DESIGN OF A CHATBOT

Submitted as a project report for mini project
For the partial fulfillment of the degree of
Bachelor of Technology
in
Information Technology

by

Aritra sen Roll 2021ITB047
Ananya De Roll 2021ITB050
Arkya Ghosh Roll 2021ITB069

Under the supervision of

Dr. Surajit Kumar Roy
Associate Professor
Department of Information Technology
IIEST, Shibpur

May 2023

Department of Information Technology
Indian Institute of Engineering Science and Technology, Shibpur
P.O. Botanic Garden, Howrah 711103, WB, India
+91 33 26684561-3 (3 lines)/0521-5 (5 lines) X260
https://www.iiests.ac.in/IIEST/AcaUnitDetails/IT

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

Date:

TO WHOM IT MAY CONCERN

This is to certify that Aritra Sen roll no 2021ITB047, Arkya Ghosh roll no 2021ITB069 and Ananya De roll no 2021ITB050 have done their mini project on for the partial fulfilment of the degree of B.Tech. in Information Technology.

During this period they has completed the project. The report has fulfilled all the requirements as per the regulations of the institute and has reached the standard needed for submission.

Dr. Surajit Kumar Roy

Associate Professor

Signature of HOD, IT

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our esteemed mentor, Dr. Surajit Kumar Roy, Associate Professor in the Department of Information Technology at the Indian Institute of Engineering Science and Technology, Shibpur. His guidance, support, and encouragement have been invaluable in the successful completion of this project.

We would also like to extend our gratitude to our parents, whose unwavering love and support have been our constant source of strength. Their encouragement and belief in our abilities have been instrumental in our academic pursuits.

Our sincere thanks also go to our colleagues and friends, whose feedback, suggestions, and encouragement have helped us stay motivated throughout the project. Their support has been a source of inspiration for us.

Finally, we would like to acknowledge the contributions of the various resources that have been instrumental in our project, including textbooks, research papers, and online resources. We are grateful for the knowledge and insights gained from these resources, which have been essential in our learning journey.

Once again, we express our heartfelt gratitude to everyone who has contributed to the success of this project.

INDEX

Chapter 1

Introduction

A Chatbot is a computerized program that acts like a colloquist between the human and the bot, a virtual assistant that has become exceptionally popular in recent years mainly due to dramatic improvements in the areas like artificial intelligence, machine learning and other underlying technologies such as neural networks and natural language processing.

Chatbots can be programmed to perform a variety of tasks such as answering questions, processing orders, booking appointments, providing recommendations and alternative solutions, and more. They are becoming increasingly popular in customer service, as they can quickly and efficiently handle inquiries from multiple users 24/7 without the need for human intervention. Chatbots are also used in other industries for various purposes such as healthcare, education, finance, and entertainment.

1.1 CHATBOT AND ITS TYPES

1. Rule-Based Chatbots: These chatbots use pre-defined rules and require specific inputs from the user. They work based on the decision trees and follow a set of rules to guide the conversation. They have limited capabilities and cannot understand the context of a conversation.

2. AI Chatbots: These chatbots are based on machine learning algorithms and utilize natural language processing (NLP) to understand user input. They can learn from previous interactions and improve their responses over time. These chatbots have the ability to understand natural language and can engage in more complex conversations.

3. Voice-Activated Chatbots: These chatbots use speech recognition technology to communicate with users. They work with voice assistants such as Amazon Alexa, Google Assistant, and Apple Siri. They can understand natural language and engage in conversational interaction with users.

4. Hybrid Chatbots: These chatbots combine the capabilities of rule-based and AI chatbots. They use pre-defined rules for specific situations and can also learn from user interactions. They can understand natural language and provide personalized responses based on user behaviour.

5. Contextual Chatbots: These chatbots use machine learning algorithms and NLP to understand user input along with context. They analyse the conversation history to provide personalized responses and recommendations. These chatbots provide a more personalized and engaging user experience.

6. social media Chatbots: These chatbots are integrated into social media platforms like Facebook, Twitter, and LinkedIn. They interact with users on these platforms and respond to their queries and comments in real-time. They are used for customer service, lead generation, and social engagement.

1.2 MOTIVATION

Our key motivation for choosing AI chatbot is because chatbots can learn and improve over time through user interactions, so you do not need to have a complete set of rules in place from the start. This means that you can start with a simple chatbot and gradually add more features and capabilities over time as the chatbot learns from user interactions. In this project, we have harnessed the power of HTML, CSS, Bootstrap, Node.js, and the OpenAI API to create an interactive and intelligent conversational experience.

Our goal is to develop a chatbot that seamlessly interacts with users, providing them with valuable information, assistance, and engaging conversations. By leveraging the OpenAI API, we can tap into advanced natural language processing capabilities, enabling our chatbot to understand and respond to user queries with contextually relevant and coherent answers.

With a focus on user experience, we have designed an intuitive interface using HTML, CSS, JAVASCRIPT, Vite, and Bootstrap. The chat-like layout allows users to easily input their questions and receive prompt responses from the chatbot. Through the Node.js server, we establish the backbone of our chatbot, facilitating smooth communication between the user interface and the OpenAI API. Our chatbot is equipped to handle a wide range of tasks, from answering frequently asked questions and providing recommendations to assisting with specific inquiries or guiding users through processes. We have carefully designed a conversation flow that adapts to various user inputs, ensuring a dynamic and personalized interaction.

Throughout the development process, we prioritize user input validation to ensure the reliability and security of our chatbot. Extensive testing and iterative improvements guarantee a robust and user-friendly design.

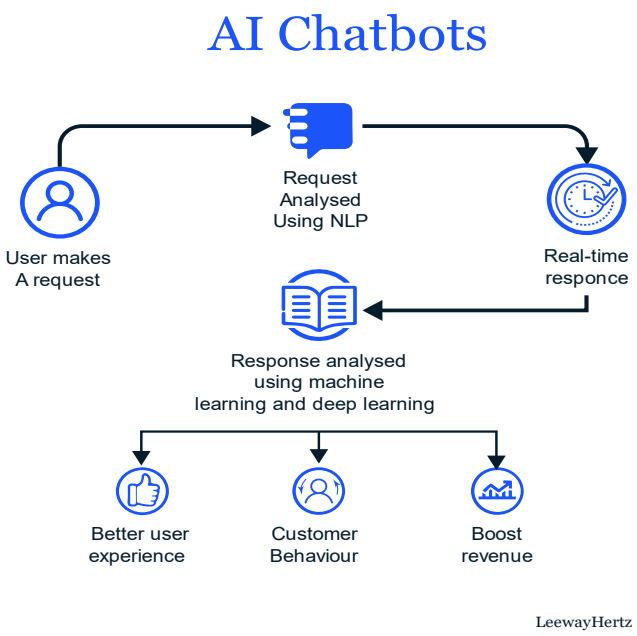
1.3 HOW DOES AI CHATBOT WORK?

AI chatbots work by using natural language processing (NLP) and machine learning (ML) algorithms to understand user input, analyze it, and generate a response. The chatbot software is trained on large amounts of data and patterns to improve its ability to understand and respond to user queries.

When a user sends a message or asks a question, the chatbot reads the text and applies NLP algorithms to understand the context, intent, and meaning of the message. Based on this analysis, the chatbot selects the appropriate response from a pre-existing database of responses, or generates a new response using ML algorithms and natural language generation (NLG) techniques.

The chatbot may use rule-based systems, pattern-matching techniques, or deep learning models to understand user input and generate a response. It may also incorporate sentiment analysis to detect the user's mood or emotions and tailor its response accordingly.

The chatbot interacts with users through a conversational interface, such as a messaging app, website, or voice assistant. The user can type or speak their questions or messages, and the chatbot responds in natural language, mimicking a human conversation. Some chatbots may also use multimedia, such as images, videos, or audio, to provide a more engaging and interactive experience for users.



1.4 BENEFITS AND USES OF AI CHATBOT

An AI-based chatbot is a chatbot that uses artificial intelligence (AI) technologies such as natural language processing (NLP), machine learning (ML), and deep learning (DL) to understand and respond to user queries and requests. They can also handle more complex requests and provide more accurate and relevant information to users. AI-based chatbots can be used in a wide range of applications, including customer service, e-commerce, healthcare, education, and many others. They can help businesses improve customer satisfaction, reduce costs, and increase efficiency by automating tasks and providing round-the-clock support. AI-based chatbots can be used in a wide range of applications, including customer service, e-commerce, healthcare, education, and many others. They can help businesses improve customer satisfaction, reduce costs, and increase efficiency by automating tasks and providing round-the-clock support.

Chapter 2

Literature Survey

2.1 CHATBOT ARCHITECTURE

Pattern Match: Chatbots use pattern-matching techniques to classify textual inputs and provide appropriate responses. Developers create repetitive behaviors using message patterns that match parts of customer messages. Chatbots can only answer questions if specific words are present in the patterns.

Algorithms: To get the chatbot ready with adequate responses, developers create a hierarchy with the permutation and combination of different patterns. Algorithms help make the structure more manageable by reducing the total number of different types of classifiers.

Chatbot architecture can vary depending on the specific use case and requirements. However, there are some common components that are typically involved in building a chatbot:

1. User Interface: This is the component that enables users to interact with the chatbot. It can be a simple text-based interface or a more advanced interface that incorporates multimedia elements such as images, videos, and audio.
2. Natural Language Processing (NLP): This is the component that enables the chatbot to understand and interpret user input. NLP involves techniques such as entity recognition, sentiment analysis, and intent classification.
3. Dialog Management: This component manages the conversation flow between the chatbot and the user. It determines the appropriate responses to user input based on the context of the conversation.
4. Backend Services: The backend services are responsible for providing the data and information needed by the chatbot to respond to user requests. These services can include databases, APIs, and other third-party services.
5. Machine Learning: Machine learning is used to improve the chatbot's performance over time by enabling it to learn from user interactions and improve its response accuracy.

2.2 CHATBOT EVALUATION AND APPLICATION

Chatbots are becoming increasingly popular and are being used in various industries such as healthcare, banking, retail, and customer service. The success of a chatbot largely depends on its ability to accurately understand and respond to user input, as well as its overall usability and user experience. Therefore, evaluating and testing chatbots is crucial to ensuring their effectiveness and usefulness.

One method of evaluating chatbots is through user testing, where a group of users interact with the chatbot and provide feedback on its performance. This feedback can be used to identify any areas where the chatbot may be lacking and make necessary improvements. Another method is through automated testing, which involves using software to simulate user interactions and test the chatbot's functionality and performance. When it comes to application, chatbots have the potential to revolutionize customer service by providing 24/7 support and reducing the workload for human agents. They can also be used for lead generation, appointment scheduling, and even as virtual assistants for tasks such as setting reminders and managing calendars.

Chatbots are rapidly gaining popularity across a wide range of industries, from healthcare to e-commerce, finance, and more. In the healthcare industry, chatbots are used to provide medical advice, symptom checking, and even diagnosis to patients. They can also be used to schedule appointments, manage patient records, and provide medication reminders. In e-commerce, chatbots can assist customers with purchasing products, tracking orders, and answering frequently asked questions. In finance, chatbots can help customers manage their accounts, provide financial advice, and assist with transactions. Chatbots are also being used in education, where they can provide personalized learning experiences, answer student queries, and assist with administrative tasks. The use of chatbots in various industries is expected to continue to grow as they become more advanced and capable of handling increasingly complex tasks.

2.3 CHATBOT CHALLENGES

While chatbots offer many benefits, they also face several challenges that need to be addressed for them to become even more useful. One major challenge is the ability of chatbots to understand and interpret natural language accurately. Chatbots need to be able to understand and respond to a wide range of user inputs and contexts, which can be difficult to achieve. Additionally, chatbots need to be able to learn and adapt to new information and user preferences over time, which can require significant amounts of data and resources. Another challenge is the need for chatbots to maintain a conversational flow and engage users in a way that feels natural and authentic. This requires chatbots to have a certain level of emotional intelligence and empathy to detect and respond appropriately to user emotions and needs. Chatbots also need to be secure and protect user data and privacy.

2.4 EXISTING PROBLEMS AND SOLUTIONS

Chatbots face several challenges that can hinder their performance and user experience. One major challenge is the ability to accurately understand and interpret user inputs, particularly in cases where users use informal language, slang, or misspell words. Additionally, chatbots must be able to respond appropriately and in a timely manner, without overwhelming users with too much information or irrelevant responses. Another challenge is maintaining context and continuity of the conversation, which can become challenging when users switch topics or ask unrelated questions. Finally, chatbots must be able to handle sensitive information, such as personal or financial data, in a secure and confidential manner.

To overcome these challenges, various solutions have been proposed and implemented. One solution is to use natural language processing (NLP) and machine learning algorithms to improve the accuracy and understanding of user inputs. Additionally, chatbots can be designed to provide relevant and concise responses, with the option for users to receive more detailed information if needed. Context management techniques, such as tracking previous interactions and using conversational history, can also help maintain continuity in the conversation. Finally, ensuring the security and privacy of user data can be achieved through robust encryption and secure data storage practices.

2.5 COMPARATIVE ANALYSIS

When it comes to chatbots, there are several options available in the market. OpenAI and Google are two of the most popular chatbot providers, but there are also other options worth considering. Here's a comparative analysis of OpenAI, Google, and other chatbot providers:

OpenAI: OpenAI is an artificial intelligence research laboratory consisting of the for-profit corporation OpenAI LP and its parent company, the non-profit OpenAI Inc. They have developed a chatbot called GPT-3 that uses natural language processing (NLP) to understand and generate human-like responses. GPT-3 is known for its ability to carry on lifelike conversations and has been used in a variety of applications, including customer service, content creation, and language translation.

Google: Google's chatbot, Google Assistant, is an AI-powered virtual assistant that is integrated into Google's products and services. It is capable of understanding natural language inputs and can perform a variety of tasks, including setting reminders, making reservations, and answering questions. Google Assistant is known for its accuracy and ease of use, and it can be accessed via voice or text commands.

Other chatbot providers: There are several other chatbot providers in the market, including Microsoft's Cortana, Amazon's Alexa, and Facebook's Messenger bot. These chatbots use a combination of NLP and machine learning algorithms to understand user inputs and generate appropriate responses. While they may not be as advanced as OpenAI or Google's chatbots, they are still effective tools for a variety of applications. In terms of comparative analysis, OpenAI's GPT-3 is considered to be the most advanced chatbot available. It has the ability to understand and generate human-like responses, and its NLP capabilities are unparalleled in the industry. Google Assistant is also a popular choice due to its accuracy and ease of use, especially for voice commands. However, it may not be as advanced as GPT-3 when it comes to carrying on complex conversations. Other chatbot providers, such as Microsoft's Cortana, Amazon's Alexa, and Facebook's Messenger bot, are also effective tools for various applications.

Chapter 3

Design and Implementation

3.1 FRONTEND

As we focus on building a chatbot which uses AI model, creating an eye catchy model, is important. Designing a good frontend is crucial for chatbots because it directly impacts the user experience and overall satisfaction with the chatbot. A well-designed frontend can make the chatbot more visually appealing and user-friendly, allowing users to easily navigate and interact with the chatbot.

A good frontend design can also enhance the chatbot's credibility, making it more trustworthy and professional. It can also improve the chatbot's ability to communicate effectively with users, by using appropriate fonts, colors, and graphics to convey information.

Furthermore, a good frontend design can help the chatbot stand out from competitors, creating a unique brand identity and building customer loyalty. This is especially important in today's market, where chatbots are becoming increasingly common and users have many options to choose from.

HTML, CSS, and JavaScript are the building blocks of the web. HTML provides the structure of a webpage, CSS provides styling and layout, and JavaScript provides interactivity and functionality.

Vite is a build tool for JavaScript projects that provides a development server, hot module replacement, and other optimizations to make development faster and more efficient. It is used to bundle the JavaScript files, optimize the code, and create the final build for deployment.

To create a chatbot using these technologies, the frontend design of the chatbot is created using HTML and CSS. JavaScript is used to handle user interactions, send requests to the backend server, and dynamically update the UI with responses from the server.

The chatbot can be integrated with a backend server that uses machine learning algorithms, natural language processing, and other AI techniques to understand user queries and provide appropriate responses. The frontend chatbot then displays these responses to the user.

Overall, the frontend technologies such as HTML, CSS, and JavaScript, combined with a build tool like Vite, can be used to create a responsive and interactive chatbot that can communicate with users in a natural and engaging way.

3.2 BACKEND

Introduction:

The OpenAI API is a powerful tool for creating intelligent and conversational chatbots. The API provides developers with access to a variety of natural language processing models, including GPT-3, which can be used to generate responses to user input. In this report, we will discuss the implementation of OpenAI API in the backend of our chatbot, its working, and possible applications.

Implementation:

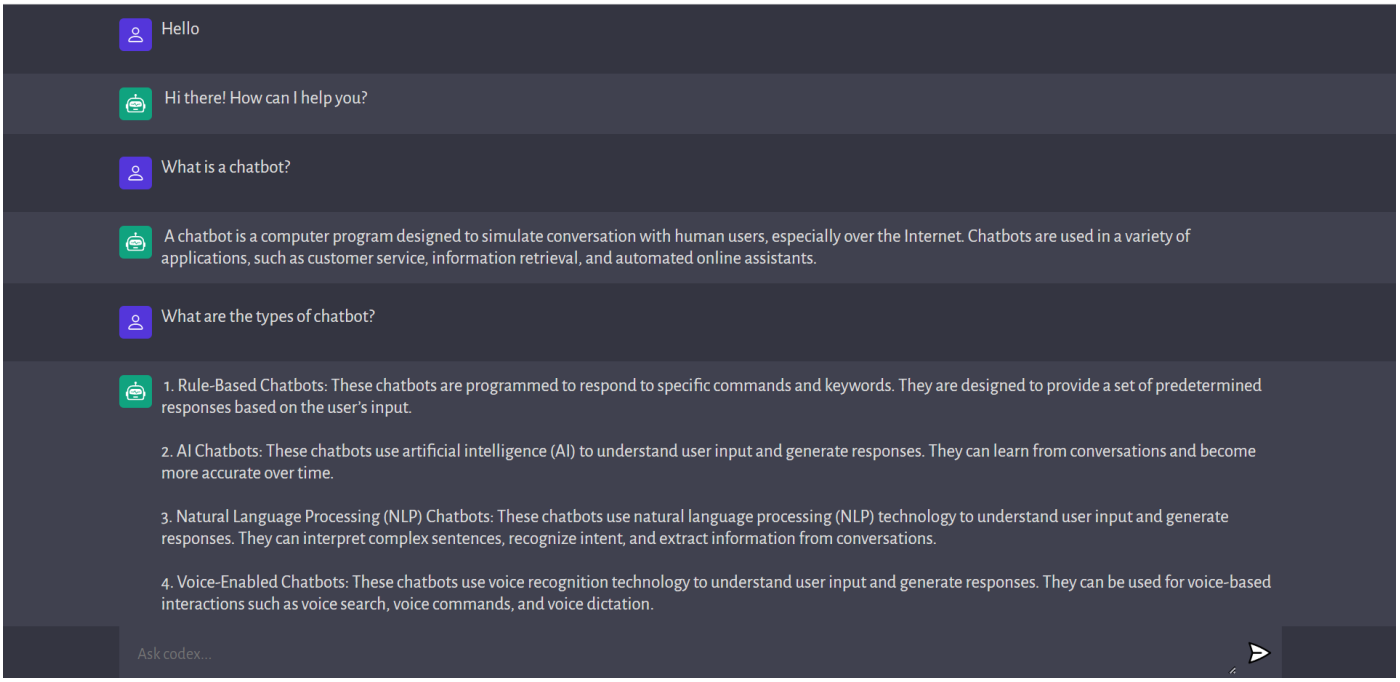
We used the OpenAI API to create a backend for our chatbot. The API allows us to send user input to OpenAI's natural language processing models, which generate a response that is returned to the user. The chatbot can use a variety of models to generate responses, including GPT-3, which is one of the most advanced natural language processing models available.

Working:

When a user sends a message to the chatbot, the message is first processed by the frontend. The frontend then sends the message to the backend, which is powered by the OpenAI API. The backend sends the user's message to the appropriate natural language processing model, which generates a response based on the input. The response is then returned to the backend, which sends it back to the frontend. The frontend displays the response to the user, who can then continue the conversation.

Applications:

The OpenAI API can be used in a variety of applications, including customer service, virtual assistants, and language translation. With the ability to generate natural and conversational responses, chatbots powered by the OpenAI API can provide users with a more engaging and helpful experience. Additionally, the API can be used to automate customer support tasks, freeing up staff to focus on more complex issues.



Chapter 4

Code and Output

HTML CODE SNIPPET

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="./assets/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Codex - Your AI</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div id="app">
      <div id="chat_container"></div>

      <form>
        <textarea name="prompt" rows="1" cols="1" placeholder="Ask codex..."></textarea>
        <button type="submit"><img src="assets/send.svg" alt="send" />
      </form>
    </div>

    <script type="module" src="./script.js"></script>
  </body>
</html>
```

The HTML markup defines the layout of the page, with a container for the chat messages (id="chat_container") and a form that allows the user to input a message to the AI assistant (with a textarea element and a submit button).

CSS CODE SNIPPET

```
.ai {
  background: #40414F;
}

.chat {
  width: 100%;
  max-width: 1280px;
  margin: 0 auto;

  display: flex;
  flex-direction: row;
  align-items: flex-start;
  gap: 10px;
}
```

```
.profile {
  width: 36px;
  height: 36px;
  border-radius: 5px;

  background: #5436DA;

  display: flex;
  justify-content: center;
  align-items: center;
}

.ai .profile {
  background: #10a37f;
}
```

```
form {
  width: 100%;
  max-width: 1280px;
  margin: 0 auto;
  padding: 10px;
  background: #40414F;

  display: flex;
  flex-direction: row;
  gap: 10px;
}
form img {
  width: 30px;
  height: 30px;
}
```

1. The .ai and .chat selectors style the elements that contain the chat messages, including their background colours, size, and layout properties.
2. The .profile selector styles the element that displays the profile image of the chat participant, including its size, shape, and background colour.
3. The .message selector styles the element that displays the actual chat message, including its size, font color, overflow behavior, and white-space handling.
4. The form selector styles the form element that contains the input field and send button, including its size, background color, and layout properties.
5. The text area selector styles the input field where the user can type their message, including its size, font color, padding, border radius, and border.

JAVASCRIPT SNIPPET

```
function loader(element) {
    element.textContent = ''

    loadInterval = setInterval(() => {
        // Update the text content of the
loading indicator
        element.textContent += '.';

        // If the loading indicator has
reached three dots, reset it
        if (element.textContent === '....') {
            element.textContent = '';
        }
    }, 300);
}
```

```
function typeText(element, text) {
    let index = 0

    let interval = setInterval(() => {
        if (index < text.length) {
            element.innerHTML +=
text.charAt(index)
            index++
        } else {
            clearInterval(interval)
        }
    }, 20)
}
```

function loader(element)

The code defines a function called "loader" that takes an HTML element as its argument. The function clears the text content of the element and then sets an interval to append a period to the text content every 300 milliseconds. If the text content reaches "....", it is reset to an empty string. The function appears to be used to create a loading animation.

function typeText(element, text)

This is a JavaScript function that types out text one character at a time on a specified HTML element. It takes two parameters: element, which is the HTML element where the text will be displayed, and text, which is the string of text to be displayed. The function uses a setInterval method to repeatedly add one character of the text to the element until the entire text has been typed out, at which point the interval is cleared. The speed of typing can be adjusted by changing the value in the setInterval method.

```
function generateUniqueId() {
   const timestamp = Date.now();
   const randomNumber = Math.random();
   const hexadecimalString =
randomNumber.toString(16);

   return `id-${timestamp}-
${hexadecimalString}`;
}

function chatStripe(isAi, value, uniqueId) {
   return (
      `
      <div class="wrapper ${isAi && 'ai'}">
         <div class="chat">
            <div class="profile">
               <img
                 src=${isAi ? bot : user}
                 alt="${isAi ? 'bot' : 'user'}"
               />
            </div>
            <div class="message"
id=${uniqueId}>${value}</div>
         </div>
      </div>
   )
}
```

```
clearInterval(loadInterval)
   messageDiv.innerHTML = " "

   if (response.ok) {
      const data = await response.json();
      const parsedData = data.bot.trim() // trims
any trailing spaces/'\n'

      typeText(messageDiv, parsedData)
   } else {
      const err = await response.text()

      messageDiv.innerHTML = "Something went
wrong"
      alert(err)
   }
}

form.addEventListener('submit', handleSubmit)
form.addEventListener('keyup', (e) => {
   if (e.keyCode === 13) {
      handleSubmit(e)
   }
})
```

1. This is a function called generateUniqueId that generates a unique ID. It does this by combining the current timestamp and a random number to create a hexadecimal string. The function then returns a string that begins with "id-" followed by the timestamp and the hexadecimal string. This ensures that each ID generated is unique, as the timestamp is constantly increasing and the random number ensures that the hexadecimal string is unique.
2. The chatStripe function takes three parameters: isAi, value, and uniqueId. It returns a string containing HTML markup for a chat message with an avatar image, a message content, and a unique ID. The isAi parameter is a boolean value that determines whether the message belongs to an AI or a user. The value parameter is a string that contains the message content. The uniqueId parameter is a string that contains a unique identifier for the message. The returned string of HTML markup will be used to display the chat messages in the chat application.
3. The handleSubmit function is called when the form is submitted or when the enter key is pressed while the form is in focus. In handleSubmit, the loader function is first called to display a loading indicator on the page. A fetch request is then made to an API with the user input from the form. If the response is successful, the parsedData from the response is extracted and passed to the typeText function, which types out the text on the page character by character. If the response is unsuccessful, an error message is displayed on the page and an alert is shown with the error message from the response.

VITE CODE SNIPPET

Counter.js

```
export function setupCounter(element) {
  let counter = 0
  const setCounter = (count) => {
    counter = count
    element.innerHTML = `count is ${counter}`
  }
  element.addEventListener('click', () =>
setCounter(counter + 1))
  setCounter(0)
}
```

```
    <img src="${javascriptLogo}" class="logo vanilla"
alt="JavaScript logo" />
    </a>
    <h1>Hello Vite!</h1>
    <div class="card">
      <button id="counter" type="button"></button>
    </div>
    <p class="read-the-docs">
      Click on the Vite logo to learn more
    </p>
  </div>
`setupCounter(document.querySelector('#counter'))
```

Vite is a build tool and development server that focuses on fast development and optimized production builds. It uses modern browser capabilities like native ES modules and dynamic imports to enable fast and efficient development.

In this specific code snippet, we have an HTML file that includes two images (JavaScript logo and Vite logo), a header ("Hello Vite!"), a button ("count is 0"), and a paragraph ("Click on the Vite logo to learn more"). The images are imported from their respective SVG files and the button functionality is implemented with a setupCounter function defined in a separate file called counter.js. When the button is clicked, the counter increases by one and the new count is displayed on the button.

Backend

```
import express from 'express'
import * as dotenv from 'dotenv'
import cors from 'cors'
import { Configuration, OpenAIApi } from 'openai'

dotenv.config()

const configuration = new Configuration({
  apiKey: process.env.OPENAI_API_KEY,
});

const openai = new OpenAIApi(configuration);

const app = express()
app.use(cors())
app.use(express.json())

app.get('/', async (req, res) => {
  res.status(200).send({
    message: 'Hello from CodeX!'
  })
})
```

```
app.post('/', async (req, res) => {
  try {
    const prompt = req.body.prompt;
    const response = await openai.createCompletion({
      model: "text-davinci-003",
      prompt: `${prompt}`, temperature: 0,
      max_tokens: 3000,
      top_p: 1,
      frequency_penalty: 0.5,
      presence_penalty: 0,
    });
  res.status(200).send({
      bot: response.data.choices[0].text
    });
  } catch (error) {
    console.error(error)
    res.status(500).send(error || 'Something went
wrong'); }
})
app.listen(5000, () => console.log('AI server started on
http://localhost:5000'))
```

This is a Node.js Express server that uses the OpenAI API to create a chatbot. It listens on port 5000 and has two routes: a GET route that returns a simple "Hello from CodeX!" message, and a POST route that takes a prompt as input and sends it to the OpenAI API to generate a response using the "text-davinci-003" model with specific parameters. The response generated by the API is returned to the client as the "bot" property of a JSON object. The server uses environment variables to get the OpenAI API key.

Package. json

```
{
  "name": "codex-server",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "server": "nodemon server"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "nodemon": "^2.0.20",
    "openai": "^3.1.0"
  }
}
```

This is a package.json file, which is used by Node.js to manage the dependencies and scripts of a project.

In this specific case, the name of the project is "codex-server" and version is "0.0.0". The private property is set to true, which means that this project is intended to be used locally and not published to any public registry.

The "type": "module" property indicates that this is a JavaScript module.

Under "dependencies", there are several packages listed that are used in this project, including cors, dotenv, express, nodemon, and openai. These packages can be installed using the npm install command.

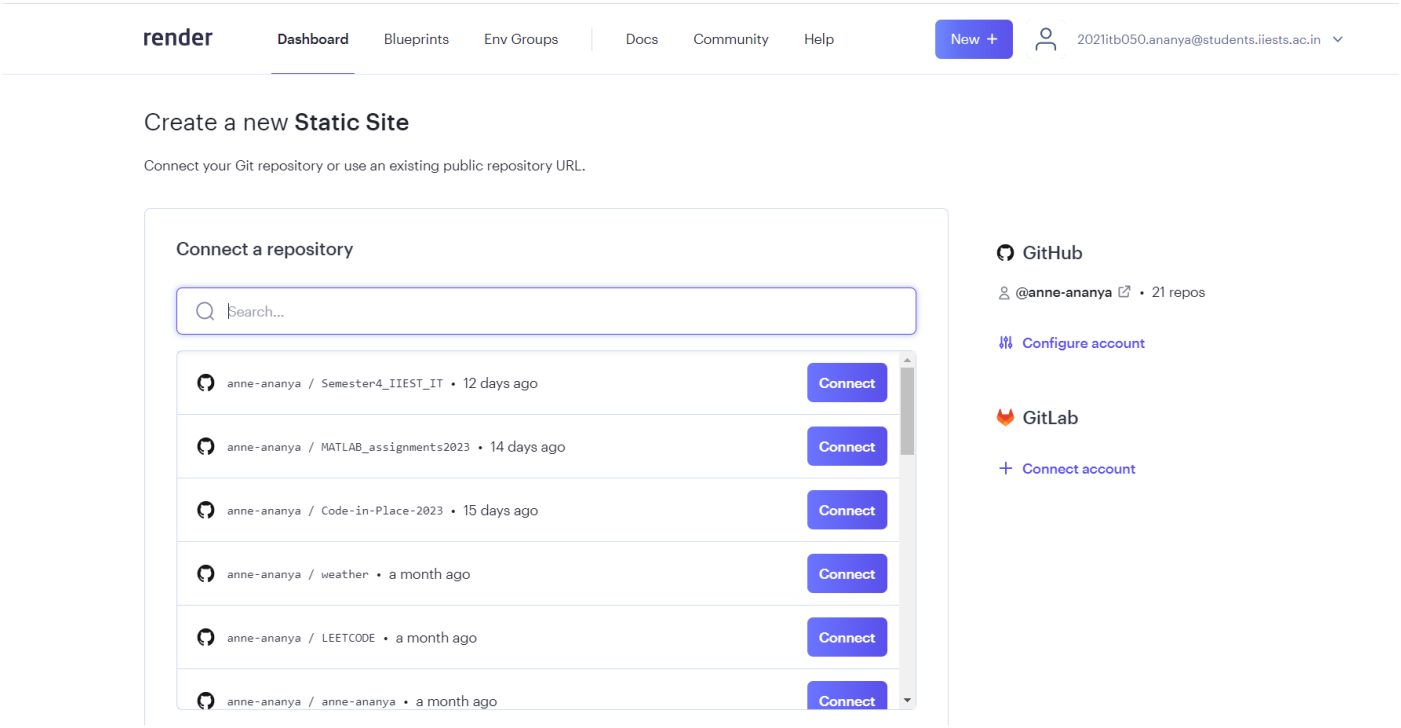
The "scripts" section includes a script named server which runs the server using nodemon, which is a tool that automatically restarts the server when changes are made to the code.

Chapter 5

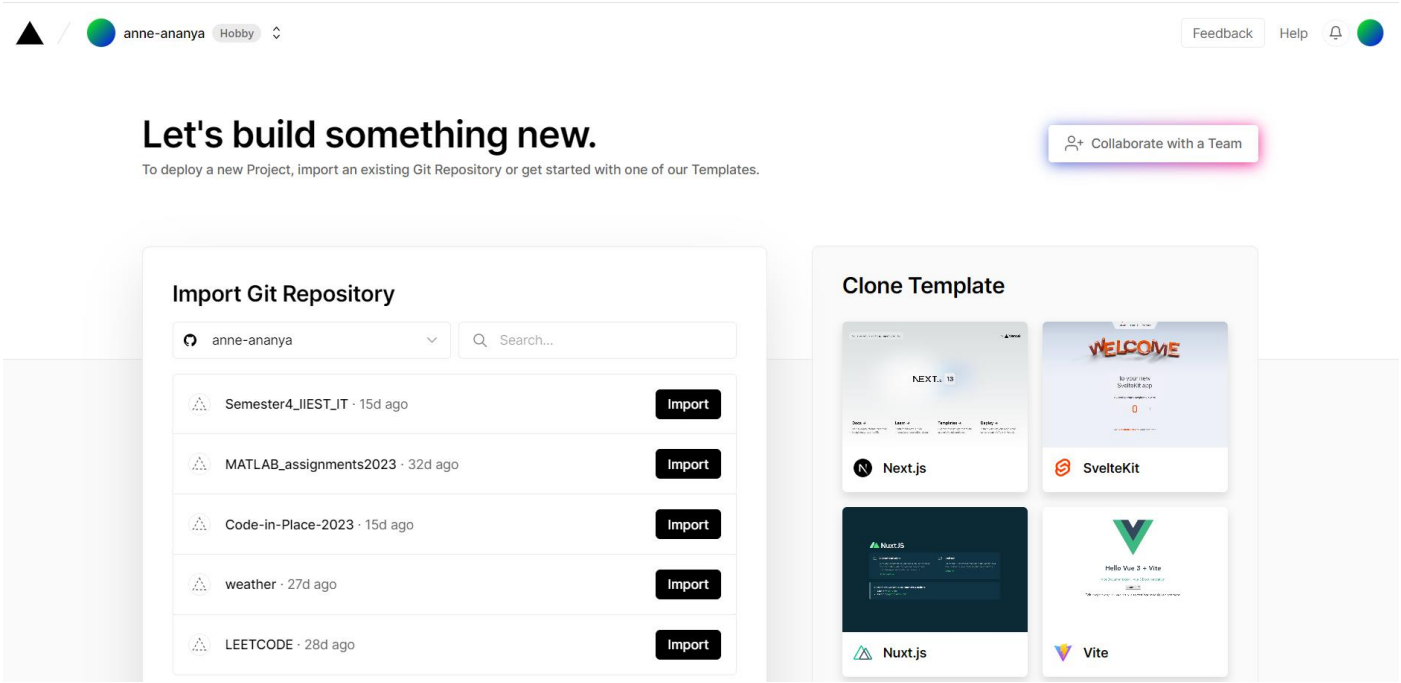
Deployment

The deployment of a chatbot's frontend and backend involves several steps and procedures to ensure that the application runs smoothly and without any issues. In our project, we designed the frontend using HTML, CSS, and JavaScript, while the backend was powered by the OpenAI API.

To deploy the chatbot's frontend, we used Render, a cloud platform that provides simple and scalable web hosting. The first step involved uploading the frontend code to a GitHub repository and connecting it to Render. We then configured the deployment settings, including specifying the type of application and selecting the appropriate runtime environment.



Next, we deployed the chatbot's backend on Vercel, another cloud platform that specializes in serverless functions. We created a serverless function to handle requests from the frontend and connect to the OpenAI API to generate responses. We also configured the necessary environment variables to store sensitive information such as API keys and access tokens.



Once both the frontend and backend were deployed, we tested the application thoroughly to ensure that it was working as intended. We also monitored the application's performance and scalability to ensure that it could handle a high volume of traffic and requests.

Throughout the deployment process, we took care to follow best practices and security protocols to protect sensitive information and prevent unauthorized access. We also documented the procedures and steps involved in the deployment process to make it easier for future developers to maintain and update the application.

Overall, the successful deployment of the chatbot's frontend and backend on Render and Vercel demonstrates the importance of careful planning, attention to detail, and adherence to best practices when developing and deploying web applications.

Chapter 6

Conclusion and Future Scope

The emergence of AI and ML has led to significant changes in various industries, including the use of chatbots as a tool to interact with customers. There are now numerous chatbot-building platforms available for different industries, such as e-commerce, banking, healthcare, and leisure, among others. Chatbots are highly effective in reaching a vast audience on messaging apps, and they are increasingly becoming more capable of gathering information from customers.

In this project, we designed a chatbot using front-end technologies such as HTML, CSS, and JavaScript, with the back-end powered by the OpenAI API. We deployed the front-end on Render and the back-end on Vercel. The process involved several steps, such as creating the user interface, integrating the OpenAI API, and deploying the application. We also had to take care to ensure that the project was scalable, secure, and reliable.

As students, this project presented us with various challenges, and we had to come up with solutions and strategies to overcome them. We learned the importance of taking a problem head-on and never backing down from learning something new. Overall, the project was a great experience that allowed us to gain hands-on experience with AI and ML tools and technologies. We believe that this experience will be beneficial to us in our future careers and endeavours.

The future of AI chatbot technology is very promising, as innovative advancements are revolutionizing the industry every year. While chatbots were once limited to pre-programmed responses, today's cutting-edge versions are capable of carrying on natural and lifelike conversations thanks to natural language processing. However, there are still key challenges that must be overcome for further development. Increased investment and new technical innovations will make AI chatbots much more secure, capable, and versatile in the years to come. These bots will become accessible, trustworthy communication tools, benefitting organizations and users in virtually every industry and niche.

Over the next decade, advancements in AI chatbots will open up many more applications beyond customer service. For example, new chatbots with advanced translation capabilities could help companies expand globally and improve international customer service. Similarly, AI chatbots will likely become more popular in human resources departments worldwide, gaining ground in employee training, IT help, and administrative assistance functions. As these bots often collect personal information, protection will be a top priority for future developments. Overall, the future of AI chatbots is exciting and holds great potential for innovation and growth.

References

1. Vite documentation
   https://vitejs.dev/
2. Chatbot basics
   https://www.oracle.com/in/chatbots/what-is-a-chatbot/
3. Youtube video
   https://www.youtube.com/live/JzPgeRJfNo4?feature=share
4. Future of chatbot

   https://link.springer.com/article/10.1007/s00607-021-01016-7

5. Open AI API
   https://platform.openai.com/docs/api-reference/introduction
6. Comparative analysis of chatbots.
   https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3563674