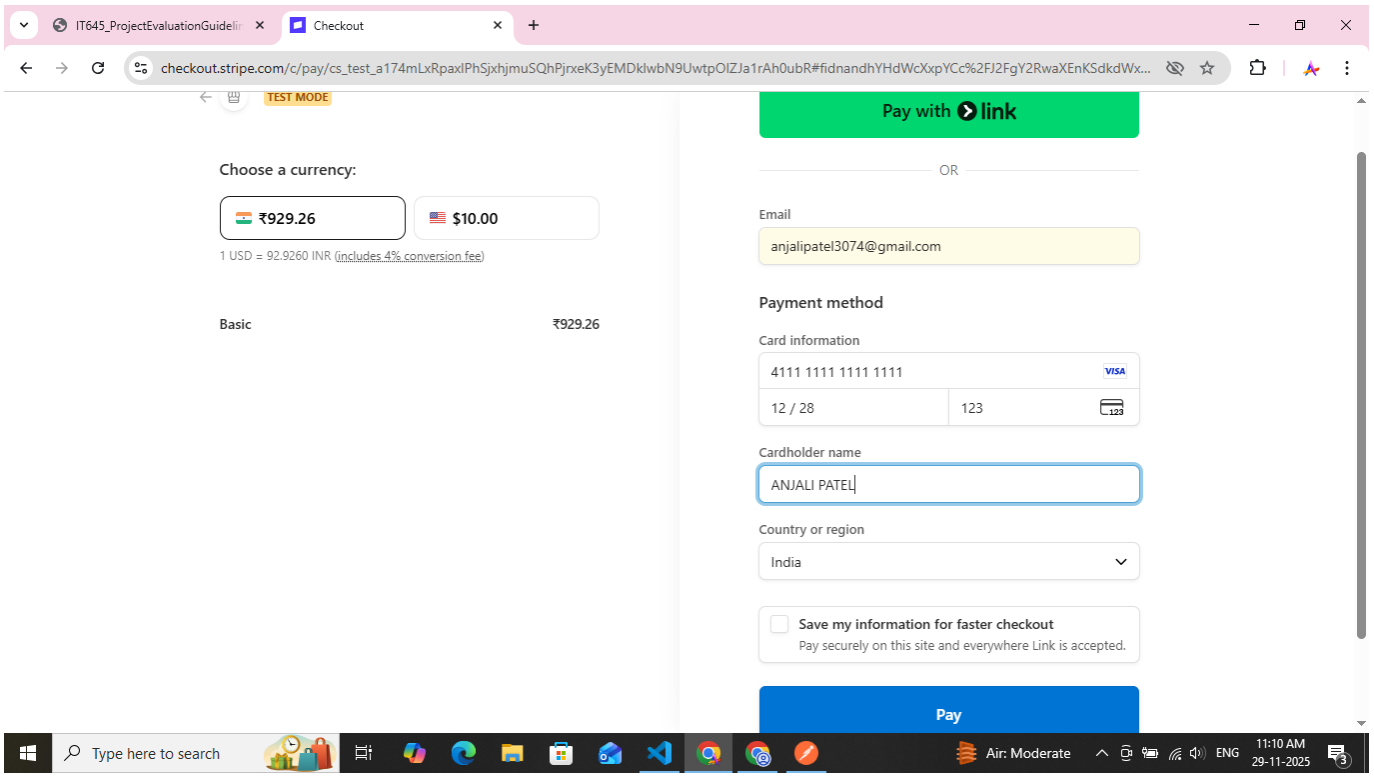
[16] get plans API 2



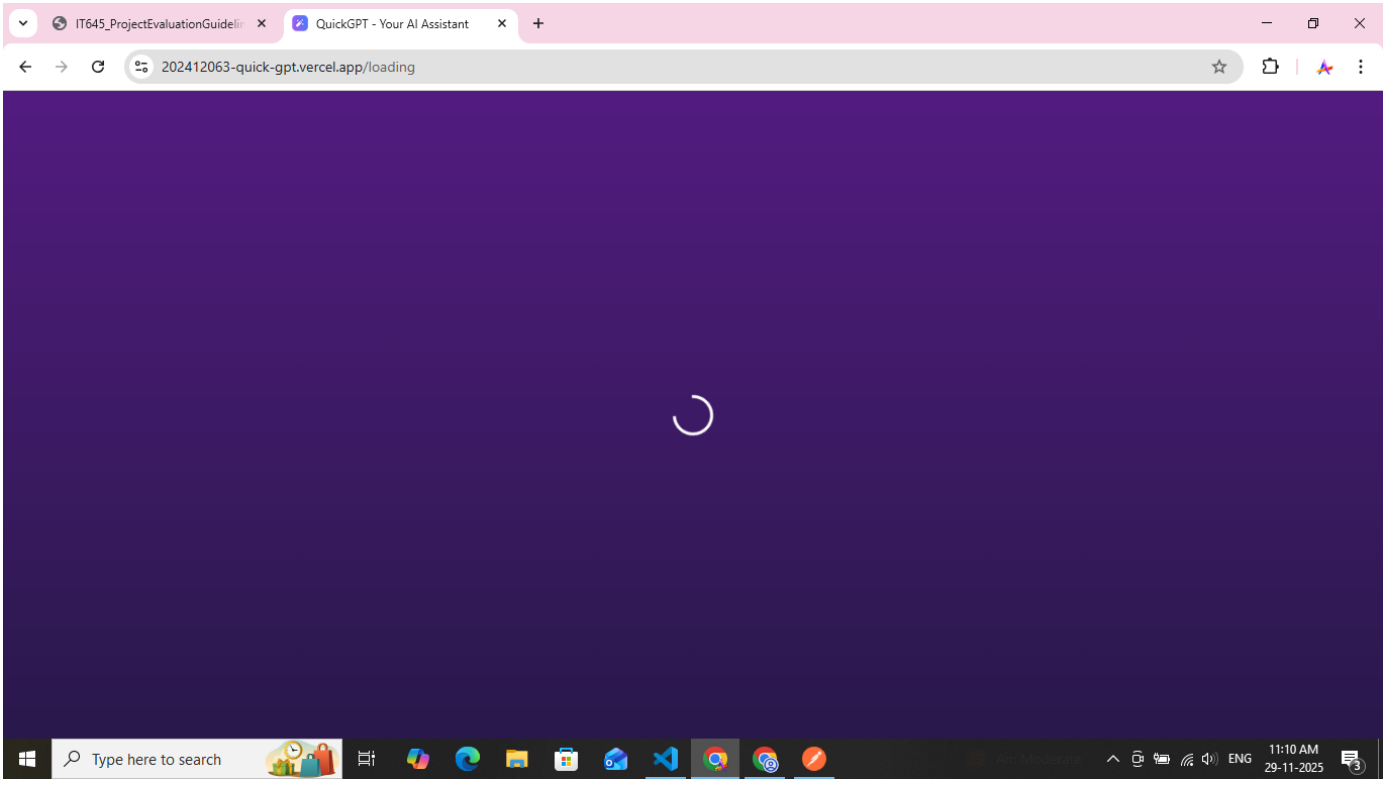
[17] purchase plan



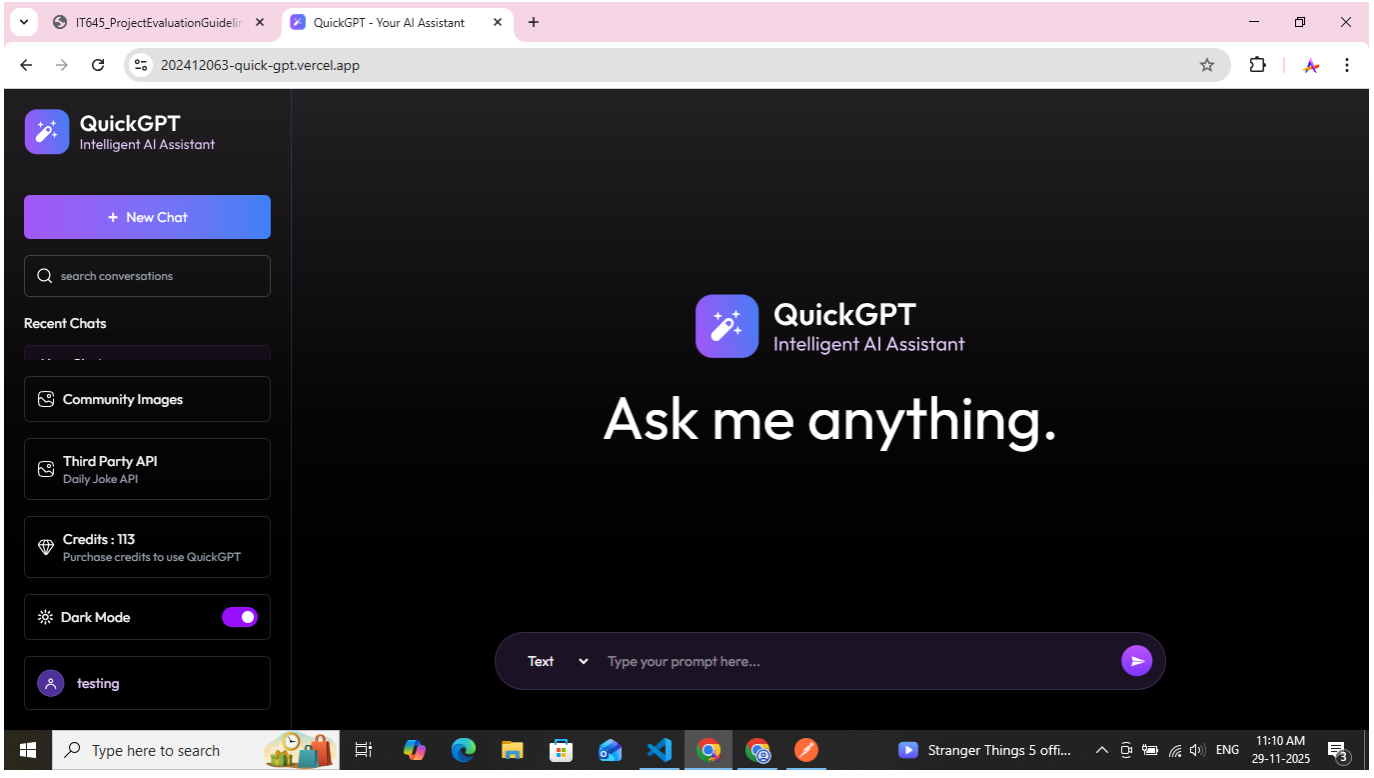
[18] payment processing



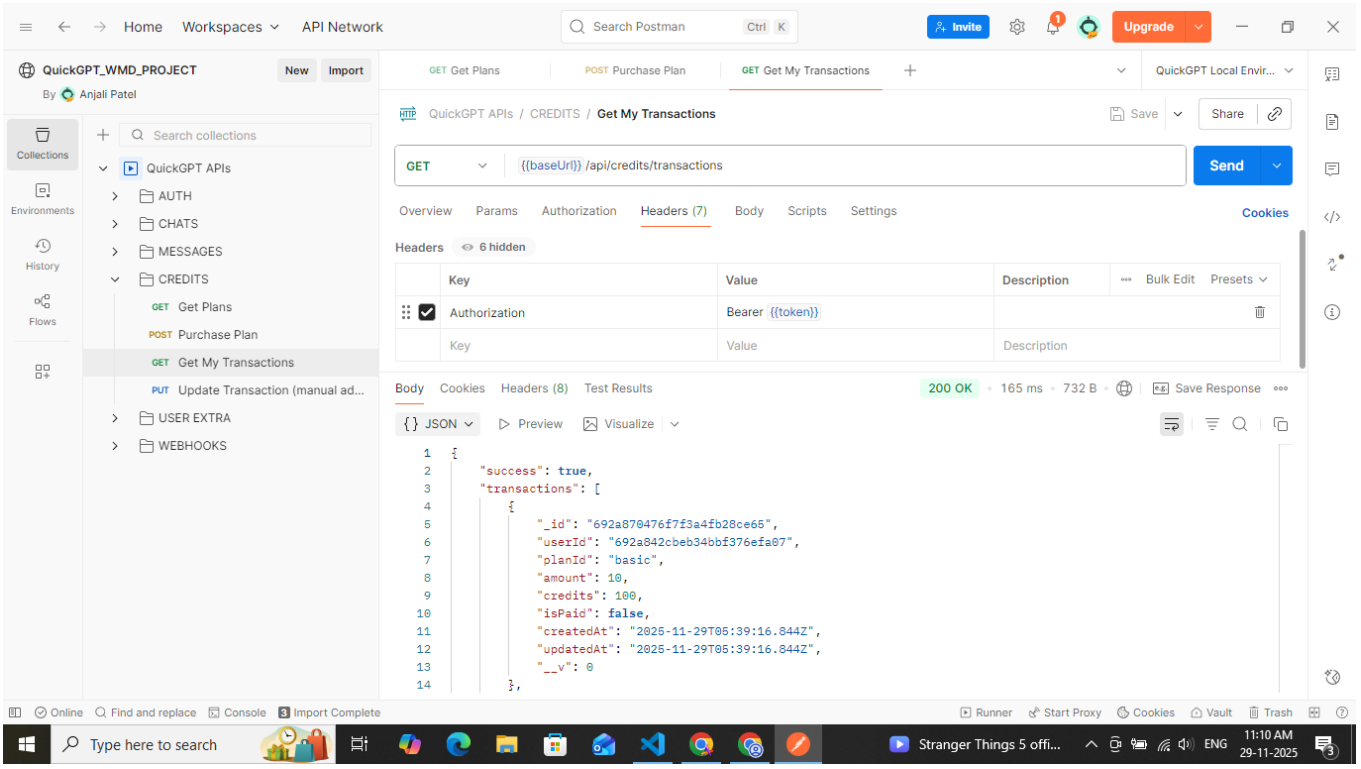
[19] redirected after sucessful payment



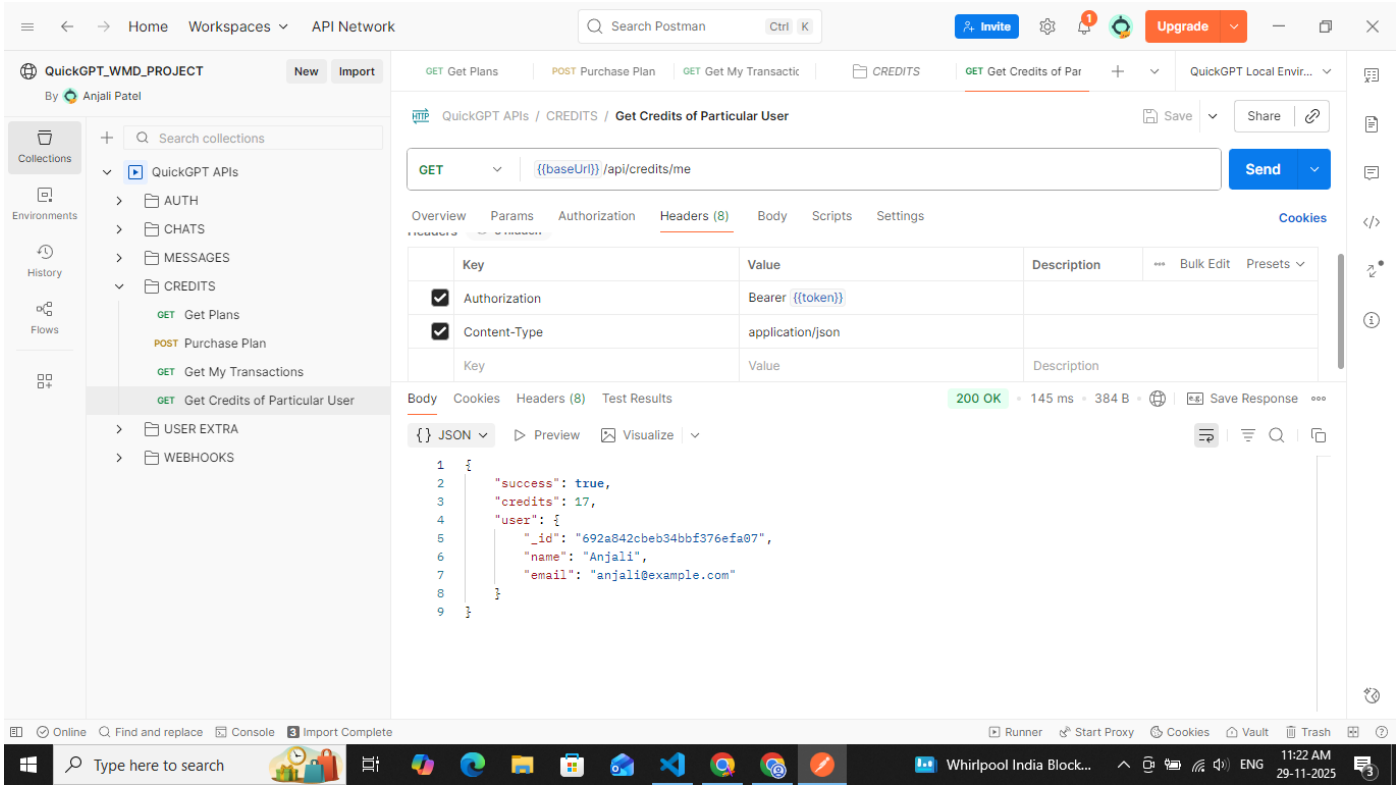
[20] updated credits



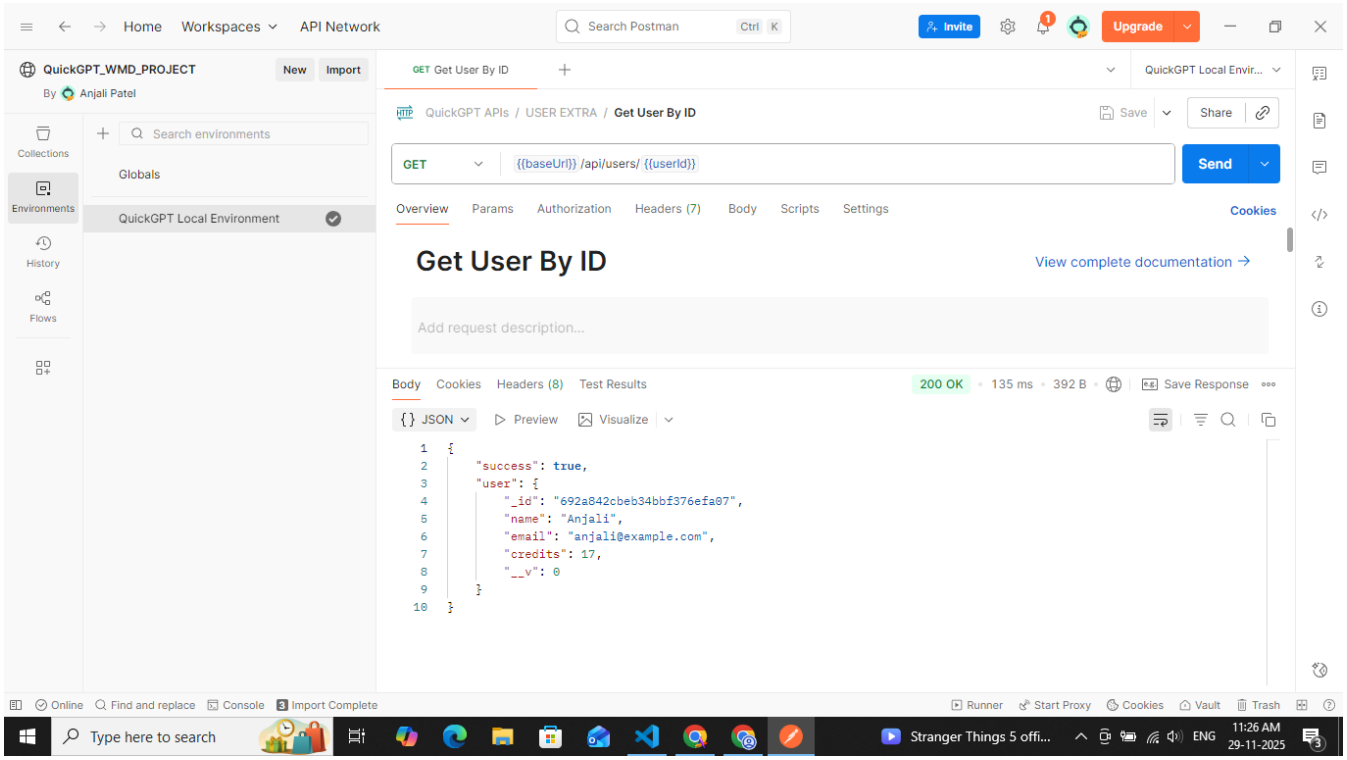
[21] get my transactions API



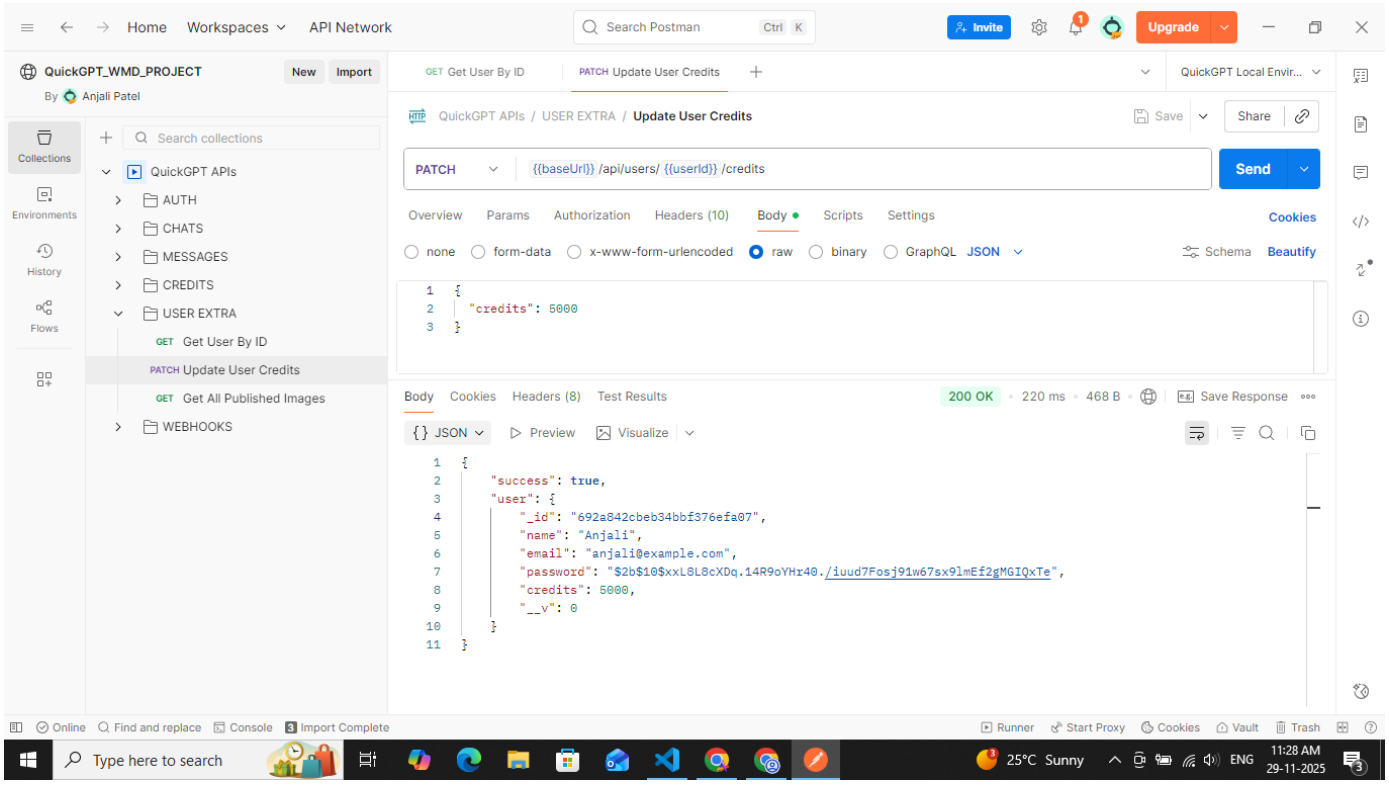
[22] get credits of particular user API



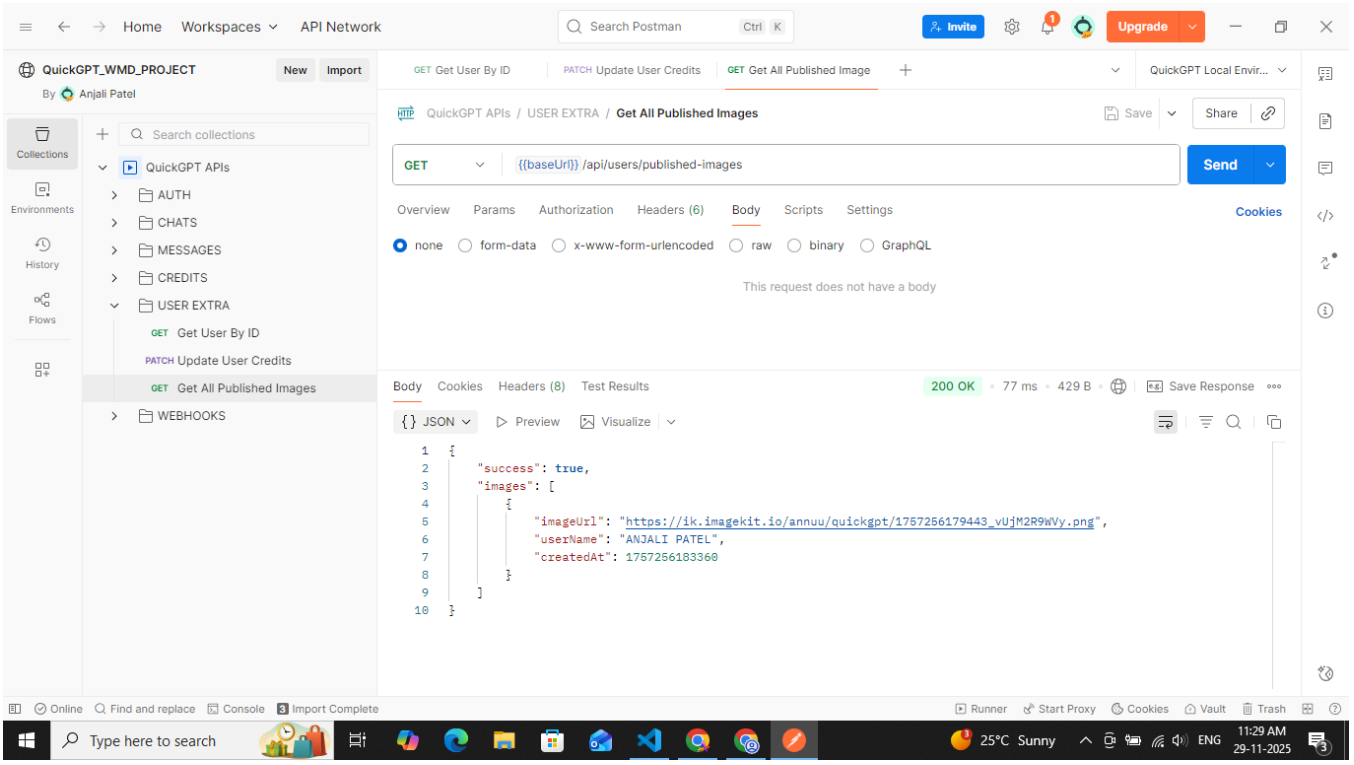
[23] get user by ID API



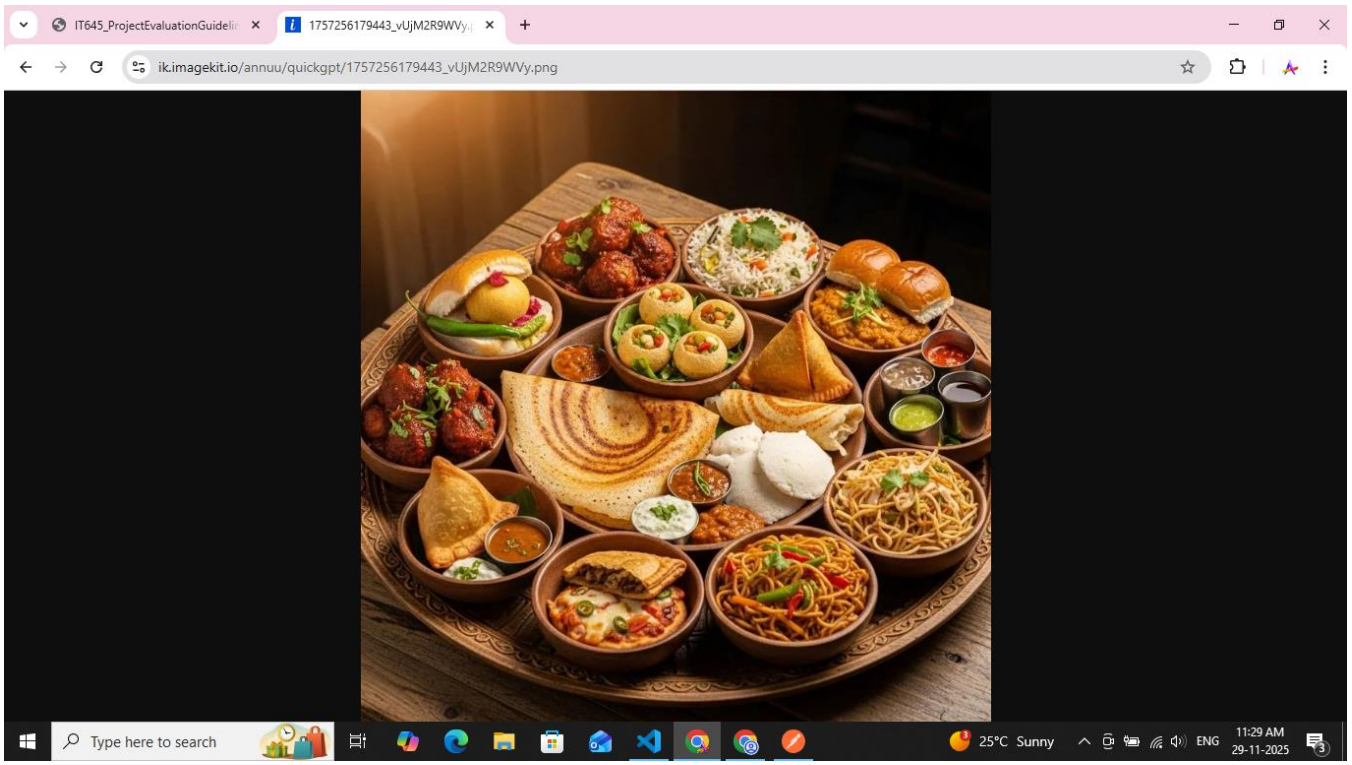
[24] update user credits API



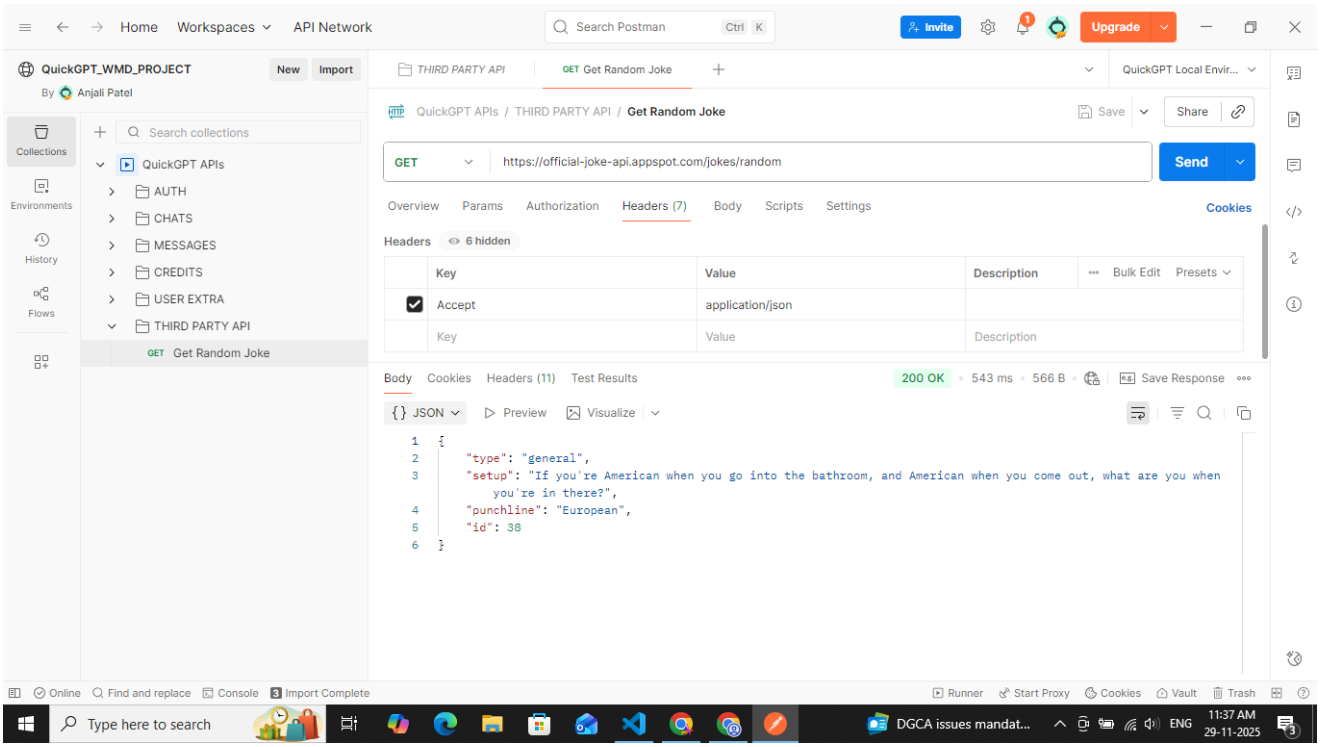
[25] get all published images API



[26] getting published image from link



[27] get random joke third party API



Justification : motivation, benefits, and need

QuickGPT is built to give users a simple and reliable way to use AI for text and image generation without dealing with complex setups or high costs. Moden AI tools are powerful but expensive, so there is a real need for a controlled, secure and scalable platform that offers these features in a practical way.

- **Motivation :**
  - Users want quick access to AI chat and image tools in one place
  - AI APIs cost money, so a credit-based model is necessary to prevent misuse
  - Developers need a real-world project that covers authentication, payments and API integration
  - Secure, fast and responsive wen apps are now standard expectations
- **Benefits :**
  - Unified platform for chat, image generation and community posts
  - Fair usage through credit-based billing
  - Safe login and secure payments using JWT and stripe
  - Scalable architecture using react, node.js, and mongoDB
  - Engaging experience through a community gallery
- **Need :**
  - MERN stack provides speed, flexibility and easy scaling
  - JWT enables clean, stateless authentication for API routes
  - Stripe handles payments safely without exposing sensitive data
  - Backend gatekeeping is essential to validate credits before calling expensive AI APIs
  - Proper architecture ensures performance, maintainability and controlled AI usage