

**MIT Art, Design and Technology University**

**MIT School of Computing, Pune**

**Department of Information Technology**

|  |
| --- |
| **Lab Manual** |

# **Practical - Web Programming**

# **Class - S.Y. (SEM-II), DA**

# **Batch - SMAD**

# **Name Abhishek Dhautre**

# **Mr.**

**A.Y. 2024 – 2025 (SEM-II)**

|  |  |
| --- | --- |
| **Assignment No 1: Project Assignment based on Unit-I and Unit-II** | |
| **Project-I** | Project Title: ReactJS E-Commerce Website with Page RoutingProject Statement: Objective: Develop a fully functional E-commerce website using ReactJS that includes page routing functionality for smooth navigation between different sections like Home, Product Listing, Product Details, Cart, and Checkout. Project Features:  1. Navigation Bar with Routing:    * Create a Navigation Bar at the top of the website with links to:      + Home: Displays featured products and general information about the store.      + Products: Lists all available products for purchase.      + Cart: Displays the products added to the shopping cart.      + Checkout: Allows the user to review and complete the purchase.      + Profile: Displays user account information and order history.    * Implement React Router for routing between these pages. 2. Home Page:    * The Home page should showcase highlighted products or categories.    * Implement a product carousel or featured section.    * Include a search bar to allow users to search for products directly. 3. Product Listing Page:    * Create a Product Listing page that fetches products dynamically from an API or static JSON file.    * Each product in the list should show:      + Product image.      + Product name.      + Price.      + A button to add the product to the cart.    * Implement React Router links on product items to navigate to the Product Details page. 4. Product Details Page:    * When a user clicks on a product, navigate them to the Product Details page.    * The Product Details page should display:      + Full-size product image.      + Detailed product description.      + Price.      + Add to Cart button.    * Allow users to select product quantity before adding it to the cart. 5. Cart Page:    * Implement a Cart page that displays products added to the cart.    * Each cart item should include:      + Product name.      + Product quantity.      + Price per item and total cost for the item.      + A button to remove items from the cart.    * Display a total price for all items in the cart.    * Include buttons for Continue Shopping and Proceed to Checkout. 6. Checkout Page:    * Create a Checkout page where users can review their cart, enter shipping information, and confirm the order.    * Display an order summary with total cost, shipping details, and payment options.    * Provide a Place Order button to finalize the transaction. 7. User Profile Page:    * Create a Profile page where users can view their account information and order history.    * Include a form to update the profile details (name, email, shipping address). 8. State Management with Context API/Redux:    * Use React Context API or Redux to manage the global state of the cart, allowing users to add, remove, and update quantities of products across different pages.    * Ensure that the cart data is preserved during navigation between pages. 9. Responsive Design:    * Ensure the website is responsive and works well on both desktop and mobile devices using CSS Flexbox or CSS Grid.  Technical Requirements:ReactJS: For building the user interface.React Router: For implementing page routing functionality.State Management: Use React Context API or Redux to manage cart state and other global data.API Integration: Fetch products from a static JSON file or a mock API (e.g., JSONPlaceholder or a custom API).Responsive Styling: Use CSS/SCSS to ensure the website is mobile-friendly.Optional: Add localStorage or sessionStorage to persist cart data between page reloads.Expected Outcome: By completing this project, you will:   * Learn how to implement page routing in a ReactJS application using React Router. * Build a dynamic E-commerce website with features like product listings, cart management, and checkout functionality. * Gain experience with state management and ensuring the cart data persists across different pages. * Develop a responsive, user-friendly interface that functions well on both mobile and desktop screens. |
| **Assignment No 2: Project Assignment based on Unit-III, IV and V** | |
| **Project-II** | Project Title: React Native E-Commerce ApplicationProject Statement: Objective: Build a fully functional E-commerce mobile application using React Native that integrates navigation, core components, user interaction, dynamic list rendering, and state management. Project Features:  1. User Authentication:    * Implement a simple login screen where users can enter their credentials (username and password).    * After successful login, navigate to the Home Screen. 2. Navigation:    * Set up react-navigation to allow users to navigate between the following screens using StackNavigator:      + Home Screen: Displays a list of products.      + Product Details Screen: Shows details of the selected product.      + Cart Screen: Displays products added to the shopping cart.      + Profile Screen: Displays user profile details.      + Settings Screen: Allows the user to modify their account settings. 3. Core Components & UI:    * Use core React Native components such as View, Text, Image, StyleSheet, and Button to build a user-friendly interface.    * Apply Flexbox for layout, ensuring responsiveness across different screen sizes. 4. Product List and Cart:    * On the Home Screen, render the list of products using FlatList. Each item should display a product image, name, and price.    * Implement a button for each product to add it to the Cart. 5. Handling User Interactions:    * Implement touch events to handle interactions such as adding products to the cart, navigating to the product details, and updating the cart.    * Use onPress to trigger actions like adding a product to the cart or navigating between screens. 6. Cart Management:    * Use React Context API for global state management to handle the cart’s data (adding/removing products).    * Show the cart’s content and total price on the Cart Screen. 7. Product Details:    * When a user clicks on a product in the Home Screen, navigate to the Product Details Screen and display detailed information about the product, such as the description, image, and price. 8. Responsive Design:    * Ensure that the app is responsive, and elements adjust properly to different screen sizes using Flexbox. 9. Rendering Lists with SectionList:    * On the Home Screen, group products by categories (e.g., electronics, clothing) and use SectionList to render them in separate sections with headers.  Technical Requirements:React Native with Expo CLI for development and testing.React Navigation for screen transitions.State Management using React Context API or useState, useReducer.Dynamic data rendering using FlatList and SectionList.Styling using StyleSheet and Flexbox for layout.Expected Outcome: By completing this project, students will gain hands-on experience with:   * Setting up and using React Native and React Navigation. * Building a fully interactive mobile application with dynamic data rendering and state management. * Applying Flexbox for responsive layouts and managing user interactions. * Creating a simple e-commerce app with multi-screen navigation, product lists, and state-driven cart functionality. |

# 

## **Experiment No.1**

## 

## **Problem Statement:**

Design and develop a DIY Repair Web Application using ReactJS that assists users in performing household and gadget repairs with step-by-step guides, parts finder, and AR-assisted walkthroughs.

## **Objective:**

To implement a responsive web application using ReactJS and React Router that allows users to navigate between different pages including Home, Guides, Cart, About Us, Contact, Registration, and Login.

## **Theory:**

ReactJS is a JavaScript library used to build interactive UIs. It works with a component-based architecture and uses a virtual DOM to optimize rendering. With React Router, navigation between pages becomes seamless without page reloads. State management can be achieved using Context API or Redux for shared data like cart items.

## **Code:**

A. Home page:

code:

import React from 'react';

export default function Home() {

return (

<div>

<h1>Welcome to FixIt</h1>

<p>Your DIY repair assistant for home gadgets and appliances.</p>

</div>

);

}

## 

## **Output:**

1. Index/Home page output:

Displays a welcome message and brief description of the app.

## **Code:**

B. menu/product page:

code:

import React from 'react';

const guides = [

{ id: 1, title: "Fix a Leaking Tap", level: "Beginner" },

{ id: 2, title: "Replace Phone Screen", level: "Advanced" },

];

export default function Guides() {

return (

<div>

<h2>Repair Guides</h2>

<ul>

{guides.map(guide => (

<li key={guide.id}>{guide.title} - {guide.level}</li>

))}

</ul>

</div>

);

}

## **Output:**

List of repair tutorials with difficulty levels.

## **Code:**

C. cart page:

code:

import React, { useContext } from 'react';

import { CartContext } from './CartContext';

export default function Cart() {

const { cartItems } = useContext(CartContext);

return (

<div>

<h2>Parts Cart</h2>

{cartItems.map((item, index) => (

<p key={index}>{item.name} - ₹{item.price}</p>

))}

</div>

);

}

## **Output:**

Displays parts added to the cart with pricing.

## **Code:**

D. about us page:

code:

export default function About() {

return (

<div>

<h2>About FixIt</h2>

<p>We provide step-by-step DIY repair guides with AR support to empower home users.</p>

</div>

);

}

## **Output:**

Overview of FixIt’s mission and purpose.

## **Code:**

E. contact us page:

code:

export default function Contact() {

return (

<div>

<h2>Contact Us</h2>

<p>Email: support@fixitapp.com</p>

</div>

);

}

## **Output:**

Shows contact email for support.

## **Code:**

F. registration page:

code:

export default function Register() {

return (

<form>

<h2>Register</h2>

<input type="text" placeholder="Name" /><br />

<input type="email" placeholder="Email" /><br />

<input type="password" placeholder="Password" /><br />

<button>Sign Up</button>

</form>

);

}

## **Output:**

A basic form to register a new user.

## **Code:**

G. login page:

code:

export default function Login() {

return (

<form>

<h2>Login</h2>

<input type="email" placeholder="Email" /><br />

<input type="password" placeholder="Password" /><br />

<button>Login</button>

</form>

);

}

## **Output:**

Login form for returning users.

## **Conclusion:**

The DIY Repair Assistant web app was successfully implemented using ReactJS. It includes routing between multiple pages like Home, Guides, Cart, and more. This project showcases the use of dynamic components, user input forms, and shared state, providing users with a user-friendly experience for accessing repair help.

## **Experiment No.2**

## **Problem Statement:**

Design and implement a DIY Repair Assistant mobile app using React Native that provides users with repair guides, AR support, and parts management using a multi-screen navigation system.

## **Objective:**

To build a mobile application with React Native and React Navigation that supports login, multi-screen navigation, cart management, and responsive UI for users seeking DIY repair support.

## **Theory:**

React Native allows for the creation of cross-platform mobile apps using JavaScript. Components like View, Text, FlatList, and Button enable native-like UIs. react-navigation is used for seamless navigation. State is managed using Context API or React hooks like useState and useReducer.

## **Code:**

A. Home page:

code:

import React from 'react';

import { View, Text } from 'react-native';

export default function HomeScreen() {

return (

<View style={{ padding: 20 }}>

<Text style={{ fontSize: 24 }}>Welcome to FixIt</Text>

<Text>Your mobile DIY repair assistant.</Text>

</View>

);

}

## **Output:**

Displays welcome message and app introduction.

## 

## **Code:**

B. menu/product page:

code:

import React from 'react';

import { View, Text, FlatList } from 'react-native';

const data = [

{ id: '1', title: 'Fix Washing Machine Leak' },

{ id: '2', title: 'Repair Phone Charging Port' },

];

export default function RepairListScreen() {

return (

<View>

<Text style={{ fontSize: 20 }}>Repair Guides</Text>

<FlatList

data={data}

renderItem={({ item }) => <Text>{item.title}</Text>}

keyExtractor={item => item.id}

/>

</View>

);

}

## **Output:**

List of repair tutorials displayed dynamically.

## **Code:**

C. cart page:

code:

import React, { useContext } from 'react';

import { View, Text } from 'react-native';

import { CartContext } from './CartContext';

export default function CartScreen() {

const { cartItems } = useContext(CartContext);

return (

<View>

<Text>Parts Cart</Text>

{cartItems.map((item, index) => (

<Text key={index}>{item.name} - ₹{item.price}</Text>

))}

</View>

);

}

## **Output:**

Displays list of selected parts and prices.

## **Code:**

D. about us page:

code:

import React from 'react';

import { View, Text } from 'react-native';

export default function AboutScreen() {

return (

<View>

<Text>About FixIt</Text>

<Text>We help you fix things at home using step-by-step tutorials and AR overlays.</Text>

</View>

);

}

## **Output:**

Information about FixIt's purpose and vision.

## **Code:**

E. contact us page:

code:

import React from 'react';

import { View, Text } from 'react-native';

export default function ContactScreen() {

return (

<View>

<Text>Contact Us</Text>

<Text>Email: support@fixitapp.com</Text>

</View>

);

}

## **Output:**

Displays contact information.

## **Code:**

F. registration page:

code:

import React from 'react';

import { View, TextInput, Button } from 'react-native';

export default function RegisterScreen() {

return (

<View>

<TextInput placeholder="Name" />

<TextInput placeholder="Email" keyboardType="email-address" />

<TextInput placeholder="Password" secureTextEntry />

<Button title="Register" onPress={() => {}} />

</View>

);

}

## **Output:**

Registration form for new users.

## **Code:**

G. login page:

code:

import React from 'react';

import { View, TextInput, Button } from 'react-native';

export default function LoginScreen() {

return (

<View>

<TextInput placeholder="Email" keyboardType="email-address" />

<TextInput placeholder="Password" secureTextEntry />

<Button title="Login" onPress={() => {}} />

</View>

);

}

## **Output:**

Login form for existing users.

## **Conclusion:**

The mobile version of the DIY Repair Assistant app was successfully built using React Native. It supports user login, repair guide browsing, parts management, and responsive design, fulfilling core requirements of a multi-screen, interactive mobile application.