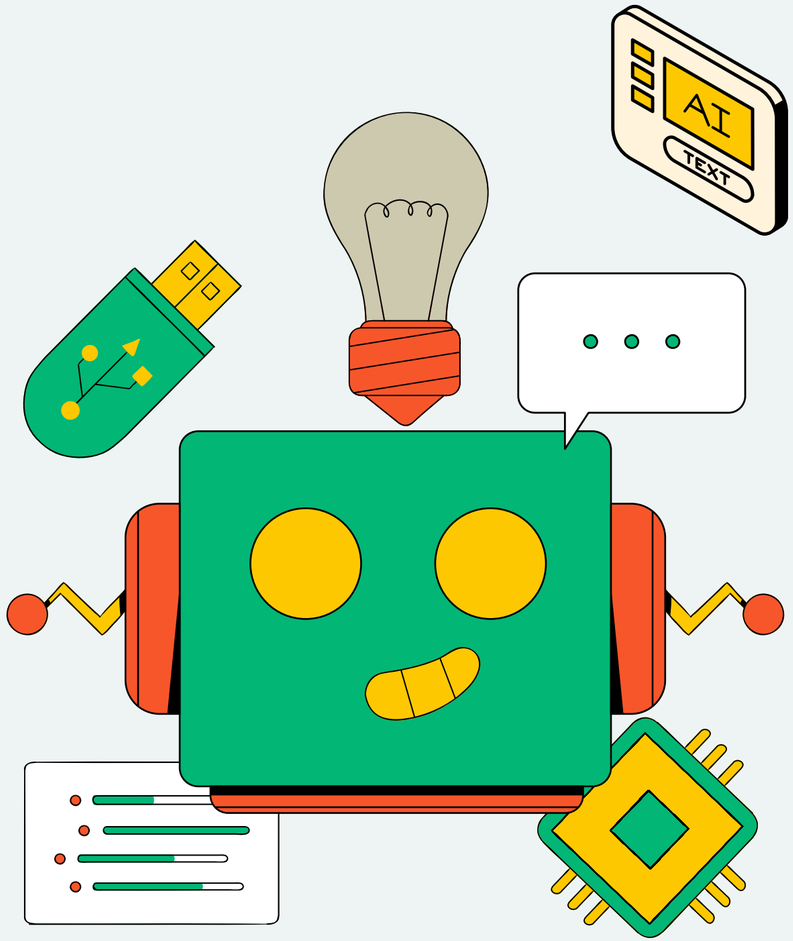
INNOVATION X – TEAM



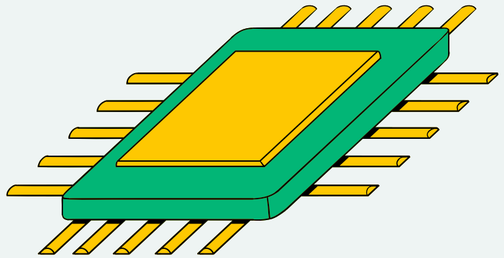
VERATILE AI ASSISTANT

DevNet Assisiate,Software Engineering Course

PRESENTATION

PRESENTED BY:
INNOVATION X – TEAM

AI-ASSISTANT



AI CHAT BOT - CONTENT

- Information about our application
- The reasons our team decided on these specific features in our application Git / Github of our project.
- Application code including comments and documentation
- Problems we faced during this project
- Future enhancements
- Conclusion

Tip: Use links to go to a different page inside your presentation. How: Highlight text, click on the link symbol on the toolbar, and select the page in your presentation you want to connect.

INTRODUCTION - AI-DESIGNER TEXT BOT

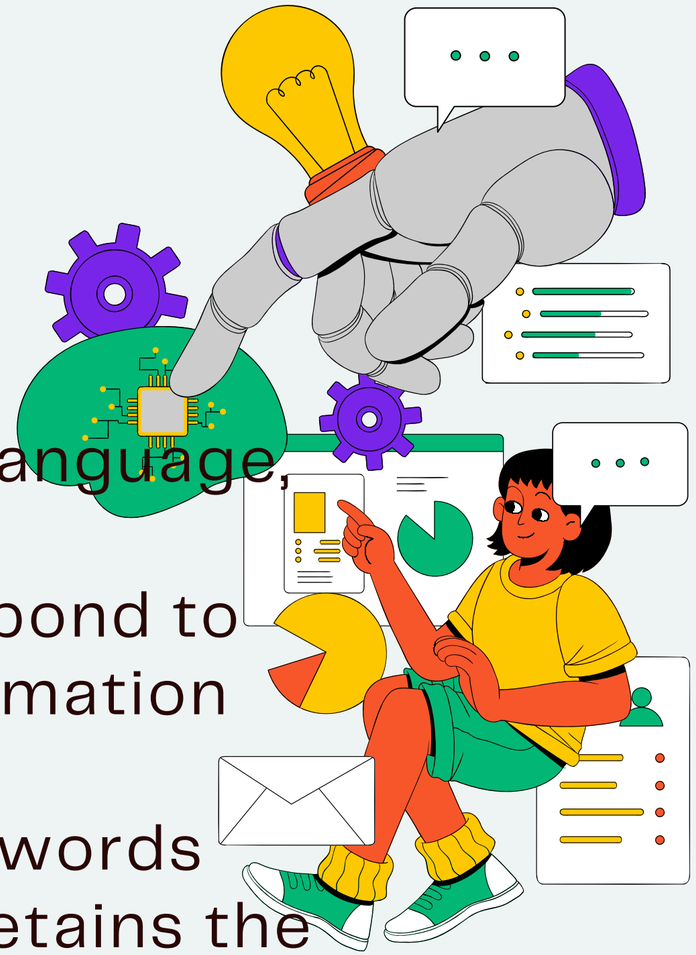
Our AI-powered chatbot that has several functionalities. It can check the grammar of the text, paraphrase sentences, chat with users, and translate text into other languages

Versatile AI Assistant Our project introduces an innovative AI assistant designed to streamline various text-based tasks. It's equipped with a suite of tools that enhance user interaction through text summarization, sentence restructuring, conversational engagement, and multilingual translation.

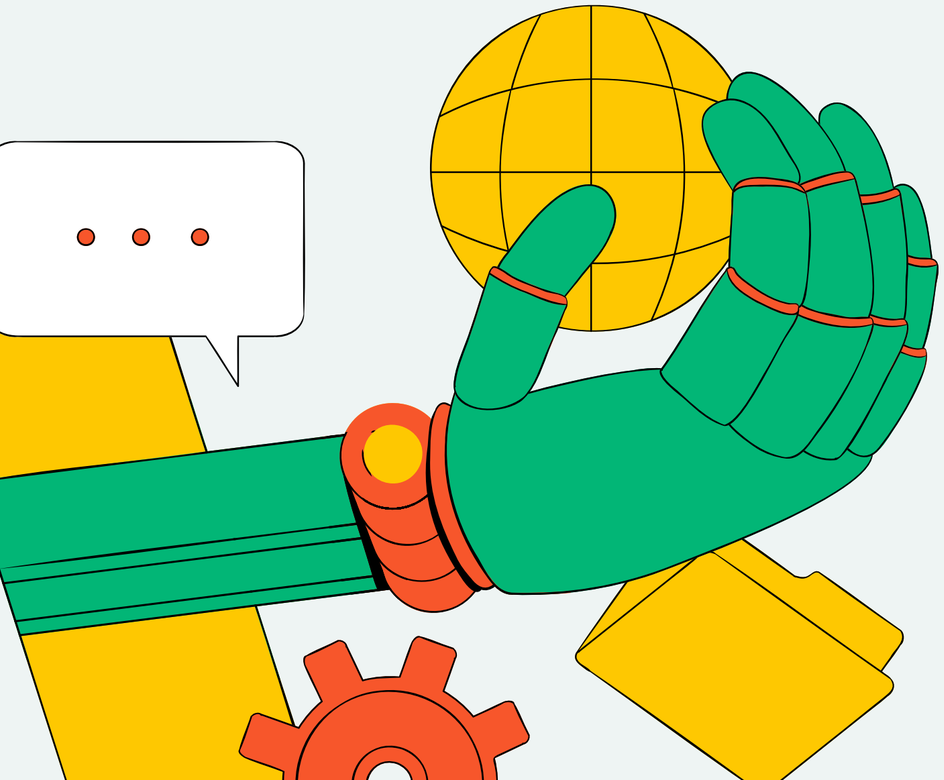
WHAT ARE THE KEY FEATURES?

Key Features:

- Checking Grammar: Checks the grammar.
- Translation: Provide a line of text and a target language, and our AI will deliver an accurate translation.
- Interactive Q&A: Our AI is programmed to respond to user queries, offering instant support and information retrieval.
- Text Refinement: The paraphrasing function rewords existing text to produce a new rendition that retains the original intent and meaning.



MAIN REASONS WHY OUR TEAM SELECTED THIS OPTION?



Integrating advanced language models like GPT-3/GPT-4 into our project underscores our commitment to leveraging state-of-the-art technology. The integration process involves adapting the CISCO net acad Software Engineering Lab 4.9.2 Python framework to incorporate the GPT-3/GPT-4 REST API, a task that demands profound technical acumen.
It's a key driver of AI applications, including natural language processing, image recognition, and recommendation systems.

Add ChatBot Functionality: Our ChatBot is adept at processing natural language, making it ideal for applications such as conversational agents, digital aides, and translation tools. It’s also capable of learning user preferences to offer a personalized experience.a little bit of body text

PROJECT PLAN

01

STEP-1

ACreating a new repository for the project and inviting all collaborators

02

STEP-2

Choosing a methodology for our project

03

STEP-3

Checking each other subteam's job and making it more precise

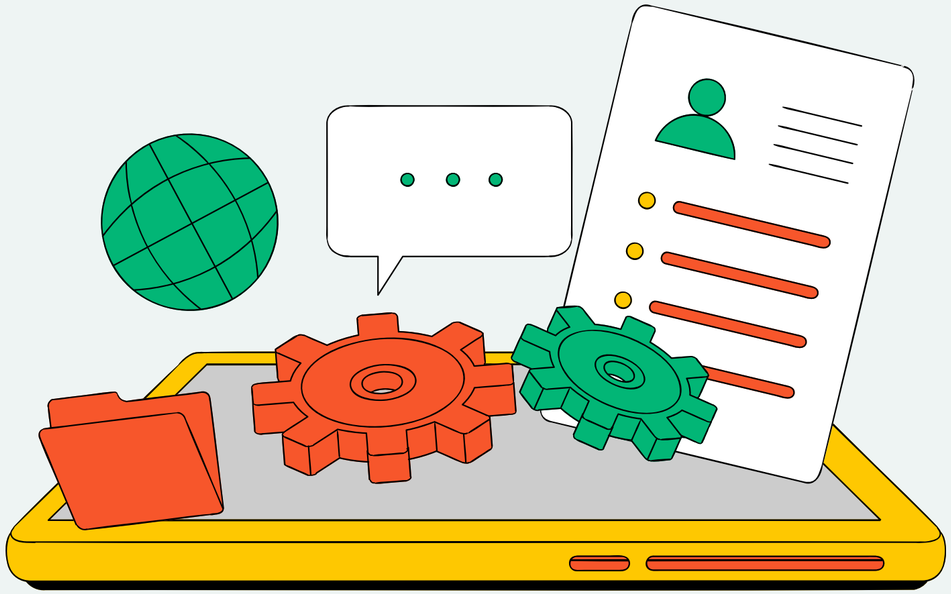
CREATING REPOSITORY DOCUMENTARY

Creating Branches: In GitHub, branches serve as separate workspaces to safeguard the primary code. Team members can establish branches for developing new features or addressing bugs, ensuring no overlap in their tasks. To initiate a branch, one should go to the GitHub repository of the project, designate a name for the branch, and select “Create branch.” This allows them to commence work on their assignments independently of the main code.

Adding Team Members: Project leads or admins have the authority to grant team members collaborative roles in the project. As collaborators, they gain the ability to read, write, and contribute to the repository, which includes branching, committing, and updating changes. To include a team member, navigate to the repository’s “Settings,” choose “Collaborators,” input the individual’s GitHub handle or email, and click “Add collaborator.” Once the invitee consents, they’re set to start branching and contributing to the project.

CHOOSING A METHODOLOGY

Agile



Employing the Agile methodology empowers our team to maintain adaptability during development, with consistent evaluations and retrospectives that pinpoint enhancement opportunities and permit timely modifications. This strategy ensures the punctual delivery of a superior Python text editor application, equipped with four key functionalities and the incorporation of the GPT API.

CHECKING EACH OTHER'S WORK

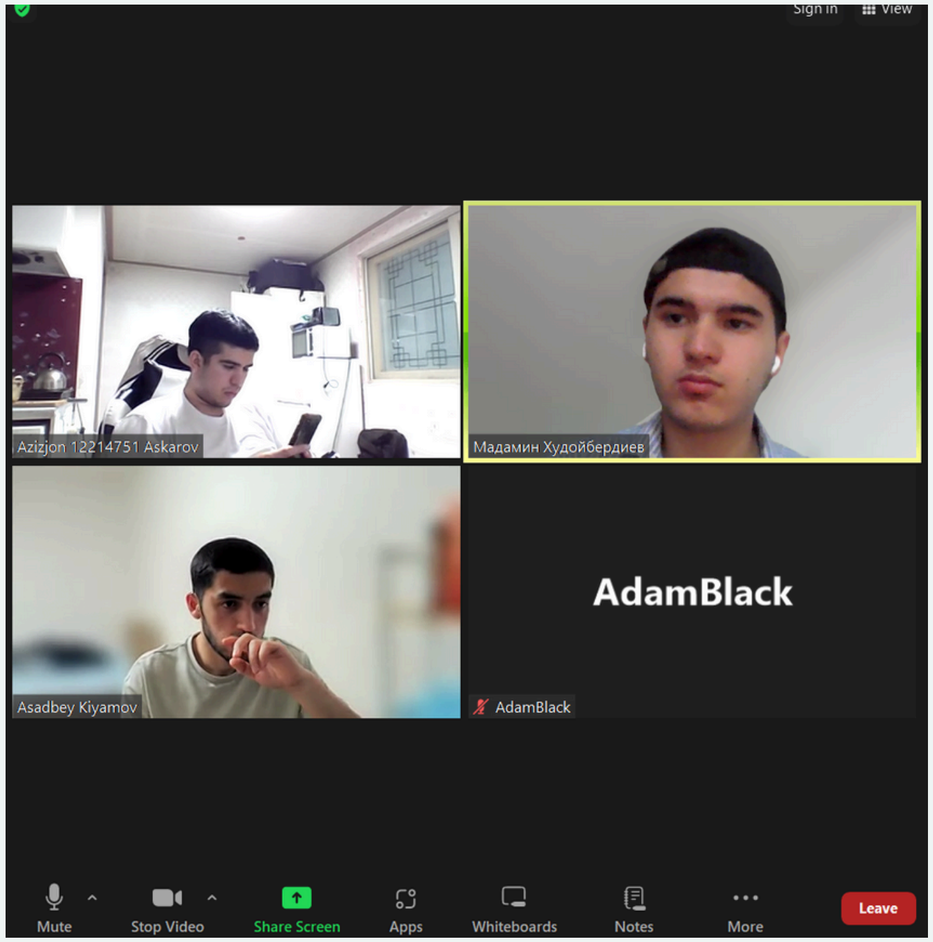
In a team setting, the practice of checking each other's work is invaluable. It fosters a culture of collaboration and continuous improvement, ensuring that tasks are completed with the highest quality. This process not only helps to catch errors and oversights but also encourages knowledge sharing and skill development among team members. By reviewing each other's work, team members can learn from different approaches and techniques, which can lead to innovative solutions and increased efficiency. Moreover, it builds trust and accountability, as team members become more invested in the collective success of the group. Ultimately, this collaborative effort contributes to the overall resilience and adaptability of the team, enabling them to tackle complex challenges with confidence.

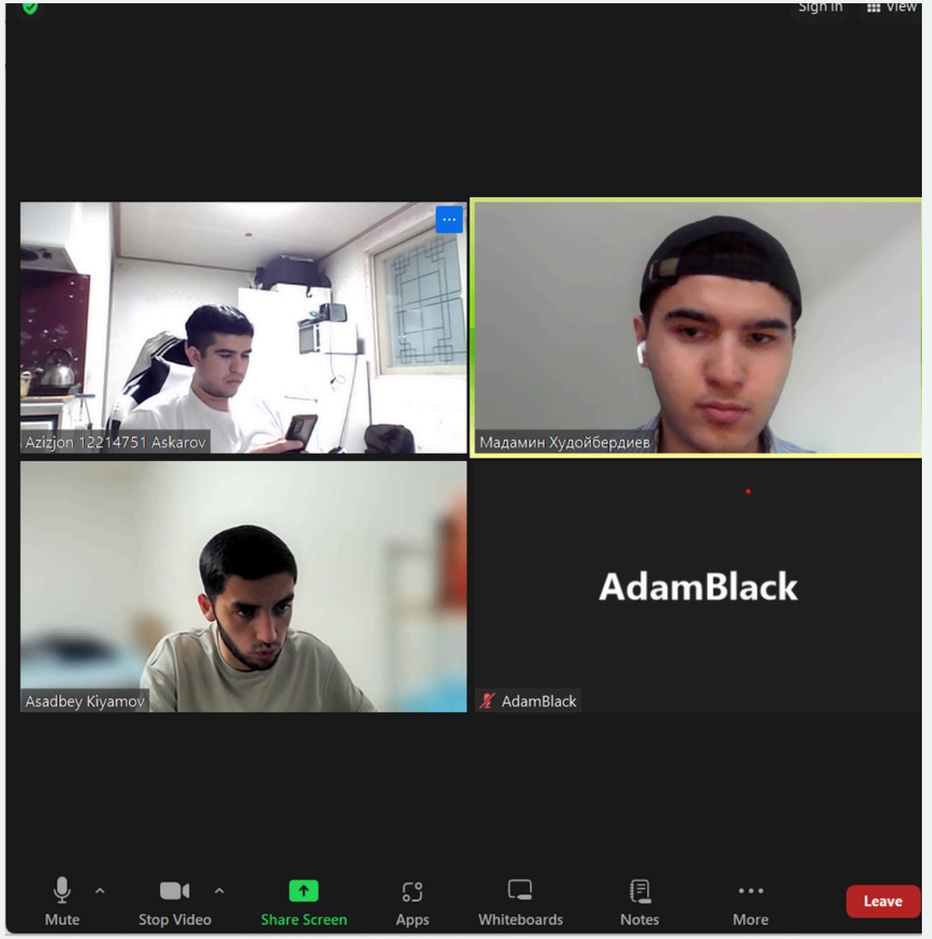
EFFICIENCY

ethical & philosophical considerations

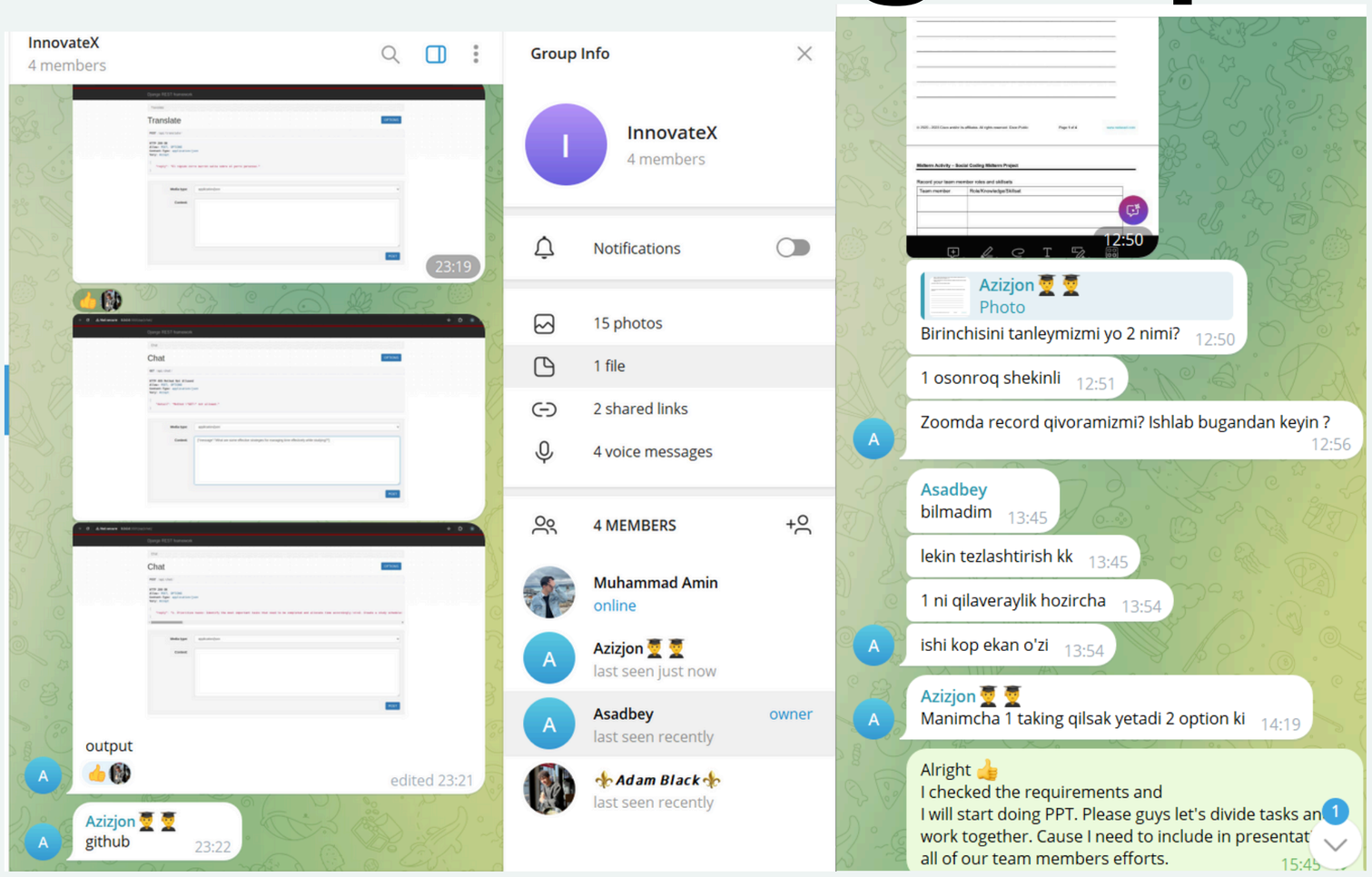
dividing into teams

Daily zoom meetings





Social Media/ group discussion



APPLICATION CODE COMMENTS AND DOCUMENTATION

IN OUR PROJECT WE USED PYTHON PROGRAMMING LANGUAGE;Django/Rest framework/ SqLite3

```python
class ChatView(APIView):
    """
    API View to handle chat interactions using OpenAI's ChatCompletion.
    """
    def post(self, request):
        """
        Receives a POST request with a user message, processes it through the GPT-3.5-turbo mo
        and returns the AI-generated response.
        """
        print("Received data:", request.data)  # Debugging line
        serializer = ChatSerializer(data=request.data)
        if serializer.is_valid():
            user_message = serializer.validated_data['message']
            print("Validated message:", user_message)  # Debugging line
            response = openai.ChatCompletion.create(
                model="gpt-3.5-turbo",
                messages=[{"role": "user", "content": user_message}]
            )
            return Response({'reply': response['choices'][0]['message']['content']})
        else:
            print("Serializer errors:", serializer.errors)  # Debugging line
        return Response(serializer.errors, status=400)
```

```python
class TranslateView(APIView):
    """
    API View to handle requests to translate text using OpenAI's GPT-4 model.
    """
    def post(self, request):
        """
        Receives a POST request with text, translates it using the GPT-4 model, and returns the tr
        """
        serializer = ChatSerializer(data=request.data)
        if serializer.is_valid():
            user_message = serializer.validated_data['message']
            response = openai.ChatCompletion.create(
                model="gpt-4",
                messages=[
                    {"role":"system", "content": system},
                    {"role": "user", "content": user_message}
                ]
            )
            return Response({'reply': response['choices'][0]['message']['content']})
        else:
            print("Serializer errors:", serializer.errors)
        return Response(serializer.errors, status=400)

system = "You are helpful assistant to translate any text that required from user with asked langu
```

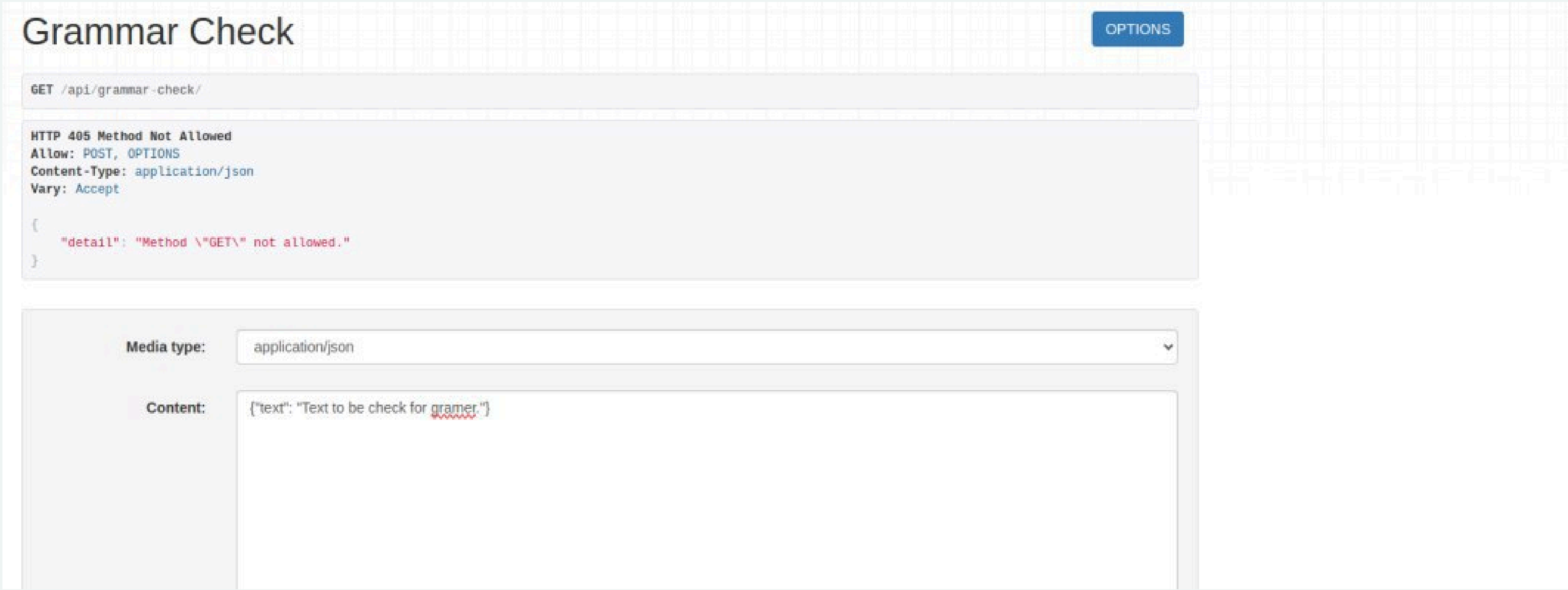
```python
class ParaphraseView(APIView):
    """
    API View to handle requests to paraphrase text using the GPT-3.5-turbo model.
    """
    def post(self, request):
        """
        Receives a POST request with text, paraphrases it using the GPT-3.5-turbo model, and
        """
        serializer = ChatSerializer(data=request.data)
        if serializer.is_valid():
            user_message = serializer.validated_data['message']
            response = openai.ChatCompletion.create(
                model="gpt-3.5-turbo",
                messages=[
                    {"role": "system", "content": "You are helpful to paraphrase any text th
                    {"role": "user", "content": user_message}
                ]
            )
            return Response({'reply': response['choices'][0]['message']['content']})
        else:
            print("Serializer errors:", serializer.errors)
        return Response(serializer.errors, status=400)
```

```python
class GrammarCheckView(APIView):
    """
    API View to handle grammar checking requests using OpenAI's GPT-4 model.
    """
    def post(self, request):
        """
        Receives a POST request with text, checks and corrects grammar using the GPT-4 model,
        and returns the corrected text.
        """
        serializer = TextSerializer(data=request.data)
        if serializer.is_valid():
            text_to_check = serializer.validated_data['text']
            response = self.check_grammar_with_gpt4(text_to_check)
            return Response(response)
        else:
            print("Serializer errors:", serializer.errors)
        return Response(serializer.errors, status=400)

    def check_grammar_with_gpt4(self, text):
        """
        Uses the GPT-4 model to check and correct grammar in the provided text.
        """
        response = openai.ChatCompletion.create(
            model="gpt-4",
            messages=[
                {"role": "system", "content": "You are a helpful assistant tasked with checking and correcting grammar."},
                {"role": "user", "content": text},
                {"role": "assistant", "content": ""},
            ],
        )
```

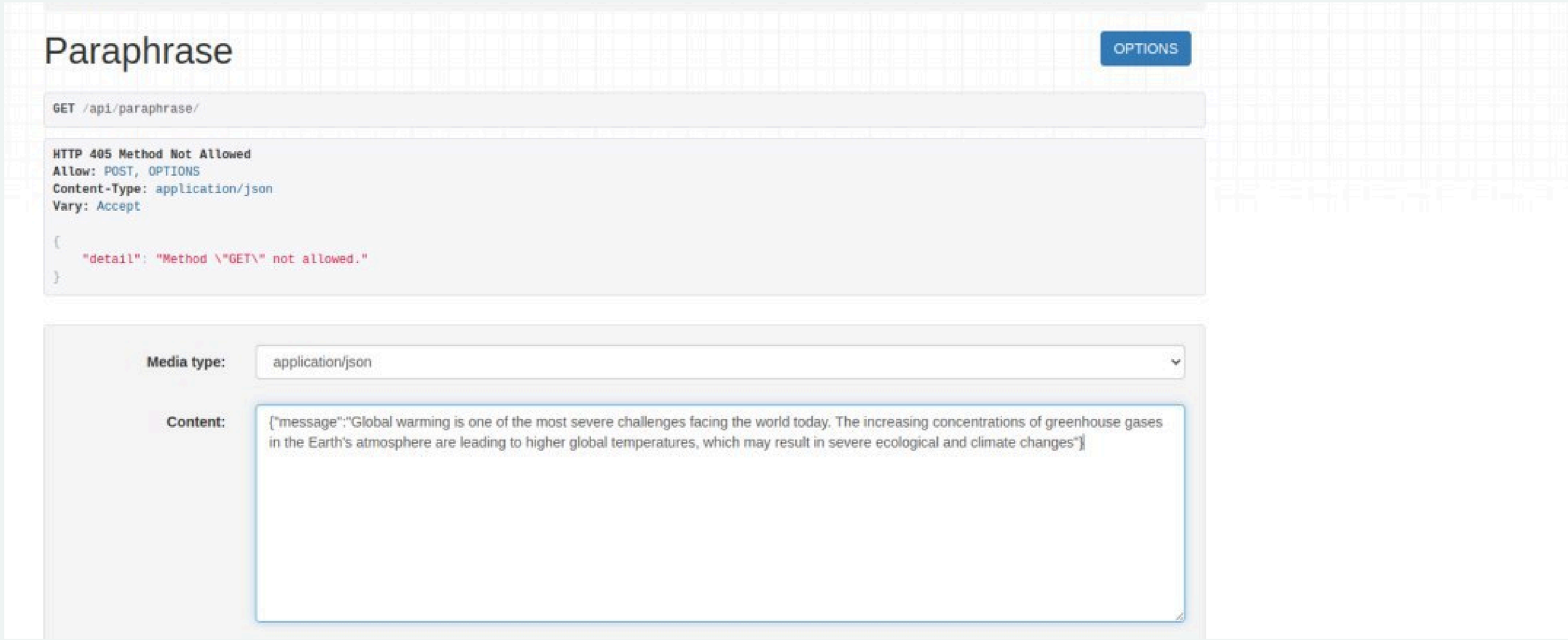
OUTPUT OF THE CODE

Grammar Check



OUTPUT OF THE CODE

Paraphrase



OUTPUT OF THE CODE

Translate

Translate

OPTIONS

```
GET /api/translate/
```

```
HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Method \"GET\" not allowed."
}
```

Media type: application/json

Content:

```
{"message":"The quick brown fox jumps over the lazy dog. Translate this to Spanish language"}
```

POST

OUTPUT OF THE CODE

Chat

Chat

OPTIONS

```
GET /api/chat/
```

```
HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Method \"GET\" not allowed."
}
```

Media type: application/json

Content:

```
{"message":"What are some effective strategies for managing time effectively while studying?"}
```

POST

PROBLEMS

PROBLEM 1: INTEGRATION COMPLEXITY

- Issue: Integrating different components of the chatbot, such as natural language processing (NLP), backend logic, and user interface, proved to be complex and time-consuming.

PROBLEM 2: SCALABILITY CONCERNS

- **Issue**: As the user base and functionality of the chatbot expanded, concerns regarding scalability emerged. We realized that our initial architecture might not efficiently handle a growing number of users and requests.

PROBLEM 3: TRAINING DATA LIMITATIONS

- Issue: Developing robust conversational abilities for the chatbot required large volumes of diverse training data, which was challenging to acquire and curate.

FUTURE ENHANCEMENTS

Multimodal Interaction:

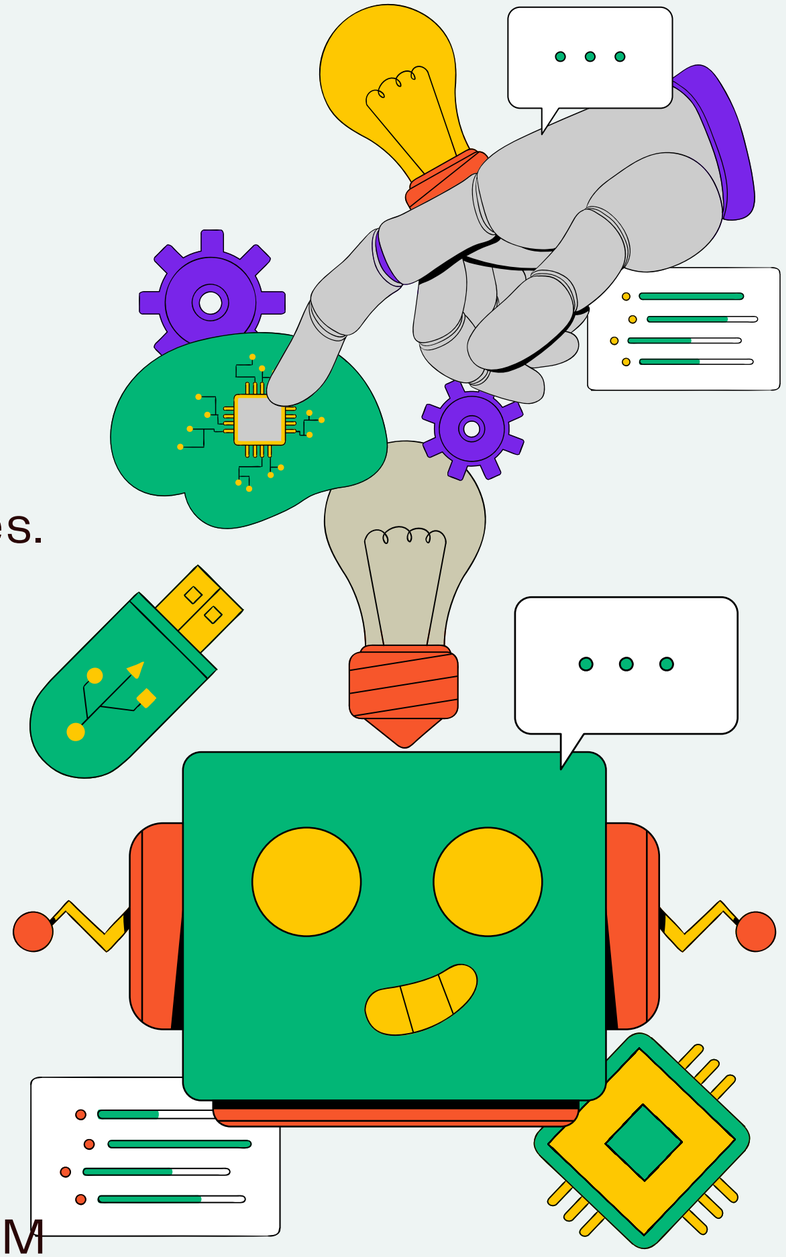
- Enhancement: Integrate support for multimodal interaction, enabling users to interact with the chatbot through various channels such as text, voice, and images.

Personalization and Context Awareness:

- Enhancement: Implement advanced techniques for personalization and context awareness, allowing the chatbot to adapt its responses based on user history, preferences, and contextual cues.

Integration with External Systems and APIs:

- Enhancement: Expand the chatbot's capabilities by integrating with external systems and APIs, such as CRM platforms, e-commerce databases, and IoT devices.



CONCLUSION

Throughout this project, we embarked on a journey of discovery and innovation in AI-driven chatbot development. We learned invaluable lessons in API integration, leveraging AI technologies, and problem-solving that have significantly enriched our skill sets and understanding of modern software development practices.

As we conclude this project, we take pride in our accomplishments and look forward to applying our newfound knowledge and skills to future endeavors. Together, we have not only built innovative solutions but also fostered a culture of continuous learning and growth within our team.

Key learnings:

- API Integration and AI Technologies: We delved into the intricacies of integrating diverse APIs and harnessing the power of AI to create intelligent chatbot solutions. This experience has broadened our technical expertise and equipped us with the tools to tackle complex integration challenges.
- Creative Problem-Solving: Faced with various technical hurdles, we demonstrated resilience and creativity in finding effective solutions. From modular development approaches to advanced techniques like transfer learning, our problem-solving journey has been both challenging and rewarding.
- Collaborative Teamwork: Our success would not have been possible without the dedication and collaborative efforts of every team member. Through open communication, shared goals, and mutual support, we navigated through obstacles and achieved our objectives with determination and teamwork.

Thank you to everyone involved for your hard work, creativity, and unwavering commitment to excellence. Let's continue to push boundaries, explore new horizons, and make a lasting impact in the world of AI-driven innovation.