ChatBot Project Report

This project report provides a comprehensive overview of the development and implementation of a chatbot using HTML, CSS, and JavaScript. The chatbot currently has limited capabilities, with plans to integrate 7 or more AI models to expand its functionality and cater to users' diverse needs.

The project overview section delves into the background and context of the chatbot project. It highlights the objectives, methodologies, and technologies used in the development process. The implementation details section provides a deeper insight into the technical aspects of building the chatbot, including the programming languages, frameworks, and libraries utilized.

Furthermore, the future enhancements section outlines the roadmap for the chatbot's growth. It discusses the plans to integrate additional AI models and improve the chatbot's natural language processing capabilities. The challenges faced section reflects on the obstacles encountered during the development journey and the strategies employed to overcome them.

In conclusion, this project report showcases the progress made in creating a functional chatbot using HTML, CSS, and JavaScript. With future enhancements and advancements in AI integration, the chatbot holds the potential to become an even more effective and versatile conversational agent.

Created by

Yashwant Sharma, Karan Sharma

MCA (AI)

Shoolini University

Introduction

Welcome to the ChatBot Project Report, Yashwant Sharma and Karan Sharma! This report provides an in-depth exploration of the development and implementation of our chatbot at Shoolini University.
In the following sections, we will cover various aspects of the project, including the project overview, objectives, methodology, technologies used, implementation details, future enhancements, challenges faced, and conclusion. Each section will provide valuable insights into our journey of creating and deploying the chatbot.

We'll delve into the objectives, methodologies, and technologies used to develop our chatbot. By understanding the goals we aimed to achieve, the approach we took, and the tools we utilized, you'll gain a comprehensive understanding of our project.

In addition, we'll discuss the future enhancements we have in mind to further improve the chatbot's functionality and user experience. Furthermore, we'll shed light on the challenges we encountered during the development process and how we overcame them.

So, let's get started on this exciting journey through the ChatBot Project Report!

Table of Contents

Project Overview

1 **The primary focus of this project**

Creating a chatbot that could engage in basic conversations and provide users with a convenient way to access information or assistance.

2 **Development approach**

The chatbot was developed using a combination of HTML, CSS, and JavaScript to deliver a user-friendly and responsive interface.

3 **Current chatbot capabilities**

The chatbot can respond to a limited set of questions.

4 **Future plans for customization**

We have plans to integrate 7 or more AI models into the chatbot for user customization.

Objectives

The main objectives of the AI-Powered Chatbot project are as follows:

1. **Develop a user-friendly and visually appealing chatbot interface**
2. **Implement basic conversational functionality using JavaScript**
3. **Integrate seven or more AI models to provide users with customization options**
4. **Enhance the chatbot's natural language processing capabilities**
5. **Improve the overall user experience through personalized interactions**

Methodology

1 **Design the chatbot interface**

Using HTML and CSS to create an engaging user experience

2 **Implement the chatbot's basic functionality**

Using JavaScript, including handling user inputs and generating responses

3 **Define a set of prompts and replies**

To enable the chatbot to engage in conversations

4 **Utilize regular expressions**

For text preprocessing and pattern matching

5 **Incorporate text-to-speech functionality**

To provide an audio output for the chatbot's responses

6 **Integrate multiple AI models**

To offer users a choice of conversational styles and capabilities

Technologies Used

Front-end

HTML, CSS, and JavaScript were the primary technologies used for the chatbot's development. These languages were chosen for their widespread adoption, ease of use, and ability to create responsive and interactive user experiences.

Back-end

The chatbot's logic and response generation were handled entirely on the client-side, leveraging JavaScript's capabilities. This approach was selected to maintain a simple and efficient architecture, without the need for a dedicated server-side component.

Future Integrations

The project has been designed with the flexibility to incorporate 7 or more AI models in the future, allowing the chatbot to expand its capabilities and cater to a wider range of user needs.

Implementation Details

The implementation of the AI-Powered Chatbot involves several key components:

HTML Structure:

- The chatbot interface is structured using HTML, with a container div housing the chat window and an input field for user messages

- The chat window is implemented as a div element with a messages div to display the conversation history

- An image element is used to display the chatbot's avatar

CSS Styling:

- CSS is used to style the chatbot interface, providing an appealing and user-friendly design

- Flexbox is utilized for layout and positioning of elements, ensuring a responsive design

- Custom styles are applied to user and bot messages to differentiate between the two

JavaScript Functionality:

- JavaScript is used to implement the chatbot's core functionality

- Arrays of prompts and corresponding replies are defined to handle different user inputs

- Event listeners are used to capture user input and trigger the appropriate response generation

- Regular expressions are employed for text preprocessing and pattern matching

- The compare function is used to find matching replies based on user input

- The DOM is dynamically updated to display user and bot messages in the chat window

- Text-to-speech functionality is incorporated using the Web Speech API

Future Enhancements

1

AI Model Integration

Expand the chatbot's capabilities by integrating 7 or more AI models, allowing users to choose the model that best suits their needs and preferences.

2

Natural Language Processing

Implement more advanced natural language processing techniques to enhance the chatbot's understanding of user input and provide more contextual and relevant responses.

3

Personalization

Develop features that enable the chatbot to learn and adapt to individual user preferences, creating a more personalized experience over time.

Challenges Faced

Technical Complexity

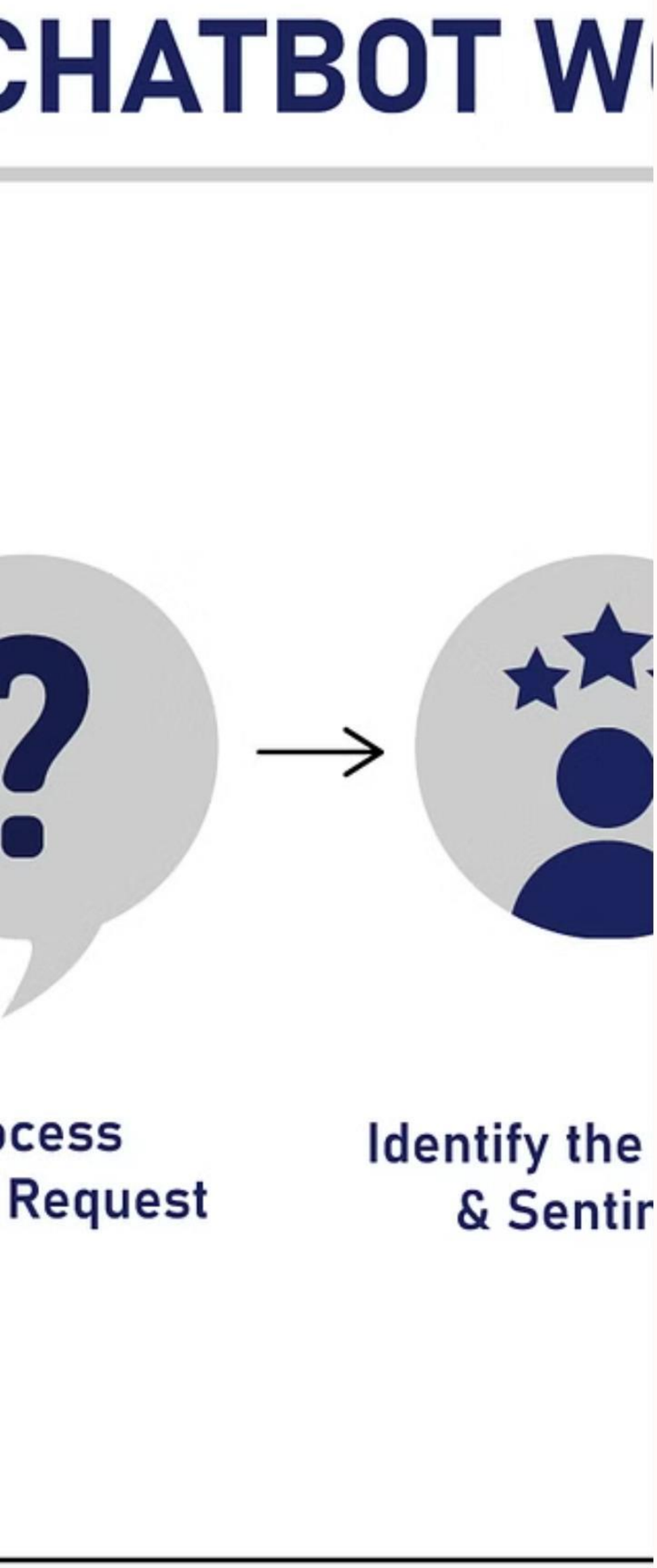
Integrating the various technologies (HTML, CSS, JavaScript) and ensuring seamless functionality required careful planning and coordination among the development team.

User Experience

Designing a user-friendly interface that could effectively guide users through the chatbot's capabilities was an iterative process that involved constant feedback and refinement.

Scalability

Preparing the chatbot's architecture to accommodate future enhancements, such as the integration of multiple AI models, required a forward-thinking approach to ensure scalability and maintainability.



Conclusion

Valuable Learning Experience

Showcasing the potential of combining HTML, CSS, and JavaScript to create engaging and responsive user interfaces

Future Enhancements

Promising opportunities to expand the chatbot's functionality and provide users with a more comprehensive and personalized assistant

Advanced AI Integration

Integrating advanced natural language processing algorithms and machine learning models to enhance the chatbot's ability to understand and respond to user queries

Overcoming Challenges

Gaining valuable insights into the challenges faced during the development process and devising effective solutions

Creating Intelligent Interfaces

A significant milestone in our journey towards creating intelligent and interactive conversational interfaces

MY HTML Code

```
<!DOCTYPE html>

<html>
<head>
<title>Chatbot</title>
<link rel="icon" href="bot.png" />
<link rel="stylesheet" href="style.css" />
</head>
<body>
<div id="container" class="container">
<div id="chat" class="chat">
<div id="messages" class="messages"></div>
<input id="input" type="text" placeholder="Say something..." autocomplete="off"
autofocus="true" />
</div>
<img src="bot.png" alt="Robot cartoon" height="500vh">
</div>
</body>
<script type="text/javascript" src="index.js" ></script>
<script type="text/javascript" src="constants.js" ></script>
<script type="text/javascript" src="speech.js" ></script>
</html>
```

My CSS Code

```
* {
  box-sizing: border-box;
}

html {
  height: 100%;
}

body {
  font-family: 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue', Arial, Helvetica,
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background-color: rgba(55, 234, 15, 0.5);
  height: 100%;
  margin: 0;
}

span {
  padding-right: 15px;
  padding-left: 15px;
}

.container {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100%;
}

.chat {
  height: 300px;
  width: 50vw;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

::-webkit-input-placeholder {
  color: .711;
}

input {
  border: 0;
  padding: 15px;
  margin-left: auto;
```

```
  border-radius: 10px;
}

.messages {
  display: flex;
  flex-direction: column;
  overflow: scroll;
  height: 90%;
  width: 100%;
  background-color: white;
  padding: 15px;
  margin: 15px;
  border-radius: 10px;
}

#bot {
  margin-left: auto;
}

.bot {
  font-family: Consolas, 'Courier New', Menlo, source-code-pro, Monaco,
  monospace;
}

.avatar {
  height: 25px;
}

.response {
  display: flex;
  align-items: center;
  margin: 1%;
}

/* Mobile */

@media only screen and (max-width: 980px) {
  .container {
    flex-direction: column;
    justify-content: flex-start;
  }
  .chat {
    width: 75vw;
    margin: 10vw;
  }
}
```

MY JS Code for reply

```
// Options the user could type in
const prompts = [
["hi", "hey", "hello", "good morning", "good afternoon"],
["how are you", "how is life", "how are things"],
["what are you doing", "what is going on", "what is up"],
["how old are you"],
["who are you", "are you human", "are you bot", "are you human or bot"],
["who created you", "who made you"],
[
"your name please",
"your name",
"may i know your name",
"what is your name",
"what call yourself"
],
["i love you"],
["happy", "good", "fun", "wonderful", "fantastic", "cool"],
["bad", "bored", "tired"],
["help me", "tell me story", "tell me joke"],
["ah", "yes", "ok", "okay", "nice"],
["bye", "good bye", "goodbye", "see you later"],
["what should i eat today"],
["bro"],
["what", "why", "how", "where", "when"],
["no","not sure","maybe","no thanks"],
[""],
["haha","ha","lol","hehe","funny","joke"]
]
// Possible responses, in corresponding order
const replies = [
["Hello!", "Hi!", "Hey!", "Hi there!","Howdy"],
[
"Fine... how are you?",
"Pretty well, how are you?",
"Fantastic, how are you?"
],
[
"Nothing much",
"About to go to sleep",
"Can you guess?",
"I don't know actually"
],
["I am infinite"],
["I am just a bot", "I am a bot. What are you?"],
["The one true God, JavaScript"],
["I am nameless", "I don't have a name"],
["I love you too", "Me too"],
["Have you ever felt bad?", "Glad to hear it"],
["Why?", "Why? You shouldn't!", "Try watching TV"],
["What about?", "Once upon a time..."],
["Tell me a story", "Tell me a joke", "Tell me about yourself"],
["Bye", "Goodbye", "See you later"],
["Sushi", "Pizza"],
["Bro!"],
["Great question"],
["That's ok","I understand","What do you want to talk about?"],
["Please say something :("],
["Haha!","Good one!"]
]
// Random for any other user input
const alternative = [
"Same",
"Go on...",
"Bro...",
"Try again",
"I'm listening...",
"I don't understand :/"
]
// Whatever else you want :)
const coronavirus = ["Please stay home", "Wear a mask", "Fortunately, I don't have COVID", "These are uncertain times"]
```

JS: For Index Page

```
document.addEventListener("DOMContentLoaded", () => {
 const inputField = document.getElementById("input");
inputField.addEventListener("keydown", (e) => {
if (e.code === "Enter") {
let input = inputField.value;
inputField.value = "";
output(input);
 }
 });
});
function output(input) {
let product;
// Regex remove non word/space chars
// Trim trailing whitespce
// Remove digits - not sure if this is best
// But solves problem of entering something like 'hi1'
let text = input.toLowerCase().replace(/[^\w\s]/gi, "").replace(/[\d]/gi,
"").trim();
text = text
 .replace(/ a /g, " ") // 'tell me a story' -> 'tell me story'
 .replace(/i feel /g, "")
 .replace(/whats/g, "what is")
 .replace(/please /g, "")
 .replace(/ please/g, "")
 .replace(/r u/g, "are you");
if (compare(prompts, replies, text)) {
// Search for exact match in prompts
product = compare(prompts, replies, text);
```

```
 } else if (text.match(/thank/gi)) {

product = "You're welcome!"

 } else if (text.match(/(corona|covid|virus)/gi)) {

// If no match, check if message contains coronavirus

product = coronavirus[Math.floor(Math.random() * coronavirus.length)];

 } else {

// If all else fails: random alternative

product = alternative[Math.floor(Math.random() * alternative.length)];

 }

// Update DOM

addChat(input, product);

}

function compare(promptsArray, repliesArray, string) {

let reply;

let replyFound = false;

for (let x = 0; x < promptsArray.length; x++) {

for (let y = 0; y < promptsArray[x].length; y++) {

if (promptsArray[x][y] === string) {

let replies = repliesArray[x];

reply = replies[Math.floor(Math.random() * replies.length)];

replyFound = true;

// Stop inner loop when input value matches prompts

break;

 }

 }

if (replyFound) {

// Stop outer loop when reply is found instead of interating through the entire
array

break;
```

```
 }

 }

return reply;

}

function addChat(input, product) {

const messagesContainer = document.getElementById("messages");

let userDiv = document.createElement("div");

userDiv.id = "user";

userDiv.className = "user response";

userDiv.innerHTML = <img src="user.png" class="avatar"><span>${input}</span>;

messagesContainer.appendChild(userDiv);

let botDiv = document.createElement("div");

let botImg = document.createElement("img");

let botText = document.createElement("span");

botDiv.id = "bot";

botImg.src = "bot-mini.png";

botImg.className = "avatar";

botDiv.className = "bot response";

botText.innerText = "Typing...";

botDiv.appendChild(botText);

botDiv.appendChild(botImg);

messagesContainer.appendChild(botDiv);

// Keep messages at most recent

messagesContainer.scrollTop = messagesContainer.scrollHeight -
messagesContainer.clientHeight;

// Fake delay to seem "real"

setTimeout(() => {

botText.innerText = ${product};

textToSpeech(product)

 }, 2000)
```

JS for Voice

```
// Text to Speech

const synth = window.speechSynthesis;

const textToSpeech = (string) => {
  let voice = new SpeechSynthesisUtterance(string);
  voice.text = string;
  voice.lang = "en-US";
  voice.volume = 1;
  voice.rate = 1;
  voice.pitch = 1; // Can be 0, 1, or 2
  synth.speak(voice);
}
```