

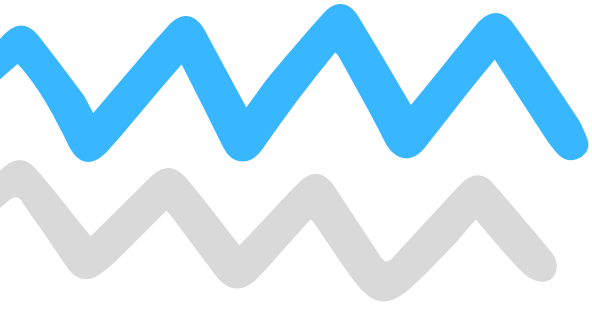


CHATBOT IN PYTHON

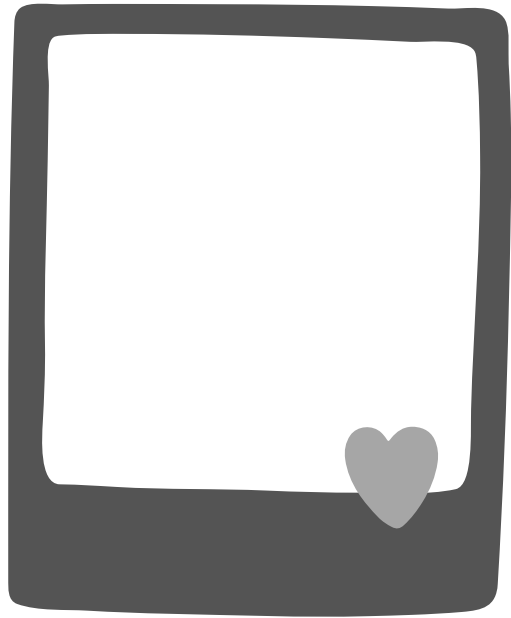
Artificial Intelligence

Phase 5

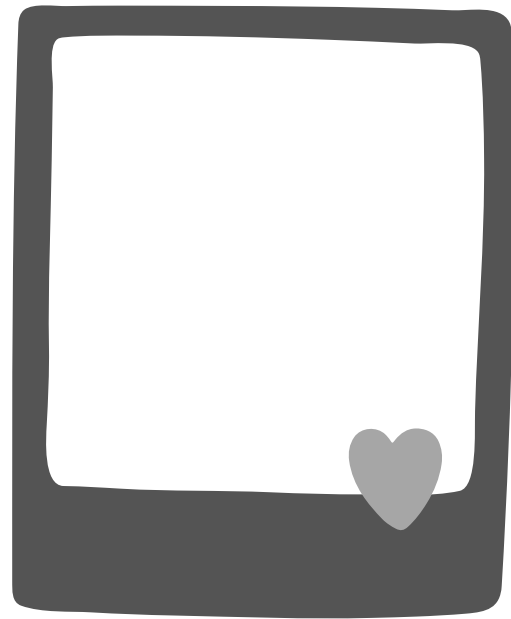




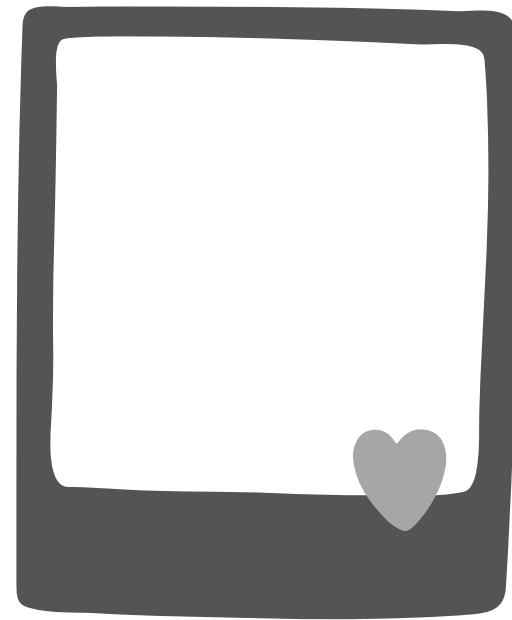
Our Team



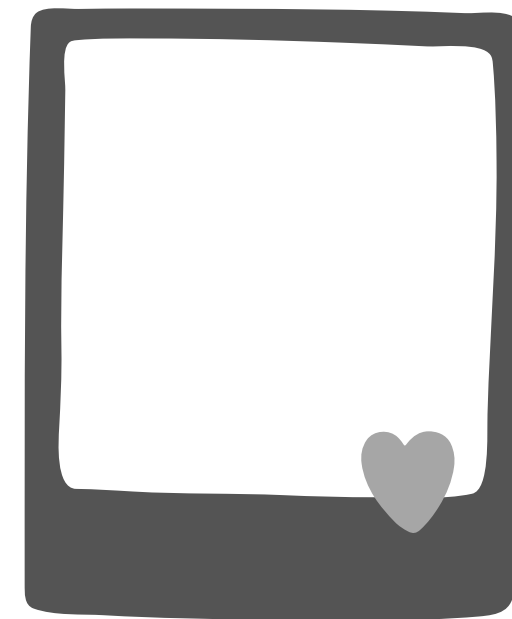
SANSHA MJ
au711021104042



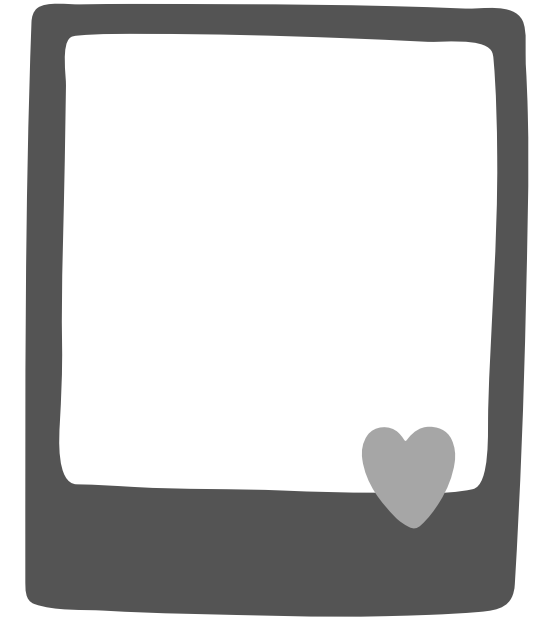
PRAVEEN R
au711021102034



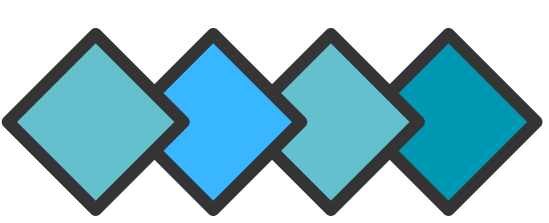
PRADEEP S
au711021102033



GOKUL R
au711021102010



**UDHAYA
KUMAR V**
au711021102054



Topic



Problem Statement



Design Thinking Process



Libraries and Integration of NLP Techniques



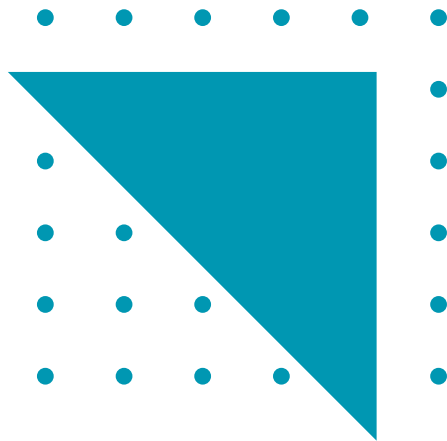
User Interaction and Web Application Integration



Innovative Techniques and Approaches



Codes



Problem Statement

The goal of this project is to build an AI-powered chatbot that can interact with users, answer their queries, and provide relevant information or assistance. The chatbot will be integrated into a web application, enhancing user engagement and providing a seamless user experience.

Design Thinking Process

- 1. Functionality:** The chatbot's scope includes answering common questions, providing guidance, and directing users to appropriate resources.
- 2. User Interface:** The chatbot will be integrated into a web application, providing a user-friendly interface for interactions.
- 3. NLP Integration:** NLP techniques, including text tokenization, entity recognition, and sentiment analysis, will be implemented to understand and process user input effectively.
- 4. Responses:** The chatbot will offer accurate answers, suggestions, and assistance, ensuring a personalized and informative interaction with users.
- 5. Integration:** The chatbot will seamlessly integrate into the web application, enabling smooth communication between the user and the chatbot.
- 6. Testing and Improvement:** Continuous testing and refinement will be conducted to enhance the chatbot's performance based on user interactions and feedback.

Libraries and Integration of NLP Techniques

- Python was used as the primary programming language.
- Libraries such as transformers were employed for GPT-3 integration.
- Flask was used for developing the web application.
- NLP techniques such as text tokenization, entity recognition, and sentiment analysis were integrated using libraries like NLTK and spaCy.

User Interaction and Web Application Integration

The chatbot interacts with users through a user-friendly interface integrated into a web application. Users can input their queries or requests, and the chatbot processes the input using NLP techniques to provide relevant and helpful responses. The web application provides a seamless and intuitive platform for users to engage with the chatbot and obtain the information they need

Innovative Techniques and Approaches

- Advanced NLP techniques were used to enhance the chatbot's understanding of user input and to generate contextually appropriate responses.
- The integration of GPT-3 facilitated the provision of more accurate and natural language-based interactions.
- Iterative testing and user feedback were utilized to continuously improve the chatbot's performance and user experience.

Codes

Dataset link <https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot>

app.py

```
from flask import Flask, request, render_template
import openai
```

```
api_key = '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'
openai.api_key = api_key
```

```
with open('dialogs.txt', 'r', encoding='utf-8') as file:
```

```
dataset = file.readlines()
app = Flask(__name__)
def chat_with_bot(user_input, dataset):
    found_match = False
    for line in dataset:
        if '|' in line:
            question, answer = line.strip().split('|')
            if question.lower() == user_input.lower():
                found_match = True
    return answer
# If no match is found, use GPT-3 to generate a response
if not found_match:
    response = openai.Completion.create(
        engine="gpt-3.5-turbo-instruct-0914", # You can choose the appropriate engine
        prompt=f"You: {user_input}\nBot:",
```

```
max_tokens=50
)
return response.choices[0].text
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.form['user_input']
    if user_input.lower() == 'bye':
        bot_response = "Goodbye!"
    else:
        bot_response = chat_with_bot(user_input, dataset)
    return bot_response
if __name__ == '__main__':
    app.run(debug=True)
```


index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Chatbot</title>
  <style>
    @import url('https://fonts.googleapis.com/css2?
family=Poppins:wght@200;300;400;500;700;800;900&display=swap');


    {
      margin: 0;
      padding: 0;
      font-family: 'Poppins', sans-serif;
    }

    body {
      overflow: hidden;
    }
```

```
.container {  
  width: 100%;  
  height: calc(100vh);  
  background-color: #000;  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  color: #fff;  
}  
.title {  
  padding: 20px;  
  text-align: center;  
}
```



```
#chat-container {  
  width: 100%;  
  border: 1px solid #ddd;  
  flex: 1;  
  display: flex;  
  flex-direction: column;  
  text-align: center;  
  overflow-y: auto;  
  padding-bottom: 20px;  
}  
#chat-log {  
  flex: 1;  
  padding: 20px;  
}
```



```
#user-input {  
width: 80%;  
padding: 10px;  
border: none;  
font-size: 16px;  
outline: none;  
background: #f2f2f2;  
}
```


```
#submit-button {  
background-color: #212121;  
color: #fff;  
border: none;  
padding: 10px 20px;  
cursor: pointer;  
font-size: 16px;  
}
```

```
#user-input {  
width: 80%;  
padding: 10px;  
border: none;  
font-size: 16px;  
outline: none;  
background: #f2f2f2;  
}
```

```
#submit-button {  
background-color: #212121;  
color: #fff;  
border: none;  
padding: 10px 20px;  
cursor: pointer;  
font-size: 16px;  
}
```



```
.message {  
    padding: 10px;  
    border-radius: 5px;  
    margin: 5px;  
}  
  
.user-message {  
    background-color: #fcf6ee;  
    color: #000;  
    text-align: right;  
}  
  
.bot-message {  
    background-color: #ffffff;  
    color: #000;  
    text-align: left;  
}
```



```
</style>
</head>
<body>
  <div class="container">
    <div class="title">
      <h1>IBMBot</h1>
      <p>The Chatbot</p>
    </div>
    <div id="chat-container">
      <div id="chat-log"></div>
      <form id="chat-form" method="POST" action="/chat">
        <input type="text" name "user_input" id="user-input" placeholder="You: "
autocomplete="off" required>
        <input type="submit" value="Submit" id="submit-button">
      </form>
    </div>
  </div>
</body>
</html>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
<script>
```

```
const chatLog = document.getElementById('chat-log');
```

```
const userInputElement = document.getElementById('user-input');
```

```
const chatForm = document.getElementById('chat-form');
```

```
chatForm.addEventListener('submit', async (e) => {
```

```
  e.preventDefault();
```

```
  const user_input = userInputElement.value;
```

```
  userInputElement.value = '';
```

```
  chatLog.innerHTML += <div class="message user-message">You: ${user_input}
```

```
</div>;
```

```
const response = await fetch('/chat', {
  method: 'POST',
  body: new URLSearchParams({ user_input }),
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8',
  },
});
const botResponse = await response.text();
chatLog.innerHTML += <div class="message bot-message">IBMBot: ${botResponse}</div>;

if (botResponse.toLowerCase() === 'goodbye!') {
  userInputElement.disabled = true;
}
});
</script>
</body>
</html>
```

**THANK YOU
SO MUCH!**

