Experiment No - 01

Roll No.	52
Name	Kshitij Nangare
Class	D15B
Subject	Full Stack Development
Lab Outcome	Build responsive and interactive UIs using Tailwind CSS
Date of Performance/ Submission	28/07/2025 11/08/2025
Signature & Grades	

Aim : Build responsive and interactive UIs using Tailwind CSS.

Theory :

Tailwind CSS is a **utility-first CSS framework** that provides pre-defined classes to style elements directly within HTML, enabling developers to build modern, responsive, and customizable user interfaces without writing custom CSS from scratch. Unlike traditional CSS frameworks such as Bootstrap, which offer pre-built components, Tailwind gives granular control over styling, allowing for **highly customizable designs**.

1. Responsive Design

Responsive design ensures that a website or application adjusts seamlessly to various screen sizes and devices (mobiles, tablets, desktops). Tailwind CSS simplifies responsive design through **breakpoint prefixes** such as:

- `sm:` for small screens
- `md:` for medium screens
- `lg:` for large screens
- `xl:` for extra-large screens

This **mobile-first approach** means styles are applied first for smaller devices and then scaled up for larger screens.

Example:

```
<div class="text-sm md:text-lg lg:text-xl">Responsive Text</div>
```

In the above code, the font size changes according to the screen size.

2. Interactivity

Interactive elements make the UI more engaging for the user. Tailwind CSS supports hover, focus, active, and dark mode states directly through class names.

- **Hover effects:** `hover:bg-blue-500` changes the background color on hover.
- **Focus states:** `focus:ring-2` adds a focus ring on input fields.

- **Transitions and animations:** Tailwind provides classes like `transition`, `duration-300`, and `ease-in-out` to create smooth effects.

Example:

```
<button class="bg-green-500 hover:bg-green-700 text-white px-4 py-2
rounded transition duration-300">
  Click Me
</button>
```

3. Advantages of Tailwind CSS in UI Development

- **Faster Development:** Ready-to-use utility classes avoid writing repetitive CSS code.
- **Highly Customizable:** Developers can configure themes, colors, and spacing in the `tailwind.config.js` file.
- **Consistency:** Ensures uniform spacing, colors, and font sizes across the project.
- **Performance:** Tailwind's build process removes unused CSS (tree-shaking), reducing file size.
- **Component Reusability:** Classes can be composed into reusable components without global CSS conflicts.

4. Real-World Use Case

In modern full-stack development, Tailwind CSS integrates seamlessly with frontend frameworks like **React, Vue, and Next.js**. It allows developers to create **responsive dashboards, forms, landing pages, and web apps** quickly.

For example, a **login page** built with Tailwind CSS can have:

- Mobile-first responsive layout
- Smooth hover and focus states
- Minimalistic design with consistent spacing
- Light and dark theme toggle

5. Tailwind + Vite

By combining **Tailwind CSS** with **Vite**, developers can achieve a **fast, efficient, and scalable UI development workflow**. Tailwind handles styling in a utility-first manner, while Vite ensures fast builds and smooth development experience.

Why Tailwind + Vite?

- **Speed:** Vite starts instantly, and Tailwind classes require no CSS file switching.
- **Scalability:** Easy to maintain as the project grows.
- **Consistency:** Utility classes enforce consistent design patterns.
- **Mobile-first:** Tailwind's responsive utilities make adapting to different screen sizes straightforward.

---

6. Setup Instructions

1. Initialize Vite Project

npm create vite@latest my-project

cd my-project

npm install

2. Install Tailwind CSS

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

3. Configure tailwind.config.js
Edit the content section to include HTML/JS/TS files:

```
export default {
  content: [
    "./index.html",
```

```
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

4. Add Tailwind Directives to CSS
In src/index.css (or src/style.css):

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

5. Run the Development Server

```
npm run dev
```

Source Code :

SessionCard.jsx

```
const SessionCard = ({
    session,
    isEditable = false,
    onEdit = () => {},
    onPublish = () => {},
    onUnpublish = () => {},
    onDelete = () => {},
    onLike = () => {},
    isProcessing = false,
    hasLiked = false
}) => {
    return (
        <div className="bg-white rounded-lg shadow-md overflow-hidden border
border-gray-200 hover:shadow-xl transition-all duration-300 flex flex-col h-full">
            {/* Status Bar (only visible if editable) */}
            {isEditable && (
                <div className={`p-3 flex justify-between items-center text-sm font-medium ${
                    session.status === 'published' ? 'bg-green-50 text-green-700' : 'bg-yellow-50
text-yellow-700'
                }`}>
                    <span className={`px-2 py-0.5 rounded-full ${
                        session.status === 'published'
                            ? 'bg-green-100 text-green-800'
                            : 'bg-yellow-100 text-yellow-800'
                    }`}>
                        {session.status === 'published' ? 'Published' : 'Draft'}
                    </span>

                    <div className="text-gray-500">
                        <Clock className="inline-block h-3.5 w-3.5 mr-1 align-middle" />
                        <span className="align-middle">
                            {new Date(session.updatedAt ||
session.created_at).toLocaleDateString('en-IN', {
                                day: 'numeric',
                                month: 'short',
                                year: 'numeric'
                            })}
                        </span>
                    </div>
                </div>
            )}

            {/* Session Image or Placeholder */}
            <div className="h-48 bg-gray-100 overflow-hidden relative group">
                {session.imageUrl ? (
                    <img
```

```
                    src={session.imageUrl}
                    alt={session.title}
                    className="w-full h-full object-cover transition-transform duration-300
group-hover:scale-105"
                    onError={(e) => {
                      e.target.onerror = null;
                      e.target.src =
"https://flimp.net/wp-content/uploads/2024/06/Why-Employee-Wellness-Is-Good-for-Business-
Benefits-and-ROI.png";
                    }}
                  />
                ) : (
                  <div className="w-full h-full flex items-center justify-center text-gray-400
bg-gray-200">
                    <PlayCircle className="h-16 w-16 text-gray-300" />
                  </div>
                )}

                {/* Overlay for quick view on hover for non-editable cards */}
                {!isEditable && (
                  <Link
                    to={`/sessions/${session._id}`}
                    className="absolute inset-0 bg-black bg-opacity-40 flex items-center
justify-center opacity-0 group-hover:opacity-100 transition-opacity duration-300"
                    aria-label={`View ${session.title}`}
                  >
                    <Eye className="h-10 w-10 text-white animate-pulse" />
                  </Link>
                )}
              </div>

              {/* Session Content */}
              <div className="p-4 flex-grow flex flex-col">
                <div className="flex justify-between items-start mb-2">
                  <h3 className="font-semibold text-lg text-gray-800 leading-tight pr-2
line-clamp-2">
                    {session.title}
                  </h3>
                  {/* Like button and count */}
                  <button
                    onClick={() => onLike(session._id)}
                    disabled={isProcessing}
                    className="flex items-center text-sm text-gray-500 hover:text-red-500
transition-colors duration-200 disabled:opacity-50 disabled:cursor-not-allowed shrink-0 ml-2"
                    aria-label={hasLiked ? "Unlike session" : "Like session"}
                  >
                    <Heart
                      className={`h-5 w-5 mr-1 ${
                        hasLiked ? 'text-red-500 fill-red-500' : 'text-gray-400'
                      } transition-all duration-200`}
```

```
                    />
                    <span className="text-base font-medium">{session.likes}</span>
                </button>
            </div>

            <p className="text-sm text-gray-600 mb-3 line-clamp-2 min-h-[2.5rem]">
                {session.description || 'No description available.'}
            </p>

            <div className="flex items-center text-sm text-gray-500 mb-3">
                <UserIcon className="h-4 w-4 mr-1.5 text-gray-400" />
                <span className="font-medium">{session.user_id?.name ||
'Anonymous'}</span>
            </div>

            {/* Tags section with line-clamp */}
            {session.tags && session.tags.length > 0 && (
                <div className="flex flex-wrap gap-2 mb-4 overflow-hidden" style={{ maxHeight:
'3.5rem' }}> {/* Adjusted height for 2 lines of tags */}
                    {session.tags.map(tag => (
                        <span key={tag} className="bg-indigo-50 text-indigo-700 text-xs
font-medium px-2.5 py-1 rounded-full whitespace-nowrap">
                            {tag}
                        </span>
                    ))}
                </div>
            )}

        </div>
    );
};
```

DashboardPage.jsx

```
const DashboardPage = () => {

    return (
        <div className="min-h-screen bg-gray-50">
            {/* Header */}
            <Navbar />

            {/* Main Content */}
            <main className="max-w-7xl mx-auto px-6 py-8">

                {/* Tab Navigation */}
                <div className="border-b border-gray-200 mb-6">
                    <nav className="-mb-px flex space-x-8">
                        <button
```

```
                        onClick={() => setActiveTab('all')}
                        className={`whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm
${
                            activeTab === 'all'
                                ? 'border-indigo-500 text-indigo-600'
                                : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300'
                        }`}
                    >
                        All Sessions
                    </button>
                    <button
                        onClick={() => setActiveTab('my')}
                        className={`whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm
${
                            activeTab === 'my'
                                ? 'border-indigo-500 text-indigo-600'
                                : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300'
                        }`}
                    >
                        My Sessions
                    </button>
                </nav>
            </div>

            {/* Search Bar (Only for All Sessions tab) */}
            {activeTab === 'all' && (
                <div className="relative mb-6">
                    <div className="absolute inset-y-0 left-0 pl-3 flex items-center
pointer-events-none">
                        <Search className="h-5 w-5 text-gray-400" aria-hidden="true" />
                    </div>
                    <input
                        type="text"
                        name="search"
                        id="search"
                        className="block w-full pl-10 pr-3 py-2 border border-gray-300
rounded-md leading-5 bg-white placeholder-gray-500 focus:outline-none
focus:ring-indigo-500 focus:border-indigo-500 sm:text-sm"
                        placeholder="Search sessions by title or tag..."
                        value={searchTerm}
                        onChange={(e) => setSearchTerm(e.target.value)}
                    />
                </div>
            )}

            {/* Sessions Display */}
            {activeTab === 'all' ? (
                <section>
```

```
                    <h2 className="text-xl font-semibold text-gray-900 mb-4">All Wellness
Sessions</h2>
                    {isLoading ? (
                        <LoadingSpinner />
                    ) : allSessions.length === 0 ? (
                        <div className="bg-white p-8 rounded-lg shadow-sm text-center">
                            <p className="text-gray-500">No sessions found matching your
search.</p>
                            <p className="text-gray-400 text-sm mt-2">Try a different keyword or
check back later!</p>
                        </div>
                    ) : (
                        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
                            {allSessions.map(session => ( // Use `allSessions` here, as filtering is
done on backend
                                <SessionCard
                                    key={session._id}
                                    session={session}
                                    isEditable={false} // Public sessions are not editable from here
                                    onLike={handleLikeUnlike}
                                    hasLiked={user && session.likedBy &&
session.likedBy.includes(user.id)} // Check if current user liked it
                                    isProcessing={processingCardId === session._id}
                                />
                            ))}
                        </div>
                    )}
                </section>
            ) : (
                <section>
                    <MySessionsPage />
                </section>
            )}
        </main>
    </div>
  );
};

export default DashboardPage;
```

Output :

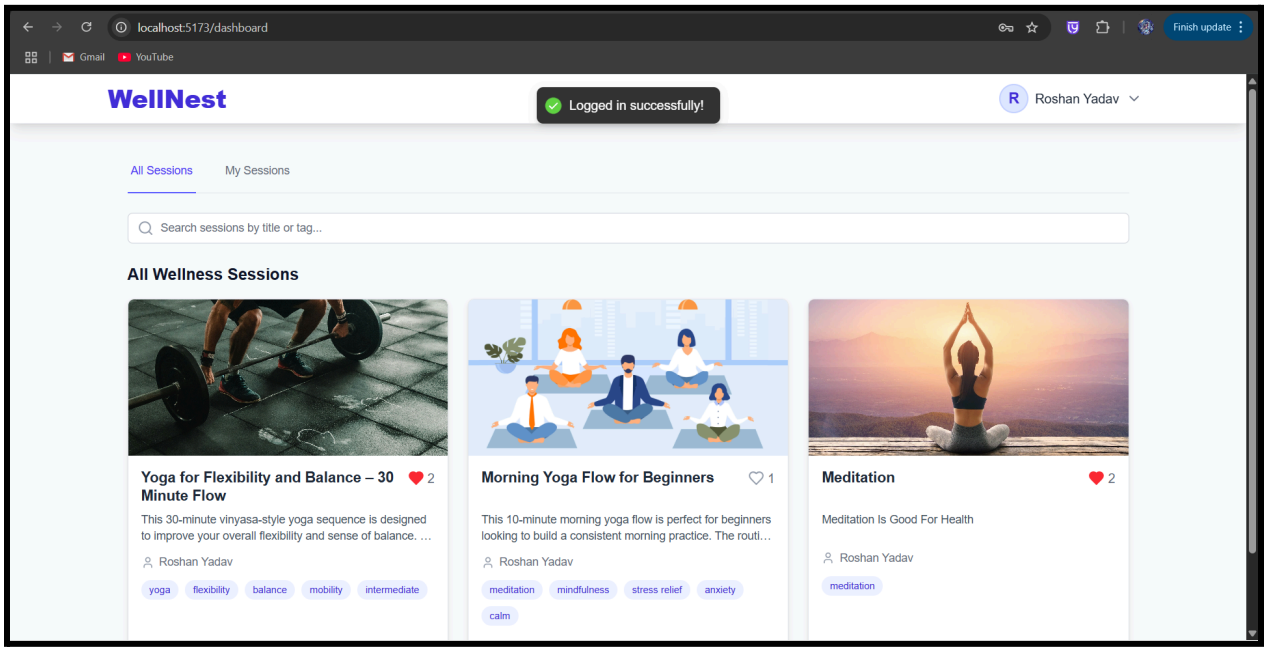

Figure 1

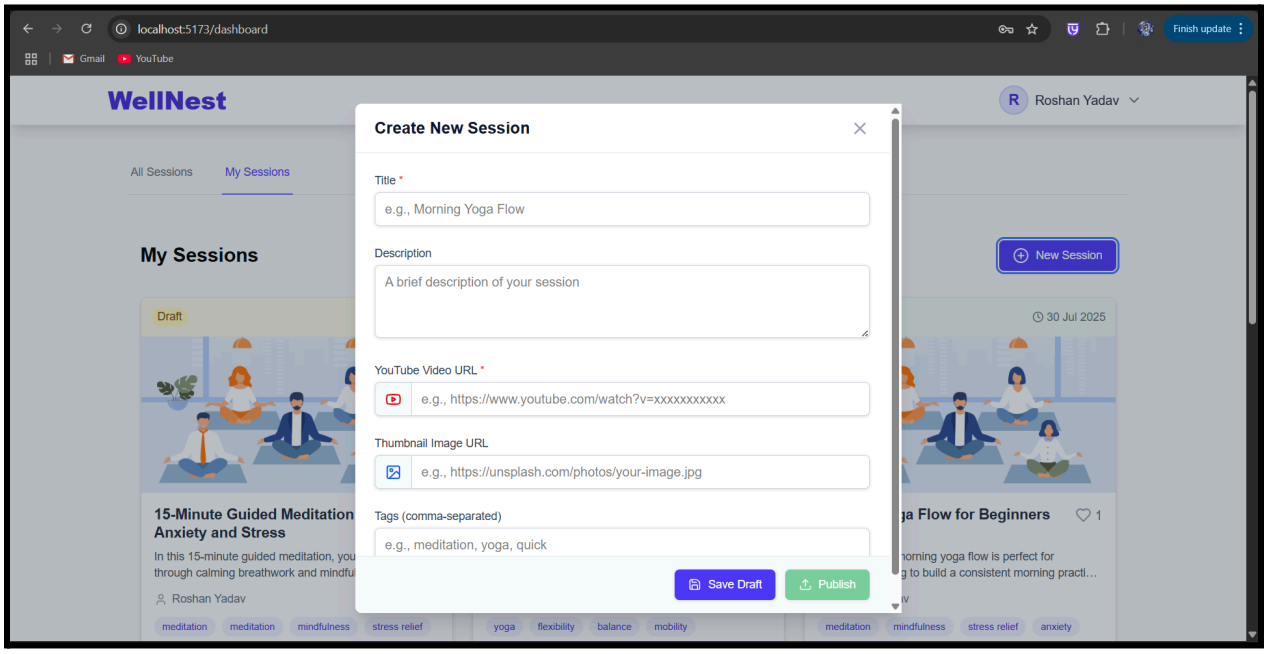

Figure 2

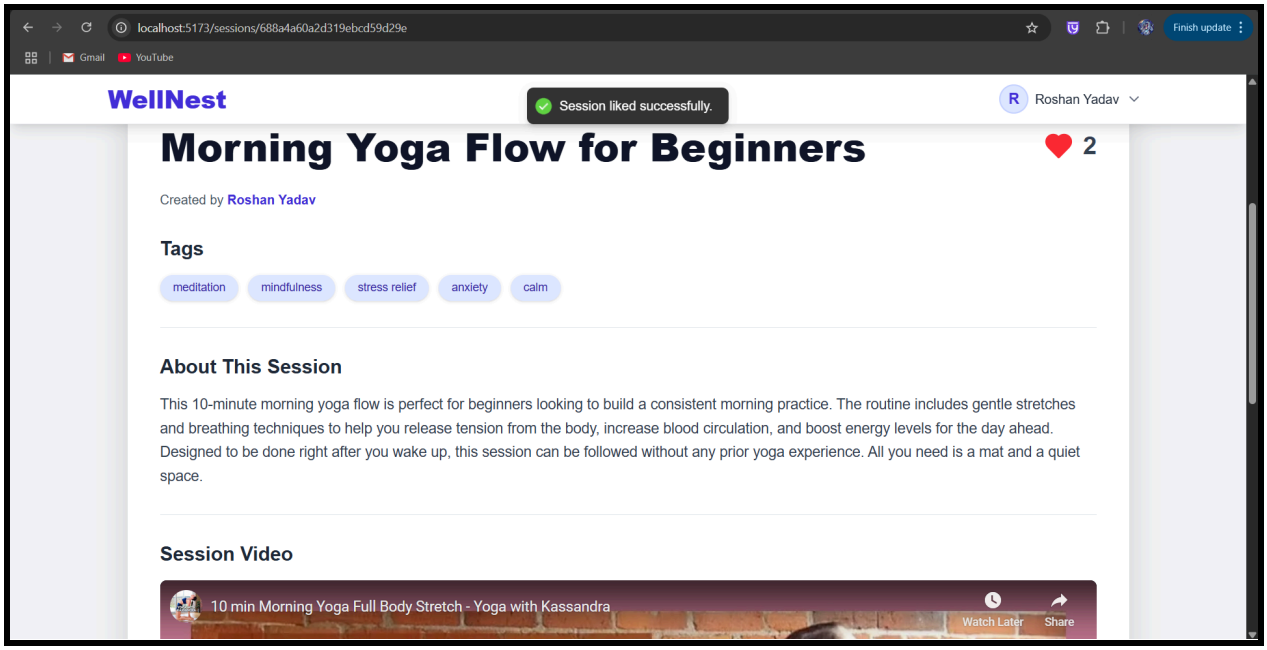

Figure 3

Conclusion

Using Tailwind CSS in responsive UI design enhances **developer productivity**, **code maintainability**, and **user experience**. Its utility-first approach allows for building clean, scalable, and interactive designs faster than writing traditional CSS. By combining responsive breakpoints with interactive states, developers can create **modern, adaptive, and user-friendly web interfaces** for any device.

Additional Implementation (30% Extra): JWT Authentication

As part of the 30% extra work requirement, I implemented **JWT (JSON Web Token) Authentication** in my responsive website project built with Tailwind CSS. JWT is a widely used method for securely transmitting information between a client and server.

Why JWT?

- Provides a **stateless authentication mechanism**.
- Ensures secure communication by signing tokens with a **secret key**.
- Eliminates the need for storing session data on the server.
- Token can carry user details (payload) that can be verified by the backend.

🛠️ **Implementation Details**

1. **Token Generation**
    - When a user successfully logs in, a JWT token is generated.
    - The token contains user information and an expiry time of **1 hour**.
    - The token is signed using a secret key stored in `.env` file.

2. **Token Verification**
    - For every protected route, the token is verified.
    - If the token is valid and not expired, the user gains access.
    - If invalid/expired, access is denied.

Code :

```
import jwt from 'jsonwebtoken';
import dotenv from 'dotenv';

dotenv.config();

// Generate JWT Token (expires in 1h)
const generateToken = (user) => {
  return jwt.sign(
    { user },
    process.env.JWT_SECRET,
    { expiresIn: '1h' }
  );
};
```

```
// Verify JWT Token
const verifyToken = (token) => {
  try {
    return jwt.verify(token, process.env.JWT_SECRET);
  } catch (err) {
    return null;
  }
};

export { generateToken, verifyToken };
```

Working Flow

1. **User Login** → Server generates JWT token and sends it to client.
2. **Client Requests Protected Data** → Client sends token in headers.
3. **Server Verifies Token** → If valid, access is granted; else denied.

Outcome

By adding JWT authentication, the website now supports **secure login and protected routes**, making it closer to a production-ready full-stack application instead of just a static responsive design.