Experiment no. 01

AIM: Build Responsive and Interactive UIs using Tailwind CSS

THEORY:

Prerequisites

Before you start working with Tailwind CSS, make sure you have the following:

- **Basic Knowledge** of HTML, CSS, and JavaScript
- **Installed Software & Tools**:
    - Node.js (with npm for packages)
    - Code Editor (VS Code preferred)
    - Web Browser (Chrome/Edge/Firefox)
- **Development Environment Setup** (Git Recommended)
- **Tailwind CSS Knowledge (Basic)**
- **Package Installation Skills**
- **Build Tool Familiarity**

🔹 What is TailwindCSS?

Tailwind CSS is a **utility-first CSS framework** used to design responsive and interactive UIs faster. It provides predefined low-level classes (e.g., `flex`, `grid`, `bg-blue-500`, `hover:shadow-lg`) instead of large component styles.

🔹 How to install TailwindCSS?

1. Open your IDE and **terminal prompt**.

Install Tailwind CSS:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

2.

Add Tailwind directives in `input.css`:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

3.
4. Configure paths in `tailwind.config.js`.

Start development with:

```
npx tailwindcss -i ./src/input.css -o ./dist/output.css --watch
```

5.

🔹 **Why is TailwindCSS required?**

1. **Rapid UI Development** – Build interfaces faster using predefined utility classes.
2. **Responsiveness** – Built-in breakpoints (`sm:`, `md:`, `lg:`, `xl:`).
3. **Customization** – Configure themes, fonts, spacing, colors in `tailwind.config.js`.
4. **Design Consistency** – Same design patterns throughout.
5. **Small Bundle Sizes** – Removes unused CSS automatically in production.
6. **Great for Frameworks** – Works seamlessly with React, Vue, Next.js, etc.

🔹 **Types of Components in Tailwind**

1. **UI Components (HTML + Utility Classes)**
   Reusable building blocks created with utility classes directly in HTML. Examples:

buttons, cards, forms, alerts.

Example:

```
<button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2
px-4 rounded">
  Click Me
</button>
```

Benefits: Fast prototyping, customizable, no separate CSS required, clean.

2. **Extracted Component Classes using @apply**
   You can **group utility classes** into custom CSS classes.

Example (`styles.css`):

```
.btn-primary {
  @apply bg-blue-600 hover:bg-blue-700 text-white font-bold py-2 px-4
rounded;
}
```

Use in HTML:

```
<button class="btn-primary">Submit</button>
```

Benefits: Cleaner HTML, reusable, readable.

3. **Component Files in Frameworks (React, Vue, etc.)**
   Works perfectly with **JSX/TSX templates** in frameworks.

Example (React):

```
function Button({ label }) {
  return (
```

```
    <button className="bg-green-500 hover:bg-green-600 text-white py-2
px-4 rounded">
      {label}
    </button>
  );
}
```

Benefits: Supports component-based development, responsive directly in template.

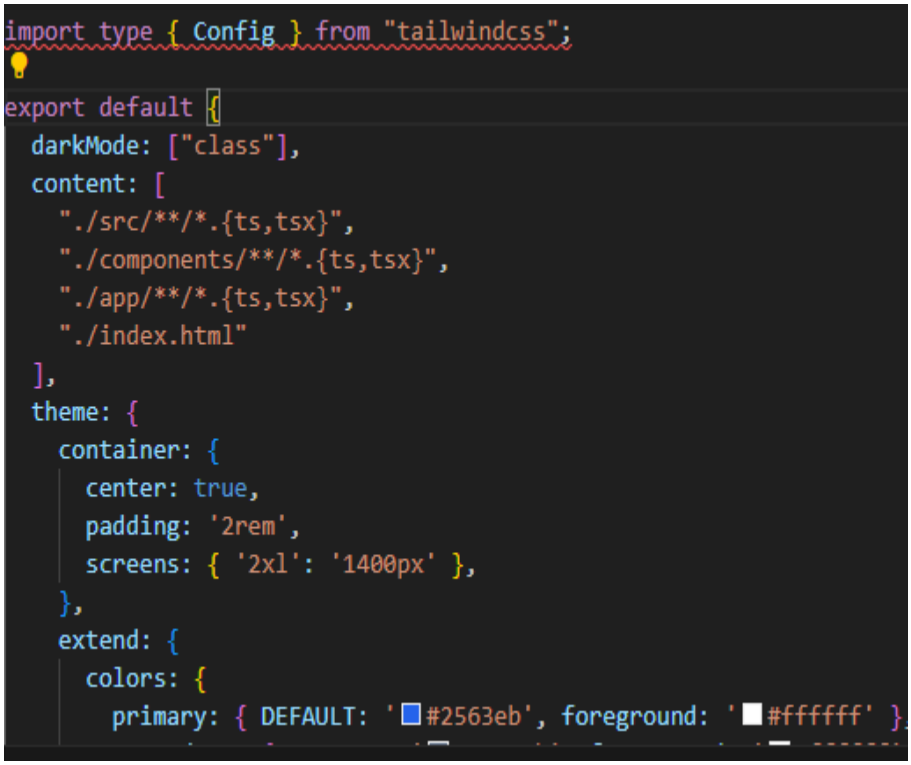
4. **Tailwind UI (Paid Library)**
   A premium collection of ready-made UI components.

👉 Example Modal:

```
<div class="relative bg-white rounded-lg shadow p-6">
  <h3 class="text-lg font-medium text-gray-900">Confirm Action</h3>
  <p class="mt-2 text-sm text-gray-500">Are you sure you want to
proceed?</p>
  <div class="mt-4 flex justify-end">
    <button class="bg-blue-600 text-white px-4 py-2 rounded
hover:bg-blue-700">
      Confirm
    </button>
  </div>
</div>
```

The **30% extra work**, we enhance the project by adding **dark mode support**, **animated transitions**, and an **interactive contact form**. Dark mode is enabled using Tailwind's `dark:` variant. By toggling a button, users can switch between light and dark themes, and elements automatically adjust their colors, e.g., `bg-white dark:bg-gray-800` and `text-gray-800 dark:text-gray-200`. We also add **Framer Motion-like animations** using Tailwind's built-in `animate-bounce`, `animate-pulse`, and smooth transition classes for better user experience. Finally, we build a responsive **contact form** with fields for name, email, and message. The form uses Tailwind's focus utilities (`focus:ring-2 focus:ring-blue-500`) for accessibility and clear input feedback.

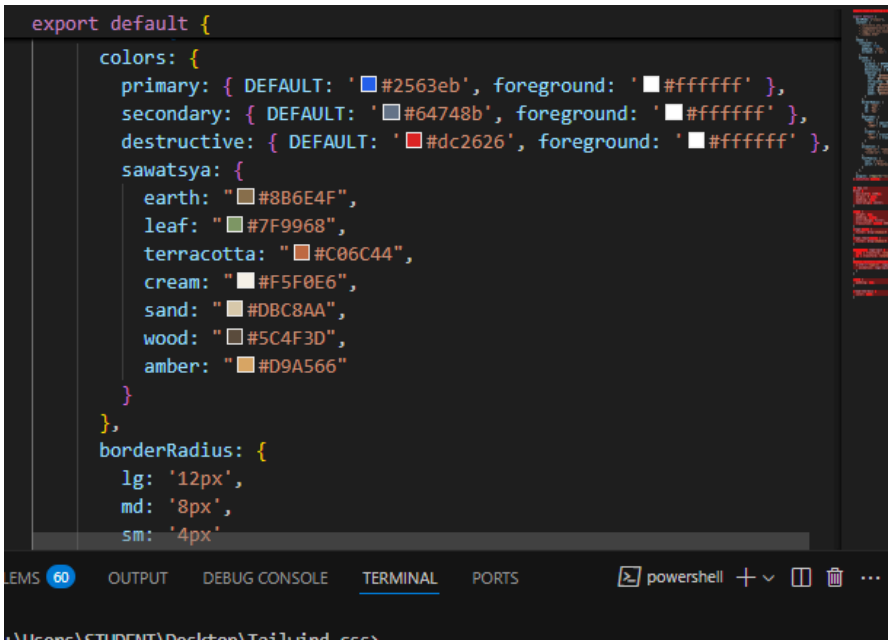
Source Code Implementation



```
import type { Config } from "tailwindcss";

export default {
  darkMode: ["class"],
  content: [
    "./src/**/*.{ts,tsx}",
    "./components/**/*.{ts,tsx}",
    "./app/**/*.{ts,tsx}",
    "./index.html"
  ],
  theme: {
    container: {
      center: true,
      padding: '2rem',
      screens: { '2xl': '1400px' },
    },
    extend: {
      colors: {
        primary: { DEFAULT: '#2563eb', foreground: '#ffffff' },
```

Fig 1.1



```
export default {
    colors: {
      primary: { DEFAULT: '#2563eb', foreground: '#ffffff' },
      secondary: { DEFAULT: '#64748b', foreground: '#ffffff' },
      destructive: { DEFAULT: '#dc2626', foreground: '#ffffff' },
      sawatsya: {
        earth: "#8B6E4F",
        leaf: "#7F9968",
        terracotta: "#C06C44",
        cream: "#F5F0E6",
        sand: "#DBC8AA",
        wood: "#5C4F3D",
        amber: "#D9A566"
      }
    },
    borderRadius: {
      lg: '12px',
      md: '8px',
      sm: '4px'
```

LEMS 60 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

Fig 1.2

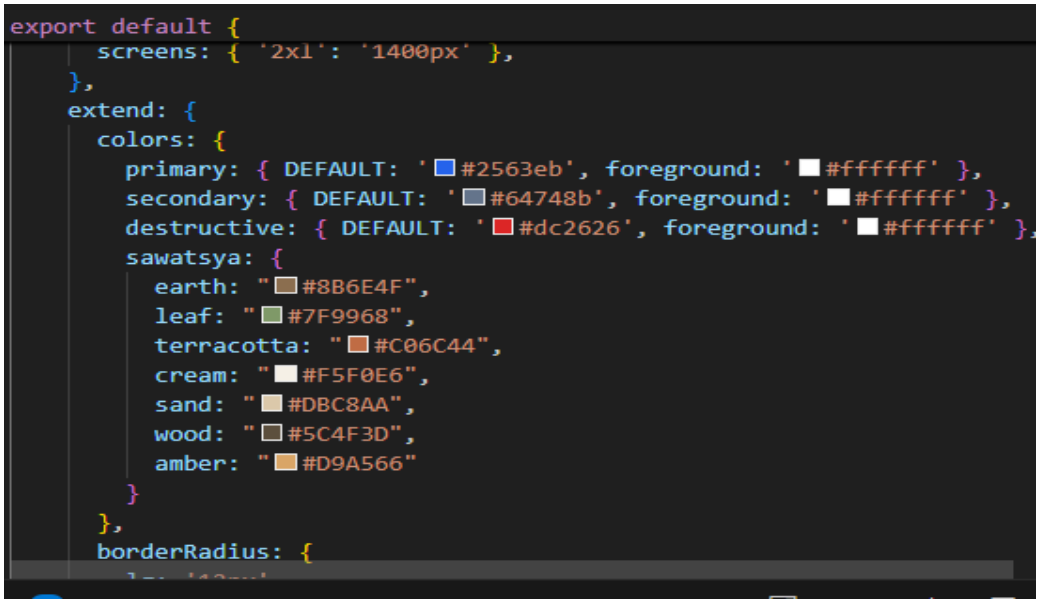

export default {
screens: { '2xl': '1400px' },
},
extend: {
colors: {
primary: { DEFAULT: '#2563eb', foreground: '#ffffff' },
secondary: { DEFAULT: '#64748b', foreground: '#ffffff' },
destructive: { DEFAULT: '#dc2626', foreground: '#ffffff' },
sawatsya: {
earth: "#8B6E4F",
leaf: "#7F9968",
terracotta: "#C06C44",
cream: "#F5F0E6",
sand: "#DBC8AA",
wood: "#5C4F3D",
amber: "#D9A566"
}
},
borderRadius: {

Fig 1.3

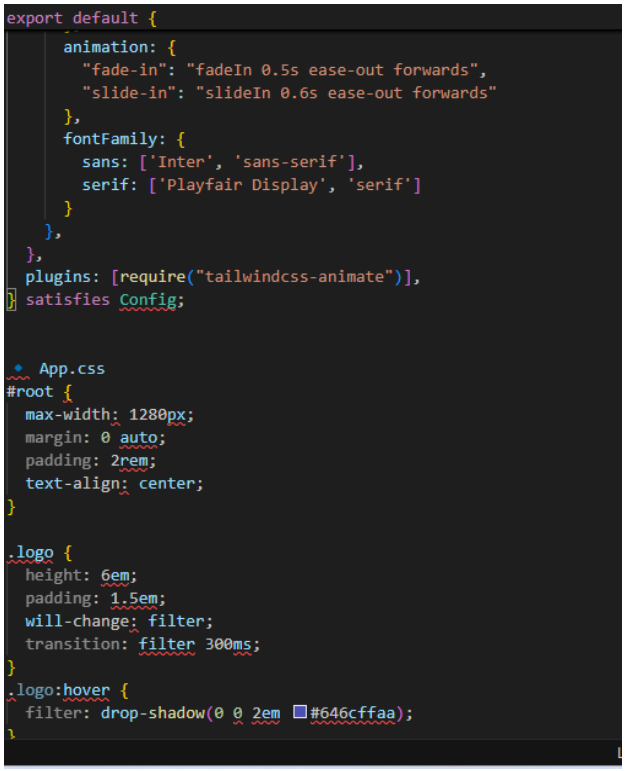

export default {
animation: {
"fade-in": "fadeIn 0.5s ease-out forwards",
"slide-in": "slideIn 0.6s ease-out forwards"
},
fontFamily: {
sans: ['Inter', 'sans-serif'],
serif: ['Playfair Display', 'serif']
}
},
},
plugins: [require("tailwindcss-animate")],
} satisfies Config;
App.css
#root {
max-width: 1280px;
margin: 0 auto;
padding: 2rem;
text-align: center;
}
.logo {
height: 6em;
padding: 1.5em;
will-change: filter;
transition: filter 300ms;
}
.logo:hover {
filter: drop-shadow(0 0 2em #646cffaa);
}

Fig 1.4

OUTPUT:

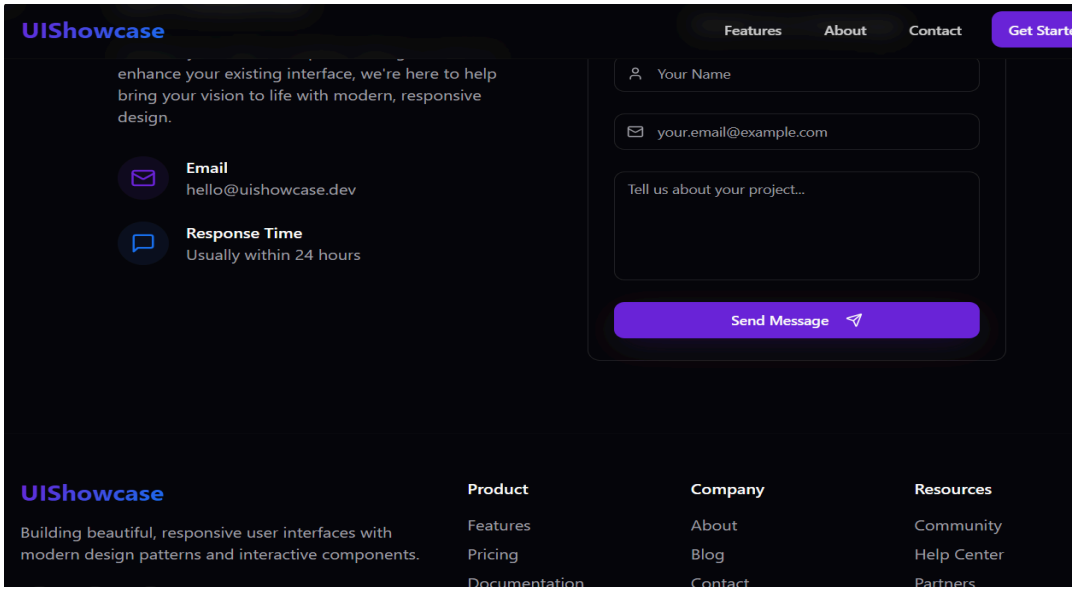


Fig 2.1

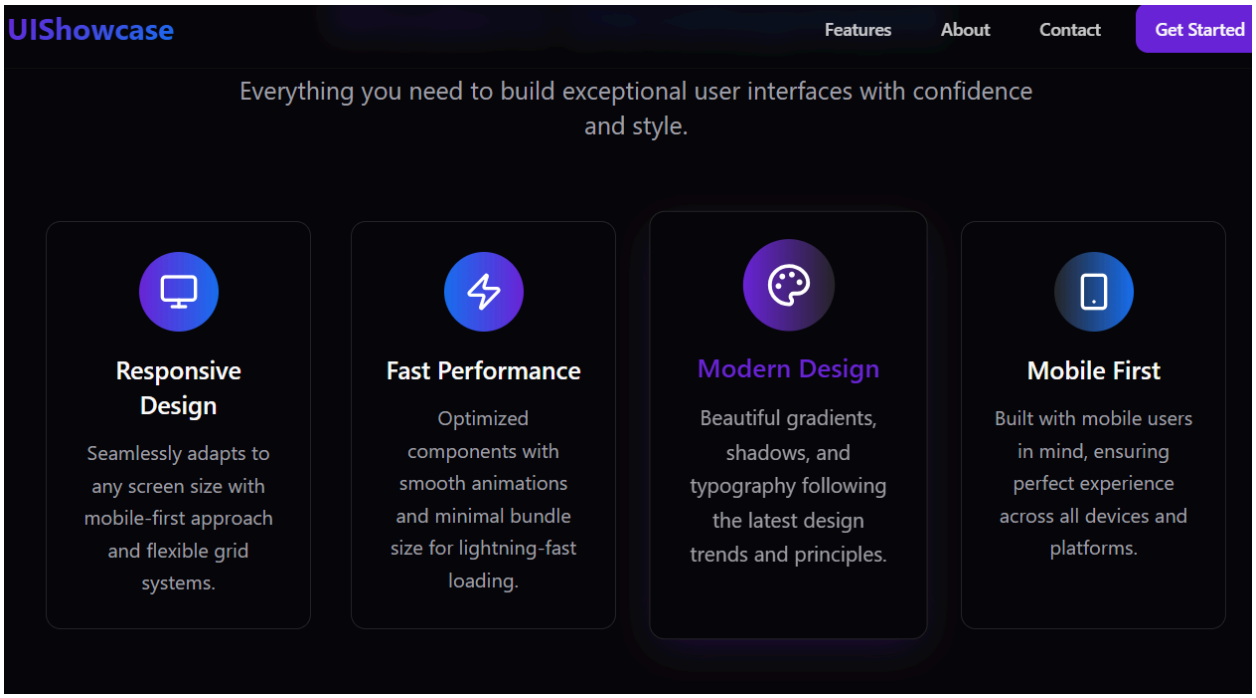


Fig 2.2

Conclusion:

Building responsive and interactive UIs with Tailwind CSS makes frontend development faster, cleaner, and more maintainable. Instead of writing long custom CSS, developers can directly apply utility-first classes that handle spacing, colors, typography, flex/grid layouts, and responsiveness through breakpoints.