

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

System Architecture

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 token
- AI answer generation and credits deducted
- Payments handled via stripe checkout
- MongoDB stores users, chats, messages, credits & transactions

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

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

▪ Chat

<u>Method</u>	<u>Endpoint</u>	<u>Description</u>
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

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

▪ Credits

<u>Method</u>	<u>Endpoint</u>	<u>Description</u>
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

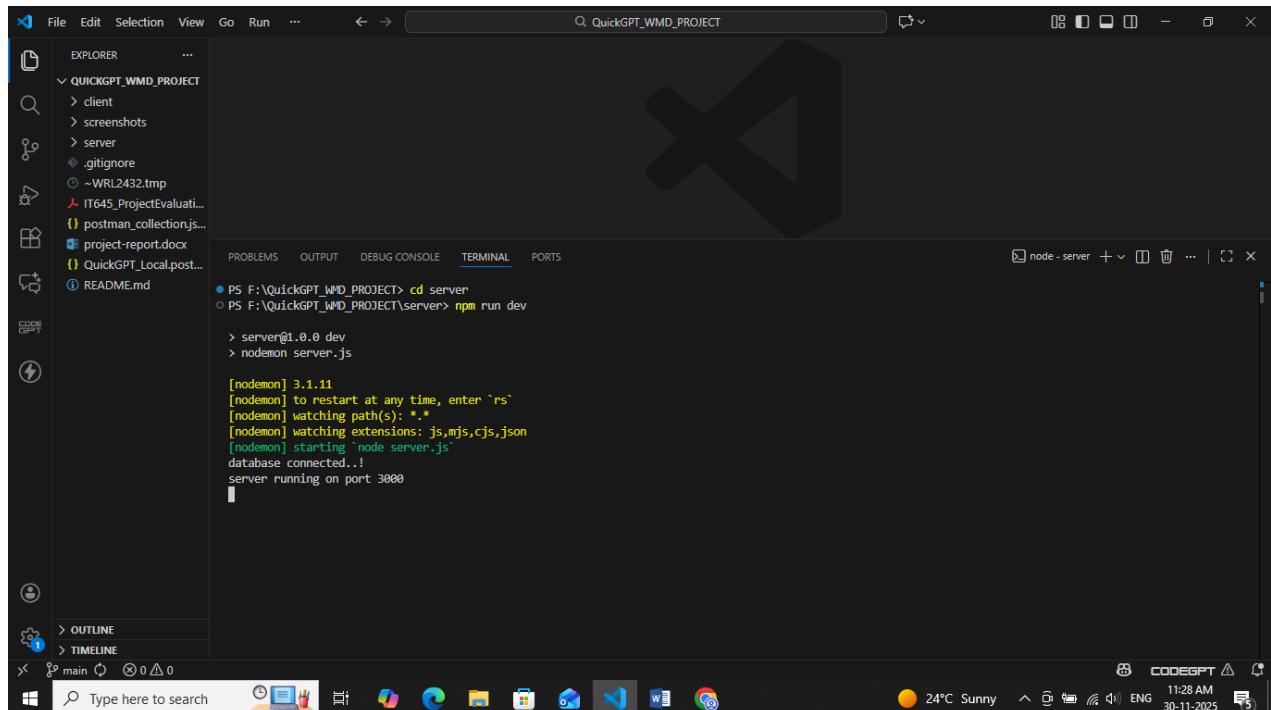
<u>Method</u>	<u>Endpoint</u>	<u>Description</u>
POST	/api/stripe/webhook	Raw webhook endpoint

▪ Third-Party API

<u>Method</u>	<u>Endpoint</u>	<u>Description</u>
GET	/api/jokes/random	Generate random joke

System Snapshots

[01] running backend



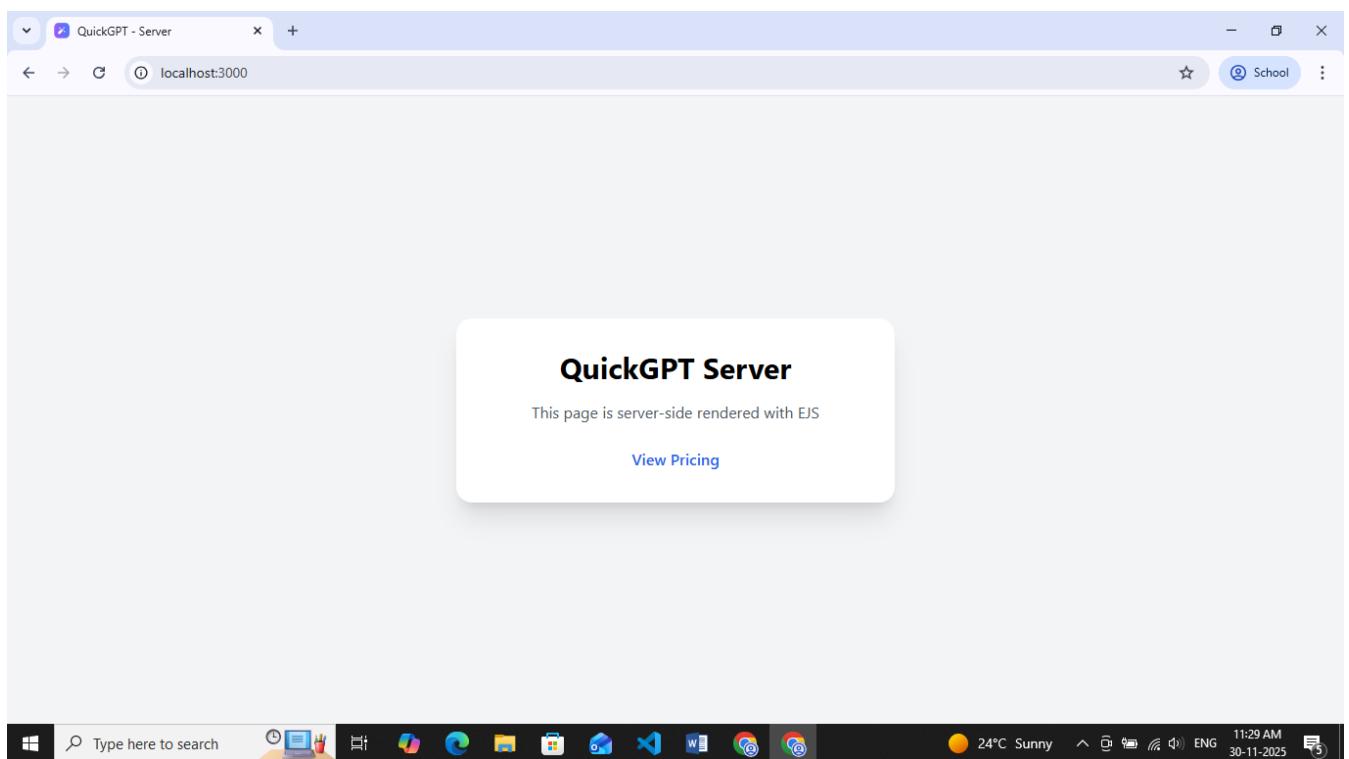
The screenshot shows the VS Code interface with the project 'QUICKGPT_WMD_PROJECT' open in the Explorer sidebar. The terminal tab is active, displaying the command-line output of the backend server starting:

```

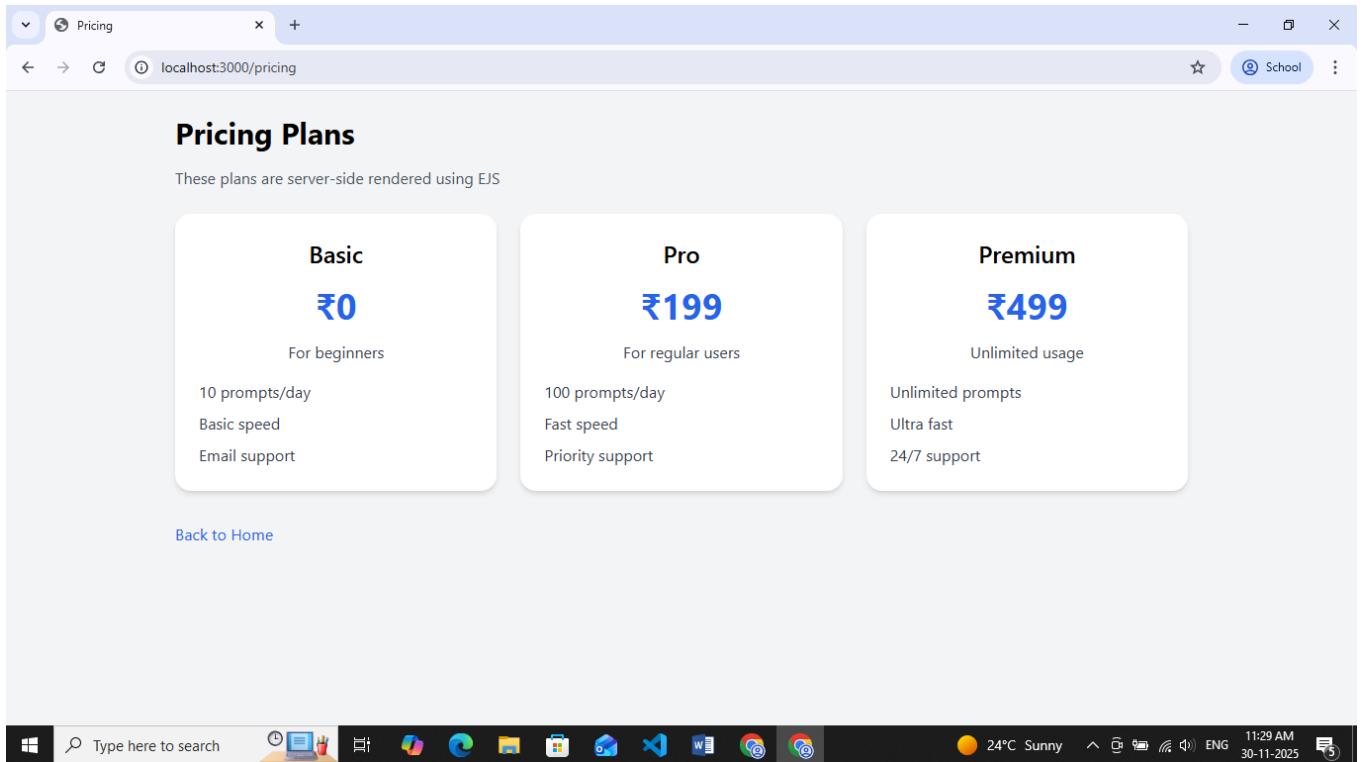
PS F:\QuickGPT_WMD_PROJECT> cd server
PS F:\QuickGPT_WMD_PROJECT\server> npm run dev
> server@1.0.0 dev
> nodemon server.js
[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
database connected..
server running on port 3000

```

[02] server-side rendering using EJS template engine



[03] server-side rendering pricing page



[04] logs

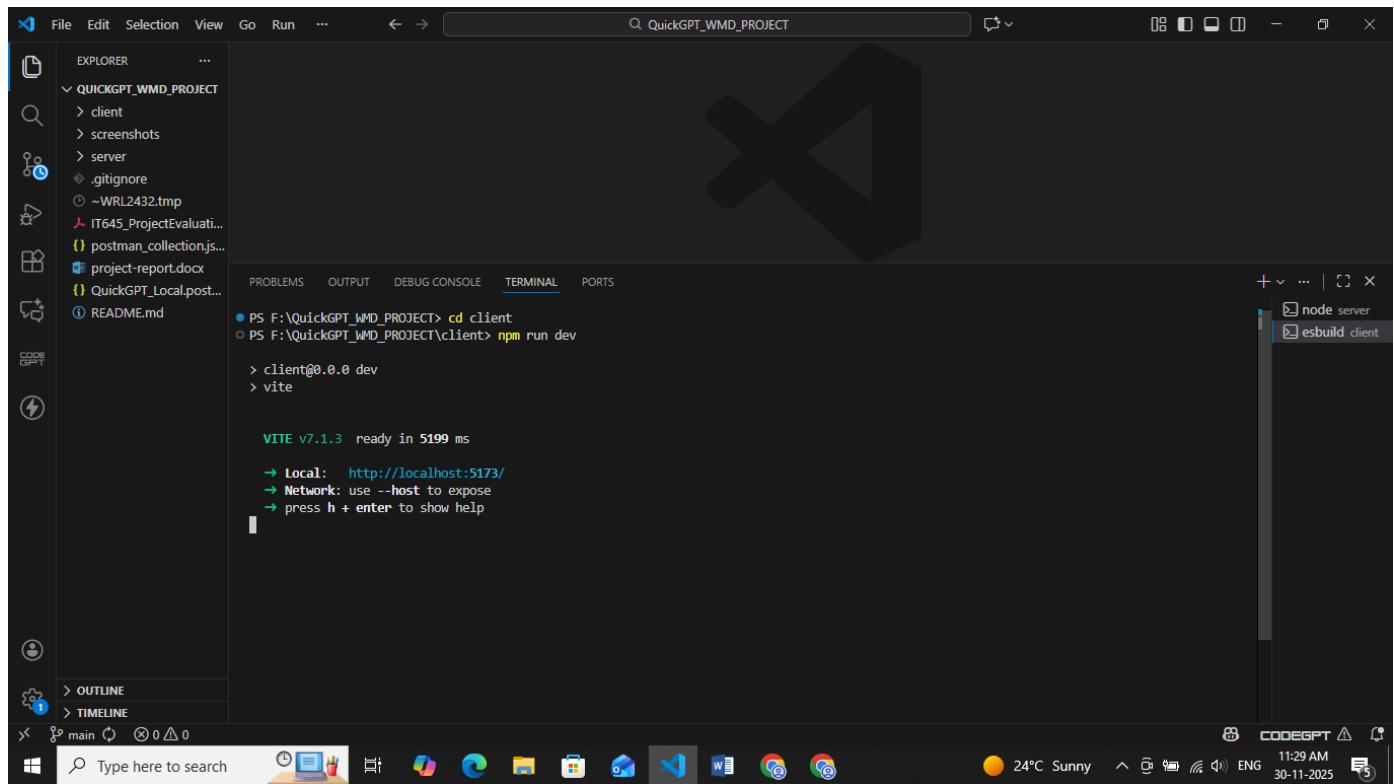
```

PS F:\QuickGPT_WMD_PROJECT> cd server
PS F:\QuickGPT_WMD_PROJECT\server> npm run dev
> server@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
database connected...
server running on port 3000
GET / 200 39.574 ms - 874
[2025-11-30T05:59:05.092Z] GET / 200 - 49ms
GET /pricing 200 3.873 ms - 3222
[2025-11-30T05:59:16.272Z] GET /pricing 200 - 6ms
GET /favicon.ico 404 1.777 ms - 1008
[2025-11-30T05:59:16.505Z] GET /favicon.ico 404 - 3ms
GET / 304 2.317 ms -
[2025-11-30T05:59:28.757Z] GET / 304 - 3ms

```

[05] running frontend



The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the following command-line session:

```

PS F:\QuickGPT_WMD_PROJECT> cd client
PS F:\QuickGPT_WMD_PROJECT\client> npm run dev

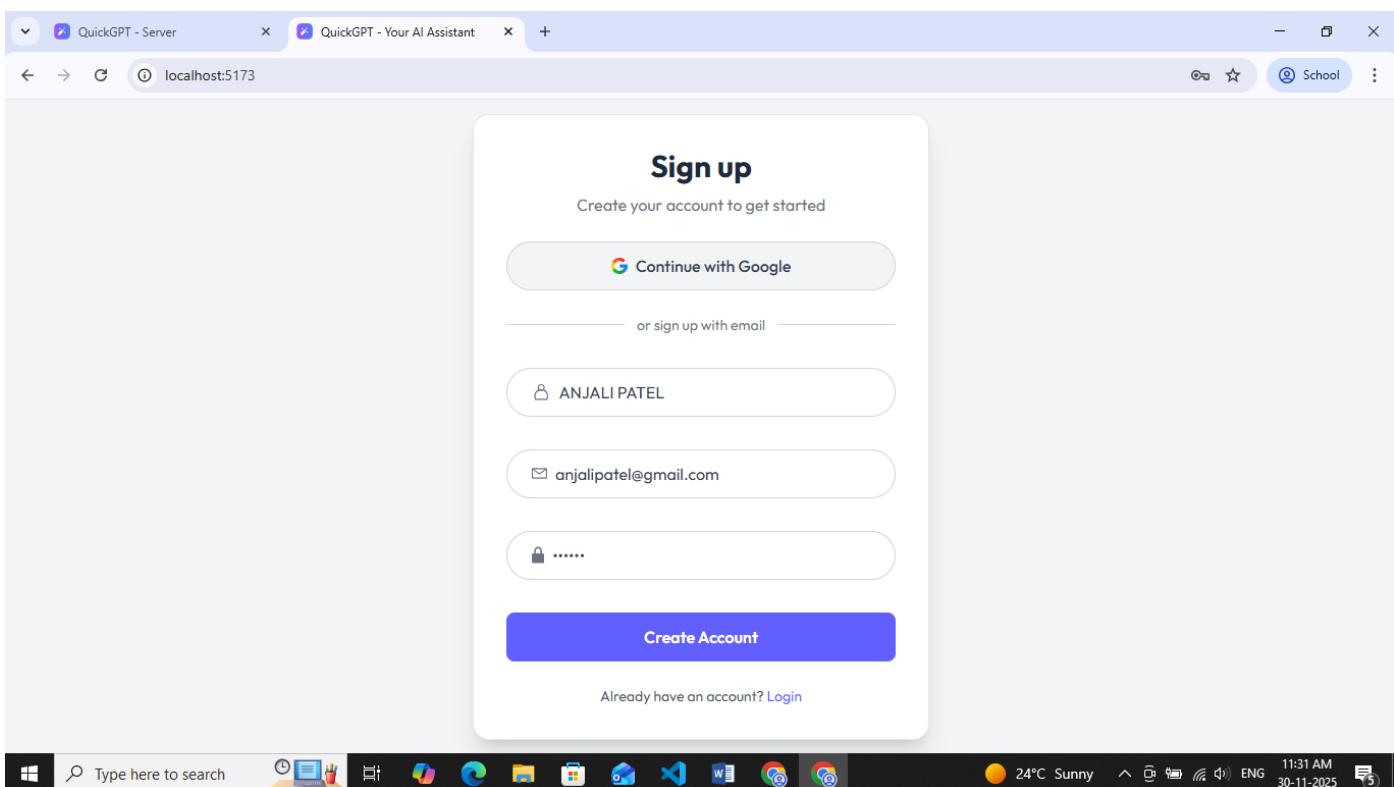
> client@0.0.0 dev
> vite

VITE v7.1.3 ready in 5199 ms

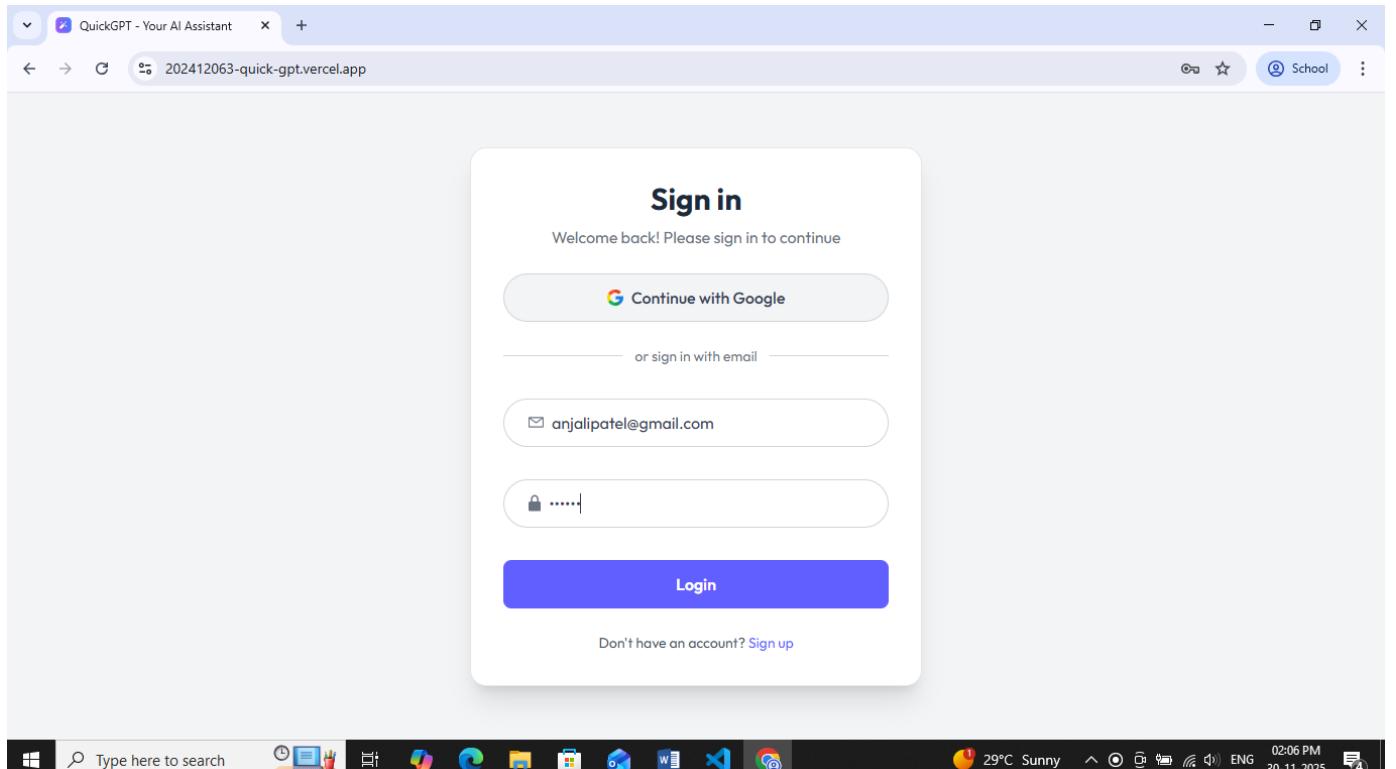
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
  
```

The Explorer sidebar on the left shows the project structure: QUICKGPT_WMD_PROJECT with client, screenshots, server, .gitignore, ~WR2432.tmp, IT645_ProjectEvaluation..., postman_collection.json, project-report.docx, QuickGPT_Local.post..., and README.md.

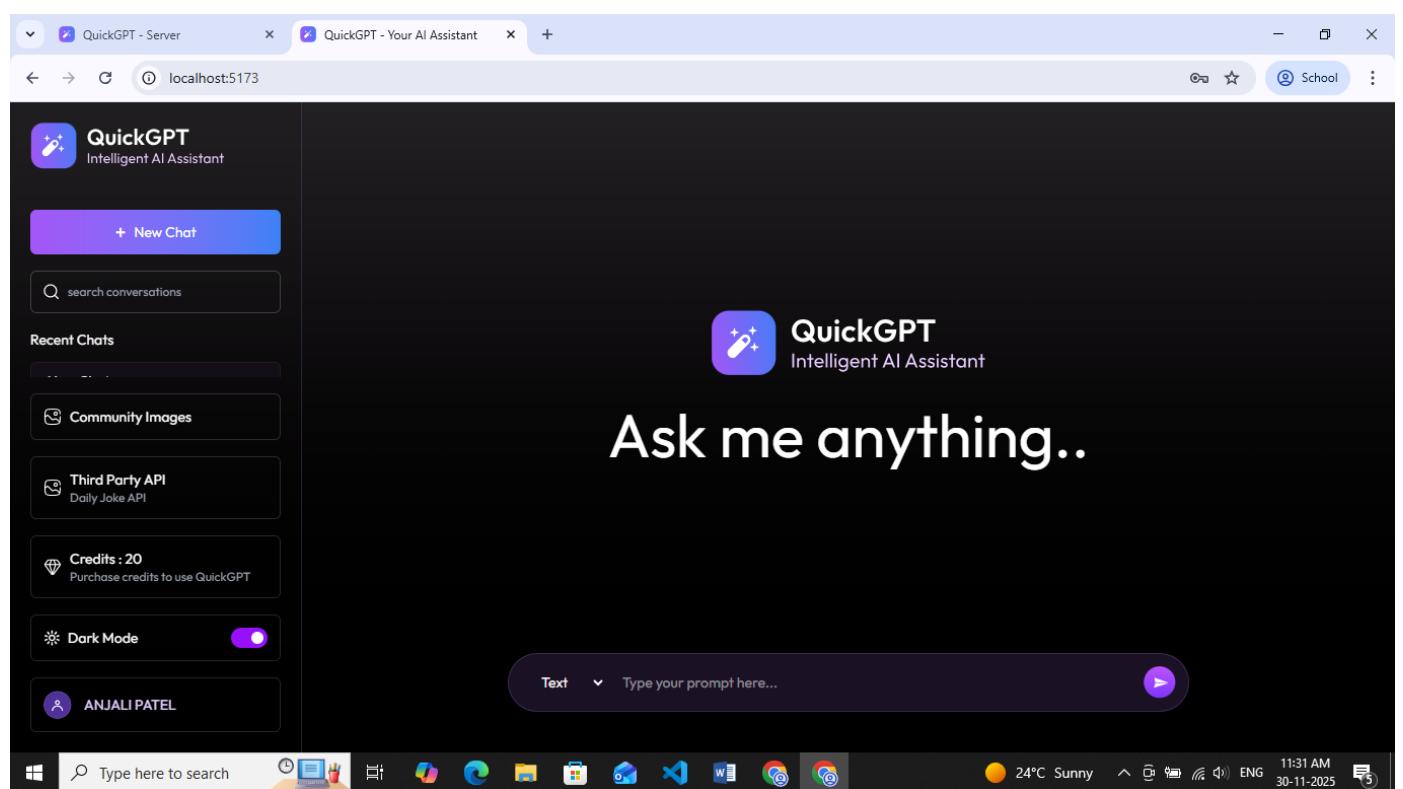
[06] registration page



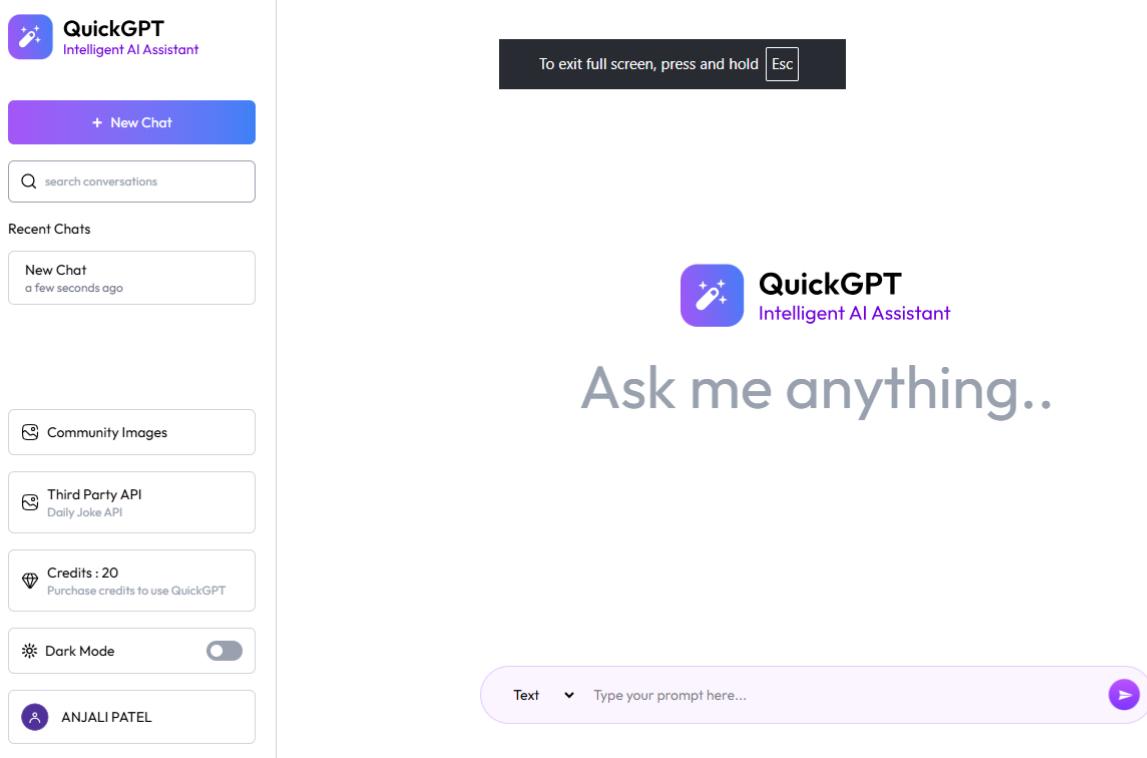
[07] login page



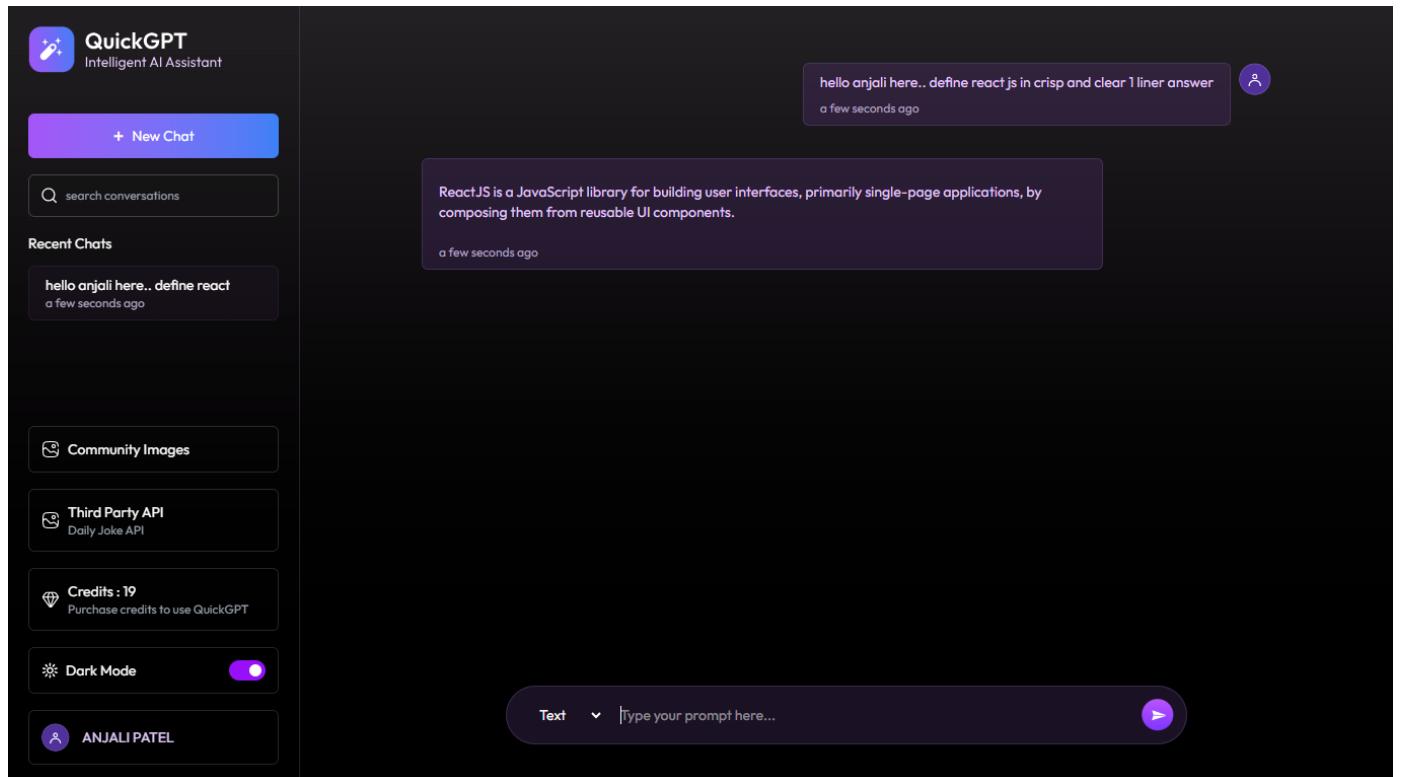
[08] home page



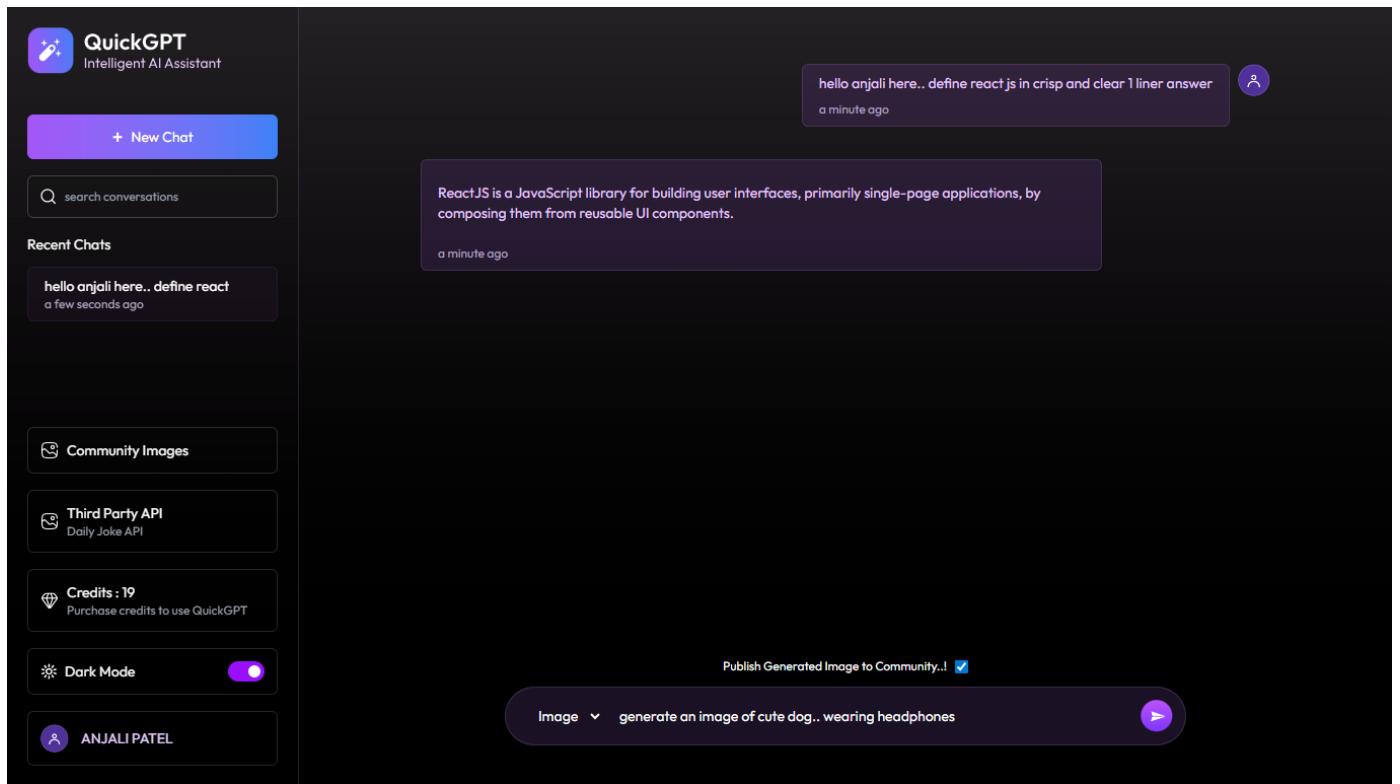
[09] home page in light mode



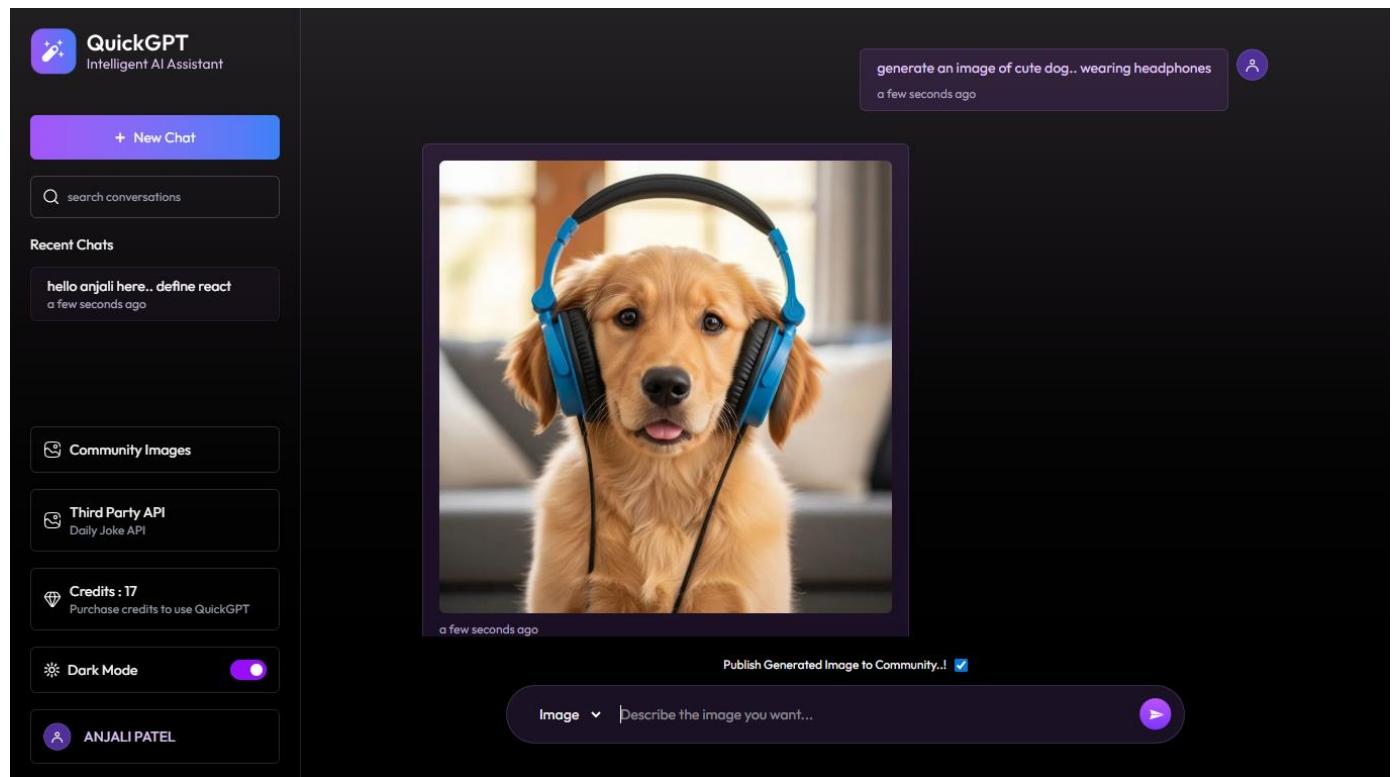
[10] text message prompt



[11] image generation prompt and adding image to community



[12] image generated



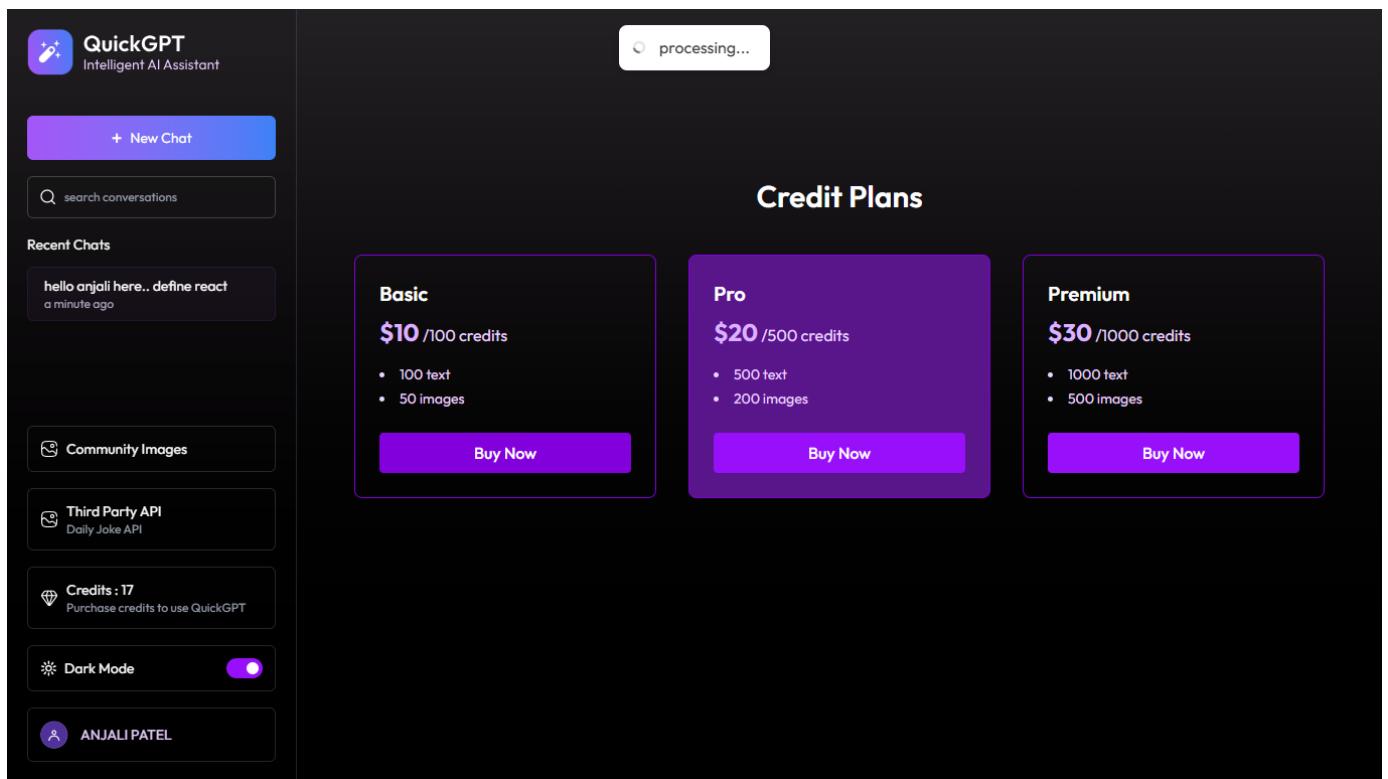
[13] community images

The screenshot shows the QuickGPT application interface. On the left sidebar, there are several buttons: '+ New Chat', 'search conversations', 'Recent Chats' (with a message 'hello anjali here.. define react'), 'Community Images', 'Third Party API', 'Credits : 17' (Purchase credits to use QuickGPT), 'Dark Mode' (switch is on), and 'ANJALI PATEL'. The main content area is titled 'Community Images' and shows two images: one of a dog wearing headphones and another of a variety of Indian food items.

[14] pre-defined credit plans

The screenshot shows the QuickGPT application interface. On the left sidebar, there are several buttons: '+ New Chat', 'search conversations', 'Recent Chats' (with a message 'hello anjali here.. define react'), 'Community Images', 'Third Party API', 'Credits : 17' (Purchase credits to use QuickGPT), 'Dark Mode' (switch is on), and 'ANJALI PATEL'. The main content area is titled 'Credit Plans' and displays three plans: 'Basic' (\$10 /100 credits, includes 100 text and 50 images), 'Pro' (\$20 /500 credits, includes 500 text and 200 images), and 'Premium' (\$30 /1000 credits, includes 1000 text and 500 images). Each plan has a 'Buy Now' button.

[15] initiate payment



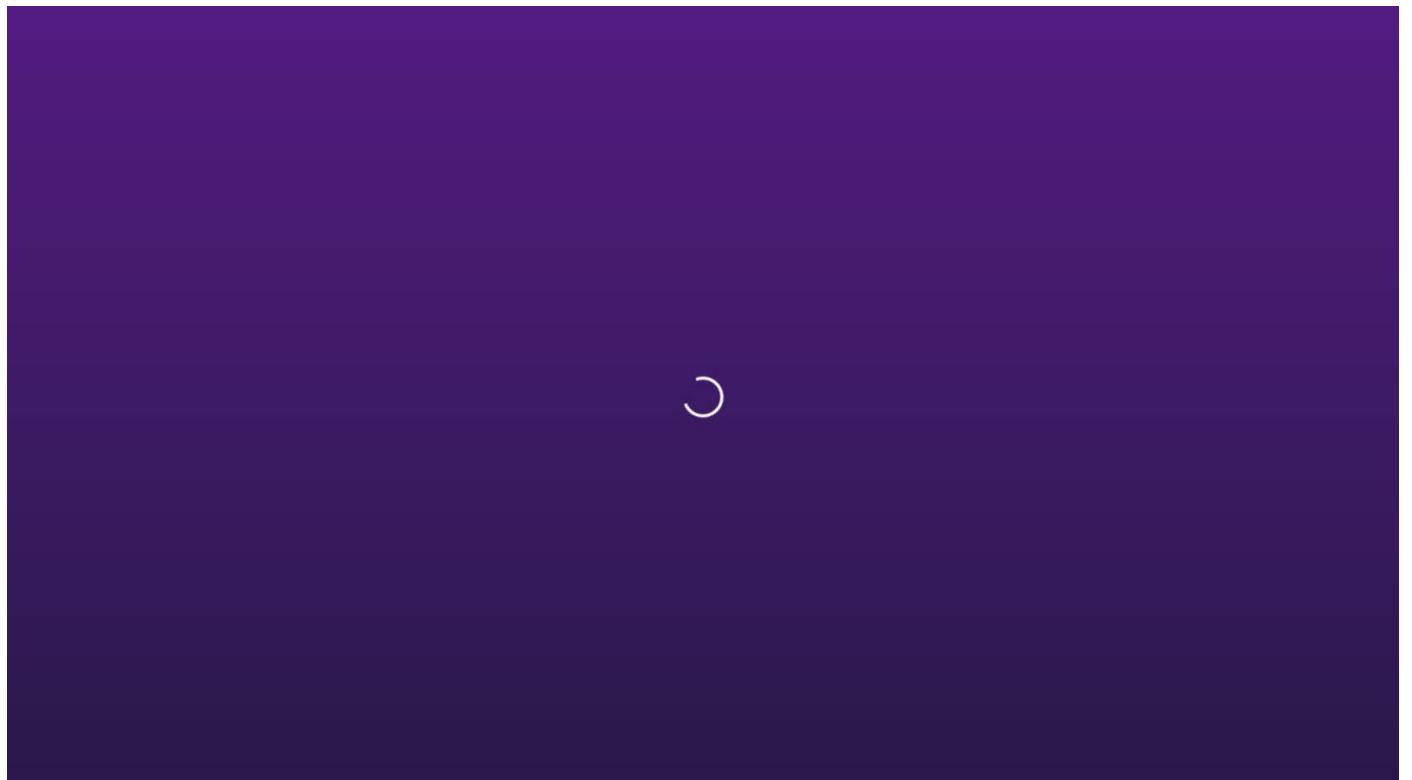
[16] stripe payment page with dummy details

The screenshot shows a Stripe payment page. At the top, there is a 'TEST MODE' button. Below it, a section for choosing a currency shows '₹929.27' (selected) and '\$10.00'. A note states '1 USD = 92.9270 INR (includes 4% conversion fee)'. To the right, a green button says 'Pay with link'. Below it, there is a 'Payment method' section with fields for 'Card information' (card number 4111 1111 1111 1111, expiration 12/28, CVC 123), 'Cardholder name' (ANJALI PATEL), 'Country or region' (India selected), and a checkbox for 'Save my information for faster checkout' (unchecked). At the bottom, a large blue 'Pay' button is visible.

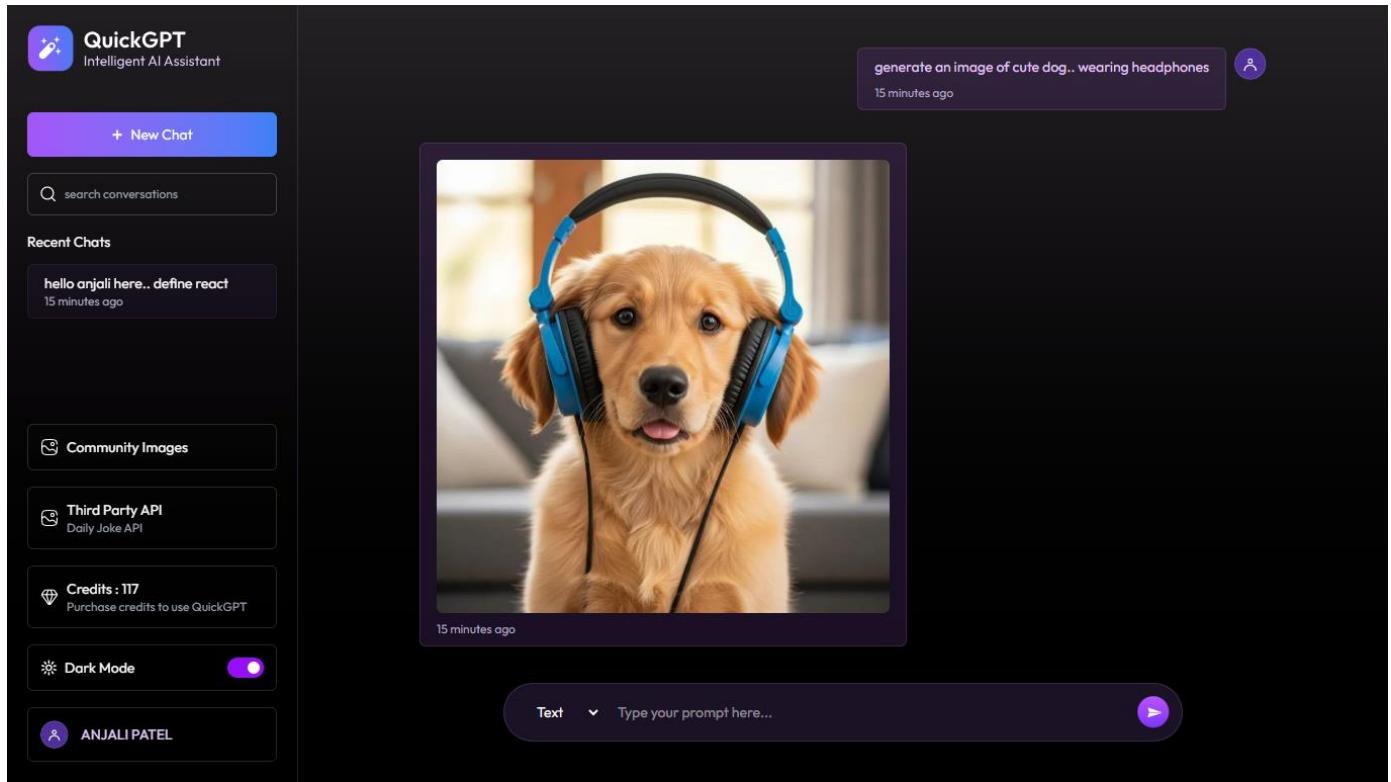
[17] payment successful

The screenshot shows a payment gateway interface. At the top left is a "TEST MODE" button. Below it, there's a "Choose a currency:" section with two options: ₹929.27 (selected) and \$10.00. A note below says "1 USD = 92.9270 INR (includes 4% conversion fee)". To the right, there's a green button labeled "Pay with link". Below it, there's an "OR" option followed by an "Email" field containing "anjalipatel3074@gmail.com". Under "Payment method", there's a "Card information" section with a sample card number "4111 1111 1111 1111", expiration date "12 / 28", and a CVC "123". The cardholder name is "ANJALI PATEL". Below that, the "Country or region" is set to "India". A "Save my information for faster checkout" button is present, along with a note: "Pay securely on this site and everywhere Link is accepted." At the bottom, a blue "Processing" button with a circular icon is shown.

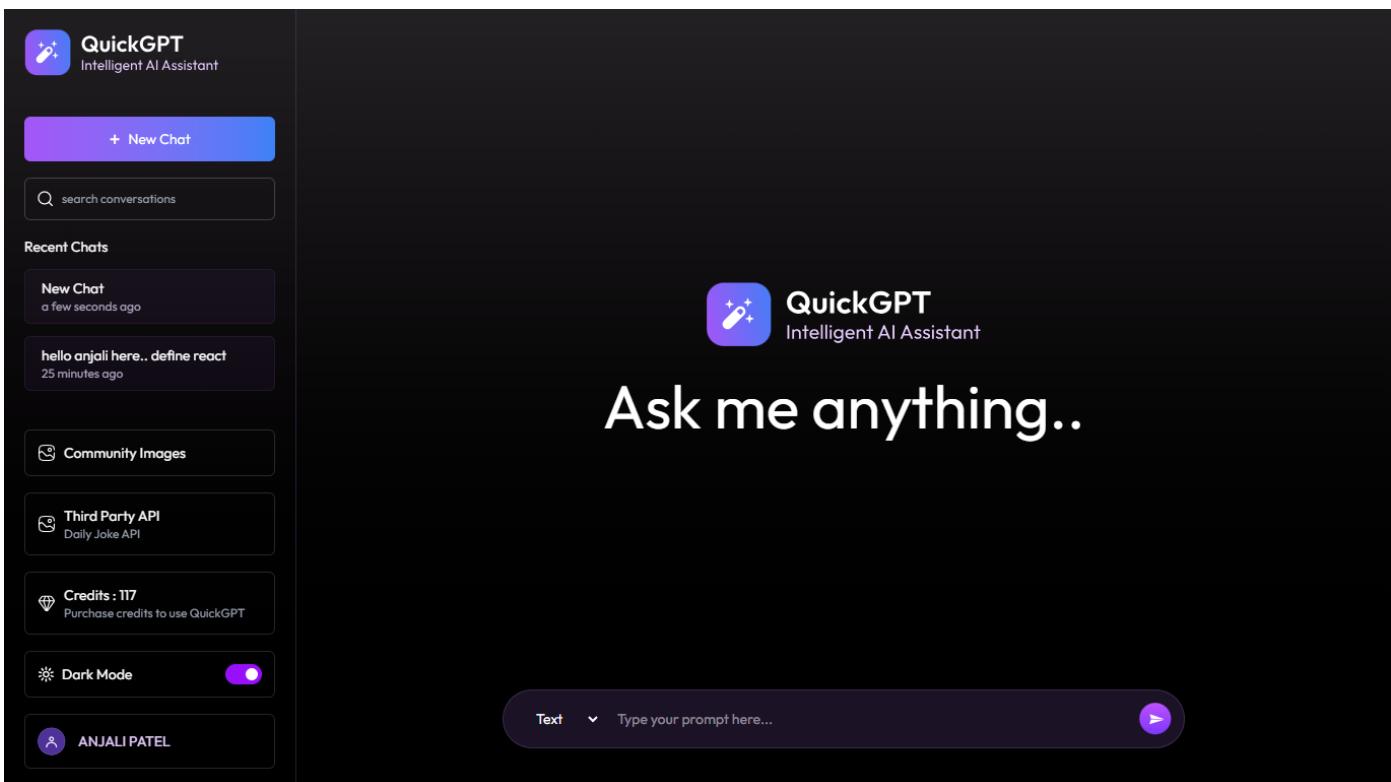
[18] redirection



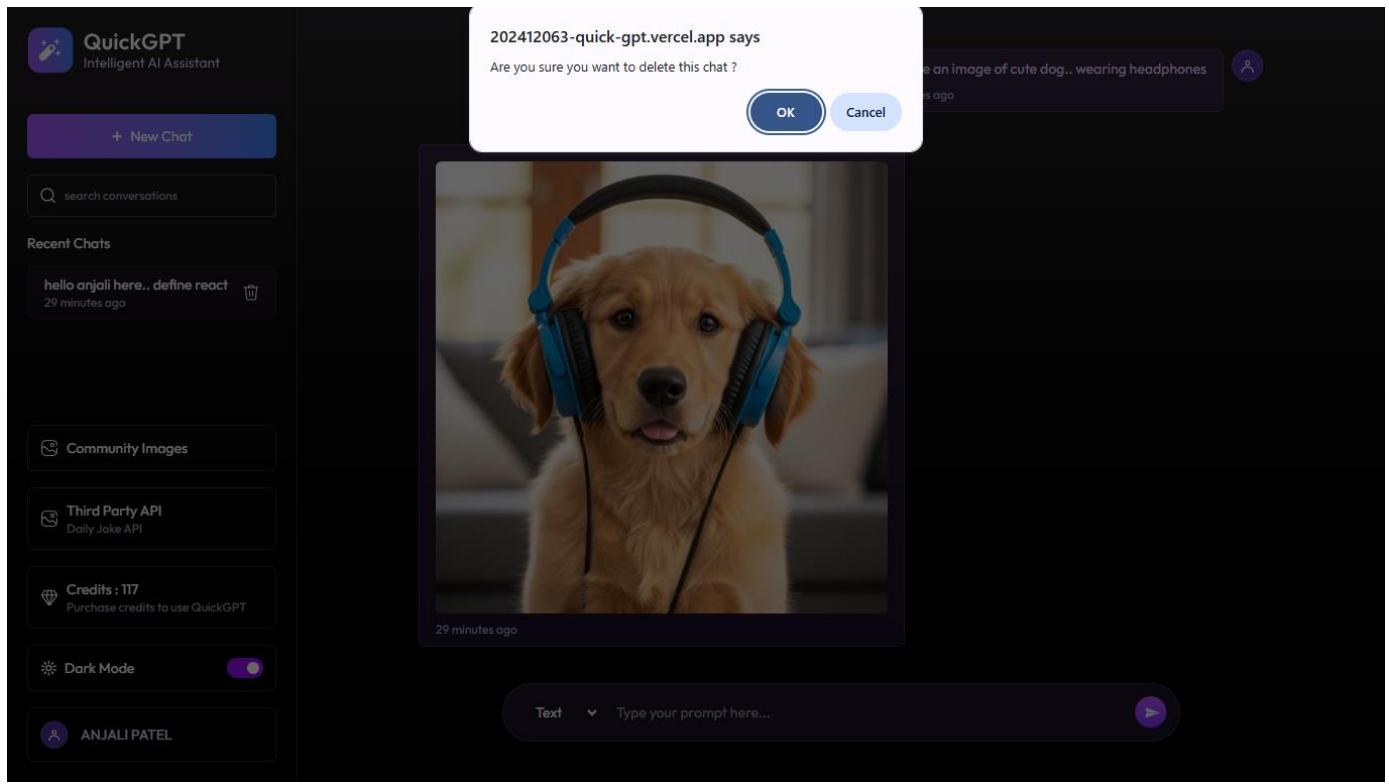
[19] updated credits



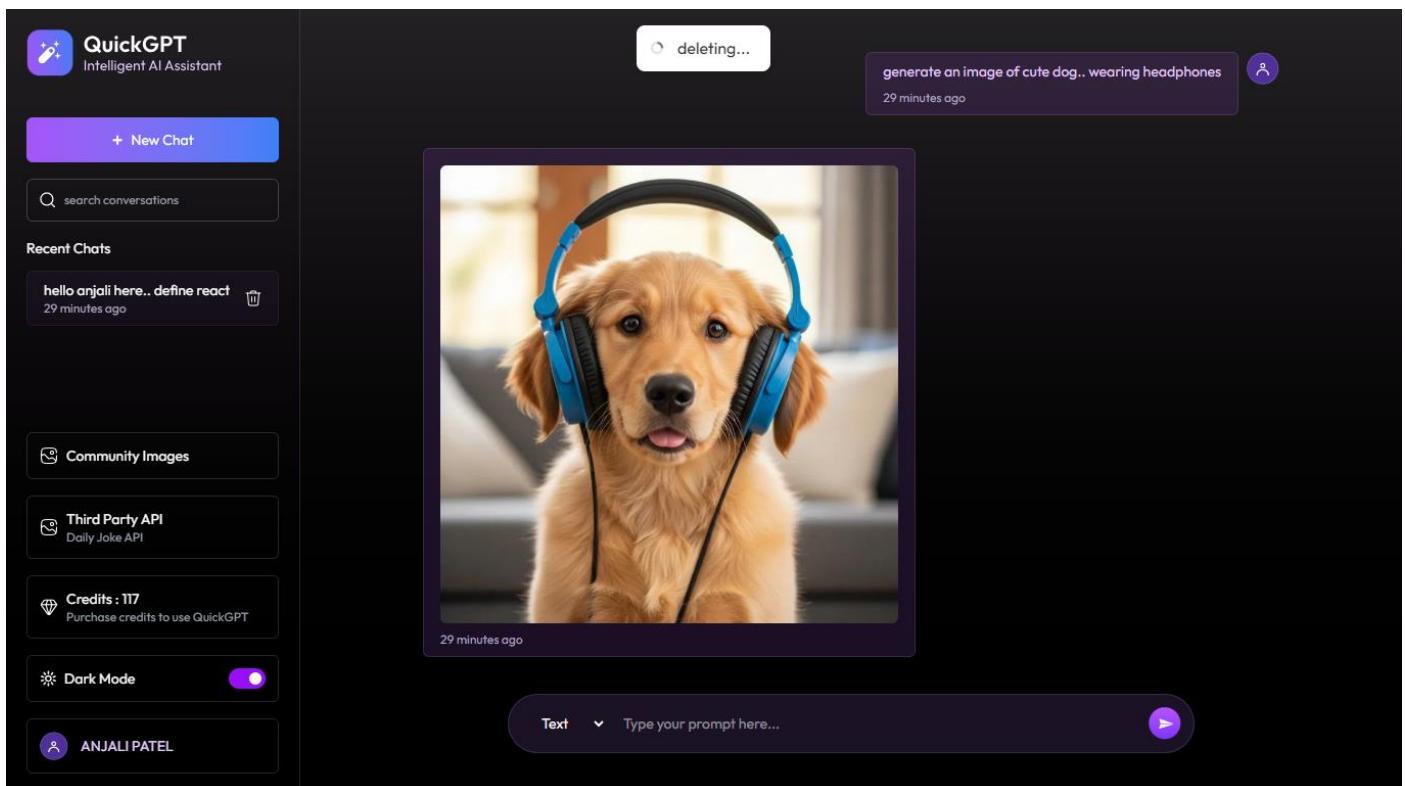
[20] new chat creation



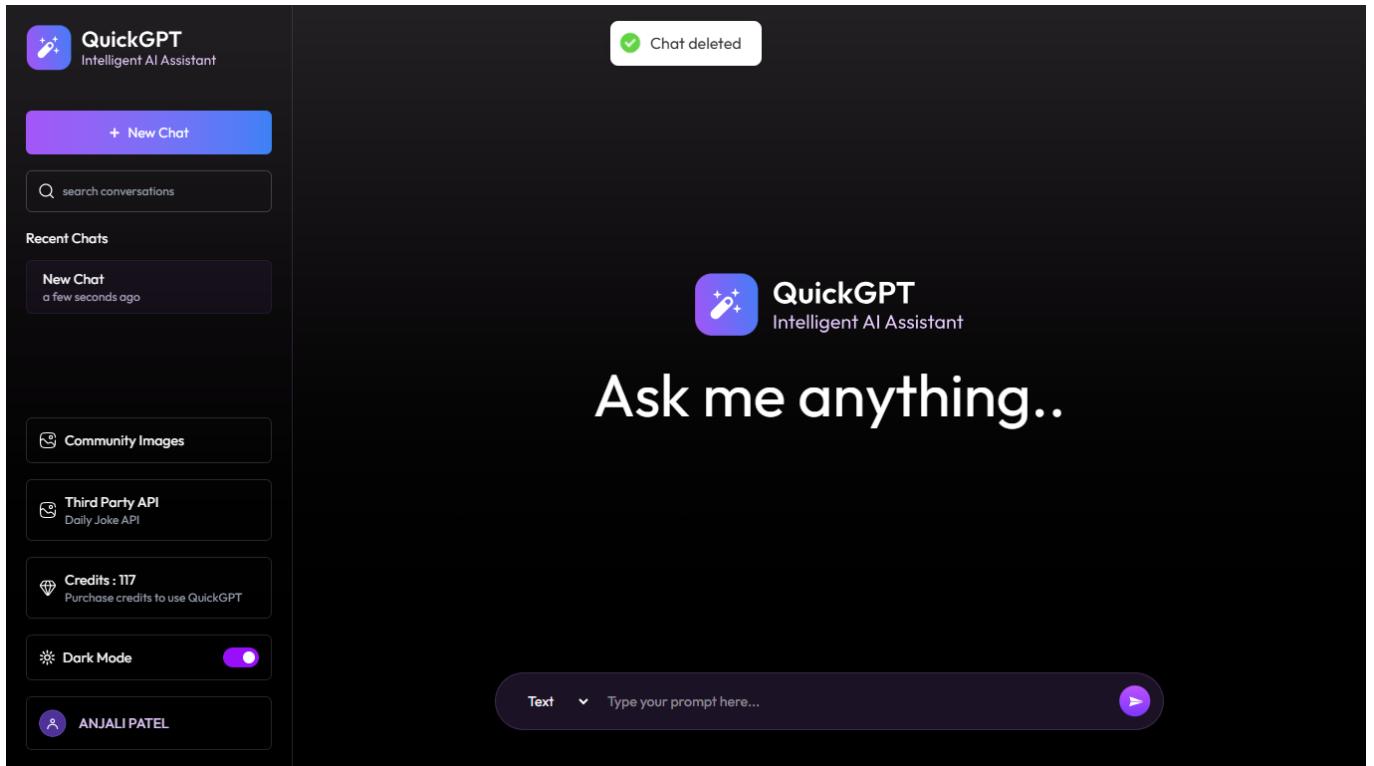
[21] delete chat



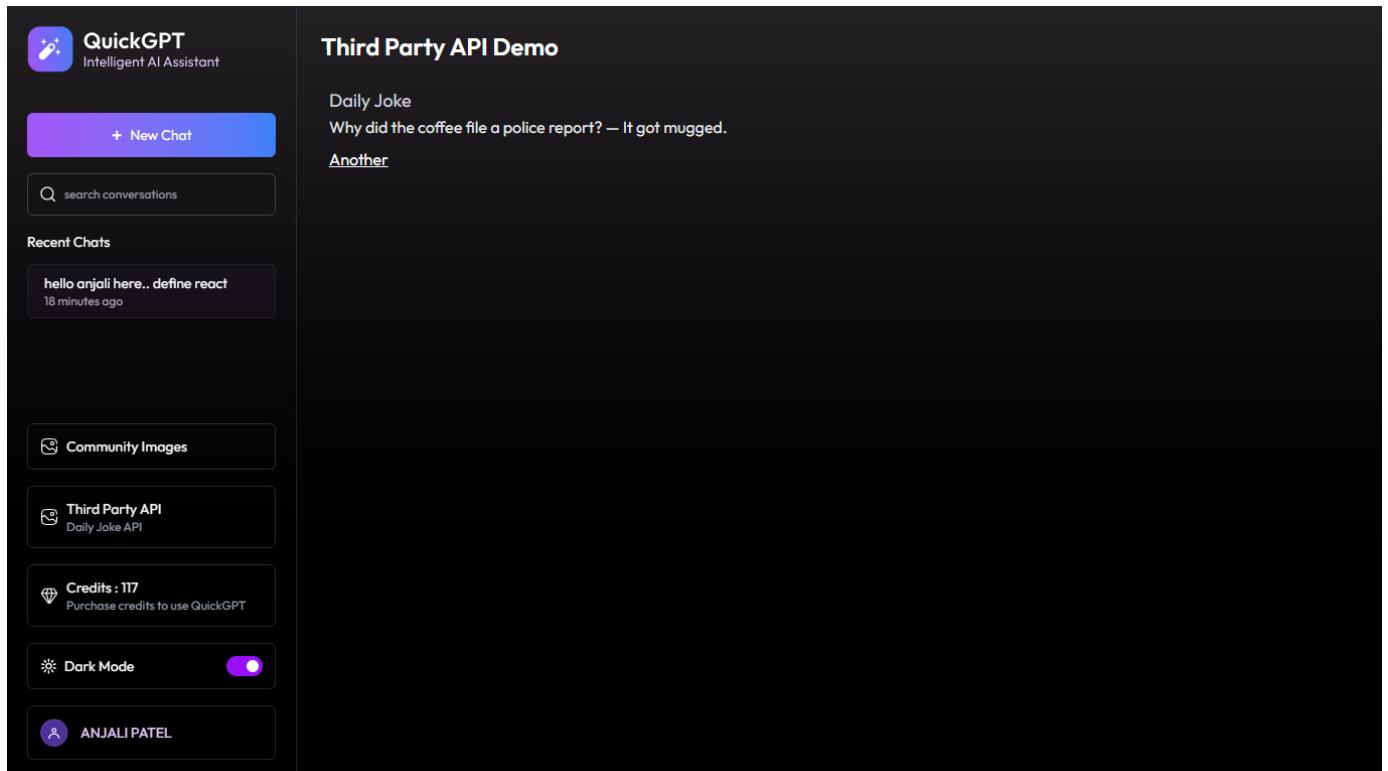
[22] deleting



[23] chat deleted



[24] third-party API



[25] random joke generation

The screenshot shows the QuickGPT AI chatbot interface. On the left sidebar, there's a logo for "QuickGPT Intelligent AI Assistant" and a purple button labeled "+ New Chat". Below that is a search bar with the placeholder "search conversations". Under "Recent Chats", there's a message from "ANJALI PATEL" that says "hello anjali here.. define react" sent 18 minutes ago. Other sidebar options include "Community Images", "Third Party API" (Daily Joke API), "Credits : 117" (Purchase credits to use QuickGPT), "Dark Mode" (with a toggle switch), and the user's name "ANJALI PATEL". The main content area has a title "Third Party API Demo" and a section titled "Daily Joke" with the text "How many seconds are in a year? – 12. January 2nd, February 2nd, March 2nd, April 2nd.... etc". There's also a link "Another".

[26] logout

The screenshot shows a sign-in page with a success message at the top: "Logged out successfully!". The main heading is "Sign in" with the sub-instruction "Welcome back! Please sign in to continue". It features a "Continue with Google" button and a "or sign in with email" link. Below this are fields for "Email id" and "Password", both with placeholder text. A large blue "Login" button is at the bottom. At the very bottom, there's a link "Don't have an account? Sign up".

Testing Snapshots

[01] register user API

The screenshot shows the Postman interface with the 'POST Register User' request selected. The 'Body' tab contains the following JSON payload:

```

1 {
2   "name": "Anjali",
3   "email": "anjali@example.com",
4   "password": "123456"
5 }

```

The 'Test Results' section shows a successful response with status code 201 Created, duration 319 ms, and size 572 B. The response body is:

```

1 {
2   "success": true,
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZC16IjY5MmE4ND0jYmVmzRiyMzNzLzIzEwNyIsImIhdCI6MTc2NDM5NDayOCwiZXhwIjoxNzY2OTg2MDI4fQ.
aSSNlW0j4puuf77i0LvKj0uASXQrODo1Y_wpgdtpNGA",
4   "user": {
5     "_id": "692a842cbbeb34bbf376efa07",
6     "name": "Anjali",
7     "email": "anjali@example.com",
8     "credits": 20
9   }
10 }

```

[02] login user API

The screenshot shows the Postman interface with the 'POST Login User' request selected. The 'Body' tab contains the following JSON payload:

```

1 {
2   "email": "anjali@example.com",
3   "password": "123456"
4 }

```

The 'Test Results' section shows a successful response with status code 200 OK, duration 230 ms, and size 466 B. The response body is:

```

1 {
2   "success": true,
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZC16IjY5MmE4ND0jYmVmzRiyMzNzLzIzEwNyIsImIhdCI6MTc2NDM5NDayOCwiZXhwIjoxNzY2OTg2MDQ0fQ.
f2FQ2Rwx0Xsw6JxIRlI6qEkMRMu3FFhuozrVBY3Iugs"
4 }

```

[03] adding token to environment variable

The screenshot shows the Postman interface with the 'Environments' tab selected. A modal window titled 'QuickGPT Local Environment' displays a table of environment variables. One variable, 'token', is highlighted and expanded to show its value as a long, encoded string. A callout bubble points to this expanded view.

Variable	Value
port	3000
baseURL	http://localhost:3000
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9eyJpZCI6IjY5MmE4NDJjYmVlM2RiYmYzNzZlZmEwNyIsImh0dC16MTc2NDM5NDA0NCwiZXhwIjoxNzY2OTg2MDQ0fQ.f2FQ2RwxbXsw6JxIRl6qEkMRMu3FFhrudrVBY3Iugsl
chatId	
userId	
transactionId	
Add variable	

[04] get logged-in user API

The screenshot shows the Postman interface with the 'GET Get Logged-in User' request selected. The 'Headers' tab is active, showing an 'Authorization' header with the value 'Bearer {{(token)}}'. The 'Body' tab shows a JSON response with the following structure:

```

1  {
2    "success": true,
3    "user": {
4      "_id": "692a842cbeb34bbf376efa07",
5      "name": "Anjali",
6      "email": "anjali@example.com",
7      "credits": 20,
8      "__v": 0
9    }
10  }
  
```

[05] create chat API

The screenshot shows the Postman interface for a 'QuickGPT_WMD_PROJECT' workspace. On the left, the 'Environments' section is expanded, showing 'QuickGPT Local Environment' selected. In the center, a 'POST Create Chat' request is being tested against 'QuickGPT APIs / CHATS / Create Chat'. The 'Headers' tab is active, containing two entries: 'Authorization: Bearer {{(token)}}' and 'Content-Type: application/json'. The 'Body' tab shows a JSON response with a success status and a new chat object. The response body is:

```

1  {
2     "success": true,
3     "chat": {
4         "userId": "692a842cbeb34bbf376efa07",
5         "userName": "Anjali",
6         "name": "New Chat",
7         "messages": [],
8         "_id": "692a84a3beb34bbf376efa0c",
9         "createdAt": "2025-11-29T05:29:07.883Z",
10        "updatedAt": "2025-11-29T05:29:07.883Z",
11        "__v": 0
12    }
13 }

```

The status bar at the bottom indicates a '201 Created' response with a duration of 135 ms and a size of 505 B.

[06] adding chatID-to environment variable

The screenshot shows the Postman interface for the same workspace. The 'Environments' section is expanded, showing 'QuickGPT Local Environment' selected. A modal window titled 'QuickGPT Local Environment' is open, displaying a table of environment variables. One row for 'chatId' is highlighted with a blue border. A callout arrow points from the text 'What's changed for variables' to this row. The modal also contains three informational boxes: 'Local by default', 'Initial Shared values', and 'Mark as sensitive'. Below the modal, the main environment table shows the following variables:

Variable	Value
port	3000
baseUrl	http://localhost:3000
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC16IjY5MmE4NDJjYmVmZiRyMjZ...
chatId	692a84a3beb34bbf376efa0c
userId	
transactionId	
Add variable	

[07] get all chats API

The screenshot shows the Postman interface with the 'QuickGPT_WMD_PROJECT' workspace selected. On the left sidebar, under 'Collections', the 'CHATS' collection is expanded, showing the 'GET Get All Chats' endpoint. The main request panel displays a GET request to `"/api/chats"`. The 'Headers' tab is active, containing a single header: 'Authorization: Bearer {{token}}'. The 'Body' tab shows a JSON response with a success status and a list of chats. The 'Test Results' tab shows a 200 OK response with 99 ms duration and 503 B size.

```

1 {
2   "success": true,
3   "chats": [
4     {
5       "_id": "692a84a3beb34bbf376efa0c",
6       "userId": "692a842cbeb34bbf376efa07",
7       "userName": "Anjali",
8       "name": "New Chat",
9       "messages": [],
10      "createdAt": "2025-11-29T05:29:07.883Z",
11      "updatedAt": "2025-11-29T05:29:07.883Z",
12      "_v": 0
13    }
14  ]

```

[08] get chat by ID

The screenshot shows the Postman interface with the 'QuickGPT_WMD_PROJECT' workspace selected. On the left sidebar, under 'Collections', the 'CHATS' collection is expanded, showing the 'GET Get Chat By ID' endpoint. The main request panel displays a GET request to `"/api/chats/{chatId}"`. The 'Headers' tab is active, containing a single header: 'Authorization: Bearer {{token}}'. The 'Body' tab shows a JSON response with a success status and a single chat object. The 'Test Results' tab shows a 200 OK response with 119 ms duration and 500 B size.

```

1 {
2   "success": true,
3   "chat": {
4     "_id": "692a84a3beb34bbf376efa0c",
5     "userId": "692a842cbeb34bbf376efa07",
6     "userName": "Anjali",
7     "name": "New Chat",
8     "messages": [],
9     "createdAt": "2025-11-29T05:29:07.883Z",
10    "updatedAt": "2025-11-29T05:29:07.883Z",
11    "_v": 0
12  }
13

```

[09] update chat name

The screenshot shows the Postman interface for a 'PUT Update Chat Name' request. The URL is `((baseUrl)) /api/chats/ {{chatId}}`. The request body is:

```

1 {
2   "name": "My New Chat"
3 }

```

The response is a 200 OK status with the following JSON data:

```

1 {
2   "success": true,
3   "chat": {
4     "_id": "692a84a3beb34bbf376efa0c",
5     "userId": "692a842cbeb34bbf376efa07",
6     "userName": "Anjali",
7     "name": "My New Chat",
8     "messages": [],
9     "createdAt": "2025-11-29T05:29:07.883Z",
10    "updatedAt": "2025-11-29T05:31:26.694Z",
11    "__v": 0
12  }
13 }

```

[10] send text message

The screenshot shows the Postman interface for a 'POST Send Text Message' request. The URL is `((baseUrl)) /api/messages/text`. The request body is:

```

1 {
2   "chatId": "fjchatId{j}",
3   "prompt": "react js definition"
4 }

```

The response is a 200 OK status with the following JSON data:

```

1 {
2   "success": true,
3   "reply": {
4     "content": "## React JS Definition:\n\nReact (also known as React.js or ReactJS) is a **free and open-source JavaScript library** for building user interfaces (UIs) or UI components. It is maintained by Facebook (Meta) and a community of individual developers and companies.\n\n**Key Characteristics and Features:**\n\n**Declarative:** You describe the desired UI state, and React efficiently updates the DOM to match that state. This simplifies reasoning about your code and makes it more predictable.\n\nInstead of directly manipulating the DOM (imperative), you describe *what* you want to see, and React figures out *how* to get there.\n\n**Component-Based:** React allows you to break down complex UIs into smaller, reusable, and independent pieces called components. Each component manages its own state and logic, making applications easier to develop, maintain, and test. Think of components as building blocks for your UI.\n\n**Virtual DOM:** React uses a virtual DOM (Document Object Model) which is a lightweight copy of the actual DOM. When data changes, React updates the"
5

```

[11] send text message API 2

POST Send Text Message

HTTP QuickGPT APIs / MESSAGES / Send Text Message

POST {{baseUrl}}/api/messages/text

Body (raw) JSON

```

1 {
  "chatId": "{{chatId}}",
  "prompt": "react js definition"
}

```

200 OK 6.37 s 4.29 KB

```

elements\n\nExample (using JSX):```\nimport React from 'react';\n\nfunction Welcome({props}) {\n  return <h1>Hello, {props.name}</h1>;\n}\n\nfunction App() {\n  return (\n    <div>\n      <Welcome name="Alice"/>\n      <Welcome name="Bob"/>\n    </div>\n  );\n}\n\nexport default App;
```\nIn this example:\n* `Welcome` is a React component that accepts a `name` prop and renders a greeting.\n* `App` is another React component that renders two `Welcome` components with different names.\nJSX is used to write the HTML-like syntax within the JavaScript code.\nIn summary, React is a powerful and flexible JavaScript library that empowers developers to create dynamic and efficient user interfaces through its component-based architecture, virtual DOM, and declarative programming style.\n`,
 "role": "assistant",
 "timestamp": 1764394372306,
 "isImage": false
}

```

## [12] generate image message

POST Send Text Message POST Generate Image Message

HTTP QuickGPT APIs / MESSAGES / Generate Image Message

POST {{baseUrl}}/api/messages/image

Body (raw) JSON

```

1 {
 "chatId": "{{chatId}}",
 "prompt": "cute dog wearing sunglasses",
 "isPublished": true
}

```

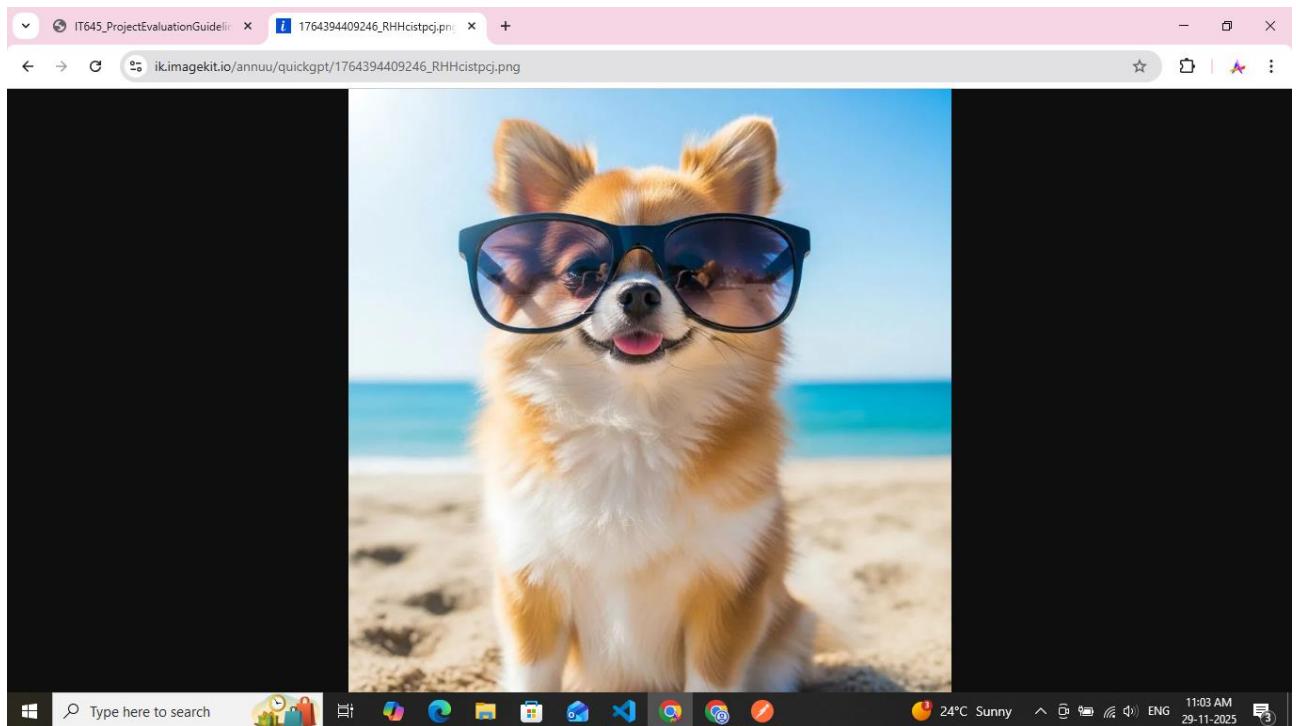
200 OK 13.59 s 452 B

```

{
 "success": true,
 "reply": {
 "role": "assistant",
 "content": "https://ik.imagekit.io/annuu/quickgpt/1764394409246_RHHcistpcj.png",
 "timestamp": 1764394411370,
 "isImage": true,
 "isPublished": true
 }
}

```

## [13] getting generated image from link



## [14] delete chat API

The screenshot shows the Postman API client interface. On the left, the sidebar displays the 'QuickGPT\_WMD\_PROJECT' workspace with collections like 'QuickGPT APIs' and 'CHATS'. Under 'CHATS', the 'DELETE Delete Chat' endpoint is selected. The request URL is `DELETE {{baseUrl}}/api/chats/ {{chatId}}`. The 'Headers' tab shows a single header: 'Authorization' with the value 'Bearer {{token}}'. The response section shows a 200 OK status with the following JSON body:

```

1 {
2 "success": true,
3 "message": "Chat deleted"
4 }

```

The Postman interface also includes sections for 'Body', 'Cookies', 'Test Results', and a preview of the JSON response. The bottom of the screen shows the Windows taskbar with various application icons and system status.

## [15] get plans API

The screenshot shows the Postman interface with the following details:

- Project:** QuickGPT\_WMD\_PROJECT
- Collection:** QuickGPT APIs
- Request:**
  - Method: GET
  - URL: {{baseUrl}}/api/credits/plans
  - Headers (6): Key, Value, Description
  - Body (JSON): A JSON object representing two plan items.
- Response:**
  - Status: 200 OK
  - Time: 8 ms
  - Size: 575 B
  - Preview: Shows the JSON response with two plan items: "basic" and "premium".

## [16] get plans API 2

The screenshot shows the Postman interface with the following details:

- Project:** QuickGPT\_WMD\_PROJECT
- Collection:** QuickGPT APIs
- Request:**
  - Method: GET
  - URL: {{baseUrl}}/api/credits/plans
  - Headers (6): Key, Value, Description
  - Body (JSON): A JSON object representing two plan items.
- Response:**
  - Status: 200 OK
  - Time: 8 ms
  - Size: 575 B
  - Preview: Shows the JSON response with two plan items: "basic" and "premium".

## [17] purchase plan

The screenshot shows the Postman interface with a collection named "QuickGPT\_WMD\_PROJECT". The "POST Purchase Plan" request is selected. The request URL is `POST {{baseUrl}} /api/credits/purchase`. The "Body" tab is active, showing the JSON payload:

```

1 {
2 "planId": "basic"
3 }

```

The response status is 200 OK, with a response time of 1.26 s and a size of 727 B. The response body is a long URL starting with `https://checkout.stripe.com/c/pay/cs_test_a174mLxRpaxlPhSjxhjmuSQhPjxxeK3yEMDklwbN9UwtpO1ZJa1rAh0ubR#fidnandhYHdWcXxpYCc%2FJ2FgY2RwaXEnKSdkdWx...`.

## [18] payment processing

The screenshot shows a browser window for "IT645\_ProjectEvaluationGuideline" with a "Checkout" tab open. The URL is `checkout.stripe.com/c/pay/cs_test_a174mLxRpaxlPhSjxhjmuSQhPjxxeK3yEMDklwbN9UwtpO1ZJa1rAh0ubR#fidnandhYHdWcXxpYCc%2FJ2FgY2RwaXEnKSdkdWx...`. The page is in TEST MODE.

Choose a currency:  ₹929.26  \$10.00  
1 USD = 92.9260 INR (includes 4% conversion fee)

Basic ₹929.26

Payment method

Email: anjalipatel3074@gmail.com

Card information: 4111 1111 1111 1111 VISA  
12 / 28 123

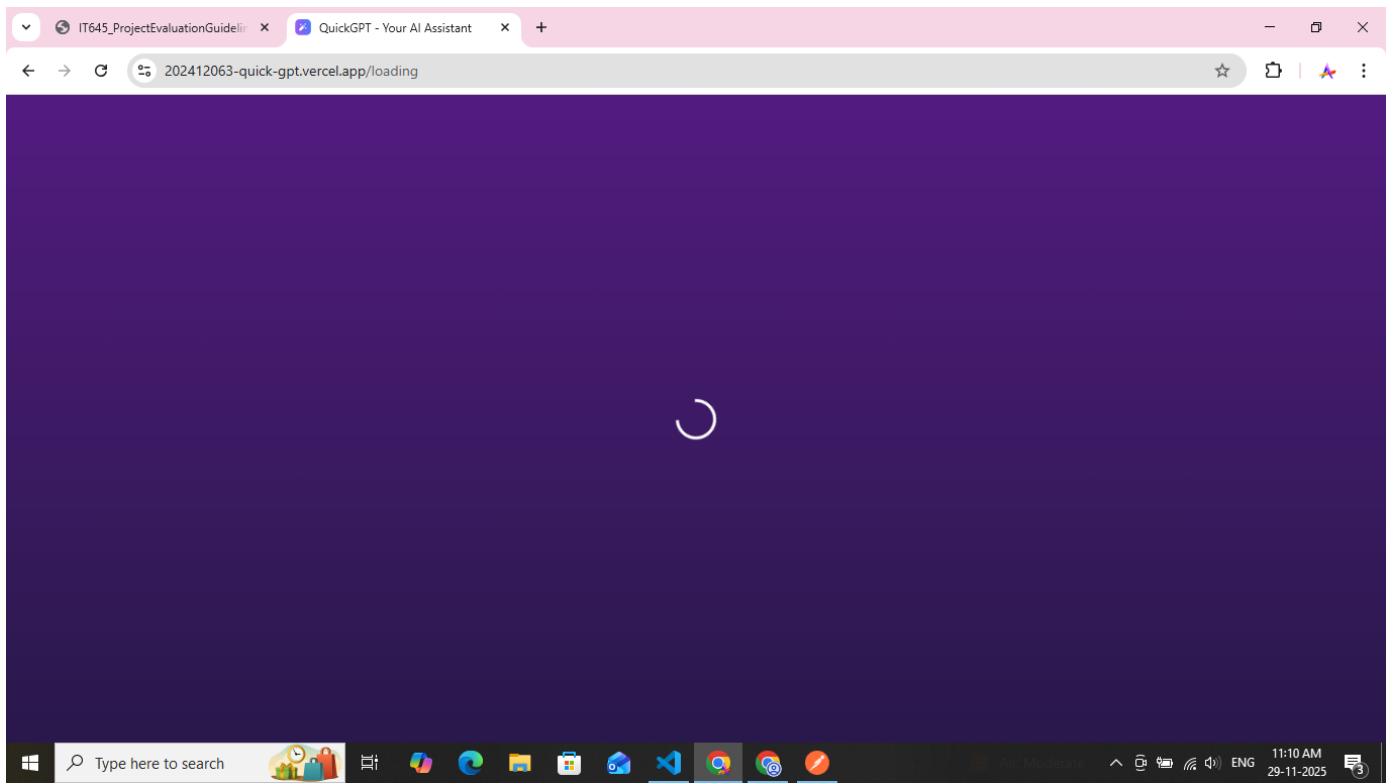
Cardholder name: ANJALI PATEL

Country or region: India

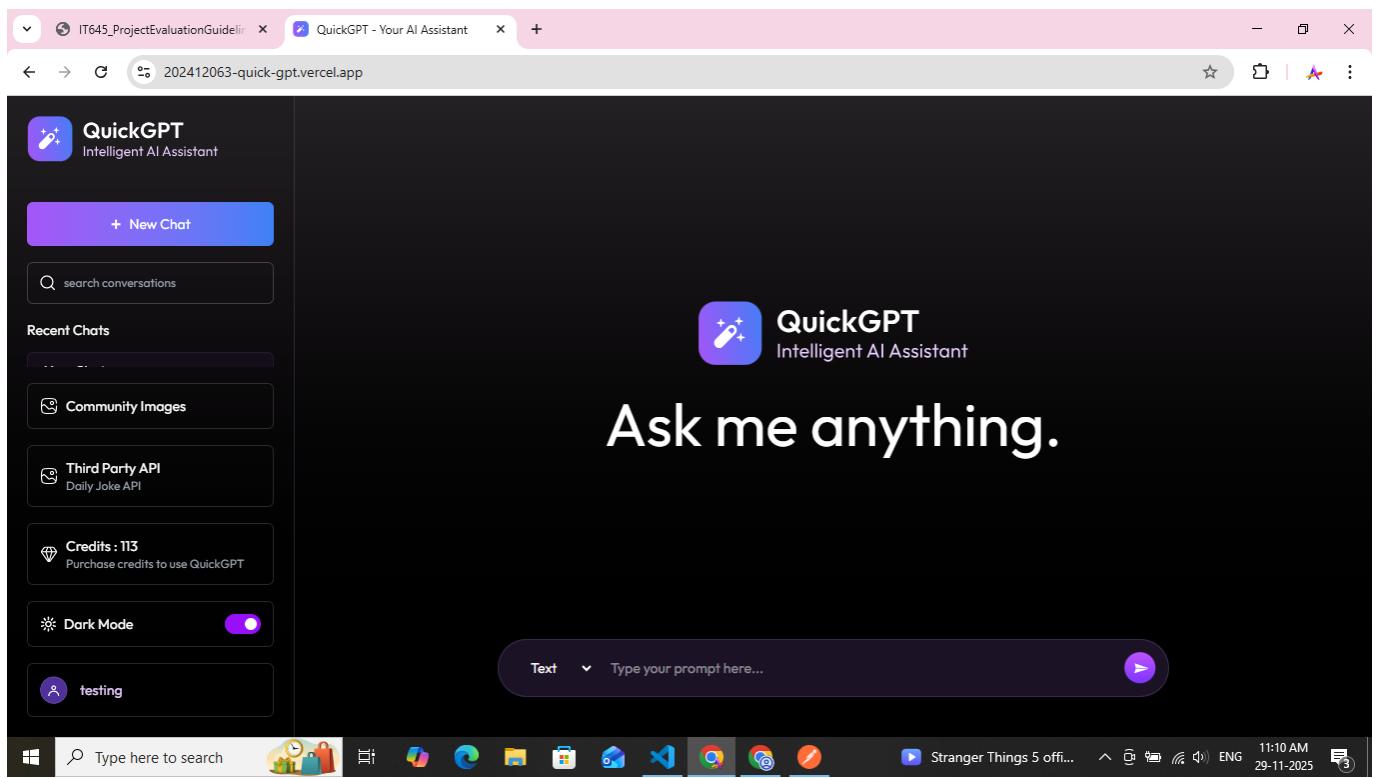
Save my information for faster checkout  
Pay securely on this site and everywhere Link is accepted.

Pay

## [19] redirected after sucessful payment



## [20] updated credits



## [21] get my transactions API

The screenshot shows the Postman interface with the following details:

- Collection:** QuickGPT\_WMD\_PROJECT
- Request Type:** GET
- URL:** {{baseUrl}}/api/credits/transactions
- Headers:**
  - Authorization: Bearer {{token}}
- Body:** (Empty)
- Cookies:** (Empty)
- Test Results:** 200 OK (165 ms, 732 B)
- JSON Response:**

```

1 {
2 "success": true,
3 "transactions": [
4 {
5 "_id": "692a870476f7f3a4fb28ce65",
6 "userId": "692a842cbeb34bbf376efa07",
7 "planId": "basic",
8 "amount": 10,
9 "credits": 100,
10 "isPaid": false,
11 "createdAt": "2025-11-29T08:39:16.844Z",
12 "updatedAt": "2025-11-29T05:39:16.844Z",
13 "__v": 0
14 }

```

## [22] get credits of particular user API

The screenshot shows the Postman interface with the following details:

- Collection:** QuickGPT\_WMD\_PROJECT
- Request Type:** GET
- URL:** {{baseUrl}}/api/credits/me
- Headers:**
  - Authorization: Bearer {{token}}
  - Content-Type: application/json
- Body:** (Empty)
- Cookies:** (Empty)
- Test Results:** 200 OK (145 ms, 384 B)
- JSON Response:**

```

1 {
2 "success": true,
3 "credits": 17,
4 "user": {
5 "_id": "692a842cbeb34bbf376efa07",
6 "name": "Anjali",
7 "email": "anjali@example.com"
8 }
9 }

```

## [23] get user by ID API

Postman screenshot showing the 'Get User By ID' API call. The request URL is `GET {{baseUrl}} /api/users/ {userId}`. The response is a 200 OK status with a response time of 135 ms and a body size of 392 B. The response body is:

```

1 {
2 "success": true,
3 "user": {
4 "_id": "692a842cbeb34bbf376efa07",
5 "name": "Anjali",
6 "email": "anjali@example.com",
7 "credits": 17,
8 "__v": 0
9 }
10 }

```

## [24] update user credits API

Postman screenshot showing the 'PATCH Update User Credits' API call. The request URL is `PATCH {{baseUrl}} /api/users/ {userId} /credits`. The response is a 200 OK status with a response time of 220 ms and a body size of 468 B. The response body is:

```

1 {
2 "success": true,
3 "user": {
4 "_id": "692a842cbeb34bbf376efa07",
5 "name": "Anjali",
6 "email": "anjali@example.com",
7 "password": "$2b$10$xxL8L8cXDq.14R9oYHr40./iuud7Fosj9iw67sx9lmEf2gMGIQxTe",
8 "credits": 5000,
9 "__v": 0
10 }
11 }

```

## [25] get all published images API

The screenshot shows the Postman interface with the 'QuickGPT\_WMD\_PROJECT' workspace selected. In the left sidebar, under 'Collections', the 'USER EXTRA' collection is expanded, and the 'GET Get All Published Images' request is highlighted. The request URL is `GET {{baseUrl}} /api/users/published-images`. The 'Body' tab is selected, showing the response body as JSON:

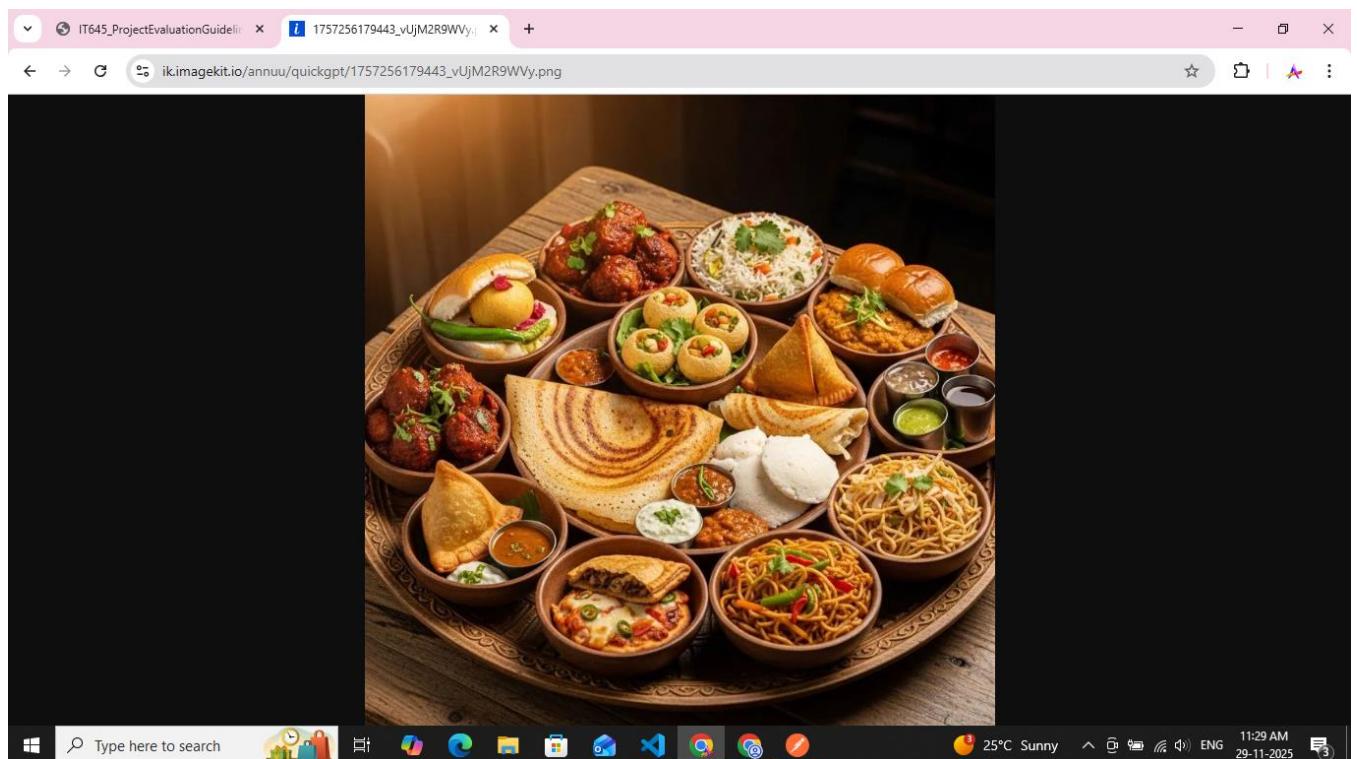
```

1 {
2 "success": true,
3 "images": [
4 {
5 "imageUrl": "https://ik.imagekit.io/annuu/quickgpt/1757256179443_vUjM2R9WVY.png",
6 "userName": "ANJALI PATEL",
7 "createdAt": 1757256183360
8 }
9]
10 }

```

The response status is 200 OK with 77 ms latency and 429 B size. The bottom status bar shows the Windows taskbar with the date and time as 29-11-2025.

## [26] getting published image from link



## [27] get random joke third party API

The screenshot shows the Postman application interface. On the left, the sidebar displays collections, environments, history, and flows. The main workspace shows a collection named "QuickGPT\_WMD\_PROJECT" with a sub-collection "THIRD PARTY API". A specific request titled "GET Get Random Joke" is selected. The request URL is "https://official-joke-api.appspot.com/jokes/random". The "Headers" tab is active, showing an "Accept" header set to "application/json". The "Body" tab shows the JSON response:

```

1 {
2 "type": "general",
3 "setup": "If you're American when you go into the bathroom, and American when you come out, what are you when
4 you're in there?",
5 "punchline": "European",
6 "id": 38
}

```

## **Third-Party libraries & tools used**

### ➤ **Frontend**

- React
- Tailwind CSS
- React router
- Context API
- axios

### ➤ **Backend**

- express
- mongoose
- cors
- jsonwebtoken
- bcryptjs
- morgan
- dotenv

### ➤ **Payments**

- stripe

### ➤ **AI**

- Google Gemini

### ➤ **Other tools**

- Postman : for API testing
- npm
- Git + GitHub
- VS code editor

## Links

➤ **GitHub repository link**

- [https://github.com/AP-anjali/QuickGPT\\_WMD\\_PROJECT](https://github.com/AP-anjali/QuickGPT_WMD_PROJECT)

➤ **Deployed website link**

- **Frontend :** <https://202412063-quick-gpt.vercel.app/>
- **Backend :** <https://quick-gpt-server-plum.vercel.app/>