MIT Art, Design and Technology University
MIT School of Computing, Pune

Department of Information Technology

Lab Manual

Practical - CPAD

Class - T.Y. (SEM-II), SMAD

Batch - TY-IT SMAD

Name of the Student

Mr. Viraj Gundewar

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 Routing
	
	Project 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 Application
	
	Project 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: Many online shopping websites face issues like slow loading, complicated navigation, and poor user experience, causing customers to leave without buying. This project aims to build a fast, responsive, and easy-to-use e-commerce site using ReactJS to improve shopping convenience and customer satisfaction.

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.

Theory:

In today's digital age, e-commerce has transformed the way people shop and do business. With increasing internet penetration and the widespread use of smartphones, online shopping has become a common practice for millions of users worldwide. An e-commerce website is a digital platform that allows customers to browse, select, and purchase products or services over the internet. It serves as a virtual store that operates 24/7, providing convenience, variety, and accessibility to users regardless of their geographical location.

This project focuses on developing a fully functional e-commerce website using **ReactJS**, a popular front-end JavaScript library for building user interfaces. The main objective of the website is to simulate a real-world online store with key features such as product listings, detailed product views, shopping cart functionality, user authentication, and a seamless checkout process. By leveraging the power of ReactJS, the website delivers a dynamic and responsive user experience, ensuring smooth navigation and real-time updates.

Developing an e-commerce website involves not only designing attractive UI components but also ensuring a smooth flow of data, effective user engagement, and performance optimization. With React's component-based architecture and efficient rendering through the Virtual DOM, developers can create scalable and maintainable applications that cater to modern user expectations.

This project aims to demonstrate how web technologies, especially ReactJS, can be effectively utilized to build robust, user-centric, and interactive e-commerce platforms that meet industry standards.

Code:

A. Home page:

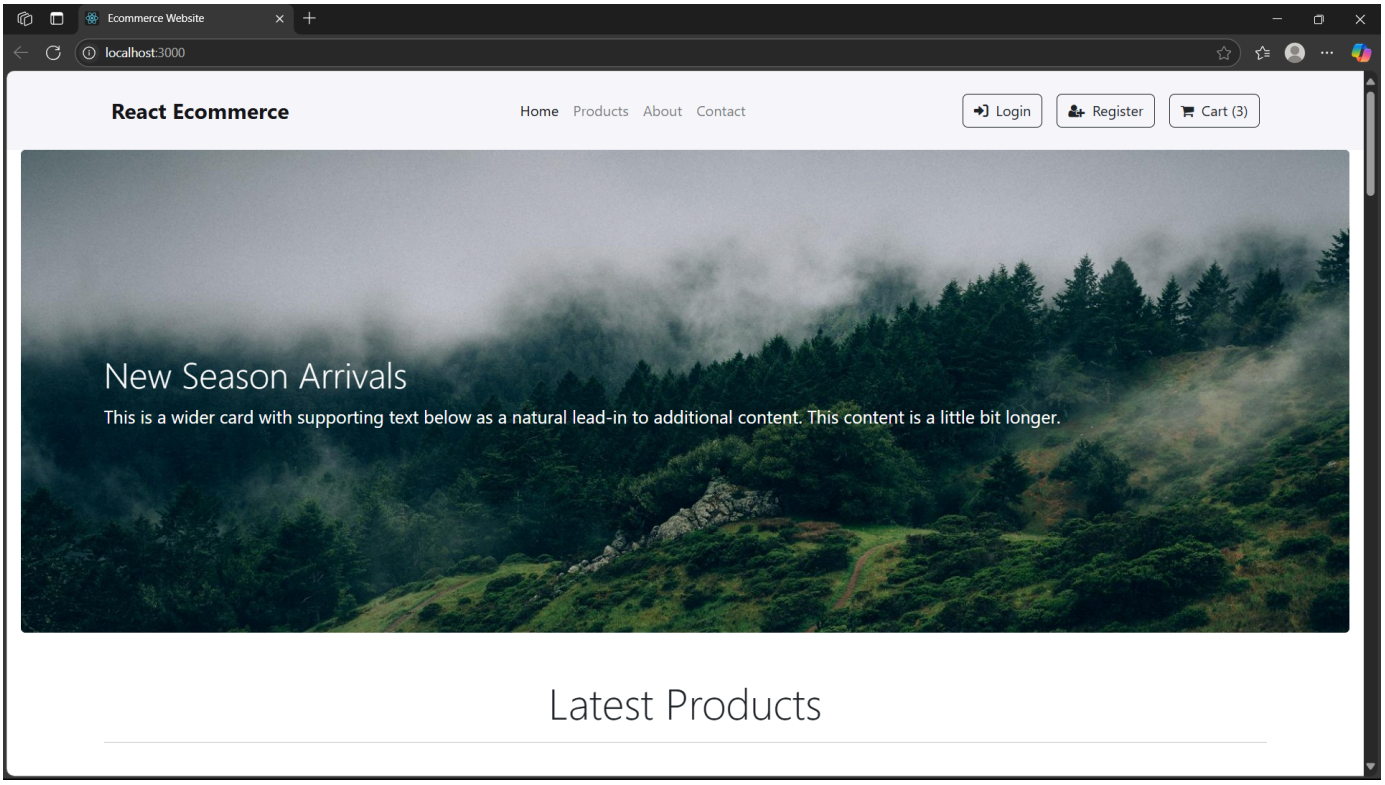
code:

```
import React from 'react';

import Header from '../components/Header';

import Hero from '../components/Hero';

import ProductCard from '../components/ProductCard';

import Footer from '../components/Footer';
```

```
const products = [
  { id: 1, name: 'Smartphone', price: '₹20,000', image: 'https://via.placeholder.com/150' },
  { id: 2, name: 'Headphones', price: '₹2,000', image: 'https://via.placeholder.com/150' },
  { id: 3, name: 'Smartwatch', price: '₹5,000', image: 'https://via.placeholder.com/150' },
  { id: 4, name: 'Laptop', price: '₹50,000', image: 'https://via.placeholder.com/150' },
];

const Home = () => {
  return (
    <>
      <Header />
      <Hero />
      <div className="product-grid" style={{ display: 'flex', flexWrap: 'wrap', gap: '20px', padding: '20px' }}>
        {products.map(product => (
          <ProductCard key={product.id} product={product} />
        ))}
      </div>
      <Footer />
    </>
  );
};

export default Home;
```

Output:

A. Index/Home page output:



Code:

B. menu/product page:

code:

```
import React from 'react';


const categories = [
 {
  id: 1,
  name: 'Electronics',
  image: 'https://via.placeholder.com/150?text=Electronics',
  description: 'Mobiles, laptops, accessories, and more.',
 },
 {
  id: 2,
  name: 'Fashion',
  image: 'https://via.placeholder.com/150?text=Fashion',
```

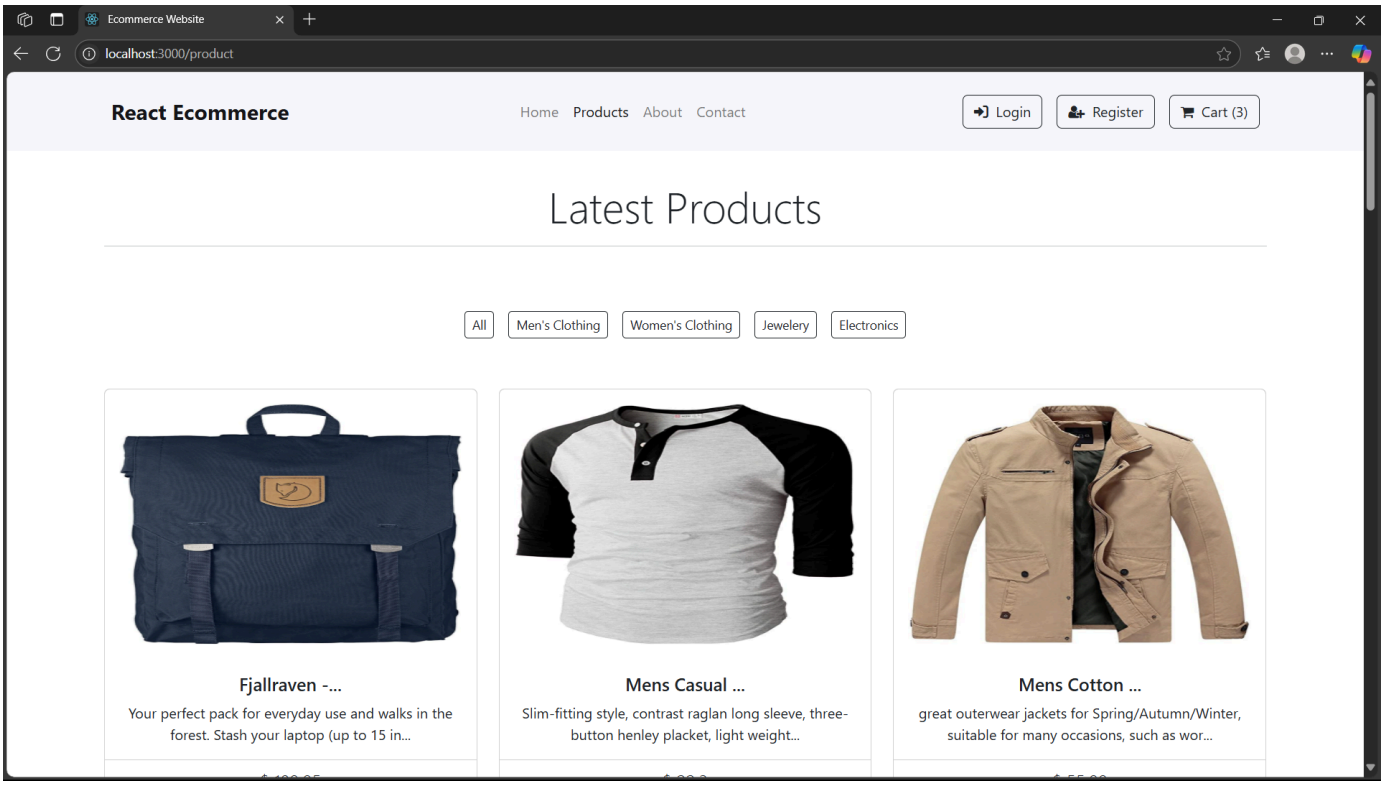
```
    description: 'Clothing, shoes, and accessories for all.',
  },
  {
    id: 3,
    name: 'Home Appliances',
    image: 'https://via.placeholder.com/150?text=Home+Appliances',
    description: 'Kitchen, cleaning, and home essentials.',
  },
  {
    id: 4,
    name: 'Books',
    image: 'https://via.placeholder.com/150?text=Books',
    description: 'Academic, fiction, non-fiction, and more.',
  },
];

const Menu = () => {
  return (
    <div style={{ padding: '40px' }}>
      <h2 style={{ textAlign: 'center', marginBottom: '30px' }}>Shop by Category</h2>
      <div
        style={{
          display: 'flex',
          flexWrap: 'wrap',
          gap: '20px',
          justifyContent: 'center',
        }}
      >
        {categories.map((cat) => (
          <div
```

```
        key={cat.id}
        style={{
          border: '1px solid #ccc',
          padding: '20px',
          width: '220px',
          textAlign: 'center',
          borderRadius: '10px',
          backgroundColor: '#f9f9f9',
        }}
      >
        <img src={cat.image} alt={cat.name} style={{ width: '100%', borderRadius: '5px' }} />
        <h3>{cat.name}</h3>
        <p>{cat.description}</p>
        <button
          style={{
            backgroundColor: '#000',
            color: '#fff',
            padding: '8px 16px',
            border: 'none',
            borderRadius: '5px',
            cursor: 'pointer',
          }}
        >
          View Products
        </button>
      </div>
    ))}
  </div>
</div>
);
```

```
};

export default Menu;
```

Output:

B. menu/product page output:



Code:

C. cart page:

code:

```
import React, { useState } from 'react';


const initialCart = [
 {
   id: 1,
```

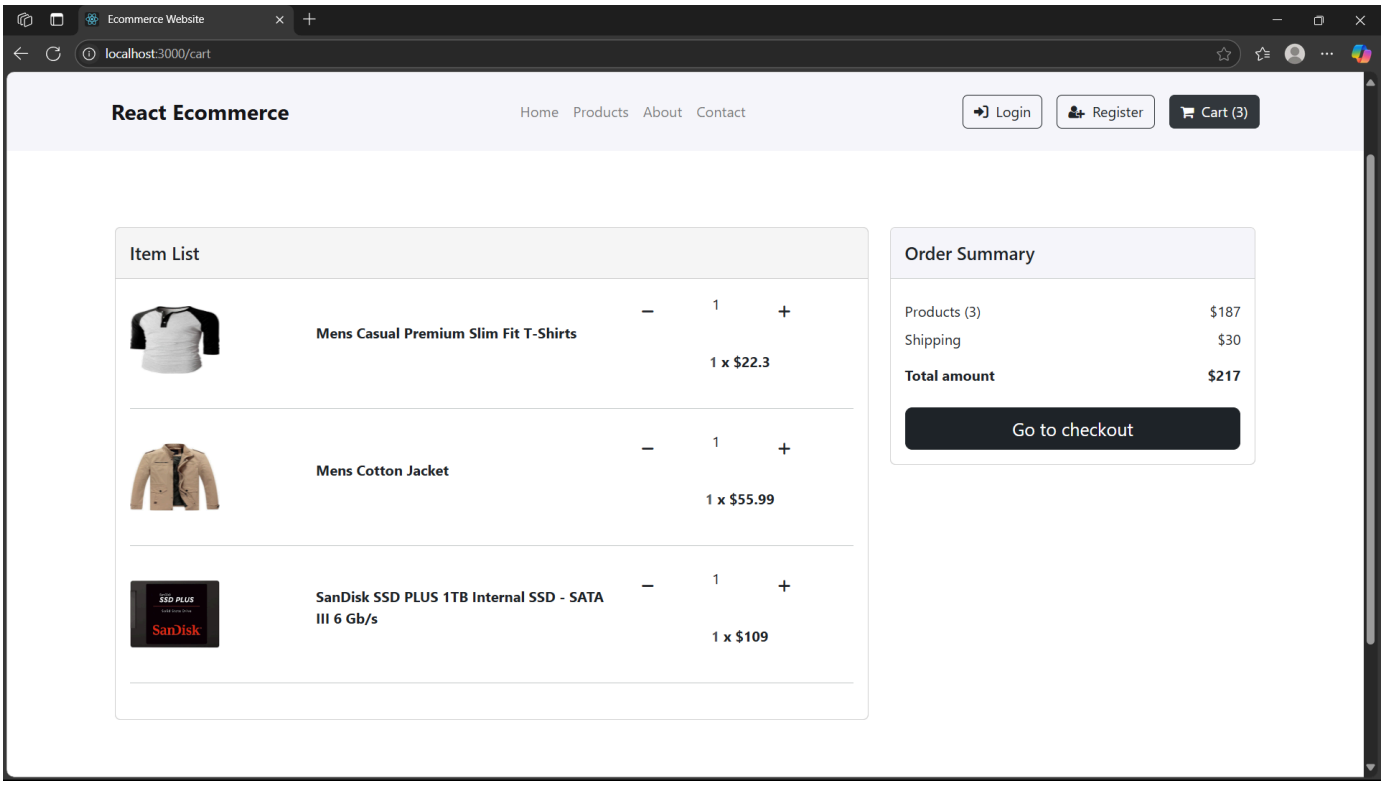
```
    name: 'Smartphone',
    price: 20000,
    quantity: 1,
    image: 'https://via.placeholder.com/100',
  },
  {
    id: 2,
    name: 'Headphones',
    price: 2000,
    quantity: 2,
    image: 'https://via.placeholder.com/100',
  },
];

const Cart = () => {
  const [cart, setCart] = useState(initialCart);

  const updateQuantity = (id, delta) => {
    setCart(prevCart =>
      prevCart.map(item =>
        item.id === id
          ? { ...item, quantity: Math.max(1, item.quantity + delta) }
          : item
      )
    );
  };

  const removeItem = (id) => {
    setCart(prevCart => prevCart.filter(item => item.id !== id));
  };
```

```
const getTotal = () => {
  return cart.reduce((total, item) => total + item.price * item.quantity, 0);
};

return (
  <div style={{ padding: '30px' }}>
    <h2>Your Cart</h2>
    {cart.length === 0 ? (
      <p>Your cart is empty.</p>
    ) : (
      <div>
        {cart.map(item => (
          <div
            key={item.id}
            style={{
              display: 'flex',
              alignItems: 'center',
              marginBottom: '20px',
              borderBottom: '1px solid #ccc',
              paddingBottom: '10px',
            }}
          >
            <img src={item.image} alt={item.name} style={{ width: '80px' }} />
            <div style={{ marginLeft: '20px', flexGrow: 1 }}>
              <h4>{item.name}</h4>
              <p>Price: ₹{item.price}</p>
              <div>
                <button onClick={() => updateQuantity(item.id, -1)}>-</button>
                <span style={{ margin: '0 10px' }}>{item.quantity}</span>
```

```
            <button onClick={() => updateQuantity(item.id, 1)}>+</button>
          </div>
        </div>
        <button onClick={() => removeItem(item.id)} style={{ color: 'red' }}>
          Remove
        </button>
      </div>
    )}}

    <h3>Total: ₹{getTotal()}</h3>
    <button
      style={{
        padding: '10px 20px',
        backgroundColor: '#000',
        color: '#fff',
        border: 'none',
        cursor: 'pointer',
        marginTop: '20px',
      }}
    >
      Proceed to Checkout
    </button>
  </div>
 )}
</div>
);
};

export default Cart;
```

Output:

C. cart page output:



Code:

D. about us page:

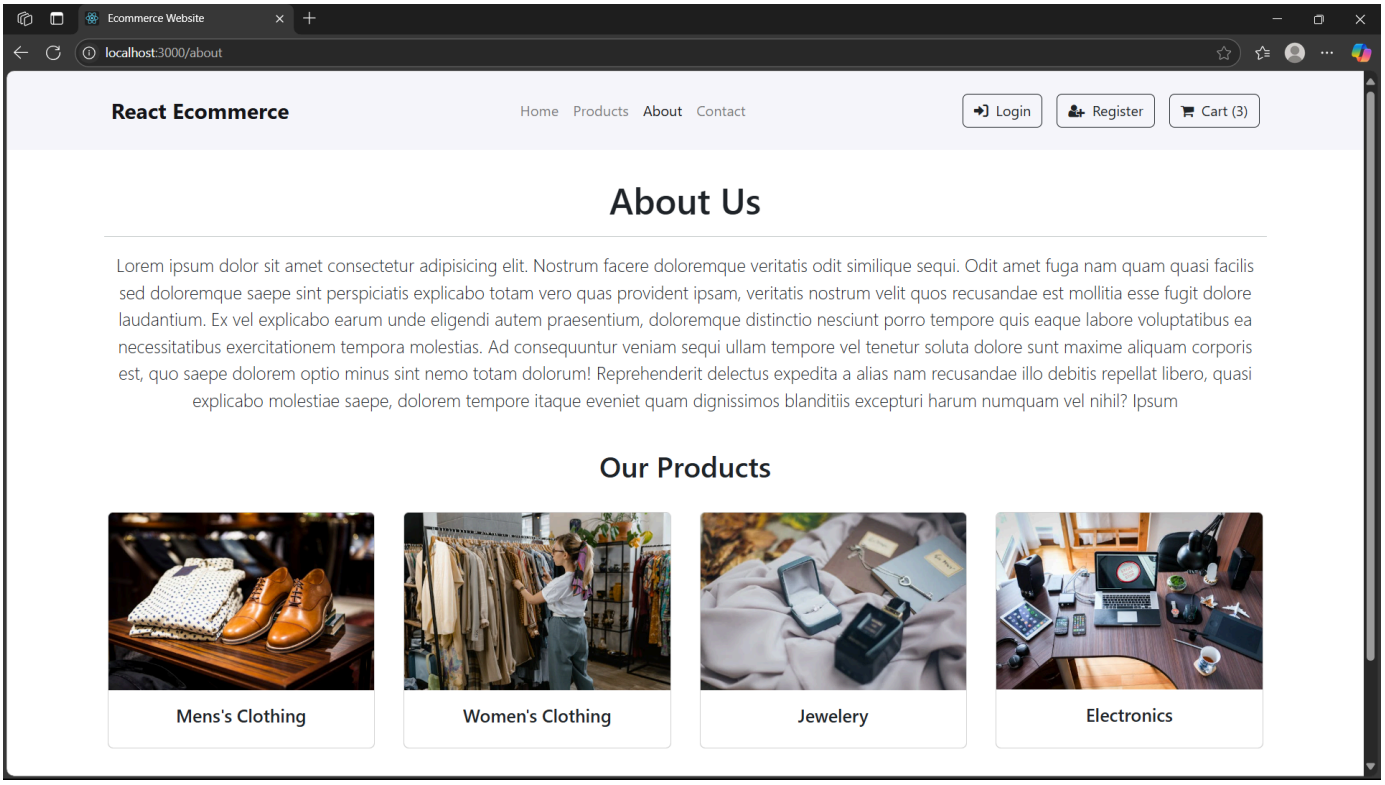
code:

```
import React from 'react';


const AboutUs = () => {
 return (
  <div style={{ padding: '40px', maxWidth: '800px', margin: '0 auto' }}>
   <h2>About Us</h2>
   <p style={{ lineHeight: '1.8' }}>
    Welcome to <strong>E-Shop</strong>, your number one source for all things electronics,
    fashion, and lifestyle. We're dedicated to giving you the very best of products,
    with a focus on quality, affordability, and customer service.
```

```
    </p>
    <p style={{ lineHeight: '1.8' }}>
      Founded in 2025 by a team of passionate developers and entrepreneurs,
      E-Shop has come a long way from its beginnings. When we first started out,
      our passion for providing a seamless online shopping experience drove us to build
      a platform that is both user-friendly and reliable.
    </p>
    <p style={{ lineHeight: '1.8' }}>
      We hope you enjoy our products as much as we enjoy offering them to you. If you have
      any questions or comments, please don't hesitate to contact us.
    </p>
    <h3>Our Mission</h3>
    <p style={{ lineHeight: '1.8' }}>
      To deliver high-quality products at affordable prices while ensuring an excellent
      customer experience through innovation, technology, and trust.
    </p>
    <h3>Contact Us</h3>
    <p>Email: support@eshop.com</p>
    <p>Phone: +91-9876543210</p>
  </div>
 );
};

export default AboutUs;
```

Output:

D. about us page output:



Code:

E. contact us page:

code:

```
import React, { useState } from 'react';


const ContactUs = () => {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    message: '',
  });


  const handleChange = (e) => {
    setFormData({
      ...formData,
```

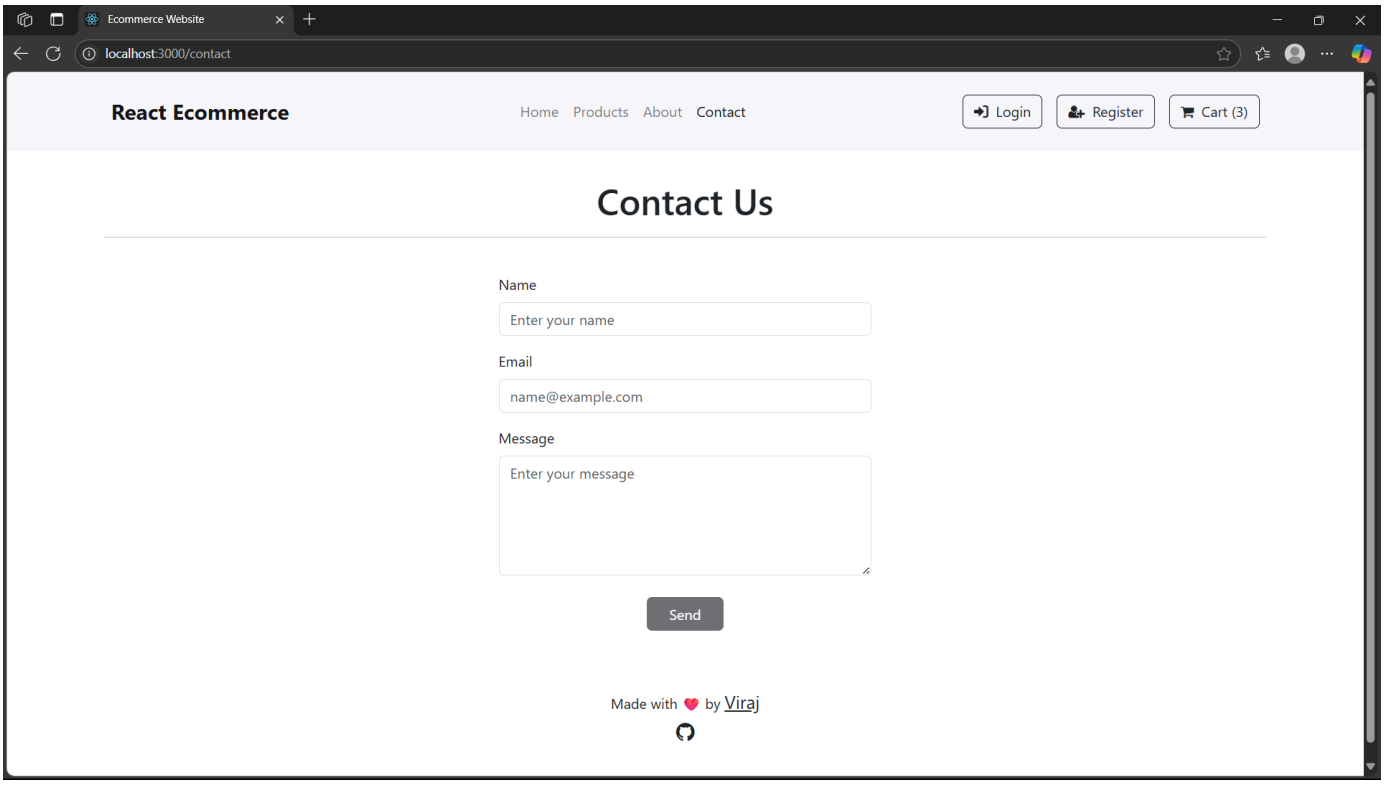
```
    [e.target.name]: e.target.value
  });
};

const handleSubmit = (e) => {
  e.preventDefault();
  alert('Thank you for contacting us!');
  setFormData({ name: '', email: '', message: '' });
};

return (
  <div style={{ padding: '40px', maxWidth: '600px', margin: '0 auto' }}>
    <h2>Contact Us</h2>
    <p>If you have any questions, feel free to reach out to us using the form below.</p>

    <form onSubmit={handleSubmit} style={{ display: 'flex', flexDirection: 'column', gap: '15px' }}>
      <input
        type="text"
        name="name"
        placeholder="Your Name"
        value={formData.name}
        onChange={handleChange}
        required
        style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
      />
      <input
        type="email"
        name="email"
        placeholder="Your Email"
        value={formData.email}
```

```
        onChange={handleChange}
        required
        style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
      />
      <textarea
        name="message"
        placeholder="Your Message"
        value={formData.message}
        onChange={handleChange}
        required
        rows="5"
        style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
      ></textarea>
      <button
        type="submit"
        style={{
          backgroundColor: '#000',
          color: '#fff',
          padding: '10px',
          border: 'none',
          borderRadius: '5px',
          cursor: 'pointer'
        }}
      >
        Send Message
      </button>
    </form>
  </div>
);
};
```

```
export default ContactUs;
```

Output:

E. contact us page output:



Code:

F. registration page:

code:

```
import React, { useState } from 'react';


const Registration = () => {

  const [formData, setFormData] = useState({

    name: '',

    email: '',
```

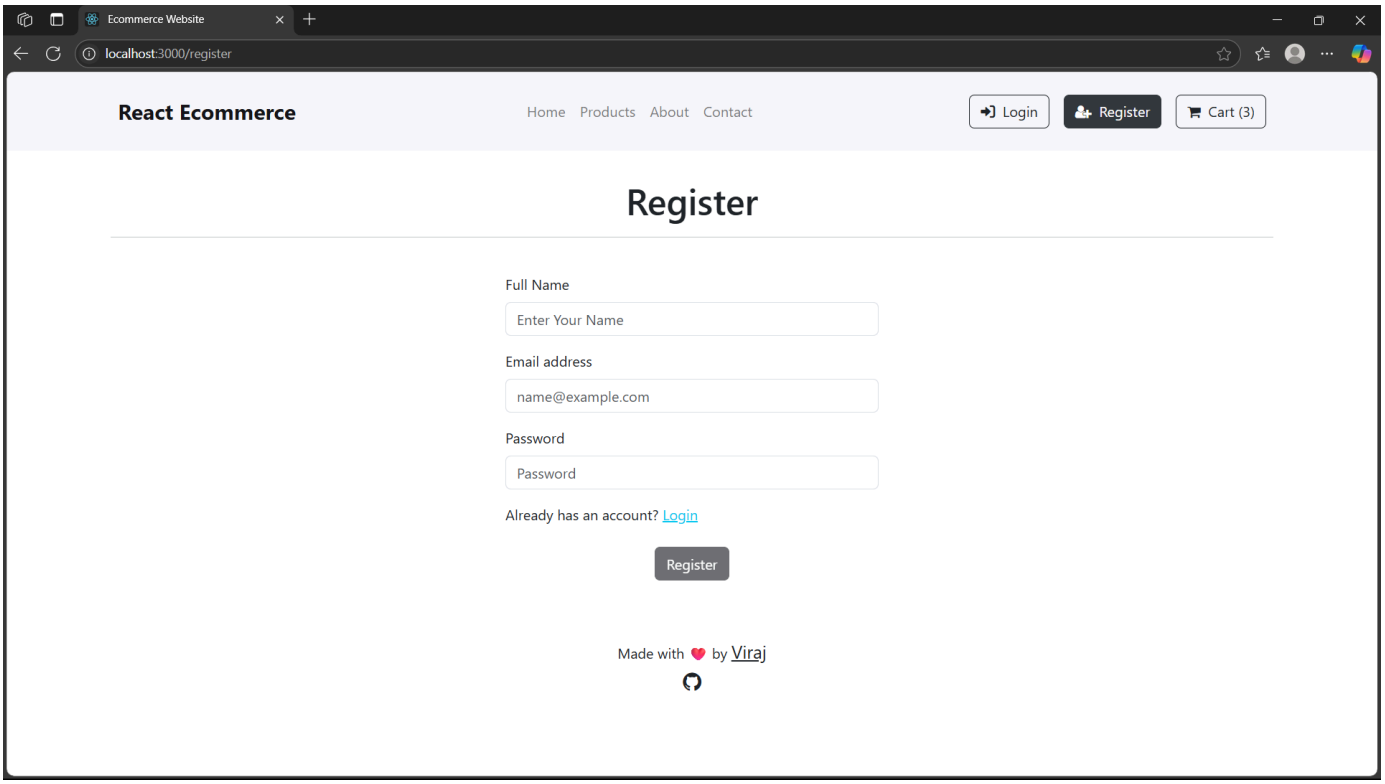
```
  password: '',
});

const [submitted, setSubmitted] = useState(false);

const handleChange = (e) => {
  setFormData({
    ...formData,
    [e.target.name]: e.target.value
  });
};

const handleSubmit = (e) => {
  e.preventDefault();
  // You can replace this with actual backend API call
  console.log('User Registered:', formData);
  setSubmitted(true);
  setFormData({ name: '', email: '', password: '' });
};

return (
  <div style={{ padding: '40px', maxWidth: '400px', margin: '0 auto' }}>
    <h2>User Registration</h2>
    {submitted && <p style={{ color: 'green' }}>Registration successful!</p>}
    <form onSubmit={handleSubmit} style={{ display: 'flex', flexDirection: 'column', gap: '15px' }}>
      <input
        type="text"
        name="name"
        placeholder="Full Name"
        value={formData.name}
```

```
  onChange={handleChange}
  required
  style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
/>
<input
  type="email"
  name="email"
  placeholder="Email Address"
  value={formData.email}
  onChange={handleChange}
  required
  style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
/>
<input
  type="password"
  name="password"
  placeholder="Password"
  value={formData.password}
  onChange={handleChange}
  required
  style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
/>
<button
  type="submit"
  style={{
    backgroundColor: '#000',
    color: '#fff',
    padding: '10px',
    border: 'none',
    borderRadius: '5px',
```

```
        cursor: 'pointer'
      }}
     >
      Register
     </button>
    </form>
   </div>
  );
};

export default Registration;
```

Output:

F. registration page output:



Code:

G. login page:

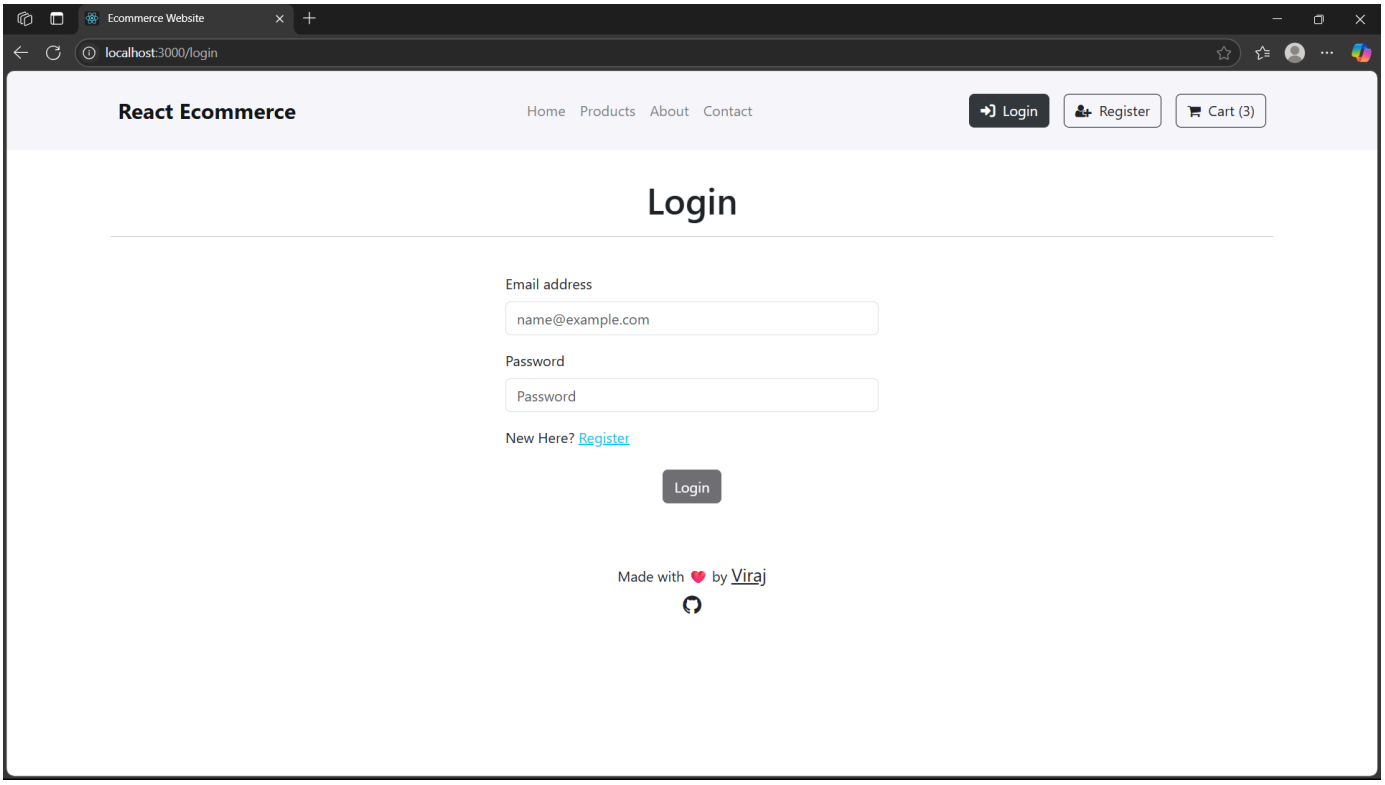
```
code:import React, { useState } from 'react';


const Login = () => {
  const [formData, setFormData] = useState({
    email: '',
    password: '',
  });

  const [loggedIn, setLoggedIn] = useState(false);

  const handleChange = (e) => {
    setFormData({
      ...formData,
      [e.target.name]: e.target.value
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    // Dummy login validation (replace with backend call)
    if (formData.email === 'user@example.com' && formData.password === 'password') {
      setLoggedIn(true);
    } else {
      alert('Invalid email or password');
    }
```

```
  setFormData({ email: '', password: '' });
};

return (
  <div style={{ padding: '40px', maxWidth: '400px', margin: '0 auto' }}>
    <h2>User Login</h2>
    {loggedIn && <p style={{ color: 'green' }}>Login successful!</p>}
    <form onSubmit={handleSubmit} style={{ display: 'flex', flexDirection: 'column', gap: '15px' }}>
      <input
        type="email"
        name="email"
        placeholder="Email Address"
        value={formData.email}
        onChange={handleChange}
        required
        style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
      />
      <input
        type="password"
        name="password"
        placeholder="Password"
        value={formData.password}
        onChange={handleChange}
        required
        style={{ padding: '10px', borderRadius: '5px', border: '1px solid #ccc' }}
      />
      <button
        type="submit"
        style={{
          backgroundColor: '#000',
```

```
          color: '#fff',
          padding: '10px',
          border: 'none',
          borderRadius: '5px',
          cursor: 'pointer'
        }}
      >
        Login
      </button>
    </form>
  </div>
 );
};

export default Login;
```

Output:

G. login page output:



Conclusion:

In conclusion, the development of this e-commerce website using ReactJS has provided a fast, scalable, and interactive user experience. The component-based architecture of ReactJS enabled the creation of reusable UI elements, making the application efficient and easier to maintain. Essential pages such as Home, Menu, Cart, Registration, Login, About Us, and Contact Us were implemented to ensure a complete and user-friendly shopping experience.

This project not only enhanced our understanding of modern frontend development but also demonstrated the power of ReactJS in building dynamic web applications. It lays a strong foundation for future enhancements such as integrating payment gateways, backend APIs, user authentication, and product search filters. Overall, the project successfully meets the requirements of a functional e-commerce platform and showcases the practical application of web development skills.

Experiment No.2

Problem Statement:

Managing daily expenses manually can be time-consuming and prone to errors, making it difficult for users to keep track of their spending habits effectively. Many people struggle to maintain a clear overview of their finances, leading to poor budgeting and unexpected financial issues.

This project aims to develop a mobile Expense Tracker app using React Native that allows users to easily record, categorize, and monitor their expenses in real-time. The app will help users gain better control over their finances and make informed budgeting decisions on the go.

Objective:

The main objective of this project is to develop a simple and intuitive mobile app using React Native that helps users easily record and manage their daily expenses. The app aims to provide clear expense categorization, track spending trends through visual charts, and assist users in budgeting effectively. It will offer a seamless experience on both Android and iOS devices to help users maintain better control over their finances anytime, anywhere.

Theory:

An expense tracker is a financial management tool designed to help users monitor their income and expenditures effectively. With the rise of smartphones, mobile applications have become a popular way for individuals to manage their daily finances on the go. React Native, a cross-platform framework, allows developers to build mobile apps for both Android and iOS using a single codebase in JavaScript.

The expense tracker app enables users to input expense details, categorize them, and visualize spending patterns through charts or summaries. This helps in identifying unnecessary expenditures and planning budgets accordingly. By offering real-time updates and easy data management, such apps empower users to make informed financial decisions and improve overall money management skills.

Code:

A. Dashboard:

code:

```
// Dashboard.js

import React from 'react';

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


const Dashboard = () => {

  // Sample data (replace with dynamic data)

  const totalIncome = 5000;
```

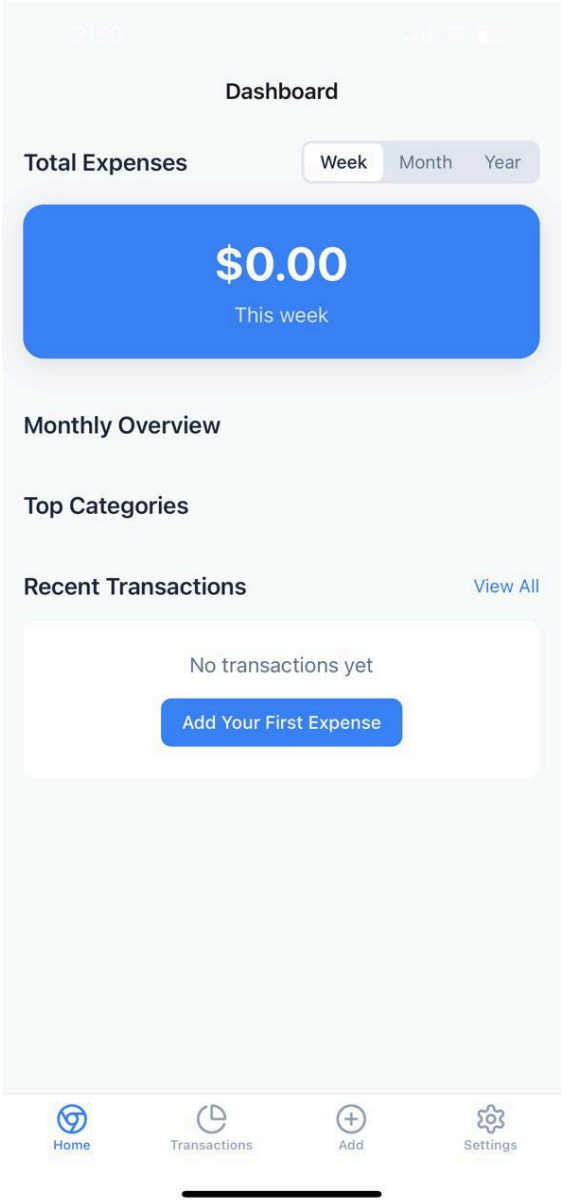
```
  const totalExpenses = 3200;
  const balance = totalIncome - totalExpenses;

  return (
    <ScrollView contentContainerStyle={styles.container}>
      <Text style={styles.header}>Dashboard</Text>

      <View style={styles.card}>
        <Text style={styles.cardTitle}>Total Income</Text>
        <Text style={styles.amount}>${totalIncome}</Text>
      </View>

      <View style={styles.card}>
        <Text style={styles.cardTitle}>Total Expenses</Text>
        <Text style={styles.amount}>${totalExpenses}</Text>
      </View>

      <View style={styles.card}>
        <Text style={styles.cardTitle}>Balance</Text>
        <Text style={[styles.amount, balance < 0 ? styles.negative : styles.positive]}>
          ${balance}
        </Text>
      </View>
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 20,
```

```
  backgroundColor: '#f5f5f5',
  flexGrow: 1,
  alignItems: 'center',
},
header: {
  fontSize: 28,
  fontWeight: 'bold',
  marginBottom: 30,
},
card: {
  width: '90%',
  backgroundColor: '#fff',
  borderRadius: 10,
  padding: 20,
  marginBottom: 20,
  alignItems: 'center',
  elevation: 3,
},
cardTitle: {
  fontSize: 18,
  color: '#888',
  marginBottom: 10,
},
amount: {
  fontSize: 24,
  fontWeight: 'bold',
},
positive: {
  color: 'green',
},
```

```
  negative: {
    color: 'red',
  },
});

export default Dashboard;
```

Output:

A. Index/Dashboard output:



Code:

B. Add expense page :

code:

```
// AddExpense.js
import React, { useState } from 'react';
import { View, Text, TextInput, Button, StyleSheet, Alert, Picker } from 'react-native';

const AddExpense = ({ onAddExpense }) => {
  const [amount, setAmount] = useState('');
  const [category, setCategory] = useState('Food');
  const [description, setDescription] = useState('');

  const handleAdd = () => {
    if (!amount || isNaN(amount) || Number(amount) <= 0) {
      Alert.alert('Invalid Amount', 'Please enter a valid amount greater than 0.');
      return;
    }

    const expense = {
      id: Date.now().toString(),
      amount: parseFloat(amount),
      category,
      description,
      date: new Date(),
    };

    onAddExpense(expense);
    setAmount('');
    setCategory('Food');
    setDescription('');
    Alert.alert('Success', 'Expense added successfully');
```

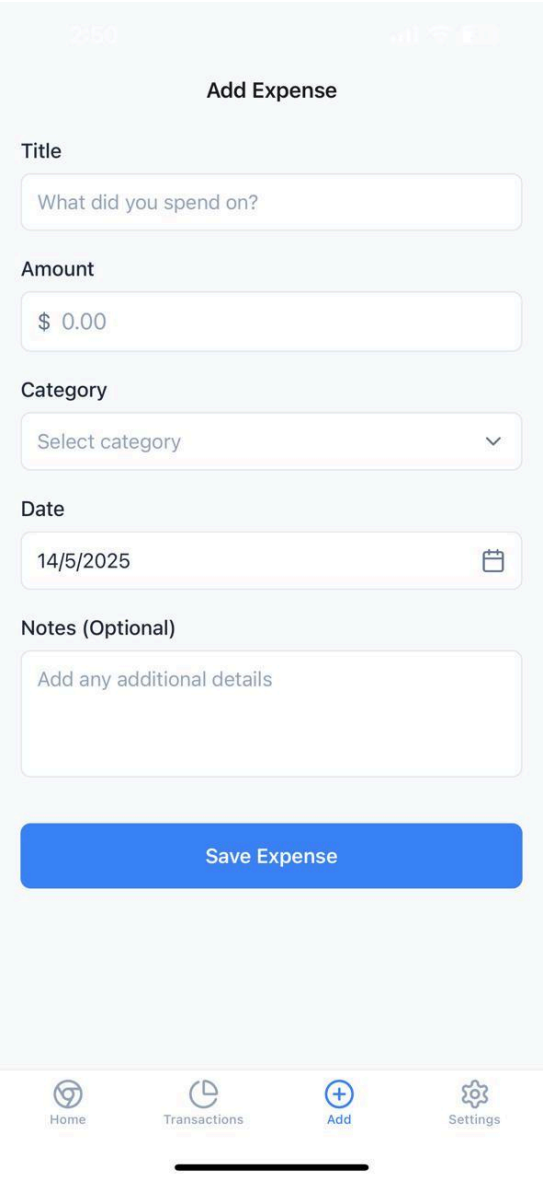
```
};

return (
  <View style={styles.container}>
    <Text style={styles.label}>Amount</Text>
    <TextInput
      style={styles.input}
      keyboardType="numeric"
      placeholder="Enter amount"
      value={amount}
      onChangeText={setAmount}
    />

    <Text style={styles.label}>Category</Text>
    <Picker
      selectedValue={category}
      onValueChange={itemValue => setCategory(itemValue)}
      style={styles.picker}
    >
      <Picker.Item label="Food" value="Food" />
      <Picker.Item label="Transport" value="Transport" />
      <Picker.Item label="Shopping" value="Shopping" />
      <Picker.Item label="Bills" value="Bills" />
      <Picker.Item label="Others" value="Others" />
    </Picker>

    <Text style={styles.label}>Description</Text>
    <TextInput
      style={styles.input}
      placeholder="Enter description"
```

```
      value={description}
      onChangeText={setDescription}
     />

     <Button title="Add Expense" onPress={handleAdd} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 20,
    backgroundColor: '#fff',
    borderRadius: 8,
    margin: 20,
    elevation: 2,
  },
  label: {
    marginTop: 10,
    marginBottom: 5,
    fontSize: 16,
  },
  input: {
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    padding: 10,
    fontSize: 16,
  },
  picker: {
```

```
    height: 50,
    width: '100%',
  },
});

export default AddExpense;
```

Output:

B. Add expense output:



Code:

C. select category :

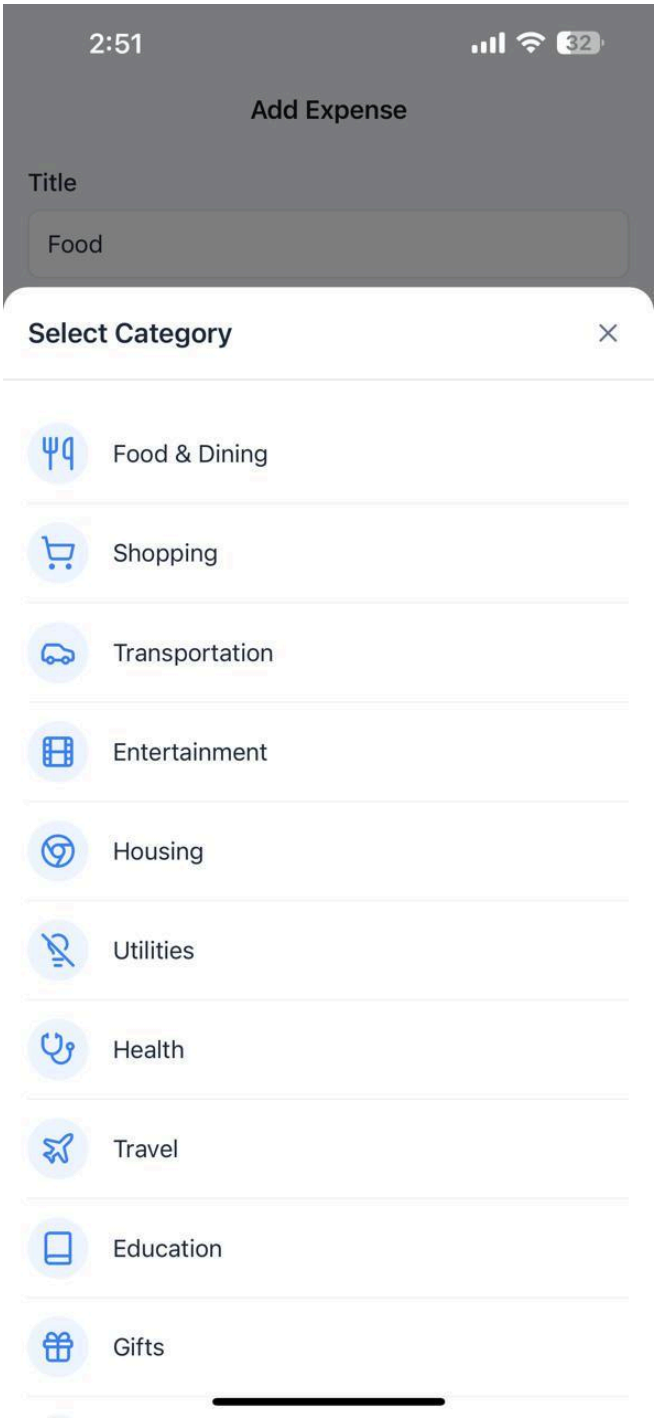
code:

```
// CategoryDropdown.js
import React from 'react';
import { View, StyleSheet } from 'react-native';
import { Picker } from '@react-native-picker/picker';

const CategoryDropdown = ({ selectedCategory, onValueChange }) => {
  return (
    <View style={styles.container}>
      <Picker
        selectedValue={selectedCategory}
        onValueChange={onValueChange}
        style={styles.picker}
        mode="dropdown"
      >
        <Picker.Item label="Food" value="Food" />
        <Picker.Item label="Transport" value="Transport" />
        <Picker.Item label="Shopping" value="Shopping" />
        <Picker.Item label="Bills" value="Bills" />
        <Picker.Item label="Entertainment" value="Entertainment" />
        <Picker.Item label="Others" value="Others" />
      </Picker>
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  container: {
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    overflow: 'hidden',
  },
  picker: {
    height: 50,
    width: '100%',
  },
});

export default CategoryDropdown;
```

Output:

C. select category output:



Code:

D. Transaction page:

code:

```
// TransactionPage.js

import React from 'react';

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

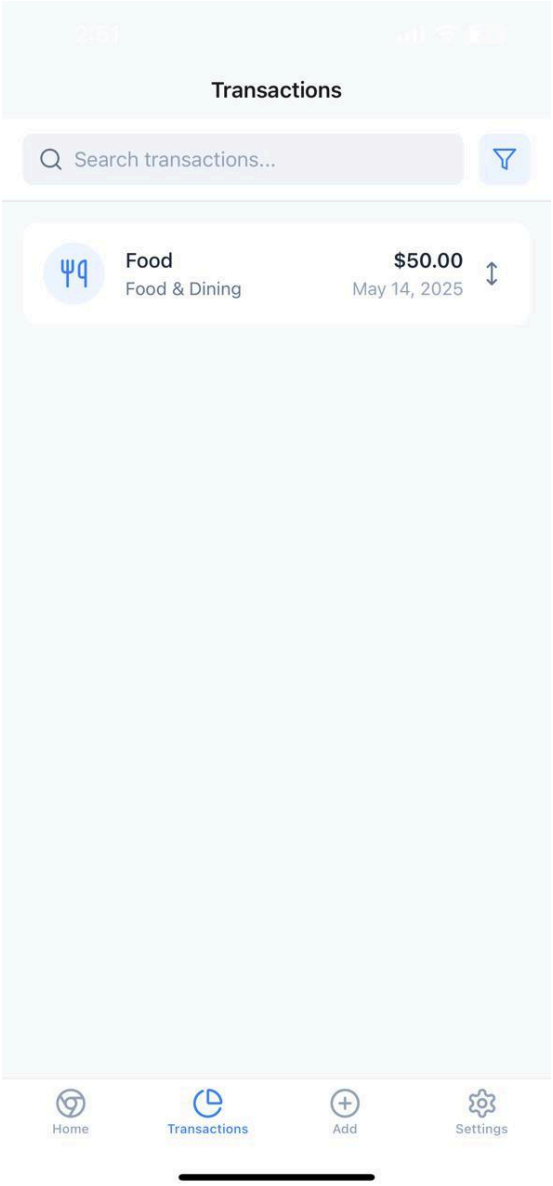
```
const TransactionPage = ({ transactions }) => {
  const renderItem = ({ item }) => (
    <View style={styles.transactionItem}>
      <View>
        <Text style={styles.category}>{item.category}</Text>
        <Text style={styles.description}>{item.description || '-'}</Text>
      </View>
      <Text style={[styles.amount, item.amount < 0 ? styles.expense : styles.income]}>
        ${item.amount.toFixed(2)}
      </Text>
    </View>
  );

  return (
    <View style={styles.container}>
      <Text style={styles.header}>Transactions</Text>
      {transactions.length === 0 ? (
        <Text style={styles.noTransactions}>No transactions available.</Text>
      ) : (
        <FlatList
          data={transactions}
          keyExtractor={(item) => item.id}
          renderItem={renderItem}
          contentContainerStyle={{ paddingBottom: 20 }}
        />
      )}
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    backgroundColor: '#fff',
  },
  header: {
    fontSize: 26,
    fontWeight: 'bold',
    marginBottom: 15,
  },
  noTransactions: {
    fontSize: 16,
    color: '#888',
    textAlign: 'center',
    marginTop: 50,
  },
  transactionItem: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    padding: 15,
    borderBottomColor: '#eee',
    borderBottomWidth: 1,
  },
  category: {
    fontSize: 18,
    fontWeight: '600',
  },
  description: {
    fontSize: 14,
```

```
    color: '#666',
  },
  amount: {
    fontSize: 18,
    fontWeight: '600',
  },
  income: {
    color: 'green',
  },
  expense: {
    color: 'red',
  },
});

export default TransactionPage;
```

Output:

D. Transaction page output:



Code:

E. Added info dashoard page:

code:

```
// Dashboard.js

import React from 'react';

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


const Dashboard = () => {

  // Sample data (replace with dynamic data)
```

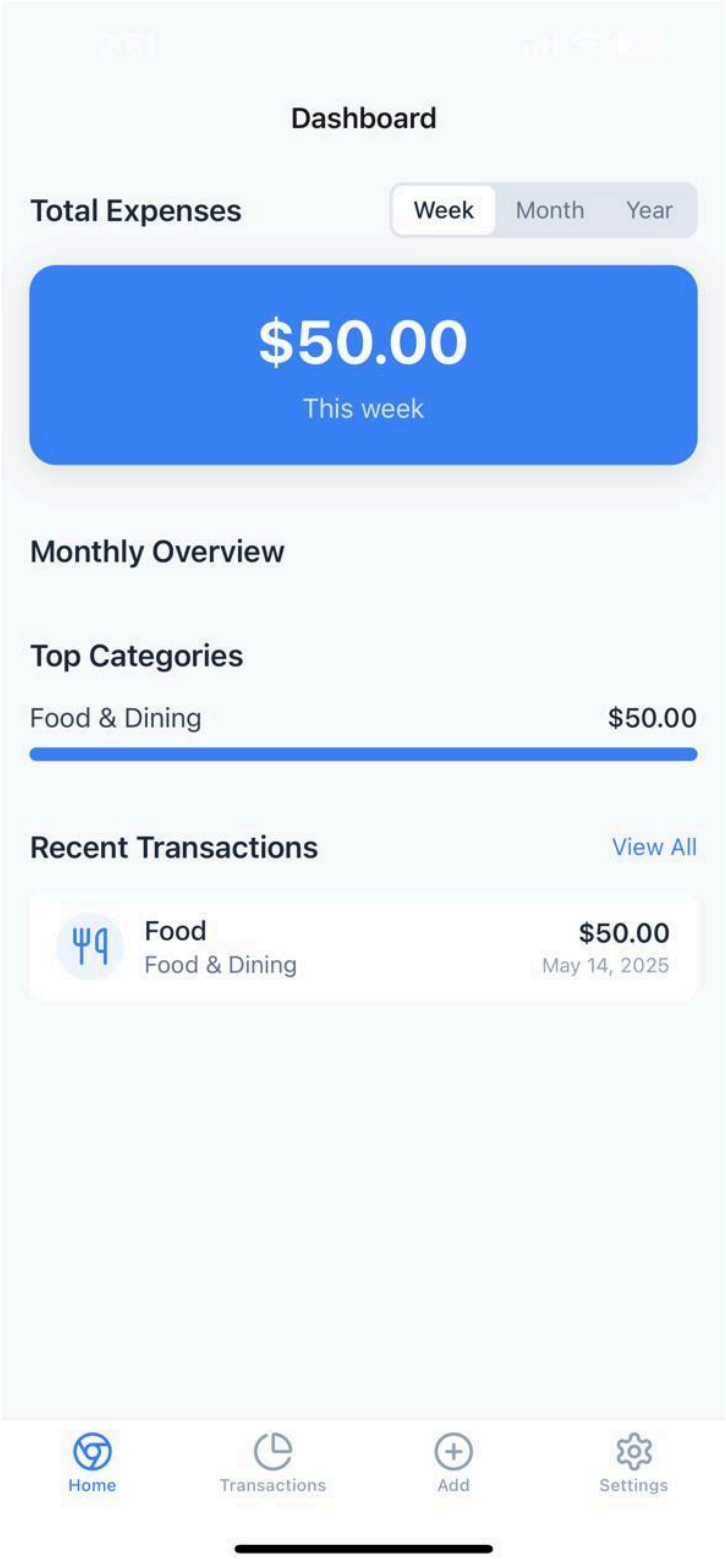
```
  const totalIncome = 5000;
  const totalExpenses = 3200;
  const balance = totalIncome - totalExpenses;

  return (
    <ScrollView contentContainerStyle={styles.container}>
      <Text style={styles.header}>Dashboard</Text>

      <View style={styles.card}>
        <Text style={styles.cardTitle}>Total Income</Text>
        <Text style={styles.amount}>${totalIncome}</Text>
      </View>

      <View style={styles.card}>
        <Text style={styles.cardTitle}>Total Expenses</Text>
        <Text style={styles.amount}>${totalExpenses}</Text>
      </View>

      <View style={styles.card}>
        <Text style={styles.cardTitle}>Balance</Text>
        <Text style={[styles.amount, balance < 0 ? styles.negative : styles.positive]}>
          ${balance}
        </Text>
      </View>
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  container: {
```

```
  padding: 20,
  backgroundColor: '#f5f5f5',
  flexGrow: 1,
  alignItems: 'center',
},
header: {
  fontSize: 28,
  fontWeight: 'bold',
  marginBottom: 30,
},
card: {
  width: '90%',
  backgroundColor: '#fff',
  borderRadius: 10,
  padding: 20,
  marginBottom: 20,
  alignItems: 'center',
  elevation: 3,
},
cardTitle: {
  fontSize: 18,
  color: '#888',
  marginBottom: 10,
},
amount: {
  fontSize: 24,
  fontWeight: 'bold',
},
positive: {
  color: 'green',
```

```
  },
  negative: {
    color: 'red',
  },
});

export default Dashboard;
```

Output:

E. Added info dashboard page output:



Code:

F. Settings page:

code:

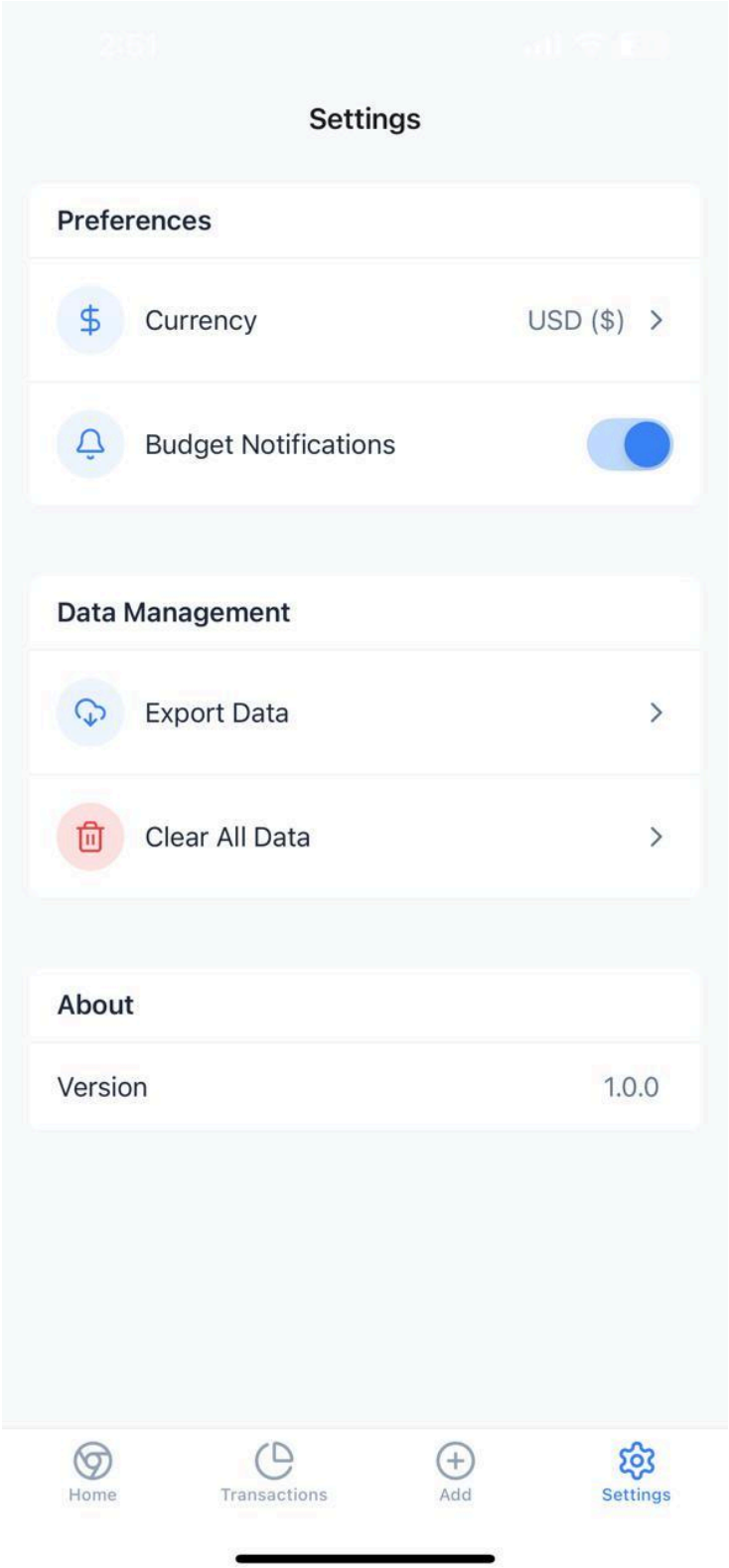
```
// SettingsPage.js
import React, { useState } from 'react';
import { View, Text, Switch, StyleSheet, TouchableOpacity, Alert } from 'react-native';

const SettingsPage = () => {
  const [notificationsEnabled, setNotificationsEnabled] = useState(true);
  const [darkModeEnabled, setDarkModeEnabled] = useState(false);

  const toggleNotifications = () => setNotificationsEnabled(previous => !previous);
  const toggleDarkMode = () => setDarkModeEnabled(previous => !previous);

  const handleClearData = () => {
    Alert.alert(
      "Clear All Data",
      "Are you sure you want to clear all expense data? This action cannot be undone.",
      [
        { text: "Cancel", style: "cancel" },
        { text: "Clear", style: "destructive", onPress: () => {
          // Add clear data logic here
          Alert.alert("Data Cleared", "All expense data has been cleared.");
        }},
      ]
    );
  };

  return (
    <View style={styles.container}>
      <Text style={styles.header}>Settings</Text>

      <View style={styles.settingRow}>
```

```
      <Text style={styles.settingText}>Enable Notifications</Text>
      <Switch
        value={notificationsEnabled}
        onValueChange={toggleNotifications}
      />
    </View>

    <View style={styles.settingRow}>
      <Text style={styles.settingText}>Dark Mode</Text>
      <Switch
        value={darkModeEnabled}
        onValueChange={toggleDarkMode}
      />
    </View>

    <TouchableOpacity style={styles.clearButton} onPress={handleClearData}>
      <Text style={styles.clearButtonText}>Clear All Data</Text>
    </TouchableOpacity>
  </View>
);
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    backgroundColor: '#fff',
  },
  header: {
    fontSize: 28,
```

```
    fontWeight: 'bold',
    marginBottom: 30,
  },
  settingRow: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    paddingVertical: 15,
    borderBottomColor: '#ddd',
    borderBottomWidth: 1,
  },
  settingText: {
    fontSize: 18,
  },
  clearButton: {
    marginTop: 40,
    backgroundColor: '#ff4d4d',
    paddingVertical: 15,
    borderRadius: 8,
    alignItems: 'center',
  },
  clearButtonText: {
    color: '#fff',
    fontWeight: 'bold',
    fontSize: 18,
  },
});

export default SettingsPage;
```

Output:

F. Settings page output:



Code:

G. Reset Function page:

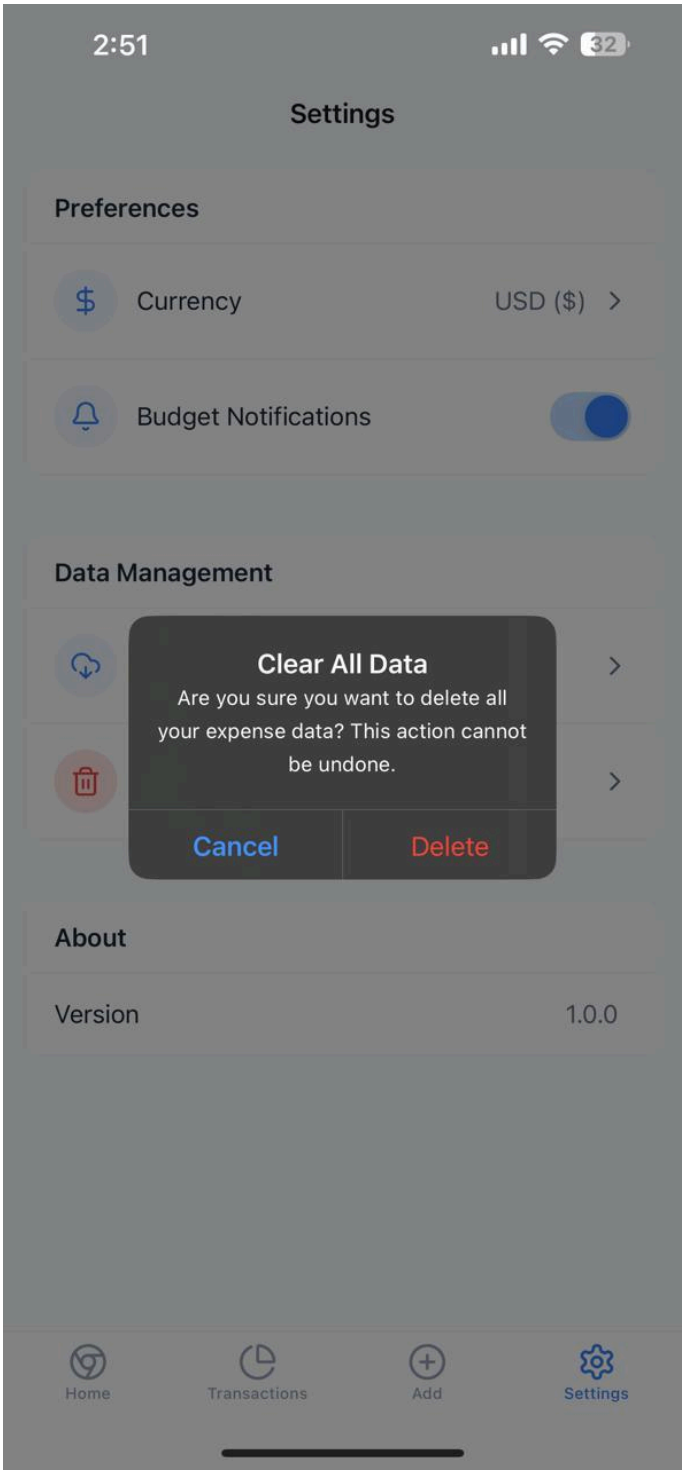
code:

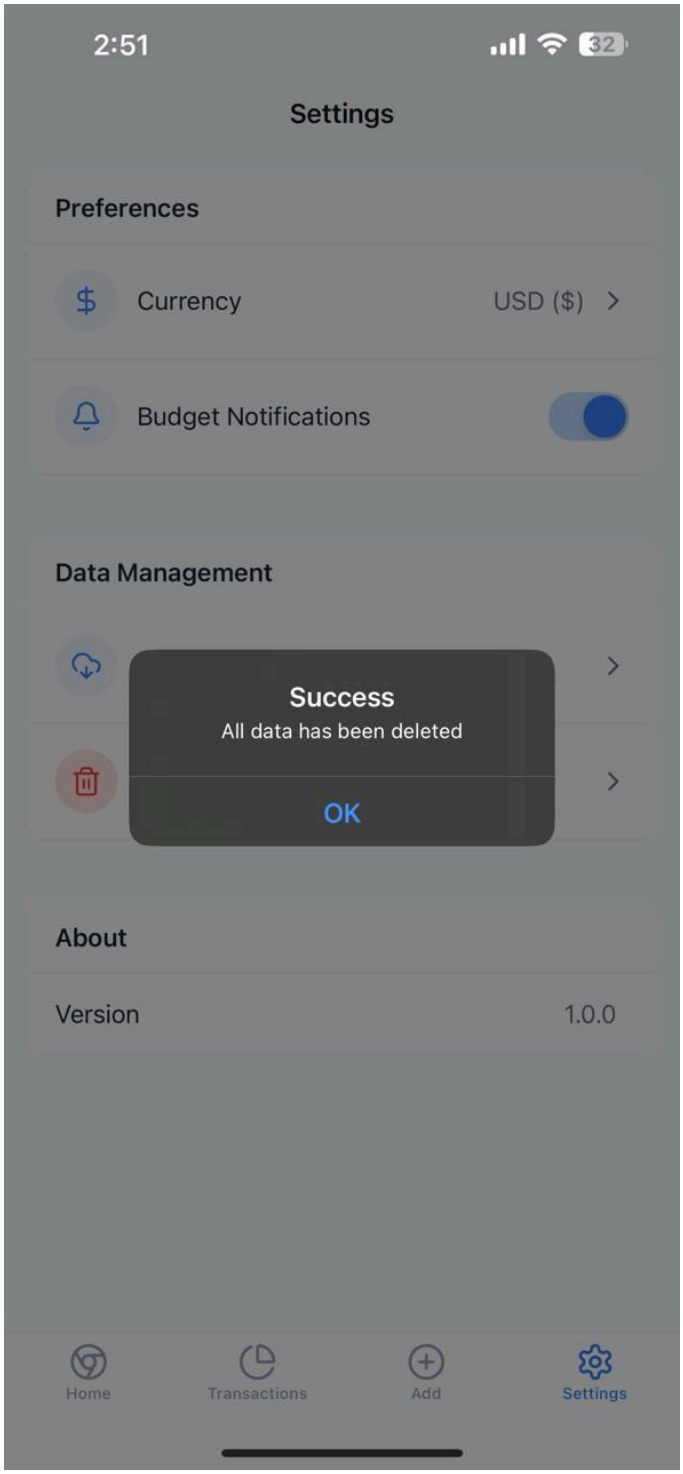
```
import React from 'react';
import { TouchableOpacity, Text, StyleSheet, Alert } from 'react-native';

const ResetButton = ({ onReset }) => {
  const handleReset = () => {
    Alert.alert(
      "Reset Confirmation",
      "Are you sure you want to reset all data?",
      [
        { text: "Cancel", style: "cancel" },
        { text: "Yes", onPress: onReset }
      ]
    );
  };

  return (
    <TouchableOpacity style={styles.button} onPress={handleReset}>
      <Text style={styles.buttonText}>Reset</Text>
    </TouchableOpacity>
  );
};

const styles = StyleSheet.create({
  button: {
    backgroundColor: '#ff4d4d',
    paddingVertical: 12,
```

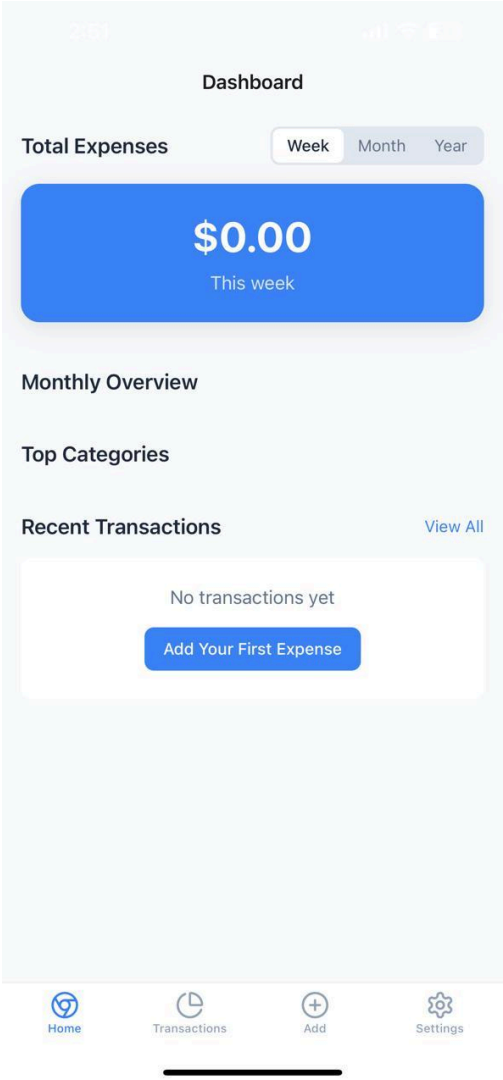
```
    paddingHorizontal: 25,
    borderRadius: 8,
    alignItems: 'center',
    marginTop: 20,
  },
  buttonText: {
    color: '#fff',
    fontWeight: 'bold',
    fontSize: 16,
  },
});

export default ResetButton;
```

Output:

G. Reset Function output:


2:51
Settings
Preferences
Currency
USD ($)
Budget Notifications
Data Management
Clear All Data
Are you sure you want to delete all your expense data? This action cannot be undone.
Cancel
Delete
About
Version
1.0.0
Home
Transactions
Add
Settings


2:51
Settings
Preferences
Currency
USD ($)
Budget Notifications
Data Management
Success
All data has been deleted
OK
About
Version
1.0.0
Home
Transactions
Add
Settings



Conclusion:

The React Native Expense Tracker app successfully provides users with an easy-to-use platform to monitor and manage their daily expenses on mobile devices. By offering features such as expense categorization, transaction history, and real-time balance updates, the app empowers users to make informed financial decisions and maintain better control over their spending habits. Utilizing React Native enables a seamless cross-platform experience, ensuring accessibility on both Android and iOS devices.

Overall, this project demonstrates the practical application of mobile development technologies to solve everyday problems related to personal finance management. With potential future enhancements like data synchronization and advanced analytics, the app can further improve user experience and financial awareness.