REPUBLIC OF TÜRKİYE
IZMIR BAKIRCAY UNIVERSITY
FACULTY OF ECONOMICS AND ADMINISTRATIVE SCIENCES

IMAGINATION AI

Aleyna Kurtkaya - 181005031
Enes Alçiçek - 181005018
Mehmet Kurttekin - 181005016
Seçil Yıldırım – 181005056

GRADUATION PROJECT

Department of Management Information Systems

June, 2023

Table of Contents

ABSTRACT

IMAGINATION AI

181005031 ALEYNA KURTKAYA

181005056 SEÇİL YILDIRIM

181005016 MEHMET KURTTEKİN

181005018 ENES ALÇİÇEK

Department of Management Information Systems

Graduation Project

Supervisor: Prof. Dr. H.Kemal İlter

"ImagiNation" is a unique project that aims to harness the power of artificial intelligence to create an engaging visual experience for children through AI-generated imagery and storytelling.The project utilizes large language models such as OpenAI to create captivating story texts for children, and generates images for these texts with artificial intelligence such as DALL-E. The goal is to stimulate children's imagination and creativity while making learning enjoyable.

A platform was created for this project. This platform can allow children aged 7-13 and parents to create stories with appropriate content and also provide eye-catching visuals for these stories. Children and parents can be able to create their own stories using this platform and interact with artificial intelligence technologies. This approach enables the project to reach a wider audience and provide a unique educational experience. Key benefits of "ImagiNation" include expanding children's imaginations, contributing to their learning processes through educational stories and images, and offering personalized learning experiences. However, potential drawbacks, such as excessive internet addiction, limited creativity, and content quality issues, must be considered and addressed by ensuring parental supervision and maintaining high content standards.

"ImagiNation" represents an innovative approach to children's education and entertainment, combining AI-driven narratives with images to create a new world of learning possibilities.

We would like to thank Professor Doctor H.Kemal İlter, who mentored and supported us in this project.

Keywords: *OpenAI, DALL-E, Ai-driven*

ÖZET

IMAGINATION AI

181005031 ALEYNA KURTKAYA

181005056 SEÇİL YILDIRIM

181005016 MEHMET KURTTEKİN

181005018 ENES ALÇİÇEK

Yönetim Bilişim Sistemleri Bölümü

Bitirme Projesi

Danışman: Prof. Dr. H.Kemal İlter

"ImagiNation", yapay zekanın gücünden yararlanarak yapay zeka tarafından oluşturulan görüntüler ve hikaye anlatımı yoluyla çocuklar için ilgi çekici bir görsel deneyim yaratmayı amaçlayan benzersiz bir projedir. Proje, çocuklar için büyüleyici hikaye metinleri oluşturmak için OpenAI gibi büyük dil modellerini kullanıyor ve bu metinler için DALL-E gibi yapay zeka ile görüntüler üretiyor. Amaç, öğrenmeyi eğlenceli hale getirirken çocukların hayal gücünü ve yaratıcılığını teşvik etmektir.

Bu proje için bir platform oluşturuldu. Bu platform, 7-13 yaş arası çocukların ve ebeveynlerin uygun içerikle hikayeler oluşturmalarına ve bu hikayeler için göz alıcı görseller sunmalarına olanak tanır. Çocuklar ve ebeveynler bu platformu kullanarak kendi hikayelerini oluşturabiliyor ve yapay zeka teknolojileri ile etkileşime geçebiliyor. Bu yaklaşım, projenin daha geniş bir kitleye ulaşmasını ve benzersiz bir eğitim deneyimi sunmasını sağlar. "ImagiNation"ın temel faydaları arasında çocukların hayal güçlerini genişletmek, eğitici hikayeler ve görsellerle öğrenme süreçlerine katkıda bulunmak ve kişiselleştirilmiş öğrenme deneyimleri sunmak yer alıyor. Bununla birlikte, aşırı internet bağımlılığı, sınırlı yaratıcılık ve içerik kalitesi sorunları gibi potansiyel dezavantajlar, ebeveyn gözetimi sağlanarak ve yüksek içerik standartları korunarak dikkate alınmalı ve ele alınmalıdır.

"ImagiNation", yeni bir öğrenme olanakları dünyası yaratmak için yapay zeka güdümlü anlatıları görüntülerle birleştirerek, çocukların eğitimine ve eğlencesine yenilikçi bir yaklaşımı temsil ediyor.

Bu projede bizlere mentorluk ve destek veren Prof. Dr. H.Kemal İlter'e teşekkür ederiz.

Anahtar Kelimeler: *OpenAI, DALL-E, Yapay Zeka*

In today's rapidly evolving world of technology, innovations in the field of artificial intelligence continue to progress without slowing down. Large language models like ChatGPT are just one example of artificial intelligence that can interact with humans and even understand their needs.

Inspired by these developments and the potential of language models such as ChatGPT, we decided to pursue a unique project. Our project aims to create a web platform to enhance children's imagination and reading habits. This platform will allow children aged 7-13 to create stories with appropriate content and will also provide attractive visuals for these stories.

Children and parents will be able to create their own stories using this platform and interact with artificial intelligence technologies. Our platform will offer customized suggestions for story creation, visual design, and reading experiences of children using powerful artificial intelligence tools such as Midjourney and Chatgpt.

This project will help children develop their imagination and increase their reading habits. Additionally, the use of our platform will contribute to the development of children's creativity, problem-solving skills, language, and vocabulary. Also, our web platform will provide creative and interactive reading experience combined with powerful artificial intelligence technologies, providing an excellent opportunity to expand children's imaginations. We hope that, through this project, we can continue to explore the potential of artificial intelligence and increase children's interest in storytelling.

We must undertake a meticulous research process for selecting story texts and artificial intelligence algorithms. It is crucial that the story texts are not only interesting and impactful but also span different categories and styles.

Throughout the development of our web platform, we will also focus on its design. Our platform will feature vibrant colors and appealing symbols. On this platform, childrens and parents will be able to create stories in various categories as they wish. In the later stages of our project, we plan to produce entertaining and educational animations. Furthermore, we aim to foster a wider community by encouraging children to contribute to our project.

Target Audience: Children aged 7-13 and parents with children aged 7-13

Project Purpose: It enables children and parents to create stories with appropriate content for children aged 7-13, and by creating visuals for these stories, it attracts attention and increases children's interaction with reading.

Process Steps:

• The user runs the system to create the content of his/her dream story by typing the words he/she wants.

• After the user enters the words, the text is generated and presented to the user.

• If there are inappropriate words for children, they are arranged by filtering on the system side.

• After the text is created, the images are created by artificial intelligence with the keywords entered for the text for the images to be added to the text and integrated into the text.

• The user can then read his/her story and create a new text by following the same steps.

• These created stories can be published publicly in the library section of our website.

• A comment and scoring system can be introduced for the stories in the library.

We plan to do these operations using the website's API. Content can be produced using tools such as chat gpt and Midjourney via the website, but if the APIs are closed, then we need to send and receive data from the website by writing simple programs.

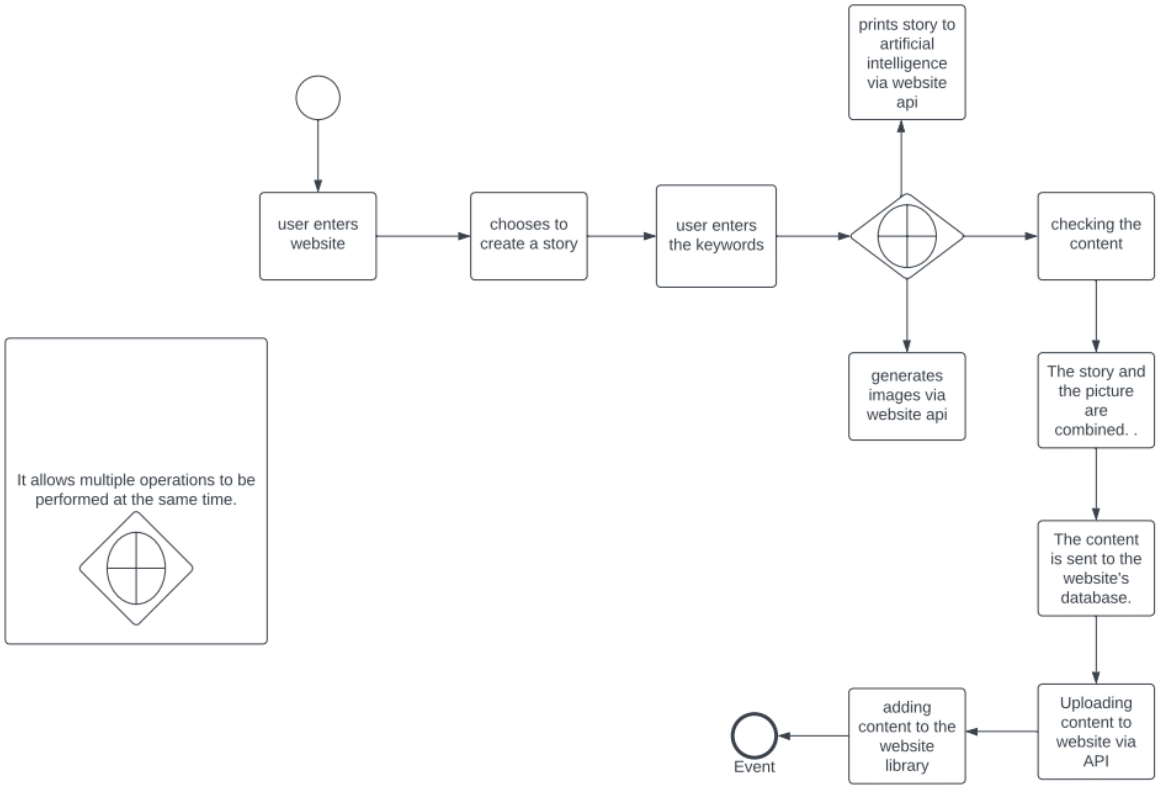


Figure 1: ImagiNation BPMN

We can summarize the potential benefits and drawbacks of our project as follows:

Benefits:

• It can expand children's imaginations and make learning enjoyable.

• Educational animations and games can contribute to children's learning processes.

• Artificial intelligence technology can provide personalized learning experiences for children.

• Through the web platform, equal educational opportunities can be provided by reaching children in different parts of the world.

Drawbacks:

• Unmonitored usage by parents could lead to excessive internet addiction in children.

• It could limit children's creativity or affect their ability to form original thoughts.

• Issues with the quality and accuracy of the content may arise.

• The use of artificial intelligence technology could contribute to a more machine-driven world rather than a human-driven one.

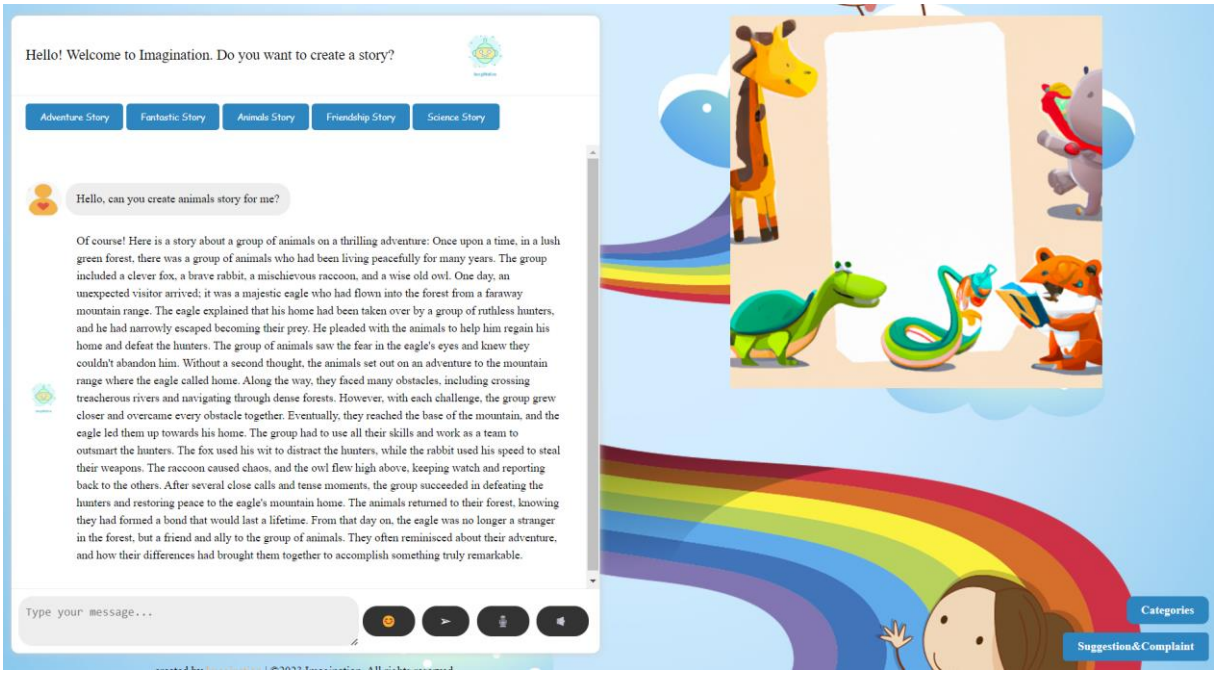
Therefore, it is important to consider the project's benefits and drawbacks while keeping it under the supervision of parents and educators. Additionally, care should be taken to ensure that the content is accurate and of high quality.

Three alternative titles for this project:

1. WonderTales: "Wonder" means "to be curious" and "Tales" means "stories." This name aims to encourage children to use their imagination and curiosity to create their own stories.
2. ImagiNation: "Imagi" means "to imagine" and "Nation" means "country." This name emphasizes the idea of children using their imagination to create their own world and stories.
3. Inkwell Adventures: "Inkwell" means "a container for ink" and "Adventures" means "exciting experiences." This name encourages children to use their pens and imagination to create their own adventures.

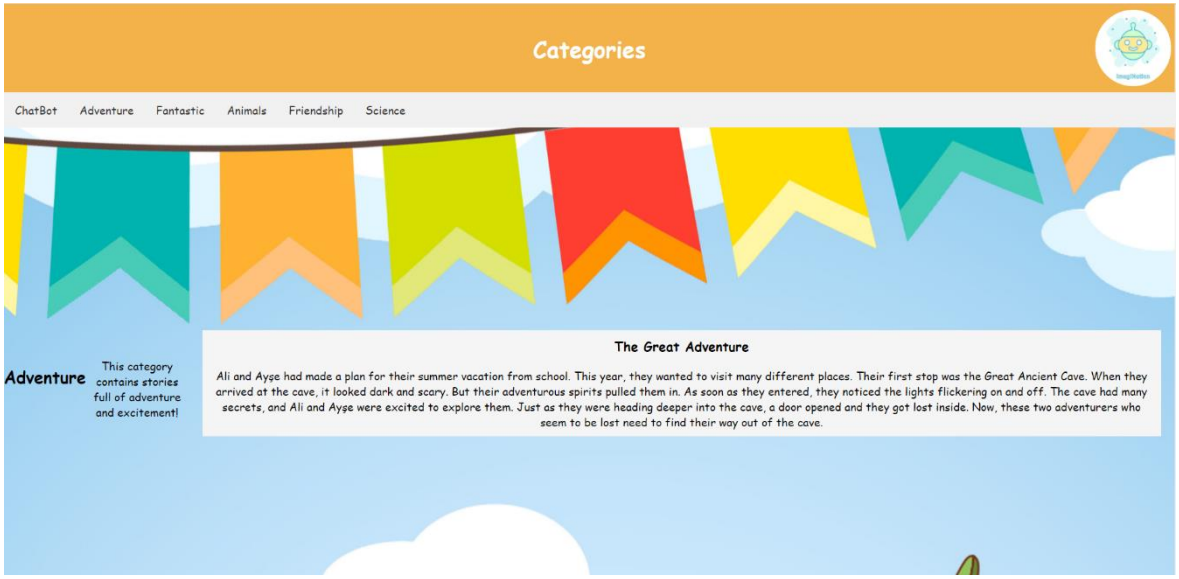
We started by brainstorming how our ideal design should look. We defined our logo and then proceeded to choose our user icon. We pondered over the design of the chatbot and we created Imagination chatbot with this concept.

Imagination Chatbot design in our website:



Our chatbot interface stands out with its vibrant colors and modern design, specially designed to provide the best story experience for children and parents. Moreover, Imagination chatbot works with artificial intelligence technology, allowing children and parents to create interactive stories. It determines the characters, plot, and development of the stories based on the input from users and offers inspiring suggestions. Imagination helps children develop their creative writing skills while also strengthening communication between families. These activities can also improve children's reading and writing abilities.

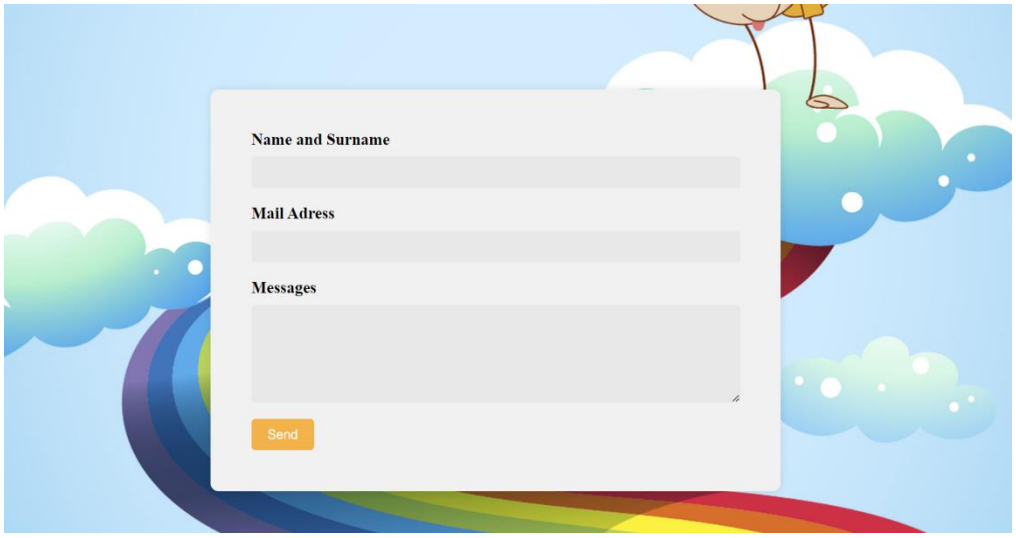
Categories page in our website:



After creating their stories, users can share them on a category page dedicated to showcasing the best stories. This page is a fantastic way to explore the various themes and styles of stories that the chatbot's users have crafted. From heartwarming tales of friendship to epic adventures through far-off lands, the category page is home to stories of all types and genres.

This feature allows children and parents to connect with each other and share their creations with the wider community. It promotes creativity and encourages children to write more, building their confidence and expanding their imaginations.

In conclusion, Imagination chatbot's category page is a fantastic way for users to showcase their stories and connect with other users.



We have created a suggestion and complaint form because the opinions of children and parents who use our platform are very important to us. Through this form, we aim to increase the satisfaction of parents and children by learning their opinions about our story creation chatbot.

Technical details:

Imagination chatbot's design incorporates HTML and CSS coding to create an engaging user interface. HTML (Hypertext Markup Language) provides the structure and layout of the chatbot, allowing for the placement of interactive elements and content. CSS (Cascading Style Sheets) is utilized to enhance the visual appeal of the interface, including the use of vibrant colors, modern design elements, and responsive layouts.

Additionally, the functionality of the chatbot is powered by JavaScript, a dynamic programming language that enables interactivity and real-time updates. JavaScript is responsible for handling user inputs, processing data, and triggering appropriate responses. It ensures smooth and seamless interactions between users and the chatbot.

To enable the chatbot's advanced capabilities and generate dynamic responses, it is connected to the OpenAI API using JavaScript. The Imagination integrates the powerful language model developed by OpenAI, allowing the chatbot to generate contextually relevant and creative story suggestions based on user inputs. The OpenAI API connection enables real-time communication with the language model, ensuring an interactive and engaging storytelling experience.

By leveraging these technologies, Imagination chatbot offers users a visually appealing and interactive platform for story creation. The combination of HTML and CSS creates an attractive and user-friendly interface, while JavaScript and the OpenAI API power the chatbot's intelligent and dynamic storytelling capabilities.

Here is the **index.html** code block (this is for the main page):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>IMAGINATION CHAT BOT</title>
  <link rel="stylesheet" href="./css/style.css">
  <link rel="shortcut icon" href="./images/img1.jpg" type="image/jpeg">
  <script src="./scripts/reader.js"></script>
</head>
<body>
  <header>
    <h1>IMAGINATION CHAT BOT</h1>
  </header>
  <div class="categories-btn">
    <a href="categories.html">Categories</a>
  </div>
  <div class="feedback-btn">
    <a href="form.html">Suggestion&Complaint</a>
  </div>
  <div class="container" id="container">
    <div class="chat-window">
      <div class="chat-header">
```

```
        <div class="renk">
          <h2>Hello! Welcome to Imagination. Do you want to create a
story?</h2>
        </div>
        <img class="profile-img" src="./images/img1.jpg" alt="Profile Image">
      </div>
      <div class="chat-body" id="chat-history">
        <div class="incoming-message">
          <div class="renk2">
            <p class="message"></p>
          </div>
          <script src="./scripts/generator.js"></script>
          <p id="output"></p>
        </div>
      </div>
      <div class="chat-footer">
        <textarea class="message-input" placeholder="Type your message..."
id="input"></textarea>
        <button class="send-btn" id="submit"
onclick="generateImage()">➢</button>
        <button for="Speech Recognition" id="speakButton"
onclick="record()"><img src="./images/mic.jpg" style="width:20px"></button>
        <button class="send-btn" onclick="openEmojiPicker()">😊</button>
        <button id="speak-button"><img src="./images/reader.jpg"
style="width:30px"></button>
      </div>
    </div>
    <p class="credit">
      created by <span>Imagination</span> |
      ©2023 Imagination. All rights reserved.
    </p>
  </div>

  <script src="./scripts/main.js"></script>
  <script>
    function openEmojiPicker() {
      window.open('https://www.emojicopy.com/', '_blank',
'width=600,height=600');
    }
  </script>
</body>
</html>
```

Here is part of the **style.css** code block (this is for the main page):

```
.categories-btn {
  position: fixed;
  bottom: 70px;
  right: 20px;
  padding: 10px 20px;
  border-radius: 10px;
  background-color: #2e86c1;
}

.categories-btn a {
  color: #fff;
  font-weight: bold;
  text-decoration: none;
}

.feedback-btn {
  position: fixed;
  bottom: 20px;
  right: 20px;
  background-color: #2e86c1;
  padding: 10px 20px;
  border-radius: 5px;
}

.feedback-btn a {
  color: #fff;
  font-weight: bold;
  text-decoration: none;
}

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  background-image: url("../images/background-image.jpg");
  background-repeat: no-repeat;
  background-position: center center;
  background-size: cover;
}

#img {
  padding-left: 65vw;
  padding-bottom: 0px;
  position: relative;
  top: -280px;
```

```
  right: 250px;
}

.container {
  float: left;
  padding: 10px;
  height: 300px;
}

.left {
  width: 25%;
}

.right {
  width: 75%;
}

.row:after {
  content: "";
  display: table;
  clear: both;
}

header {
  background-color: #f3b24a;
  color: #ffffff;
  padding: 20px;
}

.speakButton {
  width: 20px;
}

h1 {
  margin: 0;
  font-size: 28px;
  font-weight: 700;
  text-align: center;
  font-family: Comic Sans MS;
}

.container {
  max-width: 850px;
  margin: 0 auto;
  padding: 20px;
}
```

Imagination Chatbot designed for children require special attention to ensure a safe and appropriate user experience. Filtering mechanisms play a crucial role in maintaining the integrity and safety of the interactions. Some steps could be:

- Appropriate Age Group Filter

  Implementing an age group filter is essential to ensure that the chatbot caters to children of a specific age range. By setting an appropriate age filter, we can tailor the chatbot's content and responses to align with the cognitive and emotional capabilities of the target audience. This filter helps to provide age-appropriate information and avoid content that might not be suitable for younger or older children.

- Filtering Abusive and Inappropriate Words:

  One of the primary concerns when designing a chatbot for children is to prevent exposure to abusive or inappropriate language. Incorporating a word filtering mechanism allows the chatbot to automatically identify and filter out offensive or unsuitable words. This helps to maintain a respectful and safe environment for children during their interactions with the chatbot.

- Response Repetition Filter:

  To enhance the conversational experience, it is crucial to prevent the chatbot from repeatedly providing the same response. Implementing a repetition filter ensures that the chatbot generates diverse and contextually relevant replies, thereby maintaining user engagement. This filter helps to create more natural and dynamic conversations with children, avoiding monotony in the interaction.

- Content Filter:

  A content filter is necessary to assess the relevance and appropriateness of shared content within the chatbot's responses. By employing a content filtering mechanism, we can ensure that any information or media presented to children aligns with their educational, developmental, and ethical requirements. This filter prevents the dissemination of inaccurate, misleading, or harmful content.

- Topic Filter:

  In order to maintain a safe and child-friendly environment, it is essential to include a topic filter that identifies and blocks conversations about sensitive or inappropriate subjects. By implementing this filter, the chatbot can restrict discussions on topics such as violence, sexuality, or other potentially harmful content. This helps to create a secure and age-appropriate interaction for children.

Filtering mechanisms are integral to the design and development of our chatbot targeting children. By implementing appropriate filtering steps, we can ensure that the chatbot provides a safe, educational, and engaging experience. The age group filter, abusive word filter, response repetition filter, content filter, and topic filter collectively contribute to creating a secure environment and fostering positive interactions for children. It is of utmost importance to prioritize and implement these filtering mechanisms to safeguard children's well-being while using the chatbot.

Here's a basic implementation of the filtering steps using **JavaScript**:

1. **Appropriate age group filter:**

```
function ageFilter(age) {
  if (age < 7) {
    // Return appropriate message for children under 7
    return "Sorry, you must be at least 8 years old to use this chatbot.";
  }
  // Return message for older children
  return "Welcome to our chatbot!";
}
```

2. **Filtering abusive and inappropriate words:**

```
let abusiveWords = ["badword1", "badword2", "badword3"];
function filterAbusiveWords(message) {
  let words = message.split(" ");
  for (let i = 0; i < words.length; i++) {
    if (abusiveWords.includes(words[i].toLowerCase())) {
      // Replace abusive word with asterisks
      words[i] = "*".repeat(words[i].length);
    }
  }
  // Join the filtered words back into a string
  let filteredMessage = words.join(" ");
  return filteredMessage;
}
```

3. **Response repetition filter:**

```
let lastResponse = "";
function repeatFilter(message) {
  if (message === lastResponse) {
    // Return a different message instead of repeating the same reply
    return "I'm sorry, I've already answered that question. Can I help you with something
else?";
  }
  // Update the last response
  lastResponse = message;
  return message;
}
```

We wrote a Javascript code by paying attention to all of these. Here is the describing the technical details of our **main JavaScript** code:

1. **Variables:**

- **'API_KEY'**: The key used to authenticate with the OpenAI API.
- **'submitButton'**: Variable used to select the button element labeled as "submit" in the HTML.
- **'outputElement'**: Variable used to select the output element labeled as "output" in the HTML.
- **'inputElement'**: Variable used to select the input element labeled as "input" in the HTML.
- **'chatHistoryElement'**: Variable used to select the chat history element labeled as "chat-history" in the HTML.

2. **'changeInput' Function:**

- This function is used to assign a value to the **'inputElement'** input element.

3. **'msg' and 'speechSynthesis':**

- The **'msg'** variable creates a SpeechSynthesisUtterance object and assigns the text "Hello World" to it.
- The **'window.speechSynthesis.speak(msg)**' statement generates a spoken output of "Hello World" in the browser.

4. **Age Prompt and Filtering:**

- The **'userAge'** and **'agePrompted'** variables are used to track the user's age and whether the age has been prompted.
- The **'getMessage'** function performs age prompt if the user's age has not been prompted yet.

5. **API Requests and Responses:**

- The **'getMessage'** function sends the user's input to the API and processes the response.
- The **'options'** object contains the options used in the API request.
- The API request is sent using the **'fetch'** method and processed when the response is received.
- The content of the response is processed by adding user and assistant messages to the chat history.

6. **Repetition Filter:**

- The **'lastResponse'** variable stores the last response and is used to check for repetition.
- The **'repeatFilter'** function checks the message and returns a different message if it is a repeated reply.

7. **HTML Events:**

- A "click" event listener is added to the **‘submitButton’** button, calling the **‘getMessage’** function.
- The **‘clearInput’** function clears the content of the **‘inputElement’** input.

8. **‘filterInappropriateWords’ Function:**

- This function is used to filter inappropriate words.
- For example, it replaces the words in the **‘inappropriateWords’** array with asterisks.

Here is the our **main.js** code block:

```
const API_KEY = 'openai apikeyy';
const submitButton = document.querySelector("#submit");
const outputElement = document.querySelector("#output");
const inputElement = document.querySelector("#input");
const chatHistoryElement = document.querySelector("#chat-history");

function changeInput(value) {
  inputElement.value = value;
}

var msg = new SpeechSynthesisUtterance();
msg.text = "Hello World";
window.speechSynthesis.speak(msg);

let userAge;
let agePrompted = false;
let lastResponse = "";

async function getMessage() {
  console.log("clicked");

  if (!agePrompted) {
    // Check if user's age is 7 or above
    while (true) {
      userAge = parseInt(prompt("Please enter your age:"));

      if (isNaN(userAge)) {
        alert("Invalid age. Please enter a valid age.");
      } else if (userAge < 7) {
        alert("Sorry, this chatbot is only available for ages 7 and above.");
      } else {
        agePrompted = true;
        break;
      }
```

```
  }
 }

 // Inappropriate words filter
 const userInput = inputElement.value;
 const filteredInput = filterInappropriateWords(userInput);

 // Response repetition filter
 const filteredMessage = repeatFilter(filteredInput);

 const options = {
  method: "POST",
  headers: {
   Authorization: `Bearer ${API_KEY}`,
   "Content-Type": "application/json",
  },
  body: JSON.stringify({
   model: "gpt-3.5-turbo",
   messages: [
    { role: "system", content: "You are a helpful assistant." },
    { role: "user", content: filteredMessage } // Use the filtered message
   ],
   max_tokens: 100,
  }),
 };

 try {
  const response = await fetch(
   "https://api.openai.com/v1/chat/completions",
   options
  );
  const data = await response.json();
  console.log(data);

  if (data.choices[0].message.content && filteredMessage) {
   const userMessage = document.createElement("div");
   userMessage.className = "outgoing-message";
   userMessage.innerHTML = `
   <img class="profile-img" src="./images/img2.jpg" alt="Profile Image">
    <div class="message">${filteredMessage}</div>
   `;
   chatHistoryElement.appendChild(userMessage);

   const assistantMessage = document.createElement("div");
   assistantMessage.className = "incoming-message";
   assistantMessage.innerHTML = `
   <img class="profile-img" src="./images/img1.jpg" alt="Profile Image">
    <div class="message">${data.choices[0].message.content}</div>
   `;
```

```
      chatHistoryElement.appendChild(assistantMessage);

      // Update the last response
      lastResponse = filteredMessage;
    }

    inputElement.value = "";
  } catch (error) {
    console.error(error);
  }
}

submitButton.addEventListener("click", getMessage);

function clearInput() {
  inputElement.value = "";
}

function filterInappropriateWords(input) {
  // Modify this function to add your own logic for filtering inappropriate words
  // Here's an example implementation that replaces inappropriate words with asterisks
  const inappropriateWords = ["bad", "evil", "hate"];
  let filteredInput = input;

  for (let word of inappropriateWords) {
    const regex = new RegExp(word, "gi");
    filteredInput = filteredInput.replace(regex, "*".repeat(word.length));
  }

  return filteredInput;
}
function repeatFilter(message) {
  if (message === lastResponse) {
    // Return a different message instead of repeating the same reply
    return "I'm sorry, I've already answered that question. Can I help you with something
else?";
  }
  // Update the last response
  lastResponse = message;
  return message;
}
```

A story is created by connecting with Openai through the main.js codes. Afterwards, to generate a visual that corresponds to this story, we have written a JavaScript code under the title **Generator.js**. The technical details of this code are as follows:

1. Declares a constant variable **`url`** and assigns it the value `"https://api.openai.com/v1/images/generations"`. This URL represents the API endpoint for image generation.
2. Declares a constant variable **`apiKey`** and assigns it a specific API key. This key is used for authentication when making requests to the OpenAI API.
3. Retrieves the HTML element with the id "row" using **`getElementById()`** and assigns it to the variable **`row`**. This element will be used to manipulate the document's DOM.
4. Creates a new HTML container element using the **`createElement()`** method and assigns it to the variable **`cont`**. This container will be used to hold the generated image elements.
5. Defines a function named **`generateImage()`**, which is responsible for generating and displaying images based on user input.
6. Appends the **`cont`** container element to the document body using **`appendChild()`**. This ensures that the container is added to the webpage and can be seen by the user.
7. Creates an object named **`data`** that contains properties for generating the image. The **`prompt`** property retrieves the value from the HTML element with the id "input". The **`n`** property is set to 1, indicating that we want to generate a single image. The **`size`** property is set to **'512x512'**, specifying the desired size of the generated image.
8. Initiates a POST request to the specified **`url`** using the **`fetch()`** function. The request includes the necessary method, headers, and request body.
9. Sets the request method to "POST" and includes the headers required for the API request. The "Content-Type" header is set to "application/json" to indicate that the request body contains JSON data. The "Authorization" header includes the API key for authentication.
10. Converts the **`data`** object to a JSON string using **`JSON.stringify()`** and includes it in the request body. This JSON payload contains the prompt, number of images to generate, and image size.
11. Handles the response using promise chaining. The response is converted to JSON format using **`res.json()`**.
12. Logs the response data to the console for debugging purposes using **`console.log()`**.
13. Iterates through the **`data`** array received in the response and performs the following actions for each item:

- Creates a new **`img`** element using **`createElement("img")`**.
- Sets the **`src`** attribute of the **`img`** element to the URL of the generated image obtained from **`item.url`**.
- Sets the **`alt`** attribute of the **`img`** element to the string "image" to provide alternative text for accessibility.
- Sets the **`id`** attribute of the **`img`** element to "img" to uniquely identify the element.
- Appends the **`img`** element to the `cont` container element using **`appendChild()`**.

This code integrates with the OpenAI API to generate images based on user input and dynamically displays the generated images on the webpage.

Here is the **generator.js** code block:

```
const url = "https://api.openai.com/v1/images/generations";

const apiKey = "apikeyy ";

const row = document.getElementById("row");
const cont = document.createElement("container");

function generateImage() {
   document.body.appendChild(cont);

   const data = {
      prompt: document.getElementById("input").value,
      n: 1,
      size:'512x512'
   };

   fetch(url, {
      method: "POST",
      headers: {
         "Content-Type": "application/json",
         "Authorization": "Bearer " + apiKey,
      },
      body: JSON.stringify(data)
   })
      .then((res) => res.json())
      .then((data) => {
         console.log(data);
         data.data.forEach((item) => {
            const img = document.createElement("img");
            img.src = item.url;
            img.alt = "image";
            //img.classList.add("container");
            img.setAttribute("id", "img");
         cont.appendChild(img);
      })})
}
```

In the last step, we added a user's manual page to our web page. Thanks to this page, users can have information about the operation of Imagination Chatbot. This page contains the following information:

Imagination Chatbot User's Manual

1. Introduction of Main Features

This website has many features and offers you a variety of answers. Here are the main features:

Text Input: You can enter written text into Imagination Chatbot. In the text input section, you need to enter the following information about the story you want to create;

1. From which perspective do you want the story to be told (first person, third person, etc.)?

2. Details about the imaginary world in which the story takes place: geographical features, social structure, magic, etc.

3. Names of the main characters, their personality traits and their role in the story.

4. Your ideas about the challenges, obstacles and adventurous events the main characters will face.

5. The overall theme of the story.

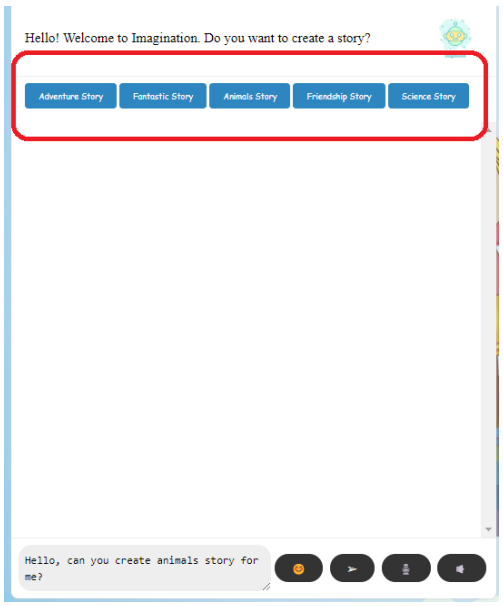
Voice Command: You can enter text to Imagination chatbot with voice commands. It recognizes the text you say using your microphone and what you say with a voice command is written in the text input section. Then, when the send button is pressed, the story and pictures will appear on the screen.

Image Output: Imagination can present the story you want as text and images. In this field, pictures related to your story will be presented.

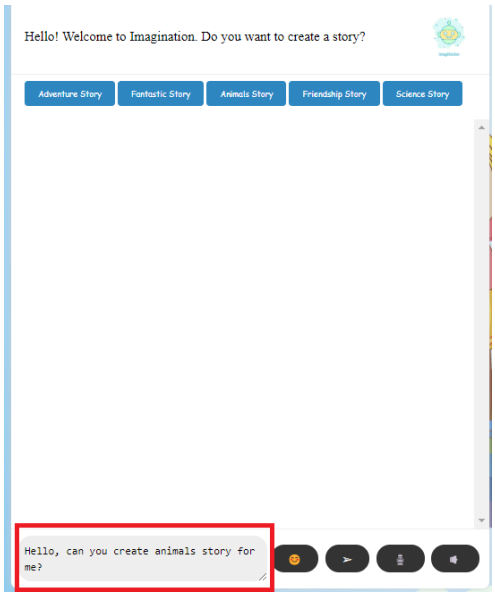
2. User Interface

The user interface determines how the website is used. Below we will introduce the components of the interface:

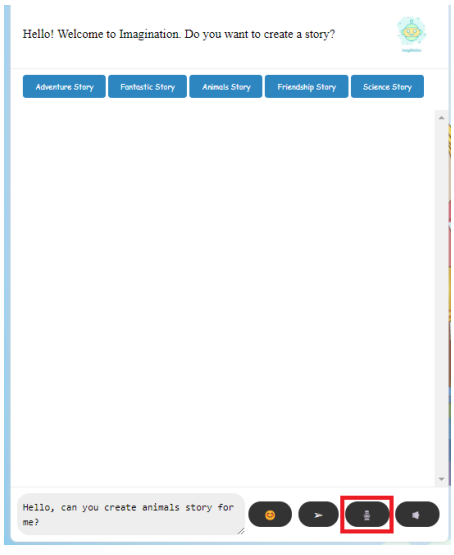
Random Story Generation Buttons: When these buttons are pressed, Imagination will generate random stories about categories and pictures about stories. (The buttons in this area do not require the submit button to be pressed.)



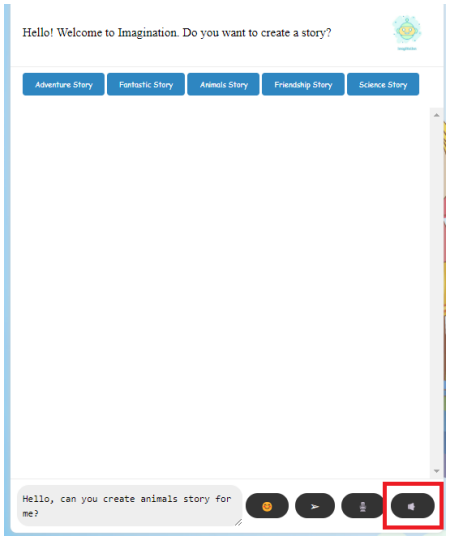
Text Input Area: This is the area where text input is done by voice command and written.



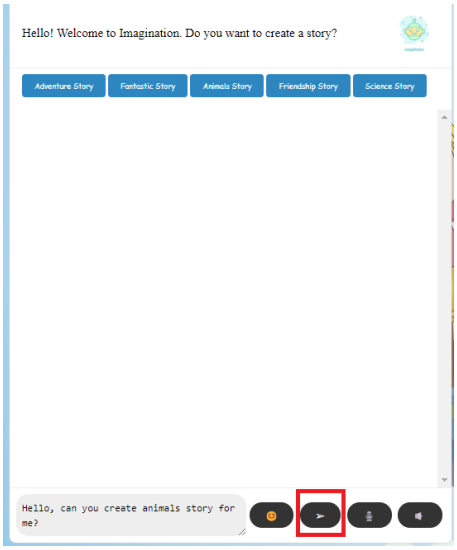
Voice Command Button: Press this button to create a story using a voice command for text input.



Speak Button: This button is used to listen to the story on the chatbot screen.



Send Button: This button is used to create written and illustrated stories from the texts you enter in the text input as text and voice commands.



3. Error Messages and Troubleshooting

Possible errors and troubleshooting steps you may encounter when using Imagination Chatbot:

1)Error: "Connection error: Check your internet connection."

Check your internet connection and try again.

If the problem persists, contact your internet provider.

2)**Error: "Failed to detect text, please try again."**

Make sure your microphone is working properly.

Try speaking more clearly and louder.

If your problems persist, you can use our contact form for support.

CONCLUSION

This report has been written to provide information about our Imagination Chatbot project that we have completed and to share the results we have achieved. In our project, we worked with the goal of creating a story using the OpenAI API and producing a suitable image for it. We used JavaScript, Html, Css and Php programming languages to realize this project.

Main.js codes were used to communicate with the OpenAI API and create text-based stories. Receiving user inputs, we sent requests to the API and obtained text-based stories thanks to OpenAI's advanced artificial intelligence models.

In Generator.js codes, we worked to produce visuals suitable for the stories we obtained. In this section, we enabled the production of visual objects using the OpenAI API. We obtained images by sending a request to the API and used these images to display on the web page.

In order to publish all these, we created a website with HTML and tried to create an eye-catching design with CSS.

This project is an important step that allows us to unleash our creativity using OpenAI's powerful artificial intelligence technologies. In addition, we aimed to provide an appropriate and safe experience for children, thanks to the filtering mechanisms used in the project.

As a result, our project was completed successfully and we achieved the results we aimed for. We created text-based stories using the OpenAI API and obtained images suitable for these stories. Users can now create their own stories using their own text input and experience them visually.

We are pleased with the success we have achieved as a result of the project. However, like any project, it can be developed further with improvements we can make in the future. We can make the user interface more user-friendly, expand the filtering options, and further enhance the user experience by adding features that users can have more control over.

REFERENCES:

1. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Agarwal, S. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165. https://arxiv.org/abs/2005.14165
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All you Need. arXiv preprint arXiv:1706.03762. https://arxiv.org/abs/1706.03762
3. https://platform.openai.com/docs/guides/completion
4. https://api.openai.com/v1/chat/completions
5. https://api.openai.com/v1/images/generations
6. https://www.d-id.com/