Experiment No: 1

Aim : Building Responsive and Interactive UIs using Tailwind CSS.

Theory:

1. Tailwind CSS – Utility-First Framework:

Tailwind CSS is a **utility-first CSS framework** that provides low-level utility classes like p-4, bg-blue-500, or text-center to style elements directly in the markup. Unlike traditional frameworks such as Bootstrap, Tailwind does not provide pre-styled components but instead enables developers to **compose custom UIs rapidly** by combining utilities.

Benefits:

- Increased development speed (no need to write custom CSS from scratch).
- Consistent styling across components.
- Full customization without overriding pre-built styles.

2. Responsive Design with Tailwind:

Responsive design ensures that the UI **adapts seamlessly to multiple devices** such as mobile, tablet, and desktop. Tailwind uses **mobile-first breakpoints**:

- sm: ≥ 640px
- md: ≥ 768px
- lg: ≥ 1024px
- xl: ≥ 1280px
- 2xl: ≥ 1536px

Example:
md:text-lg → Applies text-lg **only** on medium screens and larger.
This allows developers to create a fluid design without writing separate CSS media queries.

3. Interactive UI States:

Interactivity is achieved using **pseudo-class variants** in Tailwind:

- hover: → Styling on hover.

- focus: → Styling when an element is focused (e.g., input fields).
- active: → Styling when an element is clicked.
- disabled: → Styling for disabled elements.

These states enhance user experience without custom CSS, making buttons, inputs, and links visually responsive.

4. Component-Based Architecture:

The project uses **React (TypeScript)** to structure the application into reusable components such as:

- **Login.tsx** → Handles user authentication.
- **EditorPanel.tsx** → Provides the main coding editor.
- **Language.tsx** → Dropdown for language selection.
- **ThemeSelector.tsx** → Enables theme switching.
- **Snippet.tsx** → Saves and shares code snippets.
- **Output.tsx** → Displays compiled code output.

This modular approach ensures **maintainability and scalability**.

5. Payment Integration using LemonSqueezy (30% Extra):

For monetization, the concept of **payment gateway integration** using **LemonSqueezy** can be added. LemonSqueezy is a **SaaS payment platform** that provides APIs for:

- **Checkout pages** to accept payments securely.
- **Subscriptions & Licensing** for premium features (e.g., advanced code execution, unlimited snippets).
- **Webhooks** to handle events like successful payments or cancellations.

Why LemonSqueezy?

- Easy integration with React or Next.js.
- Supports digital product sales, recurring billing, and VAT compliance.
- Provides pre-built UI components for checkout.

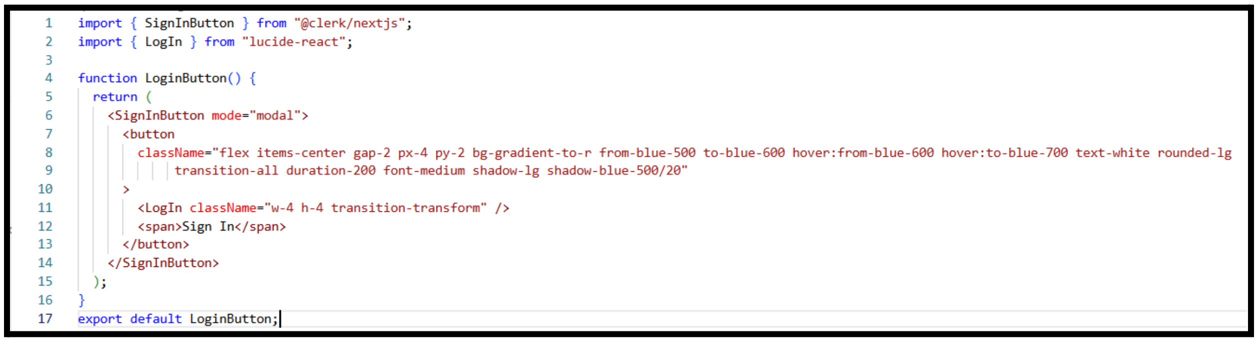
6. Deployment with Vercel:

The project is deployed on **Vercel**, a platform optimized for **Next.js and frontend apps**. Vercel enables:

- Fast global CDN delivery.
- Automatic builds and deployments from GitHub.
- Serverless functions for backend logic.

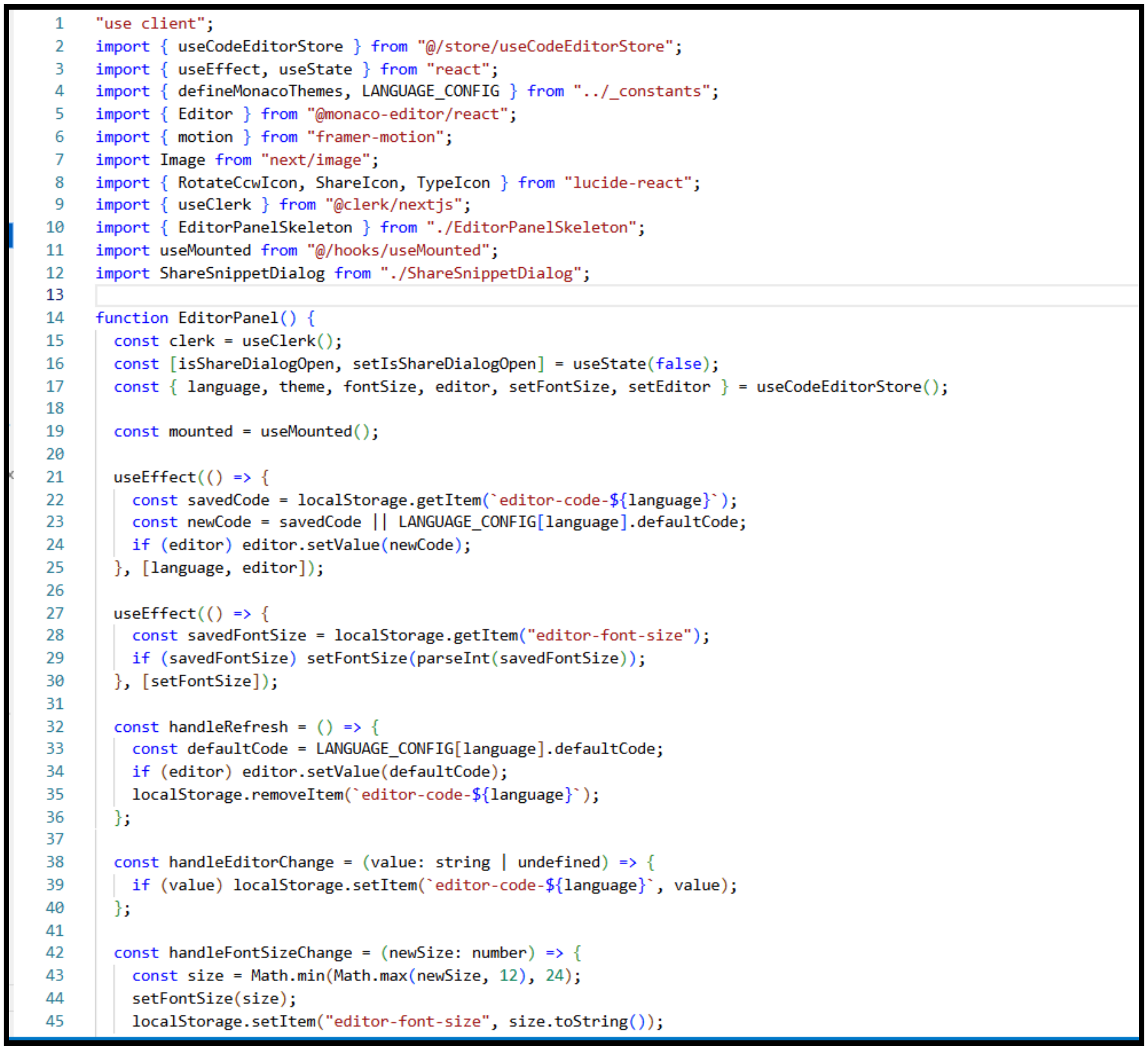
Source Code:

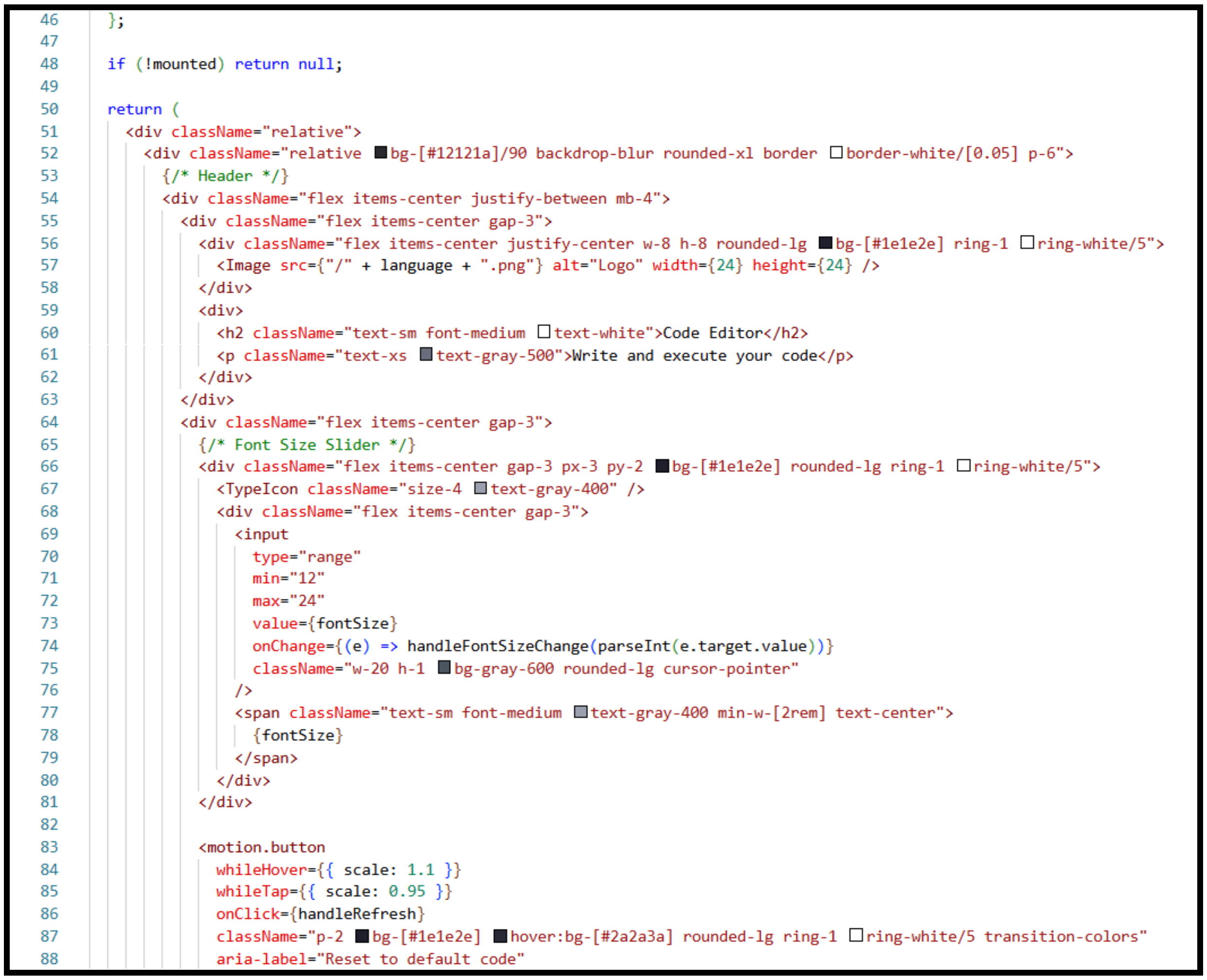
1) Login.tsx:



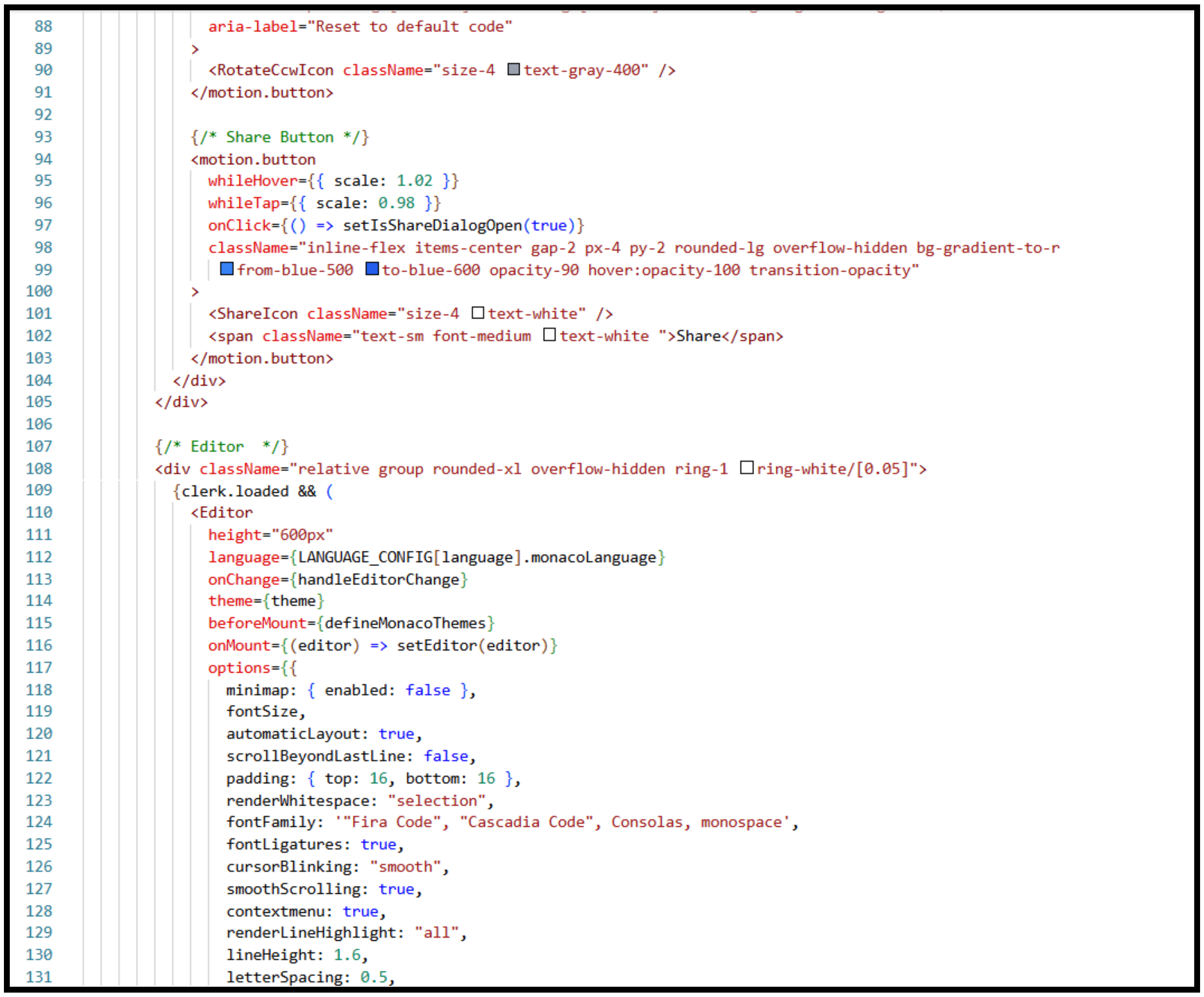
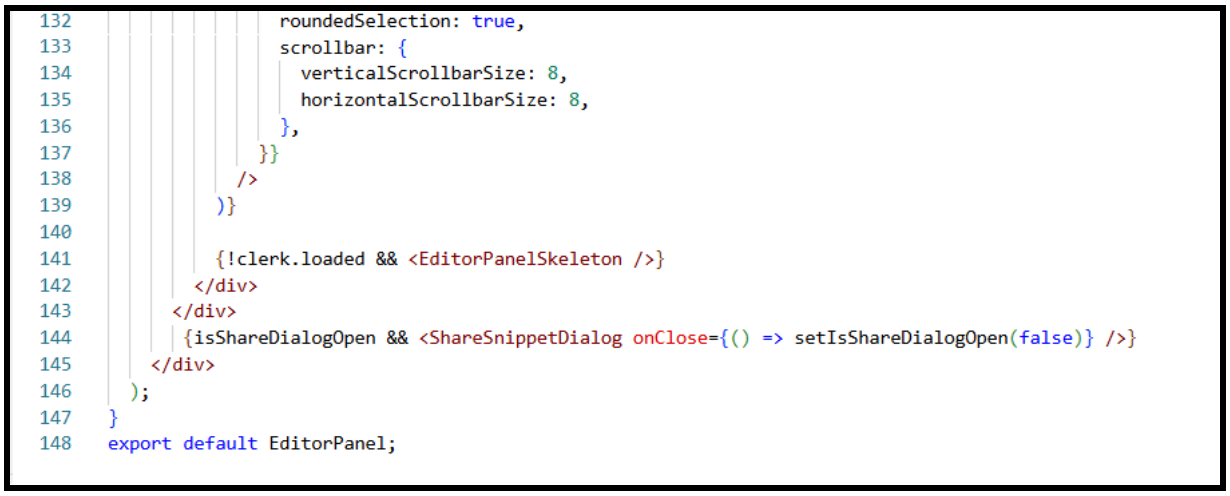
```tsx
import { SignInButton } from "@clerk/nextjs";
import { LogIn } from "lucide-react";

function LoginButton() {
  return (
    <SignInButton mode="modal">
      <button
        className="flex items-center gap-2 px-4 py-2 bg-gradient-to-r from-blue-500 to-blue-600 hover:from-blue-600 hover:to-blue-700 text-white rounded-lg
          transition-all duration-200 font-medium shadow-lg shadow-blue-500/20"
      >
        <LogIn className="w-4 h-4 transition-transform" />
        <span>Sign In</span>
      </button>
    </SignInButton>
  );
}
export default LoginButton;
```

2) EditorPanel.tsx:

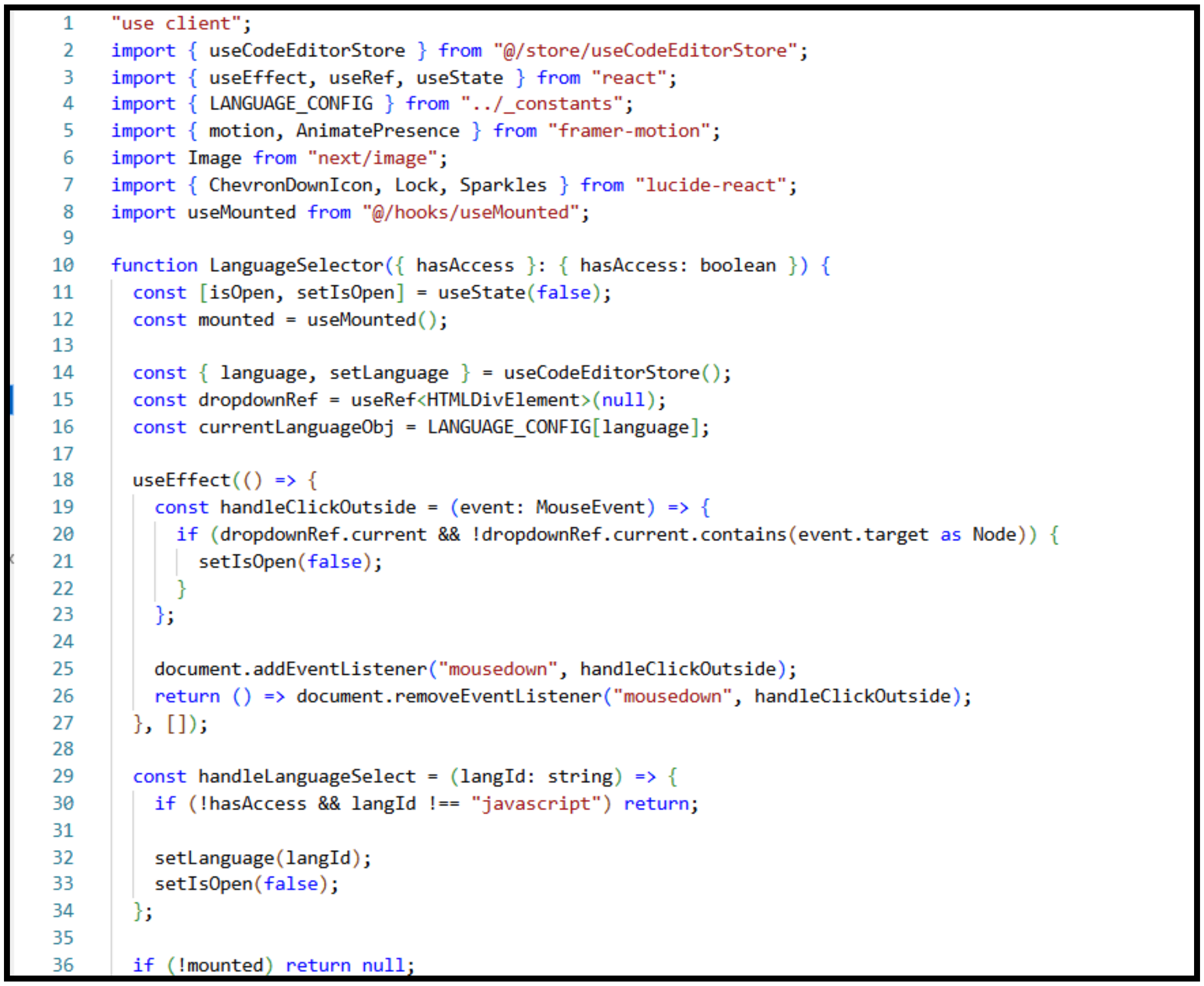

```
"use client";
import { useCodeEditorStore } from "@/store/useCodeEditorStore";
import { useEffect, useState } from "react";
import { defineMonacoThemes, LANGUAGE_CONFIG } from "../_constants";
import { Editor } from "@monaco-editor/react";
import { motion } from "framer-motion";
import Image from "next/image";
import { RotateCcwIcon, ShareIcon, TypeIcon } from "lucide-react";
import { useClerk } from "@clerk/nextjs";
import { EditorPanelSkeleton } from "./EditorPanelSkeleton";
import useMounted from "@/hooks/useMounted";
import ShareSnippetDialog from "./ShareSnippetDialog";

function EditorPanel() {
  const clerk = useClerk();
  const [isShareDialogOpen, setIsShareDialogOpen] = useState(false);
  const { language, theme, fontSize, editor, setFontSize, setEditor } = useCodeEditorStore();

  const mounted = useMounted();

  useEffect(() => {
    const savedCode = localStorage.getItem(`editor-code-${language}`);
    const newCode = savedCode || LANGUAGE_CONFIG[language].defaultCode;
    if (editor) editor.setValue(newCode);
  }, [language, editor]);

  useEffect(() => {
    const savedFontSize = localStorage.getItem("editor-font-size");
    if (savedFontSize) setFontSize(parseInt(savedFontSize));
  }, [setFontSize]);

  const handleRefresh = () => {
    const defaultCode = LANGUAGE_CONFIG[language].defaultCode;
    if (editor) editor.setValue(defaultCode);
    localStorage.removeItem(`editor-code-${language}`);
  };

  const handleEditorChange = (value: string | undefined) => {
    if (value) localStorage.setItem(`editor-code-${language}`, value);
  };

  const handleFontSizeChange = (newSize: number) => {
    const size = Math.min(Math.max(newSize, 12), 24);
    setFontSize(size);
    localStorage.setItem("editor-font-size", size.toString());
```



```
  };

  if (!mounted) return null;

  return (
    <div className="relative">
      <div className="relative bg-[#12121a]/90 backdrop-blur rounded-xl border border-white/[0.05] p-6">
        {/* Header */}
        <div className="flex items-center justify-between mb-4">
          <div className="flex items-center gap-3">
            <div className="flex items-center justify-center w-8 h-8 rounded-lg bg-[#1e1e2e] ring-1 ring-white/5">
              <Image src={"/" + language + ".png"} alt="Logo" width={24} height={24} />
            </div>
            <div>
              <h2 className="text-sm font-medium text-white">Code Editor</h2>
              <p className="text-xs text-gray-500">Write and execute your code</p>
            </div>
          </div>
          <div className="flex items-center gap-3">
            {/* Font Size Slider */}
            <div className="flex items-center gap-3 px-3 py-2 bg-[#1e1e2e] rounded-lg ring-1 ring-white/5">
              <TypeIcon className="size-4 text-gray-400" />
              <div className="flex items-center gap-3">
                <input
                  type="range"
                  min="12"
                  max="24"
                  value={fontSize}
                  onChange={(e) => handleFontSizeChange(parseInt(e.target.value))}
                  className="w-20 h-1 bg-gray-600 rounded-lg cursor-pointer"
                />
                <span className="text-sm font-medium text-gray-400 min-w-[2rem] text-center">
                  {fontSize}
                </span>
              </div>
            </div>

            <motion.button
              whileHover={{ scale: 1.1 }}
              whileTap={{ scale: 0.95 }}
              onClick={handleRefresh}
              className="p-2 bg-[#1e1e2e] hover:bg-[#2a2a3a] rounded-lg ring-1 ring-white/5 transition-colors"
              aria-label="Reset to default code"
```

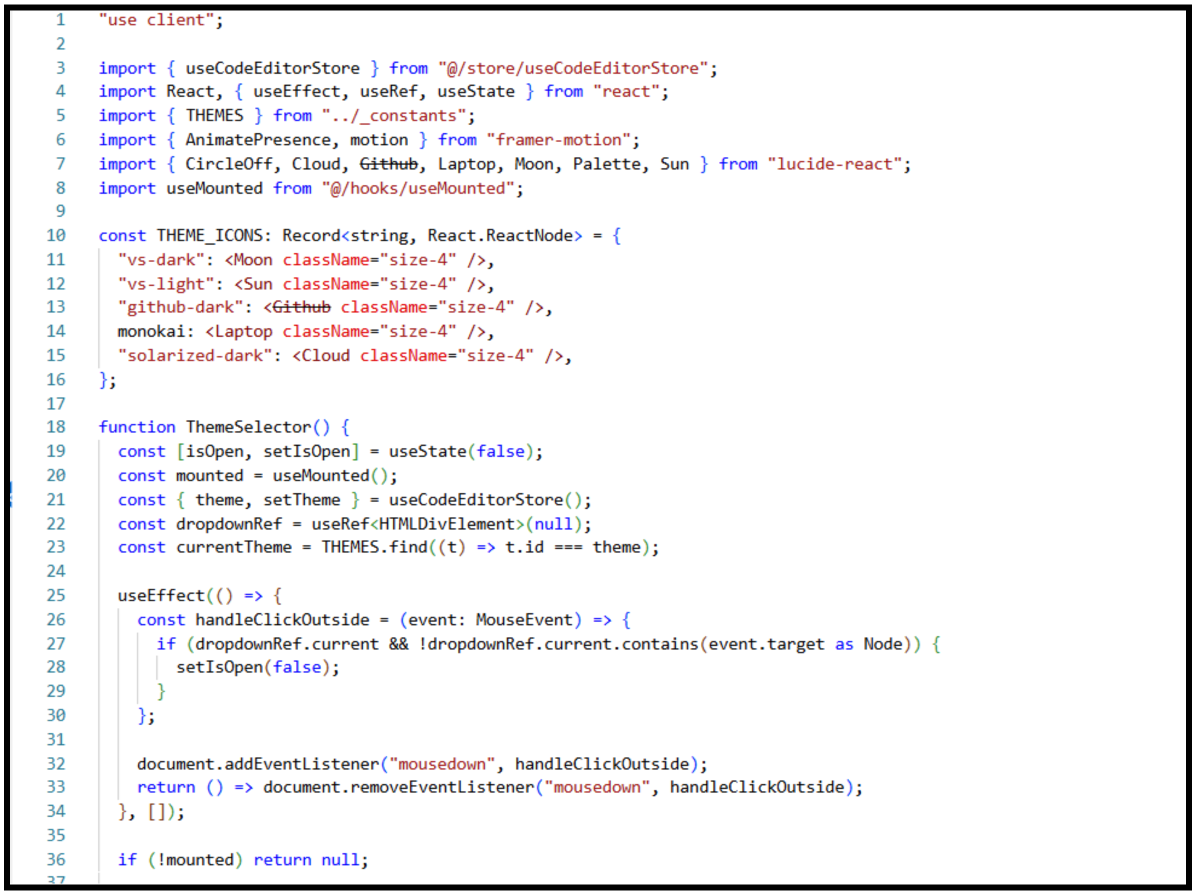

```
              aria-label="Reset to default code"
            >
              <RotateCcwIcon className="size-4 text-gray-400" />
            </motion.button>

            {/* Share Button */}
            <motion.button
              whileHover={{ scale: 1.02 }}
              whileTap={{ scale: 0.98 }}
              onClick={() => setIsShareDialogOpen(true)}
              className="inline-flex items-center gap-2 px-4 py-2 rounded-lg overflow-hidden bg-gradient-to-r
               from-blue-500 to-blue-600 opacity-90 hover:opacity-100 transition-opacity"
            >
              <ShareIcon className="size-4 text-white" />
              <span className="text-sm font-medium text-white ">Share</span>
            </motion.button>
          </div>
        </div>

        {/* Editor  */}
        <div className="relative group rounded-xl overflow-hidden ring-1 ring-white/[0.05]">
          {clerk.loaded && (
            <Editor
              height="600px"
              language={LANGUAGE_CONFIG[language].monacoLanguage}
              onChange={handleEditorChange}
              theme={theme}
              beforeMount={defineMonacoThemes}
              onMount={(editor) => setEditor(editor)}
              options={{
                minimap: { enabled: false },
                fontSize,
                automaticLayout: true,
                scrollBeyondLastLine: false,
                padding: { top: 16, bottom: 16 },
                renderWhitespace: "selection",
                fontFamily: '"Fira Code", "Cascadia Code", Consolas, monospace',
                fontLigatures: true,
                cursorBlinking: "smooth",
                smoothScrolling: true,
                contextmenu: true,
                renderLineHighlight: "all",
                lineHeight: 1.6,
                letterSpacing: 0.5,
```

```
                roundedSelection: true,
                scrollbar: {
                  verticalScrollbarSize: 8,
                  horizontalScrollbarSize: 8,
                },
              }}
            />
          )}

          {!clerk.loaded && <EditorPanelSkeleton />}
        </div>
      </div>
      {isShareDialogOpen && <ShareSnippetDialog onClose={() => setIsShareDialogOpen(false)} />}
    </div>
  );
}
export default EditorPanel;
```

3) Language.tsx:



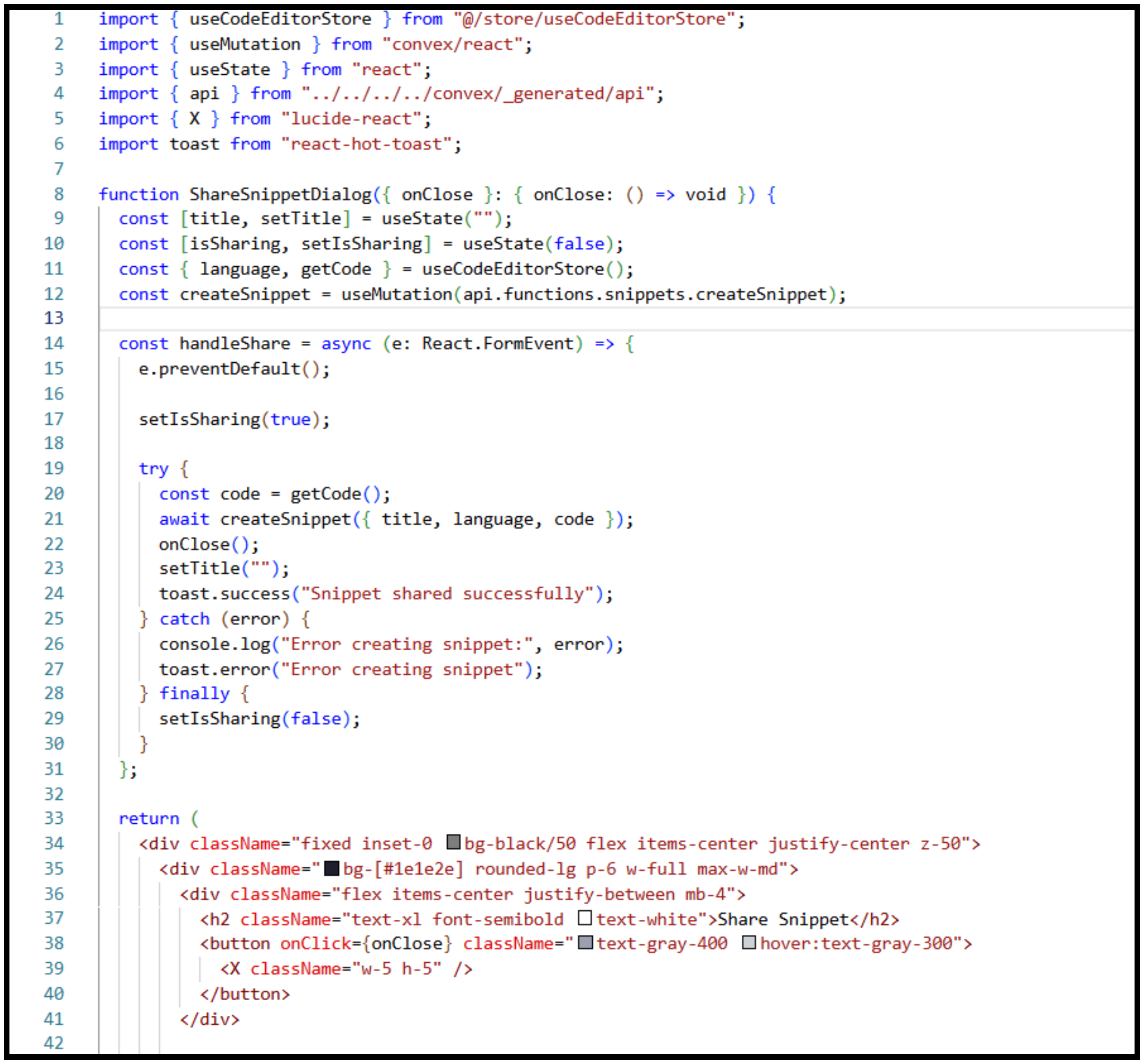
```
"use client";
import { useCodeEditorStore } from "@/store/useCodeEditorStore";
import { useEffect, useRef, useState } from "react";
import { LANGUAGE_CONFIG } from "../_constants";
import { motion, AnimatePresence } from "framer-motion";
import Image from "next/image";
import { ChevronDownIcon, Lock, Sparkles } from "lucide-react";
import useMounted from "@/hooks/useMounted";

function LanguageSelector({ hasAccess }: { hasAccess: boolean }) {
  const [isOpen, setIsOpen] = useState(false);
  const mounted = useMounted();

  const { language, setLanguage } = useCodeEditorStore();
  const dropdownRef = useRef<HTMLDivElement>(null);
  const currentLanguageObj = LANGUAGE_CONFIG[language];

  useEffect(() => {
    const handleClickOutside = (event: MouseEvent) => {
      if (dropdownRef.current && !dropdownRef.current.contains(event.target as Node)) {
        setIsOpen(false);
      }
    };

    document.addEventListener("mousedown", handleClickOutside);
    return () => document.removeEventListener("mousedown", handleClickOutside);
  }, []);

  const handleLanguageSelect = (langId: string) => {
    if (!hasAccess && langId !== "javascript") return;

    setLanguage(langId);
    setIsOpen(false);
  };

  if (!mounted) return null;
```

4) ThemeSelector.tsx:



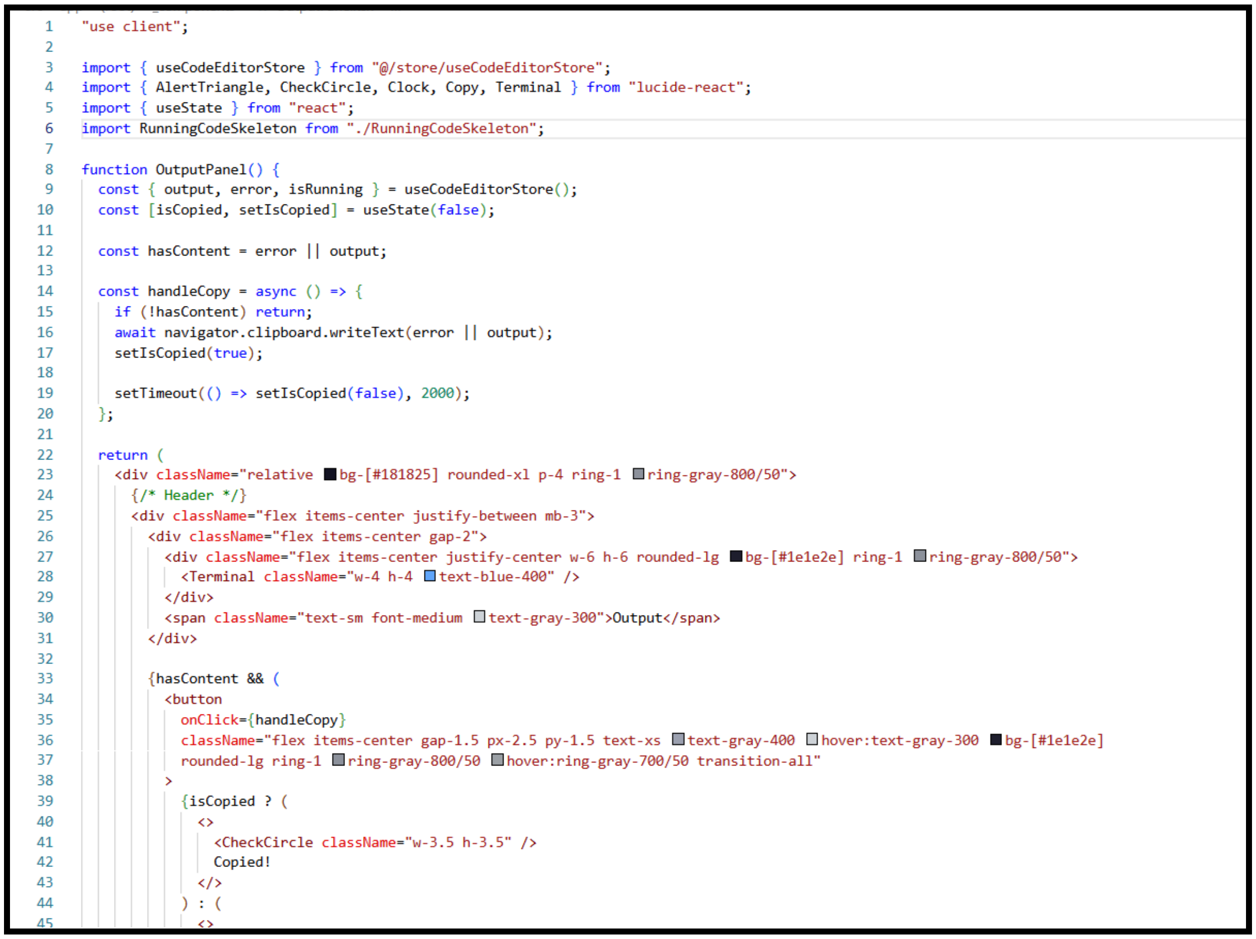
```tsx
"use client";

import { useCodeEditorStore } from "@/store/useCodeEditorStore";
import React, { useEffect, useRef, useState } from "react";
import { THEMES } from "../_constants";
import { AnimatePresence, motion } from "framer-motion";
import { CircleOff, Cloud, ~~Github~~, Laptop, Moon, Palette, Sun } from "lucide-react";
import useMounted from "@/hooks/useMounted";

const THEME_ICONS: Record<string, React.ReactNode> = {
  "vs-dark": <Moon className="size-4" />,
  "vs-light": <Sun className="size-4" />,
  "github-dark": <~~Github~~ className="size-4" />,
  monokai: <Laptop className="size-4" />,
  "solarized-dark": <Cloud className="size-4" />,
};

function ThemeSelector() {
  const [isOpen, setIsOpen] = useState(false);
  const mounted = useMounted();
  const { theme, setTheme } = useCodeEditorStore();
  const dropdownRef = useRef<HTMLDivElement>(null);
  const currentTheme = THEMES.find((t) => t.id === theme);

  useEffect(() => {
    const handleClickOutside = (event: MouseEvent) => {
      if (dropdownRef.current && !dropdownRef.current.contains(event.target as Node)) {
        setIsOpen(false);
      }
    };

    document.addEventListener("mousedown", handleClickOutside);
    return () => document.removeEventListener("mousedown", handleClickOutside);
  }, []);

  if (!mounted) return null;
```

5) Snippet.tsx:



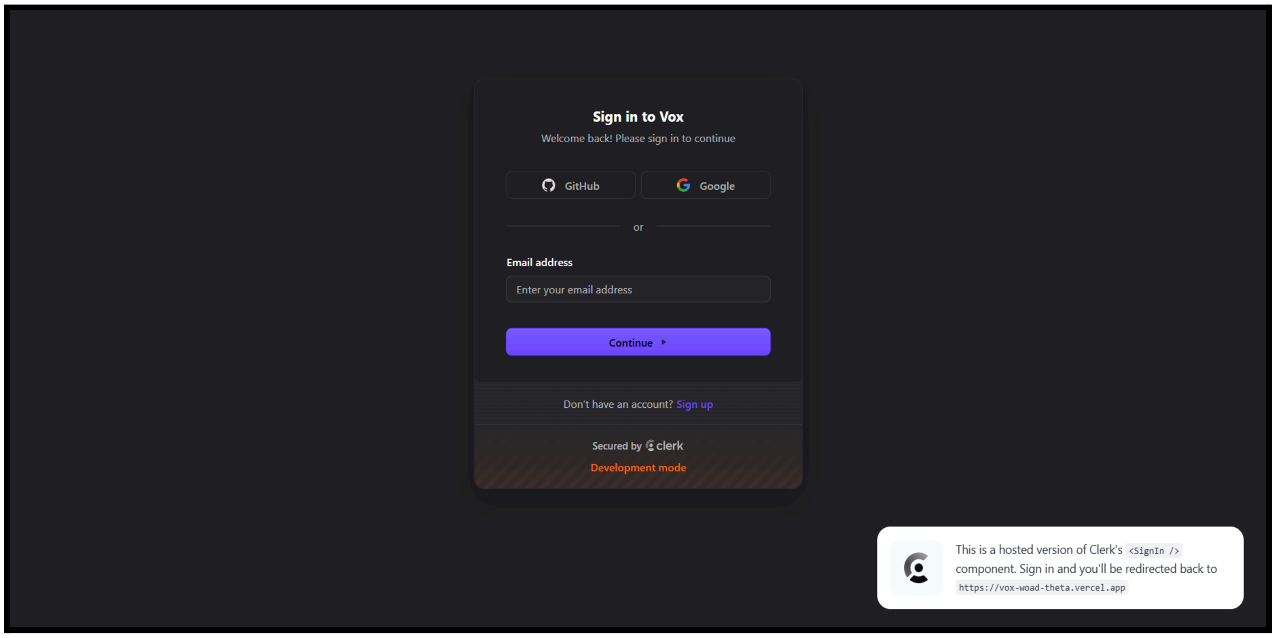
```tsx
import { useCodeEditorStore } from "@/store/useCodeEditorStore";
import { useMutation } from "convex/react";
import { useState } from "react";
import { api } from "../../../../convex/_generated/api";
import { X } from "lucide-react";
import toast from "react-hot-toast";

function ShareSnippetDialog({ onClose }: { onClose: () => void }) {
  const [title, setTitle] = useState("");
  const [isSharing, setIsSharing] = useState(false);
  const { language, getCode } = useCodeEditorStore();
  const createSnippet = useMutation(api.functions.snippets.createSnippet);

  const handleShare = async (e: React.FormEvent) => {
    e.preventDefault();

    setIsSharing(true);

    try {
      const code = getCode();
      await createSnippet({ title, language, code });
      onClose();
      setTitle("");
      toast.success("Snippet shared successfully");
    } catch (error) {
      console.log("Error creating snippet:", error);
      toast.error("Error creating snippet");
    } finally {
      setIsSharing(false);
    }
  };

  return (
    <div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50">
      <div className="bg-[#1e1e2e] rounded-lg p-6 w-full max-w-md">
        <div className="flex items-center justify-between mb-4">
          <h2 className="text-xl font-semibold text-white">Share Snippet</h2>
          <button onClick={onClose} className="text-gray-400 hover:text-gray-300">
            <X className="w-5 h-5" />
          </button>
        </div>

```

6) Output.tsx:



```
"use client";

import { useCodeEditorStore } from "@/store/useCodeEditorStore";
import { AlertTriangle, CheckCircle, Clock, Copy, Terminal } from "lucide-react";
import { useState } from "react";
import RunningCodeSkeleton from "./RunningCodeSkeleton";

function OutputPanel() {
  const { output, error, isRunning } = useCodeEditorStore();
  const [isCopied, setIsCopied] = useState(false);

  const hasContent = error || output;

  const handleCopy = async () => {
    if (!hasContent) return;
    await navigator.clipboard.writeText(error || output);
    setIsCopied(true);

    setTimeout(() => setIsCopied(false), 2000);
  };

  return (
    <div className="relative bg-[#181825] rounded-xl p-4 ring-1 ring-gray-800/50">
      {/* Header */}
      <div className="flex items-center justify-between mb-3">
        <div className="flex items-center gap-2">
          <div className="flex items-center justify-center w-6 h-6 rounded-lg bg-[#1e1e2e] ring-1 ring-gray-800/50">
            <Terminal className="w-4 h-4 text-blue-400" />
          </div>
          <span className="text-sm font-medium text-gray-300">Output</span>
        </div>

        {hasContent && (
          <button
            onClick={handleCopy}
            className="flex items-center gap-1.5 px-2.5 py-1.5 text-xs text-gray-400 hover:text-gray-300 bg-[#1e1e2e]
            rounded-lg ring-1 ring-gray-800/50 hover:ring-gray-700/50 transition-all"
          >
            {isCopied ? (
              <>
                <CheckCircle className="w-3.5 h-3.5" />
                Copied!
              </>
            ) : (
              <>
```

Output:

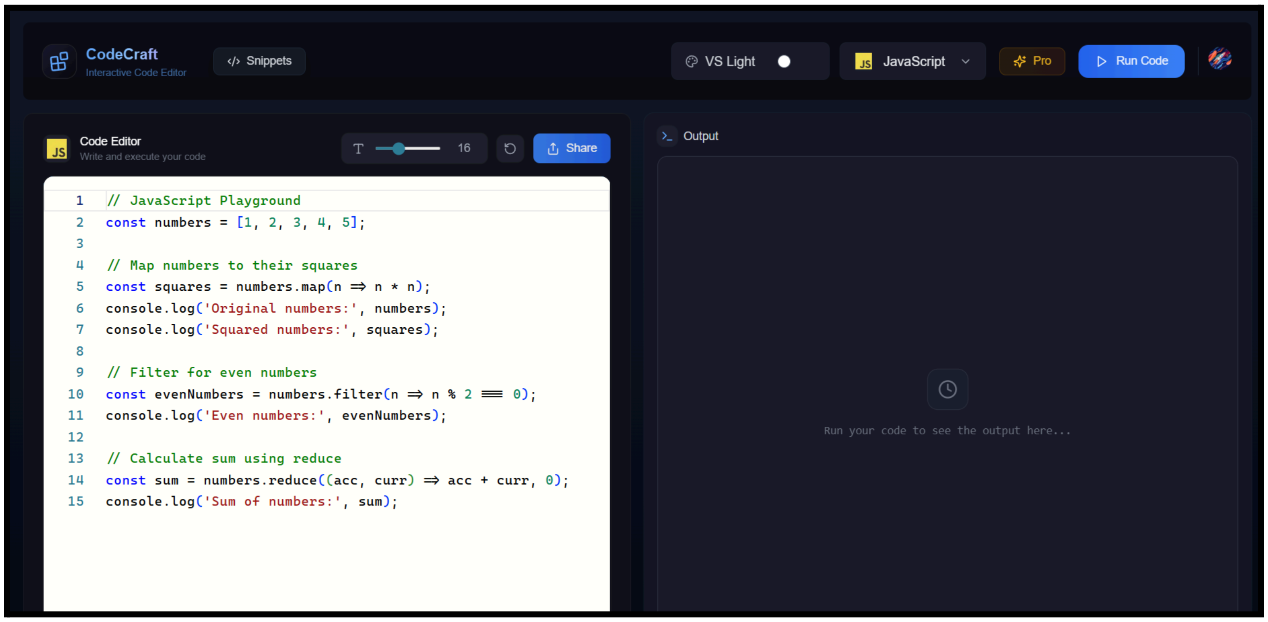


Sign in to Vox

Welcome back! Please sign in to continue

This is a hosted version of Clerk's `<SignIn />` component. Sign in and you'll be redirected back to `https://vox-woad-theta.vercel.app`

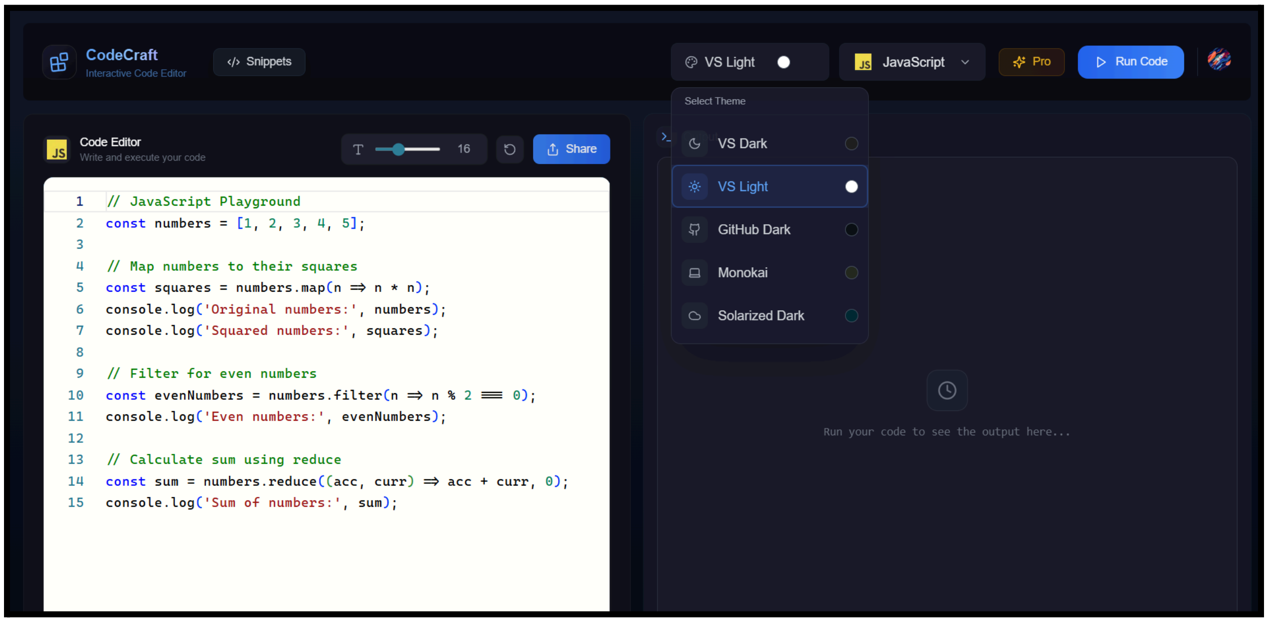
Figure 1 - Sign-in Page of the Website



```javascript
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];

// Map numbers to their squares
const squares = numbers.map(n => n * n);
console.log('Original numbers:', numbers);
console.log('Squared numbers:', squares);

// Filter for even numbers
const evenNumbers = numbers.filter(n => n % 2 === 0);
console.log('Even numbers:', evenNumbers);

// Calculate sum using reduce
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log('Sum of numbers:', sum);
```

```
Run your code to see the output here...
```

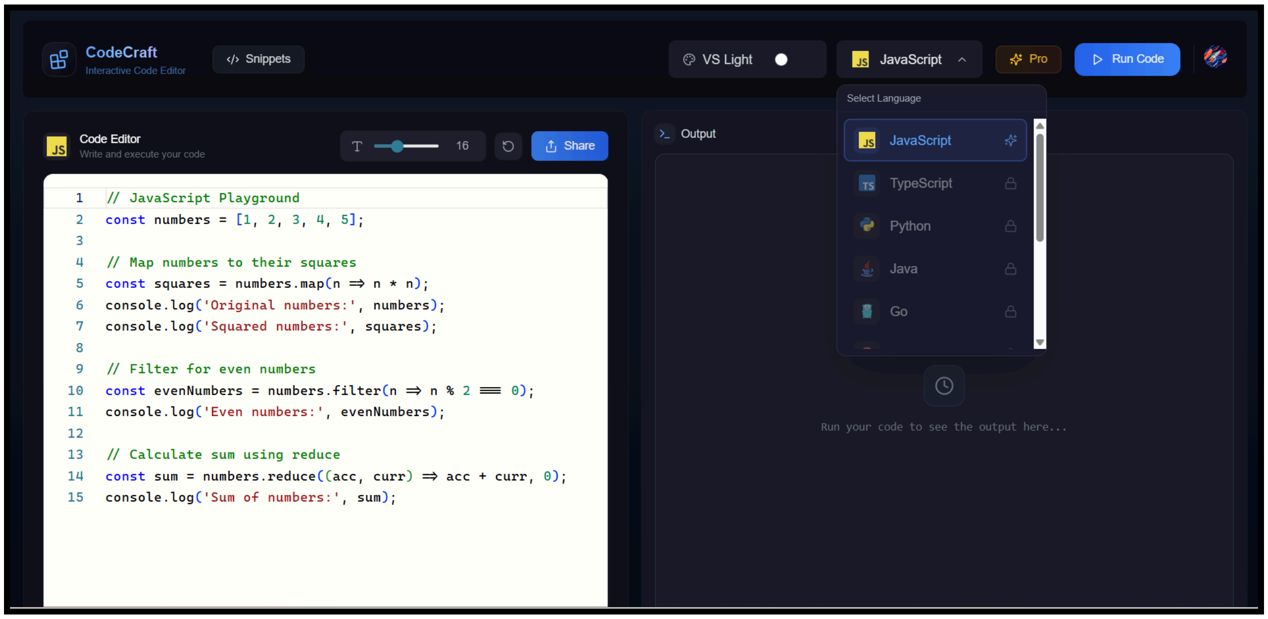
Figure 2 - Home Page of the Website



```javascript
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];

// Map numbers to their squares
const squares = numbers.map(n => n * n);
console.log('Original numbers:', numbers);
console.log('Squared numbers:', squares);

// Filter for even numbers
const evenNumbers = numbers.filter(n => n % 2 === 0);
console.log('Even numbers:', evenNumbers);

// Calculate sum using reduce
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log('Sum of numbers:', sum);
```

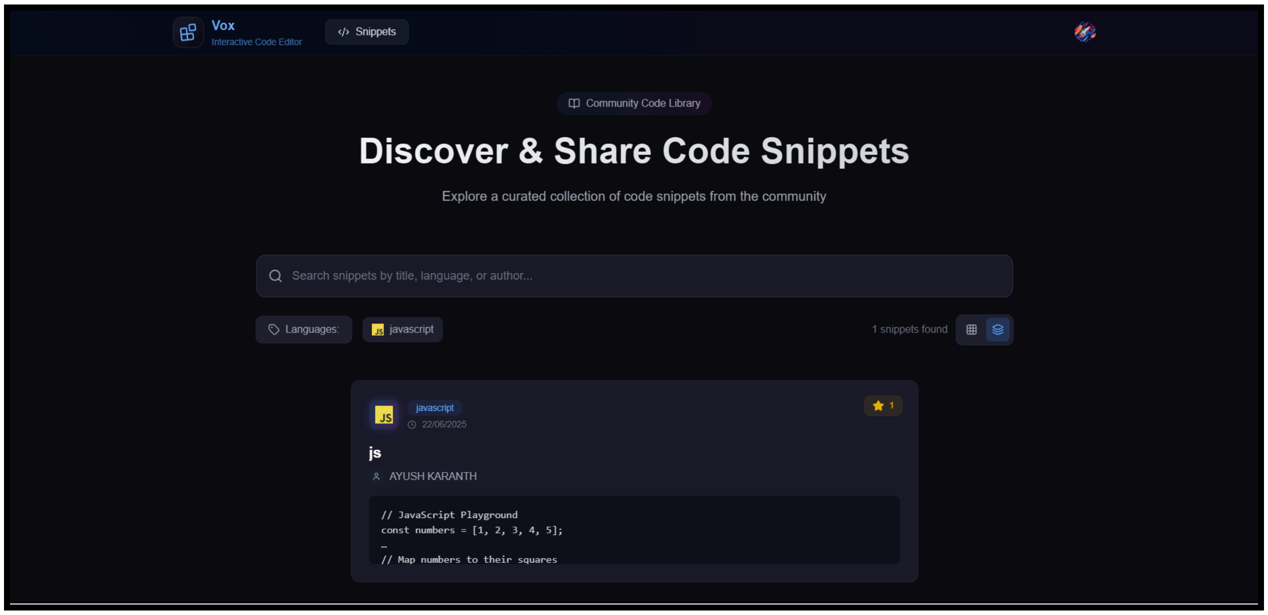
```
Run your code to see the output here...
```

Figure 3 - Color Themes Functionality



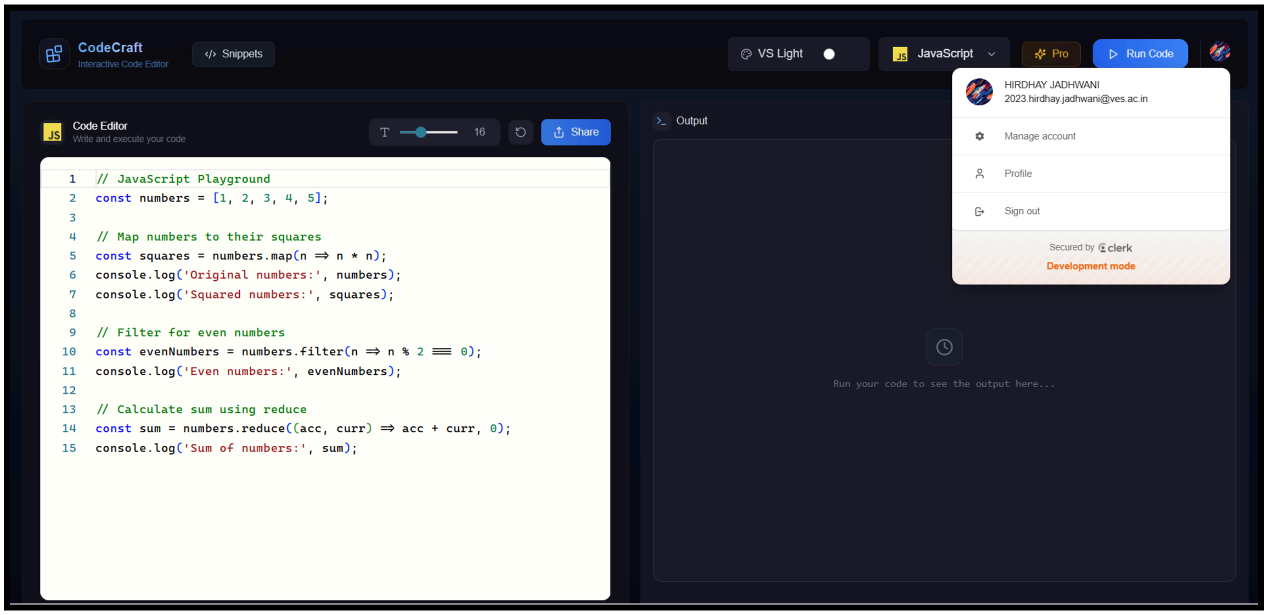
```javascript
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];

// Map numbers to their squares
const squares = numbers.map(n => n * n);
console.log('Original numbers:', numbers);
console.log('Squared numbers:', squares);

// Filter for even numbers
const evenNumbers = numbers.filter(n => n % 2 === 0);
console.log('Even numbers:', evenNumbers);

// Calculate sum using reduce
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log('Sum of numbers:', sum);
```

Figure 4 - Coding Language Functionality


Vox
Interactive Code Editor
Snippets
Community Code Library
Discover & Share Code Snippets
Explore a curated collection of code snippets from the community
Search snippets by title, language, or author...
Languages:
javascript
1 snippets found
javascript
22/06/2025
1
js
AYUSH KARANTH

```javascript
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];
…
// Map numbers to their squares
```

Figure 5 - Snippets Functionality



```javascript
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];

// Map numbers to their squares
const squares = numbers.map(n => n * n);
console.log('Original numbers:', numbers);
console.log('Squared numbers:', squares);

// Filter for even numbers
const evenNumbers = numbers.filter(n => n % 2 === 0);
console.log('Even numbers:', evenNumbers);

// Calculate sum using reduce
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log('Sum of numbers:', sum);
```

Figure 6 - Profile Functionality

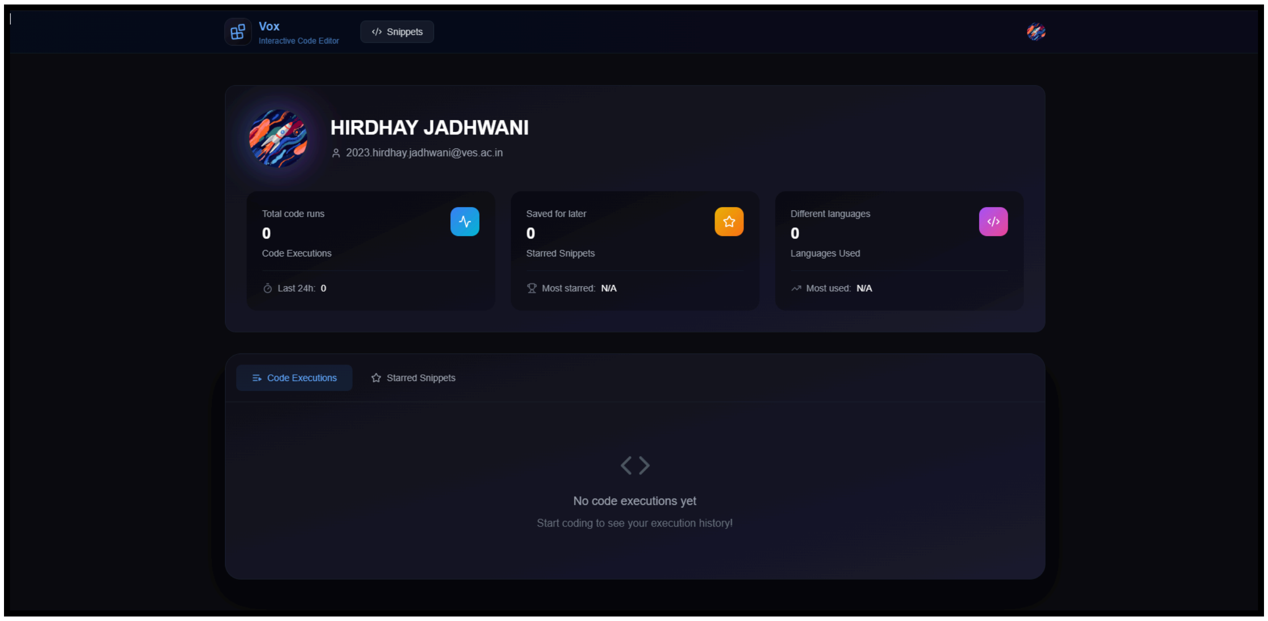
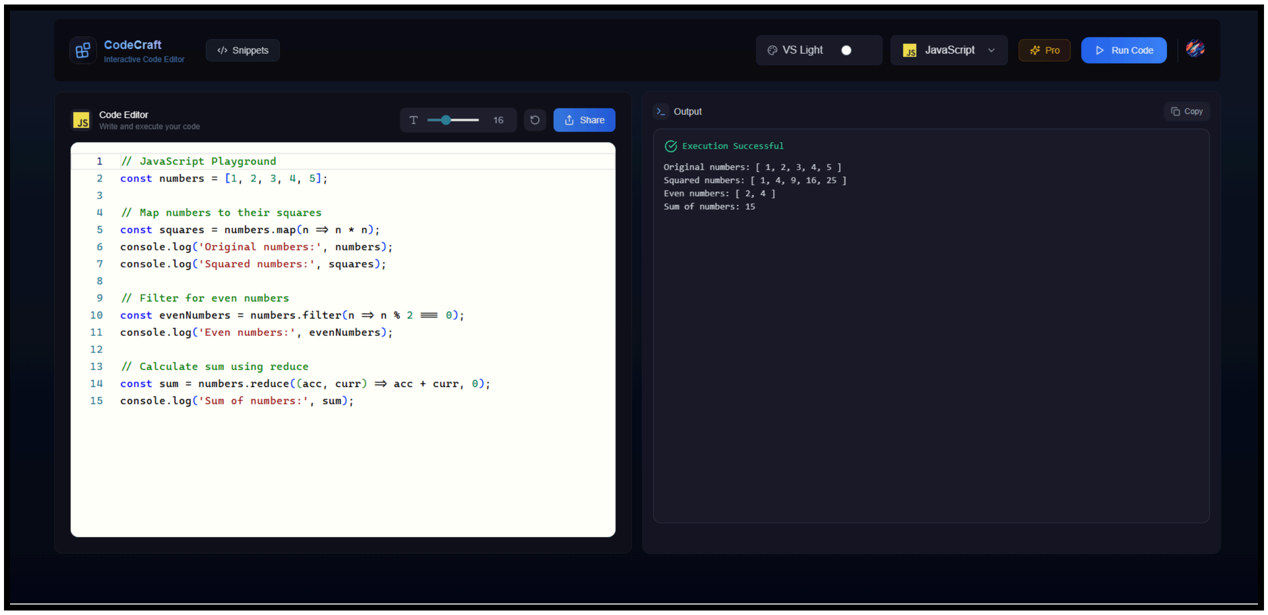


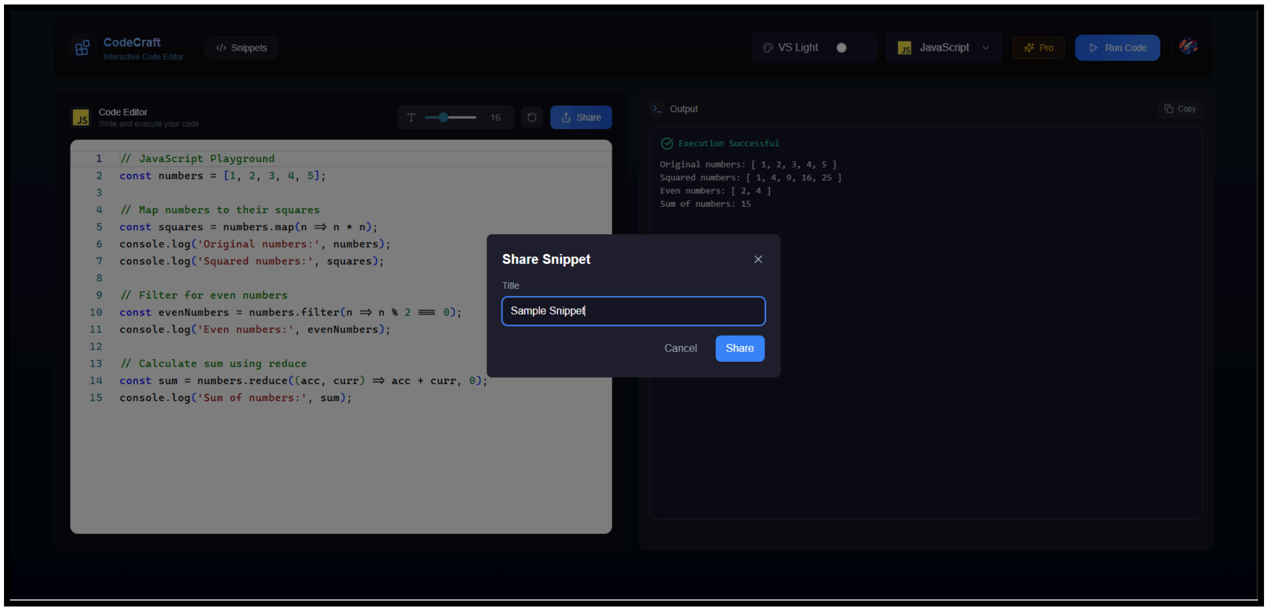
Figure 7 - Profile Page



```javascript
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];

// Map numbers to their squares
const squares = numbers.map(n => n * n);
console.log('Original numbers:', numbers);
console.log('Squared numbers:', squares);

// Filter for even numbers
const evenNumbers = numbers.filter(n => n % 2 === 0);
console.log('Even numbers:', evenNumbers);

// Calculate sum using reduce
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log('Sum of numbers:', sum);
```

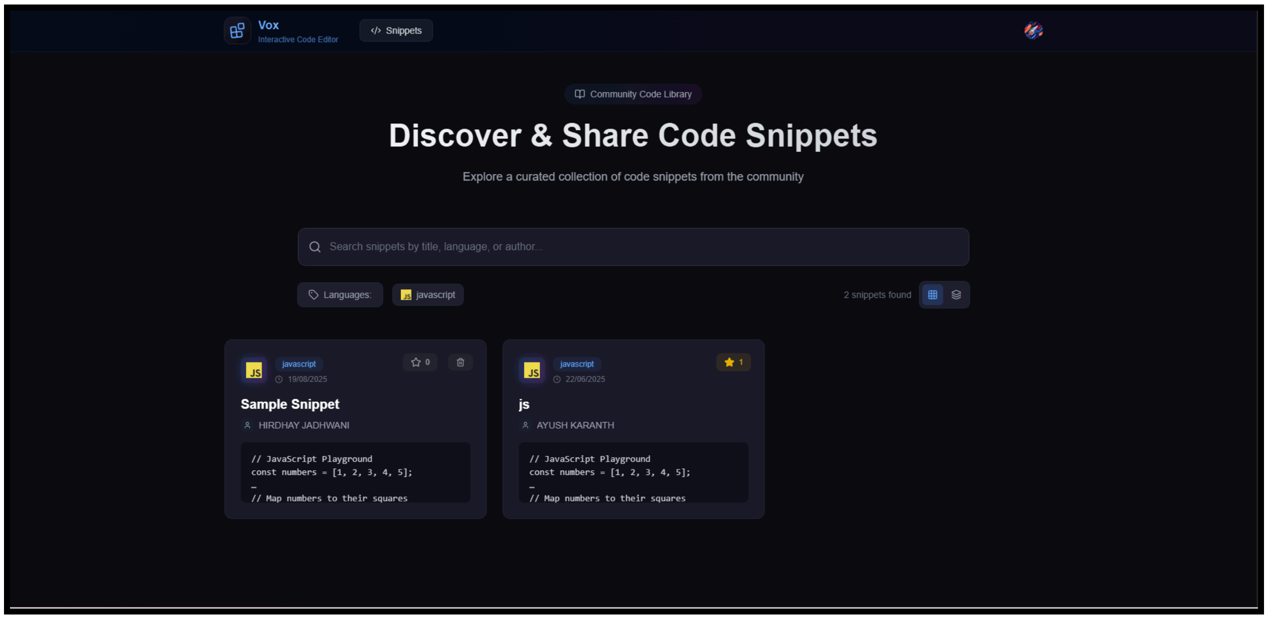
```
Execution Successful

Original numbers: [ 1, 2, 3, 4, 5 ]
Squared numbers: [ 1, 4, 9, 16, 25 ]
Even numbers: [ 2, 4 ]
Sum of numbers: 15
```

Figure 8 - Output After Hitting the Run Code button



Title: Sample Snippet

Figure 9 - Share functionality



Discover & Share Code Snippets

Explore a curated collection of code snippets from the community

javascript · 19/08/2025 · 0

Sample Snippet

HIRDHAY JADHWANI

```
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];
…
// Map numbers to their squares
```

javascript · 22/06/2025 · 1

js

AYUSH KARANTH

```
// JavaScript Playground
const numbers = [1, 2, 3, 4, 5];
…
// Map numbers to their squares
```

Figure 10 - Share Page (Similar to Snippets) where all the snippets are saved

To view this project's source code: https://github.com/Ayushkaranth/Vox
This is the live demo deployed in Vercel: https://vox-woad-theta.vercel.app/

<u>Conclusion:</u>

This experiment successfully demonstrates how **Tailwind CSS** can be used to create **responsive and interactive UIs** without writing traditional CSS. The project's **component-based architecture** ensures clean code and easy scalability. Additionally, integrating a **payment system like LemonSqueezy** introduces real-world monetization capability for premium features, making the application production-ready. Finally, deploying on **Vercel** provides a robust hosting solution for modern web applications