Webdev Finals

Balingit Luis Paulo D.

Dela Pena Dan Christian

Manarang Louie Angelo

Pineda Chester

BSCPE2-C

Main Obejctive

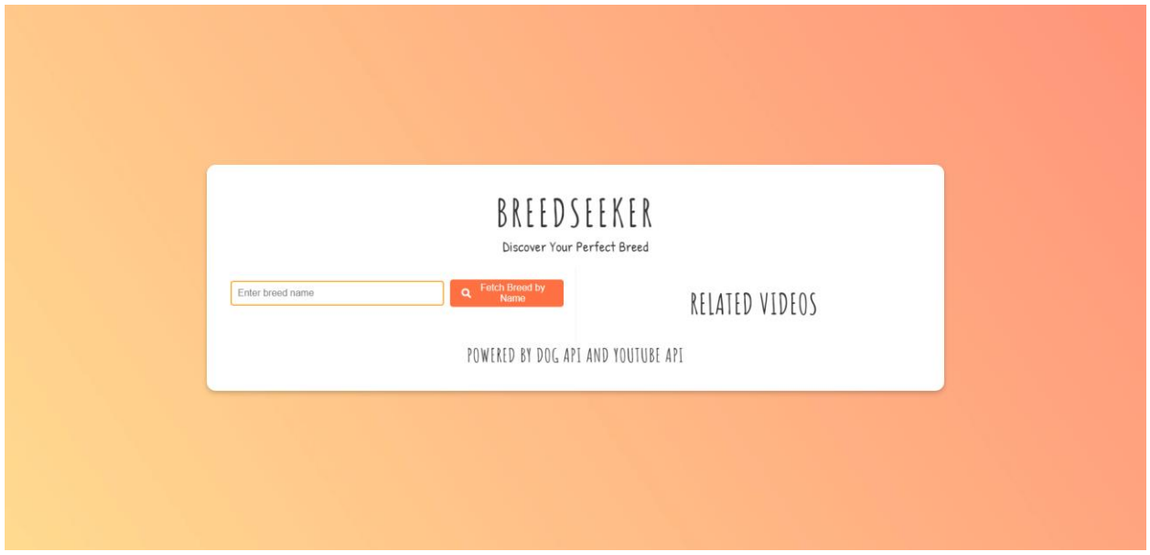
The main objective of the Breedseeker website is to create an interactive web application that allows users to search for the breed of dogs they want, retrieve detailed information about that dog, and provide real-life videos about that dog using the YouTube API.

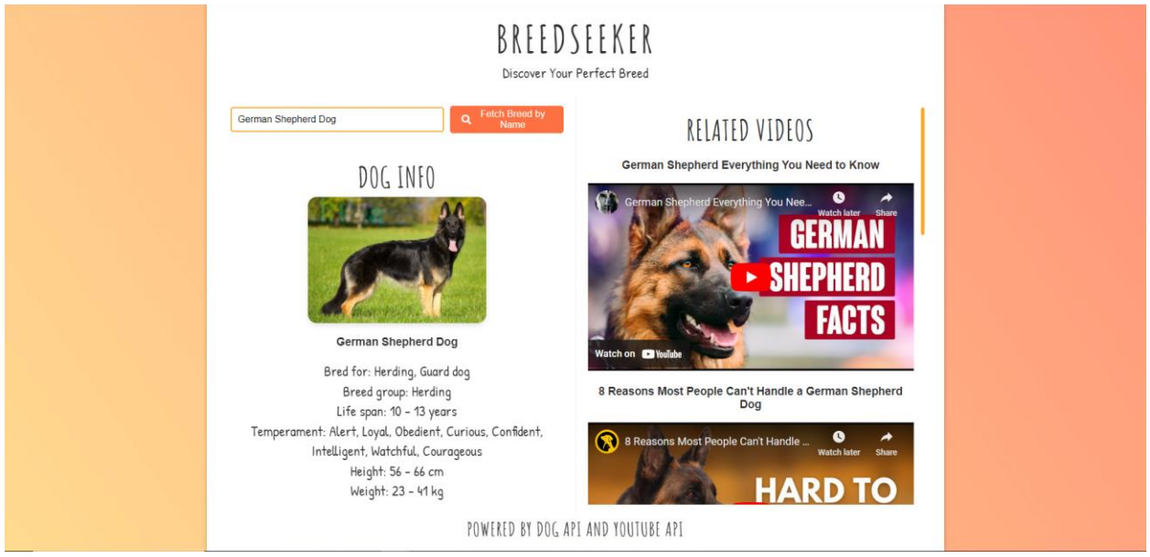
Features and Functionalities

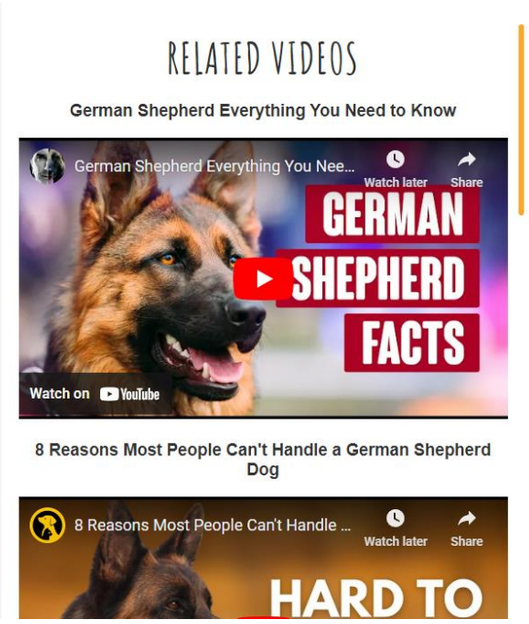
The application utilizes the Dog API to fetch information and images of various dog breeds, displaying this data dynamically on the webpage. The webpage also has proper handling to manage failed APIs. It displays an alert if the user searches for a breed of dog without entering any text in the field. It also features a responsive design, ensuring a great experience across different devices and screen sizes. We also implement a custom scrollbar to enhance the visual aesthetics of the application.

For real-time data, this application uses WebSocket's to enable real-time updates. It connects to a WebSocket server to fetch and display real-time data about dog breeds, and like the suggestion video on YouTube, it also has its own WebSocket server to handle real-time data communication, ensuring on-point and dynamic updates on the client side.

For the search and information display functionality, it allows the user to search for specific dog breeds using an input field. The search results are dynamically fetched and displayed from the Dog Api and YouTube Api. They also utilized a data list dropdown to provide an autocomplete feature, helping users quickly find and select dog breeds from the dropdown list for the dog information display shows detailed information about the searched dog breed, including an image, breed group, life span, temperament, height, and weight of the dog. Display also related Youtube videos about the searched dog breed, providing users with an additional multimedia source for learning about the breed.


BREEDSEEKER
Discover Your Perfect Breed
Enter breed name
Fetch Breed by Name
RELATED VIDEOS
POWERED BY DOG API AND YOUTUBE API


BREEDSEEKER
Discover Your Perfect Breed
German Shepherd Dog
Fetch Breed by Name
DOG INFO
German Shepherd Dog
Bred for: Herding, Guard dog
Breed group: Herding
Life span: 10 - 13 years
Temperament: Alert, Loyal, Obedient, Curious, Confident, Intelligent, Watchful, Courageous
Height: 56 - 66 cm
Weight: 23 - 41 kg
RELATED VIDEOS
German Shepherd Everything You Need to Know
German Shepherd Everything You Nee...
Watch later
Share
GERMAN
SHEPHERD
FACTS
Watch on YouTube
8 Reasons Most People Can't Handle a German Shepherd Dog
8 Reasons Most People Can't Handle ...
Watch later
Share
HARD TO
POWERED BY DOG API AND YOUTUBE API



Program Structure and Code Documentation

Program Structure

1. index.html

This file contains the HTML structure of the web application. It includes the main layout, header, search input, and sections for displaying dog information and related videos.

2. styles.css

This file contains the CSS styles for the web application. It styles the layout, fonts, buttons, and other elements to ensure a cohesive and visually appealing design. It also includes custom scrollbar styles and media queries for mobile responsiveness.

3. app.js

This file contains the client-side JavaScript code. It sets up the WebSocket connection, handles user interactions (such as fetching breed data), and dynamically updates the DOM with dog information and related videos. It also includes error handling for failed API requests.

4. server.js

This file contains the server-side code using Express.js and WebSocket. It sets up routes to fetch data from The Dog API and the YouTube Data API, handles WebSocket connections, and processes messages from the client to fetch and return breed data and related videos.

Code Documentation

Server.js

```
const express = require('express');
const axios = require('axios');
const http = require('http');
const WebSocket = require('ws');

const app = express();
const PORT = process.env.PORT || 3000;

// API keys for accessing external services
const DOG_API_KEY = 'live_Ef2OFXguvOguj9Iy1doHcSzubUBKOh0xCnsoUmwRSKirQGdFLmzRlsCgiR1Xt1Pi';
const YOUTUBE_API_KEY = 'AIzaSyAoR-pzGBTP8IkQiuLY4cjZY_KIoGp4Rso';

// Serve static files from the 'public' directory
app.use(express.static('public'));

// Route to fetch a random dog image
app.get('/dog', async (req, res) => {
  try {
     const response = await axios.get('https://api.thedogapi.com/v1/images/search', {
      headers: { 'x-api-key': DOG_API_KEY }
    });
    res.json(response.data[0]); // Send the first dog image from the response
  } catch (error) {
    console.error('Error fetching dog data:', error.response ? error.response.data : error.message);
    res.status(500).json({ error: 'Internal server error' }); // Send error response if fetching fails
  }
});

// Route to fetch YouTube videos related to dogs
app.get('/videos', async (req, res) => {
  try {
    const query = req.query.q || 'dogs'; // Default query is 'dogs' if no query is provided
    const response = await axios.get('https://www.googleapis.com/youtube/v3/search', {
      params: {
        part: 'snippet',
        q: query,
        key: YOUTUBE_API_KEY,
        type: 'video',
        maxResults: 5 // Limit the number of videos to 5
      }
    });
    res.json(response.data.items); // Send the fetched videos as response
  } catch (error) {
    console.error('Error fetching YouTube videos:', error.response ? error.response.data : error.message);
    res.status(500).json({ error: 'Internal server error' }); // Send error response if fetching fails
  }
});

// Create an HTTP server
const server = http.createServer(app);

// Set up WebSocket server on the same port as the HTTP server
```

```
const wss = new WebSocket.Server({ server });

wss.on('connection', ws => {
  // Handle incoming messages from the client
  ws.on('message', async message => {
    console.log(`Received message => ${message}`);
    let parsedMessage;
    try {
      parsedMessage = JSON.parse(message); // Try to parse the message as JSON
    } catch (e) {
      parsedMessage = message; // If parsing fails, use the message as is
    }

    // Fetch dog breed details by name
    if (parsedMessage.action === 'fetchBreedByName') {
      try {
        const breedsResponse = await axios.get('https://api.thedogapi.com/v1/breeds', {
          headers: { 'x-api-key': DOG_API_KEY }
        });
        const breed = breedsResponse.data.find(b =>
b.name.toLowerCase().includes(parsedMessage.breedName));
        if (breed) {
          const breedResponse = await
axios.get(`https://api.thedogapi.com/v1/images/search?breed_ids=${breed.id}`, {
            headers: { 'x-api-key': DOG_API_KEY }
          });
          ws.send(JSON.stringify({ breedDog: breedResponse.data[0], breedDetails: breed })); // Send breed
details and image

          // Fetch YouTube videos related to the breed
          const videoResponse = await axios.get('https://www.googleapis.com/youtube/v3/search', {
            params: {
              part: 'snippet',
              q: parsedMessage.breedName,
              key: YOUTUBE_API_KEY,
              type: 'video',
              maxResults: 5
            }
          });
          ws.send(JSON.stringify({ videos: videoResponse.data.items })); // Send the fetched videos
        } else {
          ws.send(JSON.stringify({ error: 'Breed not found' })); // Send error if breed is not found
        }
      } catch (error) {
        console.error('Error fetching breed by name:', error.response ? error.response.data :
error.message);
        ws.send(JSON.stringify({ error: 'Error fetching breed by name' })); // Send error response if
fetching fails
      }
    }
    // Fetch all dog breeds
    else if (parsedMessage.action === 'fetchBreeds') {
      try {
        const breedsResponse = await axios.get('https://api.thedogapi.com/v1/breeds', {
          headers: { 'x-api-key': DOG_API_KEY }
        });
        ws.send(JSON.stringify({ breeds: breedsResponse.data })); // Send all breeds as response
      } catch (error) {
        console.error('Error fetching breeds:', error.response ? error.response.data : error.message);
        ws.send(JSON.stringify({ error: 'Error fetching breeds' })); // Send error response if fetching
fails
      }
    }
  });

  // Send a welcome message to the client upon connection
  ws.send(JSON.stringify({ message: 'Welcome! Send "fetchData" to get dog data.' }));
});

// Start the server on the specified port
server.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

App.js

```
document.addEventListener('DOMContentLoaded', () => {
    // Get references to the DOM elements
    const dogInfo = document.getElementById('dog-info');
    const fetchBreedByNameButton = document.getElementById('fetch-breed-by-name');
    const breedNameInput = document.getElementById('breed-name');
    const breedsList = document.getElementById('breeds');
    const videoContainer = document.getElementById('video-container');

    // Set up WebSocket connection to the server
    const ws = new WebSocket('ws://localhost:3000');

     // Handle messages received from the WebSocket server
    ws.onmessage = (event) => {
        try {
            const data = JSON.parse(event.data);
            if (data.message) {
                console.log(data.message);
            } else if (data.breedDog) {
                displayDog(data.breedDog, data.breedDetails); // Display dog information
            } else if (data.breeds) {
                populateBreedsList(data.breeds); // Populate the list of breeds
            } else if (data.videos) {
                displayVideos(data.videos); // Display related videos
            } else if (data.error) {
                displayError(data.error); // Display any errors
            }
        } catch (e) {
            console.error('Error parsing message:', e); // Handle JSON parsing errors
        }
    };

     // Handle WebSocket connection open event
    ws.onopen = () => {
        console.log('WebSocket connection opened');
        ws.send(JSON.stringify({ action: 'fetchBreeds' })); // Request the list of breeds upon connection
    };

    // Handle WebSocket connection close event
    ws.onclose = () => {
        console.log('WebSocket connection closed');
    };

    // Handle the click event for fetching breed information
    fetchBreedByNameButton.addEventListener('click', () => {
        const breedName = breedNameInput.value.trim().toLowerCase();
        if (!breedName) {
            alert('Please type a dog breed or select from the list.');
            return;
        }
        ws.send(JSON.stringify({ action: 'fetchBreedByName', breedName })); // Send breed name to the
server
    });

    // Populate the datalist of breeds
    function populateBreedsList(breeds) {
        breedsList.innerHTML = '';
        breeds.forEach(breed => {
            const option = document.createElement('option');
            option.value = breed.name;
            breedsList.appendChild(option);
        });
    }

    // Display dog information including breed details if available
    function displayDog(dog, breedDetails = null) {
        let breedInfo = '';
        if (breedDetails) {
            breedInfo = `
                <h3>${breedDetails.name}</h3>
                <p>Bred for: ${breedDetails.bred_for || 'Unknown'}</p>
                <p>Breed group: ${breedDetails.breed_group || 'Unknown'}</p>
                <p>Life span: ${breedDetails.life_span || 'Unknown'}</p>
                <p>Temperament: ${breedDetails.temperament || 'Unknown'}</p>
                <p>Height: ${breedDetails.height.metric || 'Unknown'} cm</p>
                <p>Weight: ${breedDetails.weight.metric || 'Unknown'} kg</p>
```

```
            `;
        }
        dogInfo.innerHTML = `
            <h2>Dog Info</h2>
            <img src="${dog.url}" alt="Dog" width="300">
            ${breedInfo}
        `; // Update dog info in the DOM
    }

    // Display a list of related videos
    function displayVideos(videos) {
        videoContainer.innerHTML = '<h2>Related Videos</h2>'; // Clear previous videos
        videos.forEach(video => {
            const videoElement = document.createElement('div');
            videoElement.classList.add('video');
            videoElement.innerHTML = `
                <h3>${video.snippet.title}</h3>
                <iframe width="100%" height="315" src="https://www.youtube.com/embed/${video.id.videoId}"
frameborder="0" allowfullscreen></iframe>
            `;
            videoContainer.appendChild(videoElement); // Add video to the container
        });
    }

    // Display an error message
    function displayError(error) {
        console.error('Error:', error);
        alert(error);
    }
});
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BreedSeeker: Discover Your Perfect Breed</title>
    <link rel="stylesheet" href="styles.css">
    <link
href="https://fonts.googleapis.com/css2?family=Amatic+SC:wght@700&family=Patrick+Hand&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
</head>
<body>
    <video id="bgVideo">
        <source src="loop.mp4" type="video/mp4">
    </video>
    <div class="container">
        <header>
            <h1>BreedSeeker</h1>
            <p>Discover Your Perfect Breed</p>
        </header>
        <div class="content">
            <div class="left-panel">
                <div class="search-section">
                    <input type="text" id="breed-name" placeholder="Enter breed name" list="breeds">
                    <datalist id="breeds"></datalist>
                    <button id="fetch-breed-by-name"><i class="fas fa-search"></i> Fetch Breed by
Name</button>
                </div>
                <div id="dog-info"></div>
```

```
            </div>
            <div class="right-panel">
                <div id="video-container">
                    <h2>Related Videos</h2>
                </div>
            </div>
        </div>
        <footer>
            <p class="footer-text">Powered by Dog API and YouTube API</p>
        </footer>
    </div>
    <script src="js/app.js"></script>
</body>
</html>
```

Styles.css

```
body {
    font-family: 'Open Sans', Arial, sans-serif;
    color: #333;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    overflow: hidden;
    position: relative;
}

#bgVideo {
    position: absolute;
    top: 50%;
    left: 50%;
    min-width: 100%;
     min-height: 100%;
    width: auto;
    height: auto;
    z-index: -1;
    transform: translate(-50%, -50%);
}

.container {
    background-color: white;
    border-radius: 15px;
    padding: 40px 20px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.2);
    text-align: center;
    max-width: 1200px;
    width: 100%;
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 20px;
    margin-bottom: 20px;
    overflow-y: auto;
    z-index: 1;
}

h2 {
    font-family: 'Amatic SC', cursive;
    font-size: 3em;
    margin: 0;
    font-weight: 700;
    text-transform: uppercase;
    letter-spacing: 1px;
}

header h1 {
    font-family: 'Amatic SC', cursive;
    font-size: 4em;
    margin: 0;
    font-weight: 700;
    text-transform: uppercase;
    letter-spacing: 5px;
```

```css
}

header p {
    font-family: 'Patrick Hand', cursive;
    font-size: 1.5em;
    margin: 0;
    font-weight: 400;
}

.content {
    display: flex;
    width: 100%;
    margin-top: 20px;
}

.left-panel, .right-panel {
    flex: 1;
    padding: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    border-right: 1px solid #eee;
}

.right-panel {
    border-right: none;
}

.search-section {
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 10px;
    margin-bottom: 20px;
    width: 100%;
    position: relative;
}

input[type="text"] {
    padding: 10px;
    border: 2px solid #ffa726;
    border-radius: 5px;
    width: 70%;
    font-size: 1em;
    outline: none;
    transition: border-color 0.3s;
    z-index: 1;
}

input[type="text"]:focus {
    border-color: #ff7043;
}

button {
    background-color: #ff7043;
    color: white;
    border: none;
    border-radius: 5px;
    padding: 5px 20px;
    cursor: pointer;
    font-size: 1em;
    transition: background-color 0.3s;
    display: flex;
    align-items: center;
}

button i {
    margin-right: 5px;
}

button:hover {
    background-color: #ff5722;
}

#dog-info {
    margin-top: 20px;
    text-align: center;
```

```
    width: 100%;
    display: flex;
    flex-direction: column;
    align-items: center;
    line-height: 1.4;
}

#dog-info p {
    margin: 5px 0;
    font-family: 'Patrick Hand', cursive;
    font-size: 1.5em;
    margin: 0;
    font-weight: 400;
}

#dog-info img {
    border-radius: 15px;
    max-width: 100%;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
    transition: transform 0.3s;
}

#dog-info img:hover {
    transform: scale(1.05);
}

#video-container {
    max-height: 650px;
    overflow-y: auto;
    text-align: center;
    width: 100%;
    border-radius: 5px;
    padding: 10px;
    background-color: #fff;
}

.video {
    margin-bottom: 20px;
}

.footer-text {
    font-family: 'Amatic SC', cursive;
    font-size: 2em;
    margin: 0;
    font-weight: 700;
    text-transform: uppercase;
    letter-spacing: 1px;
}

::-webkit-scrollbar {
    width: 12px;
}

::-webkit-scrollbar-track {
    background: #fff;
    border-radius: 10px;
}

::-webkit-scrollbar-thumb {
    background: #ffa726;
    border-radius: 10px;
    border: 3px solid #fff;
}

::-webkit-scrollbar-thumb:hover {
    background: #ff7043;
}


@media (max-width: 768px) {
    .container {
        padding: 20px 50px;
    }

    header h1 {
        font-size: 3em;
        letter-spacing: 2px;
```

```
    }

    h2 {
        font-size: 2em;
    }

    .content {
        flex-direction: column;
    }

    .left-panel, .right-panel {
        width: 100%;
        border-right: none;
        padding: 10px;
    }

    input[type="text"] {
        width: 100%;
    }

    button {
        width: 100%;
        padding: 10px;
    }

    .search-section {
        flex-direction: column;
    }
}
```