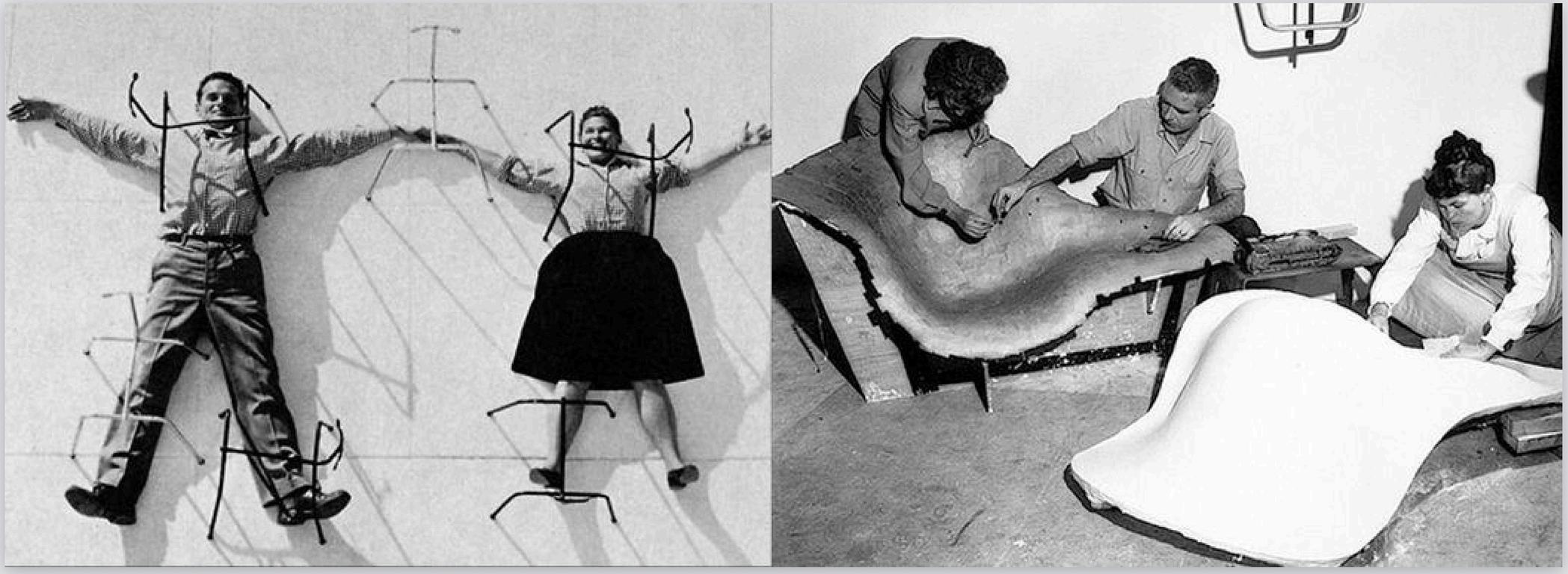
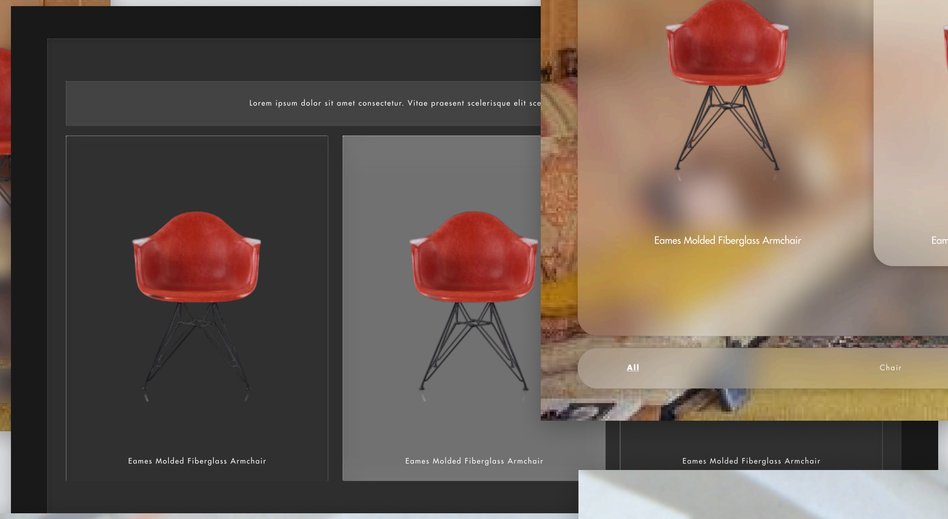
Eames Chair Card Website

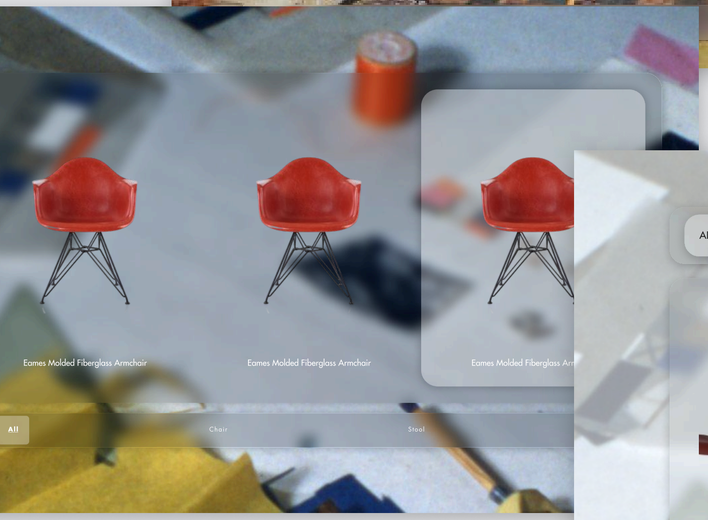
Qiyang Peng, 10/21/2025

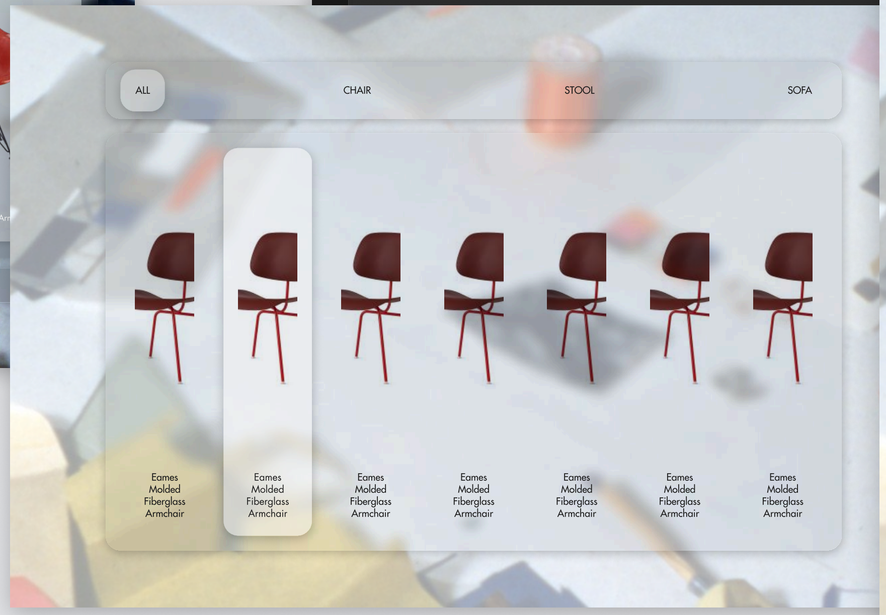


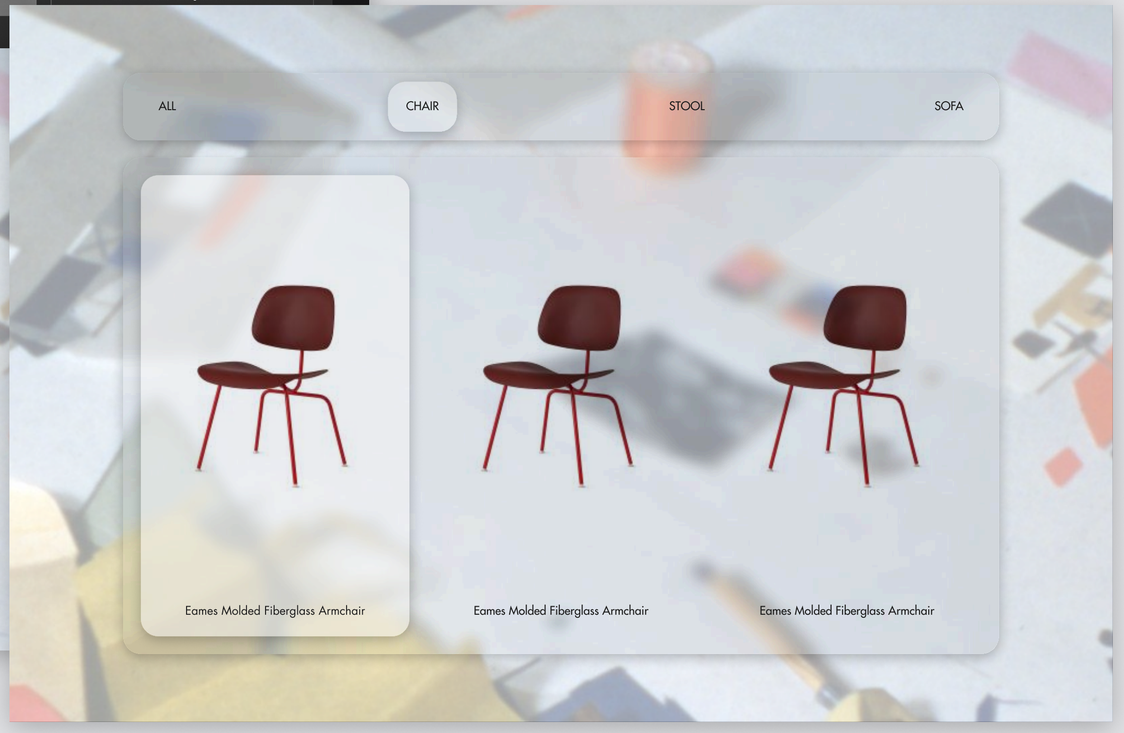

Sofa


Lorem ipsum dolor sit amet consectetur. Vitae praesent scelerisque elit sce
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair


Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
All
Chair
Stool
Sofa


Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
All
Chair
Stool


ALL
CHAIR
STOOL
SOFA
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair


ALL
CHAIR
STOOL
SOFA
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair
Eames Molded Fiberglass Armchair

ALL CHAIR STOOL SOFA

Eames Molded Fiberglass Armchair

Eames Molded Fiberglass Armchair

Eames Molded Fiberglass Armchair

Ideas & Main Features

Navigation Bar

Navigation bar enable user to navigate among different genres of the Eames chair.

Card Selection

All the product card can be selected, hovered, and de-selected.

Speaking Sound

When user pressed a card, openAI api will be called to speak about the history and the design of that product.

Code Showcasing

Play Audio

create Blob URL →

textToSpeech →

GenerateDescription →

useLLMAudio Hook →

User Click Product →

Some of My Code

generateDescription()

- **name:** Product name
- **category:** Product category
- **signal:** AbortController signal

```
const OPENAI_API_KEY = process.env.REACT_APP_OPENAI_API_KEY;

//generate product description through api call
export const generateDescription = async ({ name, category, signal }) => {
    const res = await axios.post(
        'https://api.openai.com/v1/chat/completions',
        {
            model: 'gpt-4o-mini',
            temperature: 0.7,
            messages: [
                {
                    role: 'system',
                    content:
                        'You are a concise guide who know MillerKnoll well. Write a bri
                        words) about the design history of the item. Be direct and enga
                        credible source.',
                },
                { role: 'user', content: `Item: ${name}. Category: ${category}.` },
            ],
        },
        {
            headers: {
                Authorization: `Bearer ${OPENAI_API_KEY}`,
                'Content-Type': 'application/json',
            },
            signal,
        }
    );
    return res.data.choices[0].message.content.trim();
};
```

Some of My Code

textToSpeech()

- **text:** string to convert to speech
- **signal:** AbortController signal

```
export const textToSpeech = async (text, signal) => {
    const res = await axios.post(
        'https://api.openai.com/v1/audio/speech',
        {
            model: 'tts-1',
            voice: 'alloy',
            speed: 1.1,
            input: text,
        },
        {
            headers: {
                Authorization: `Bearer ${OPENAI_API_KEY}`,
                'Content-Type': 'application/json',
            },
            responseType: 'arraybuffer',
            signal,
        }
    );
    return new Blob([res.data], { type: 'audio/mpeg' });
};
```

Some of My Code

useLLMAudio Hook

User Click Product →

// Clean previous
audio cleanup()

// Track request
reqIdRef.current += 1

// Store controller
controllerRef.current = ...

```
//useRef: store mutable references; useCallback: memorize function; useEffect: used to clean when component unmount
import { useCallback, useEffect, useRef } from 'react';
//generateDescription: call OpenAI GPT to generate text; textToSpeech: Call OpenAI TTS to convert text to audio
import { generateDescription, textToSpeech } from '../api';

export default function useLLMAudio() {
    //store audio element
    const audioRef = useRef(null);
    //store the Blob URL coming from API
    const urlRef = useRef(null);
    //store AbortController
    const controllerRef = useRef(null);
    //request ID counter for race condition prevention
    const reqIdRef = useRef(0);

    //abort the ongoing api request
    const abort = () => {
        controllerRef.current?.abort();
        controllerRef.current = null;
    };

    //clean up all the resources
    const cleanup = useCallback(() => {
        abort();
        if (audioRef.current) {
            audioRef.current.pause();
            audioRef.current.src = '';
            audioRef.current = null;
        }
        if (urlRef.current) {
            URL.revokeObjectURL(urlRef.current);
            urlRef.current = null;
        }
    }, []);

    //Stop the audio
    const stop = useCallback(() => {
        //Invalidate any on-going tasks and stop audio
        reqIdRef.current += 1;
        cleanup();
    }, [cleanup]);

    //Audio speak
    const speak = useCallback(async ({ name, category }) => {
        const myId = ++reqIdRef.current;
        //Clear any currently playing audio
        cleanup();

        const controller = new AbortController();
        controllerRef.current = controller;
```

Some of My Code

useLLMAudio Hook

Audio Ready →

// Store URL
urlRef.current = "blob:..."

// Store element
audioRef.current = new Audio()

// Play audio
audio.play()

```
//useRef: store mutable references; useCallback: memorize function; useEffect: used to clean when component unmount
import { useCallback, useEffect, useRef } from 'react';
//generateDescription: call OpenAI GPT to generate text; textToSpeech: Call OpenAI TTS to convert text to audio
import { generateDescription, textToSpeech } from '../api';

export default function useLLMAudio() {
    //store audio element
    const audioRef = useRef(null);
    //store the Blob URL coming from API
    const urlRef = useRef(null);
    //store AbortController
    const controllerRef = useRef(null);
    //request ID counter for race condition prevention
    const reqIdRef = useRef(0);

    //abort the ongoing api request
    const abort = () => {
        controllerRef.current?.abort();
        controllerRef.current = null;
    };

    //clean up all the resources
    const cleanup = useCallback(() => {
        abort();
        if (audioRef.current) {
            audioRef.current.pause();
            audioRef.current.src = '';
            audioRef.current = null;
        }
        if (urlRef.current) {
            URL.revokeObjectURL(urlRef.current);
            urlRef.current = null;
        }
    }, []);

    //Stop the audio
    const stop = useCallback(() => {
        //Invalidate any on-going tasks and stop audio
        reqIdRef.current += 1;
        cleanup();
    }, [cleanup]);

    //Audio speak
    const speak = useCallback(async ({ name, category }) => {
        const myId = ++reqIdRef.current;
        //Clear any currently playing audio
        cleanup();

        const controller = new AbortController();
        controllerRef.current = controller;
```

Some of My Code

useLLMAudio Hook

Cleanup →

// Stop audio
audioRef.current.pause()

// Free memory
URL.revokeObjectURL(urlRef.current)

// Cancel requests
controllerRef.current.abort()

```
//useRef: store mutable references; useCallback: memorize function; useEffect: used to clean when component unmount
import { useCallback, useEffect, useRef } from 'react';
//generateDescription: call OpenAI GPT to generate text; textToSpeech: Call OpenAI TTS to convert text to audio
import { generateDescription, textToSpeech } from '../api';

export default function useLLMAudio() {
    //store audio element
    const audioRef = useRef(null);
    //store the Blob URL coming from API
    const urlRef = useRef(null);
    //store AbortController
    const controllerRef = useRef(null);
    //request ID counter for race condition prevention
    const reqIdRef = useRef(0);

    //abort the ongoing api request
    const abort = () => {
        controllerRef.current?.abort();
        controllerRef.current = null;
    };

    //clean up all the resources
    const cleanup = useCallback(() => {
        abort();
        if (audioRef.current) {
            audioRef.current.pause();
            audioRef.current.src = '';
            audioRef.current = null;
        }
        if (urlRef.current) {
            URL.revokeObjectURL(urlRef.current);
            urlRef.current = null;
        }
    }, []);

    //Stop the audio
    const stop = useCallback(() => {
        //Invalidate any on-going tasks and stop audio
        reqIdRef.current += 1;
        cleanup();
    }, [cleanup]);

    //Audio speak
    const speak = useCallback(async ({ name, category }) => {
        const myId = ++reqIdRef.current;
        //Clear any currently playing audio
        cleanup();

        const controller = new AbortController();
        controllerRef.current = controller;
```

Next Steps

Reusable database

Cleaner app.js with more scalable design

More customized text

The product can contain more data in the database, so the prompt can be more accurate and customized

Quicker API Call

It takes about 3 - 5s for the user to get the audio. For example, by using shorter prompts and pre-fetch on hover/ idle.

Thank You