Experiment 1

AIM : Building Responsive and Interactive UIs using Tailwind CSS

PREREQUISITES:

Demonstrate modern web development practices, mobile-first design approach, and performance optimization techniques through the creation of a Zentry clone website.

THEORY:

What is Tailwind CSS

Tailwind CSS is a utility-first CSS framework where you style directly in your HTML/JSX using classes like `bg-blue-500` or `text-xl`.

Responsive Design with Tailwind

Tailwind uses breakpoints like `sm`, `md`, `lg`, `xl` for mobile-first responsive design.
Example: `text-sm md:text-lg lg:text-xl` changes font size based on screen size.

Interactivity with Tailwind

It provides hover, focus, and active states for interactive elements.
Example: `hover:bg-green-500` changes button color when hovered.

Animations and Transitions

Tailwind has utility classes for smooth transitions.
Example: `transition duration-300 ease-in-out` makes hover effects smooth.

Customization and Theming

Developers can extend colors, spacing, and fonts in `tailwind.config.js`.
Example: Adding a custom color like `brand-blue` for consistent branding.

Why Tailwind is Effective

It speeds up development, ensures design consistency, and avoids writing long CSS files.
Example: A button can be styled fully with `px-4 py-2 bg-blue-600 text-white rounded-lg`.

Implementation in Zentry Clone Project:

1. Hero Section Styling

```
<div className="relative min-h-screen w-full overflow-x-hidden bg-gradient-to-b from-blue-50 to-black">
  <video
    className="absolute top-0 left-0 w-full h-full object-cover z-0"
    autoPlay
    muted
    loop
  >
    <source src="hero-video.mp4" type="video/mp4" />
  </video>
</div>
```

Tailwind Classes Used:

- `relative`: Position relative for layering
- `min-h-screen`: Full viewport height
- `w-full`: Full width
- `overflow-x-hidden`: Hide horizontal overflow
- `bg-gradient-to-b from-blue-50 to-black`: Gradient background
- `absolute top-0 left-0`: Absolute positioning
- `object-cover`: Video covers container maintaining aspect ratio

2. Responsive Navigation Bar

```
<nav className="fixed top-0 z-50 w-full bg-black/80 backdrop-blur-md border-b border-white/20">
  <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <div className="flex items-center justify-between h-16">
      <div className="flex-shrink-0">
        <h1 className="text-white text-xl font-bold">Zentry</h1>
      </div>
      <div className="hidden md:block">
        <div className="ml-10 flex items-center space-x-4">
          <a className="text-white hover:text-yellow-400 transition-colors duration-300">
            Home
          </a>
        </div>
      </div>
    </div>
  </div>
</nav>
```

Responsive Features:

- `hidden md:block`: Hide on mobile, show on medium screens+
- `sm:px-6 lg:px-8`: Different padding for different screen sizes

- `max-w-7xl mx-auto`: Centered container with max width

3. Features Grid (Bento Layout)

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6 p-6">
  <div className="group relative overflow-hidden rounded-2xl bg-gradient-to-br
from-purple-600 to-blue-600 p-6 hover:scale-105 transition-all duration-300">
    <div className="absolute inset-0 bg-black/40 group-hover:bg-black/20 transition-all
duration-300"></div>
    <h3 className="relative z-10 text-white text-xl font-semibold">Feature Title</h3>
    <p className="relative z-10 text-white/80 mt-2">Feature description</p>
  </div>
</div>
```

Interactive Features:

- `hover:scale-105`: Scale on hover
- `transition-all duration-300`: Smooth transitions
- `group` and `group-hover`: Group hover effects
- `grid-cols-1 md:grid-cols-2 lg:grid-cols-3`: Responsive grid

4. Mobile-First Approach

The project follows mobile-first design:

- Base styles target mobile devices
- Larger screen styles added with breakpoint prefixes
- Touch-friendly components with adequate spacing
- Optimized typography scaling

Performance Optimization Techniques:

1. Utility Class Optimization

- Only required utility classes are included in final CSS
- Automatic purging of unused styles
- Reduced CSS bundle size

2. Custom Configuration

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{js,jsx,ts,tsx}'],
  theme: {
    extend: {
      fontFamily: {
        'custom': ['Your-Font', 'sans-serif']
      },
      animation: {
        'float': 'float 6s ease-in-out infinite'
```

```
    }
   }
  }
}
```

3. Modern CSS Features

- CSS Grid and Flexbox utilities
- Custom properties (CSS variables)
- Backdrop filters and modern visual effects

PRACTICAL IMPLEMENTATION

Project Structure:

```
zentry-clone/
├── src/
│   ├── components/
│   │   ├── Hero.jsx
│   │   ├── Navbar.jsx
│   │   ├── Features.jsx
│   │   └── Story.jsx
│   ├── styles/
│   │   └── globals.css
│   └── App.jsx
├── tailwind.config.js
└── package.json
```

Key Features Implemented:

1. **Responsive Hero Section**: Full-screen video background with overlay text
2. **Animated Navigation**: Smooth hover effects and mobile hamburger menu
3. **Interactive Bento Grid**: Hover effects with tilt animations
4. **Smooth Scroll Animations**: GSAP integration with Tailwind classes
5. **Contact Form**: Modern form design with validation states

30% Extra:

GSAP (GreenSock Animation Platform) is a powerful JavaScript library used to create high-performance animations for the web. It allows developers to animate HTML, CSS, SVG, and Canvas elements with fine-grained control. GSAP is highly optimized, ensuring smooth animations across devices and browsers. It provides advanced features like timelines, easing, and staggered animations, making it ideal for both simple transitions and complex motion graphics.

Framer Motion is a React-based animation library designed for creating interactive and declarative animations. It uses an intuitive API that integrates seamlessly with React components, enabling developers to build animations without managing complex states manually. Framer Motion supports gestures, layout animations, shared element transitions, and drag features, making it popular for modern UI/UX design in web applications.

ScrollTrigger is a GSAP plugin that links animations to the user's scroll behavior. It enables effects such as parallax scrolling, reveal-on-scroll, and pinning elements in place. By combining scroll events with GSAP timelines, developers can synchronize animations with the page's scroll position, creating immersive storytelling experiences.

Together, **GSAP, Framer Motion, and ScrollTrigger** empower developers to craft visually engaging, interactive, and smooth web animations that significantly enhance user engagement and overall digital experiences.

Source Code :

```
import React, { useRef, useState, useEffect } from 'react'
import Button from './Button'
import { TiLocationArrow } from 'react-icons/ti'
import { useGSAP } from '@gsap/react'
import gsap from 'gsap'
import {ScrollTrigger} from 'gsap/all'

gsap.registerPlugin(ScrollTrigger);

const Hero = () => {
    const [currentIndex, setCurrentIndex] = useState(1);
    const [hasClicked, setHasClicked] = useState(false);
    const [isLoading, setIsLoading] = useState(true);
    const [LoadedVideos, setLoadedVideos] = useState(0);

    const totalVideos = 4;
    const nextVideoRef = useRef(null);

    const handleVideoLoad = () => {
        setLoadedVideos((prev) => prev + 1);
    }

    const handleMiniVideoClick = () => {
        setHasClicked(true);
        setCurrentIndex(upcomingVideoIndex);
    }

    useEffect(() => {
        if (LoadedVideos === totalVideos - 1) {
            setIsLoading(false);
```

```
  return (
    <div className="relative h-dvh w-screen overflow-x-hidden">
        {isLoading && (
            <div className='flex-center absolute z-[100] h-dvh w-screen overflow-hidden bg-violet-50'>
                <div className='three-body'>
                    <div className="three-body__dot"></div>
                    <div className="three-body__dot"></div>
                    <div className="three-body__dot"></div>
                </div>
            </div>
        )}

        <div
            id="video-frame"
            className="relative z-10 h-dvh w-screen overflow-hidden rounded-lg bg-blue-75"
        >
            <div>
                <div className='mask-clip-path absolute-center absolute z-50 size-64 cursor-pointer overflow-hidden rounded-lg'>
                    <div onClick={handleMiniVideoClick} className='origin-center scale-50 opacity-0 transition-all duration-500 ease-in hover:scale-100 hover:opacity-100'>
                        <video ref={nextVideoRef} src={getVideoSrc(upcomingVideoIndex)}
                            loop
                            muted
                            id="current-video"
                            className='size-64 origin-center scale-150 object-cover object-center'
                            onLoadedData={handleVideoLoad}
                        />
                    </div>
                </div>
                <video
```

```
                    loop
                    muted
                    className='absolute left-0 top-0 size-full object-cover object-center'
                    onLoadedData={handleVideoLoad}
                />
            </div>

            <h1 className='special-font hero-heading absolute bottom-5 right-5 z-40 text-blue-75'>Re<b>a</b>lity</h1>

            <div className='absolute left-0 top-0 z-40 size-full'>
                <div className='mt-24 px-5 sm:px-10'>
                    <h1 className='special-font hero-heading text-blue-100'>
                        redefi<b>n</b>e
                    </h1>
                    <p className='mb-5 font-robert-regular max-w-64 text-blue-100'>
                        Enter the Metagame Layer <br /> Unleash the Play Economy
                    </p>
                    <Button id='watch-trailer' title='Watch Trailer' leftIcon={<TiLocationArrow />} containerClass='!bg-yellow-300 flex-center gap-1' />
                </div>
            </div>
        </div>

        {/* Black text positioned outside the video frame - this will show in clipped areas */}
        <h1 className='special-font hero-heading absolute bottom-5 right-5 !text-black'>Re<b>a</b>lity</h1>
    </div>
  )
}

export default Hero
```

```
@import "tailwindcss";

@layer theme {
  :root {
    --color-blue-50: #DFDFF0;
    --color-blue-75: #dfdff2;
    --color-blue-100: #F0F2FA;
    --color-blue-200: #010101;
    --color-blue-300: #4FB7DD;
    --color-violet-300: #5724ff;
    --color-yellow-100: #8e983f;
    --color-yellow-300: #edff66;
  }
}

@layer base {
    @font-face {
        font-family: circular-web;
        src: url('/fonts/circularweb-book.woff2') format('woff2');
    }
    @font-face {
        font-family: 'general';
        src: url('/fonts/general.woff2') format('woff2');
    }
    @font-face {
```

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

```vite.config.js
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import tailwindcss from '@tailwindcss/vite'

// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss()],
})
```

Output:

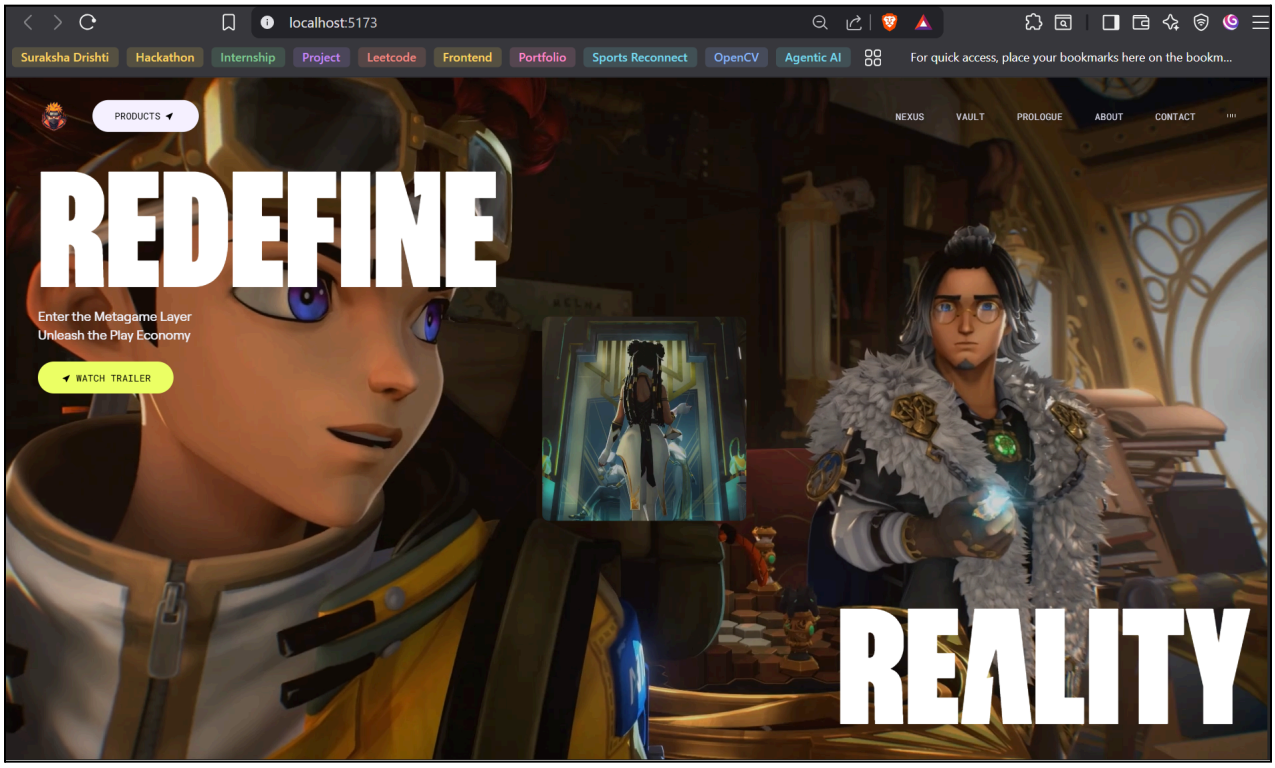

Fig.1


WELCOME TO ZENTRY
DISCOVER THE WORLD'S LARGEST SHARED ADVENTURE
The Game of Games begins—your life, now an epic MMORPG
Zentry unites every player from countless games

Fig.2


THE MULTIVERSAL IP WORLD
THE STORY OF A HIDDEN REALM

Fig.3

Fig. 4

CONCLUSION

The experiment successfully demonstrates the power and efficiency of Tailwind CSS in building modern, responsive web interfaces. Key learnings include:

Advantages Observed:

1. **Rapid Development**: Utility classes enable faster development without writing custom CSS
2. **Consistent Design System**: Pre-defined spacing, colors, and typography ensure visual consistency
3. **Responsive Design**: Built-in responsive utilities make it easy to create adaptive layouts
4. **Performance Benefits**: Smaller CSS bundle size due to utility-first approach
5. **Maintainability**: Styles are co-located with HTML, making components self-contained

Challenges Encountered:

1. **Learning Curve**: Initial time investment to learn utility class names
2. **HTML Verbosity**: Multiple classes can make HTML appear cluttered
3. **Design Constraints**: Working within the framework's design system

Best Practices Learned:

1. **Mobile-First Development**: Start with mobile styles, then enhance for larger screens
2. **Component Extraction**: Extract repeated utility patterns into reusable components
3. **Custom Utilities**: Extend Tailwind with project-specific utilities when needed
4. **Performance Optimization**: Use PurgeCSS to remove unused styles in production

The Zentry clone project successfully showcases how Tailwind CSS can be used to create visually appealing, performant, and responsive web applications while maintaining clean, maintainable code structure.

GSAP, Framer Motion, and ScrollTrigger are powerful tools that revolutionize the way animations are implemented on the web. While GSAP offers robust, high-performance animations for all types of elements, Framer Motion simplifies animation within React

applications by providing a declarative and component-driven approach. ScrollTrigger, as a plugin, extends GSAP's capabilities by connecting animations to user scrolling, resulting in highly interactive and immersive experiences. Together, these technologies not only enhance visual appeal but also improve user engagement by making interfaces dynamic, responsive, and intuitive. In essence, they form the backbone of modern web animation, combining performance, creativity, and interactivity.