**ÔN THI**

# **Đọc dữ liệu API (Mảng)**

-Chỉ đọc

## **B1: Tạo data**

|  |
| --- |
| const donutsData = [    {      id: 1,      name: 'Tasty Donut',      price: '$10.00',      image: 'https://picsum.photos/200/300',      description: 'Spicy tasty donut family',      category: 'Donut',    },  ]; |

## **B2: Tạo hàm để hiển thị**

|  |
| --- |
| const hienThi = ({ item }) => {      return (        <Text>item.name</Text>      );    }; |

## **B3 : Gắn vào view muốn hiển thị**

|  |
| --- |
| <FlatList            data={filteredDonuts} //danh sách data            keyExtractor={(item) => item.id.toString()}            renderItem={hienThi} //hamHienThi  **numColumns={2}**  **columnWrapperStyle= {styles.row}**  **showsVerticalScrollIndicator={false}**        horizontal= {true}        contentContainerStyle={styles.flatListContainer}   /> |

# **Thêm , xóa, sửa, đọc (mảng) – redux toolkit**

## **B1: Tạo slice.js**

|  |
| --- |
| import { createSlice } from '@reduxjs/toolkit';  const initialState = {    products:   [      {        id : 1,        image :require('../../assets/bifour\_-removebg-preview.png'),        price : 1800,        name : "Pinarello",        heart : false,        type : "Roadbike"      },    ],  };  const productSlice = createSlice({    name: 'products',    initialState,    reducers: {      addProduct: (state, action) => { *// Thêm sản phẩm*        state.products.push(action.payload);      },      deleteProduct: (state, action) => {*// Xóa sản phẩm*        state.products = state.products.filter(          (product) => product.id !== action.payload        );      },      updateProduct: (state, action) => {*// Sửa sản phẩm*        const { id, updatedProduct } = action.payload;        const index = state.products.findIndex((product) => product.id === id);        if (index !== -1) {          state.products[index] = { ...state.products[index], ...updatedProduct };        }      },    },  });  export const {addProduct, deleteProduct, updateProduct } = productSlice.actions;  export default productSlice.reducer; |

## **B2 :Tạo store và app.js**

|  |
| --- |
| import { configureStore } from '@reduxjs/toolkit';  import bikeReducer from '../slice/bikeSliceArray';  const store = configureStore({    reducer: {      products: bikeReducer,    },  });  export default store; |

|  |
| --- |
| import store from './redux\_toolkit/store';  import { Provider } from 'react-redux';  Sau đó bao toàn bộ App.js bởi  <Provider store={store}>  </Provider> |

## **B3 : Gắn sự kiện vào trang (cứ dùng redux tookkit là phải có cái này)**

|  |
| --- |
| import { useSelector, useDispatch } from 'react-redux';  import {addProduct,updateProduct} from '.. /bikeSliceArray'  const dispatch = useDispatch(); |

## **B4: Viết hàm**

|  |
| --- |
| //Thêm  const handleAddProduct = () => {      if (!name || !price || !image) {        Alert.alert(          'Lỗi',          'Vui lòng điền đầy đủ thông tin và chọn ảnh cho sản phẩm'        );        return;      }      const newProduct = {        id: Date.now(),        name,        price,        type,        image: { uri: image },      };      dispatch(addProduct(newProduct));      Alert.alert('Thành công', 'Sản phẩm đã được thêm');      navigation.goBack();    };  //Xóa  const handleDeleteProduct = (id) => {      dispatch(deleteProduct(id)); *// Thực hiện xóa sản phẩm*       navigation.navigate('Screen2'); *// Di chuyển về Screen2 sau khi xóa thành công*   };  //Cập nhật  const handleUpdateProduct = () => {  if (!name || !price || !image) {        Alert.alert(          'Lỗi',          'Vui lòng điền đầy đủ thông tin và chọn ảnh cho sản phẩm'        );        return;      }      const updatedProductData = {        id: bikeToEdit.id, *// ID của sản phẩm đang cần cập nhật*        updatedProduct: {          name,          price: parseFloat(price),          type,          image: image === bikeToEdit.image ? bikeToEdit.image : { uri: image },        },      };      dispatch(updateProduct(updatedProductData));      Alert.alert('Cập nhật thành công', 'Sản phẩm đã được cập nhật');      navigation.navigate('Screen2');    }; |

## **B5: Nếu hàm cập nhật có thêm**

|  |
| --- |
| const isEditMode = !!route.params?.bike; *// Kiểm tra xem có truyền thông tin sản phẩm hay không*    const bikeToEdit = route.params?.bike;    useEffect(() => {      if (isEditMode) {        setName(bikeToEdit.name);        setPrice(bikeToEdit.price.toString());        setCategory(bikeToEdit.type);        if (bikeToEdit.image?.uri) {          setImageUri(bikeToEdit.image.uri);        } else {          setImageUri(bikeToEdit.image);        }      }    }, [isEditMode, bikeToEdit]); |

## **B6 : Gắn sự kiện vào nút**

|  |
| --- |
| <TouchableOpacity onPress={()=> handleUpdateProduct()}> |

# **Hiển thị dữ liệu theo danh mục và có đổi màu , seeALL**

## **B1: Tạo data (sử dụng data ở trên)**

|  |
| --- |
| const data = [    { id: 1, name: 'Xe đạp A', type: 'xe đạp' },    { id: 2, name: 'Xe máy B', type: 'xe máy' },    { id: 3, name: 'Xe đạp C', type: 'xe đạp' },    { id: 4, name: 'Xe máy D', type: 'xe máy' },  ];  const categories = ['Tất cả', 'xe đạp', 'xe máy']; |

## **B2: Tạo useState (hooks)**

|  |
| --- |
| import React, {useState} from 'react';  // State để lưu danh mục và dữ liệu lọc  const [selectedCategory, setSelectedCategory] = useState('Tất cả');  const [filteredData, setFilteredData] = useState(data);  //Nếu có seeAll  const [showAll, setShowAll] = useState(false); |

## **B3: Viết hàm xử lí khi nút danh mục được chọn**

|  |
| --- |
| const handleCategoryPress = (category) => {    setSelectedCategory(category);    if (category === 'Tất cả') {      setFilteredData(data); // Hiển thị tất cả dữ liệu    } else {      setFilteredData(data.filter((item) => item.type === category)); // Lọc theo danh mục  }  setShowAll(false); //Nếu có seeAll  };  //Nếu có seeAll  const displayData = showAll ? filteredData : filteredData.slice(0, 4); |

## **B4: Gắn hàm đó vào nút**

|  |
| --- |
| <TouchableOpacity        onPress={() => handleCategoryPress(item)}        style={[          styles.categoryButton,          selectedCategory === item && styles.selectedCategoryButton, // Đổi màu nút được chọn        ]}   ></TouchableOpacity> |

## **B5: Thêm nút SeeAll**

|  |
| --- |
| {filteredData.length > 4 && (    <TouchableOpacity      onPress={() => setShowAll(!showAll)}      style={styles.seeAllButton}    >      <Text style={styles.seeAllText}>        {showAll ? 'Collapse' : 'See All'}      </Text>    </TouchableOpacity>  )} |

# **Chuyển màn hình**

## **B1: App.js**

|  |
| --- |
| import Screen1 from './pages/screen1';  import Screen2 from './pages/screen2';  import { NavigationContainer } from '@react-navigation/native';  import { createStackNavigator } from '@react-navigation/stack';  const Stack = createStackNavigator();  export default function App() {    return (      <NavigationContainer>        <Stack.Navigator initialRouteName="Screen1">           <Stack.Screen name="Screen1" component={Screen1} options={{ headerShown: false }}/>           <Stack.Screen name="Screen2" component={Screen2}/>        </Stack.Navigator>      </NavigationContainer>    );  } |

## **B2: Gắn sự kiện khi nhấn nút chuyển màn hình**

|  |
| --- |
| onPress={() => navigation.navigate('Profile', {name: 'Jane'})} /> |

## **B3: Nếu có truyền dữ liệu (cách để nhận dữ liệu truyền đi)**

|  |
| --- |
| const {name} = route.params |

# **Thẻ textinput**

|  |
| --- |
| const [food, setFood] = useState('');  <TextInput            style={styles.input}            onChangeText={setFood}            value={food}            placeholder="Search food"            placeholderTextColor="#ccc"  keyboardType="numeric" (nhập số)   /> |

# **Combobox**

|  |
| --- |
| import { Picker } from '@react-native-picker/picker';  const [category, setCategory] = useState('roadbike');  <Picker          selectedValue={category}          style={styles.picker}          onValueChange={(itemValue) => setCategory(itemValue)}        >          <Picker.Item label="Roadbike" value="roadbike" />          <Picker.Item label="Mountain" value="mountain" />  </Picker> |

# **Chọn ảnh**

|  |
| --- |
| import \* as ImagePicker from 'expo-image-picker';  const [imageUri, setImageUri] = useState(null);  **//Tạo hàm**  const pickImage = async () => {  *// Xin quyền truy cập ảnh*      const permissionResult = await ImagePicker.requestMediaLibraryPermissionsAsync();      if (permissionResult.granted === false) {        Alert.alert("Lỗi", "Bạn cần cấp quyền truy cập thư viện ảnh để chọn ảnh.");        return;      }  *// Mở thư viện ảnh*      const result = await ImagePicker.launchImageLibraryAsync({        mediaTypes: ImagePicker.MediaTypeOptions.Images,        allowsEditing: true,        quality: 1,      });      if (!result.canceled) {        setImageUri(result.assets[0].uri); *// Lưu đường dẫn ảnh đã chọn*      }    };  **//Gắn sự kiện**   {image ? (          typeof image === 'string' ? (  *// Nếu `image` là URL*            <Image source={{ uri: image }} style={styles.image} />          ) : (  *// Nếu `image` là local asset thông qua require*            <Image source={image} style={styles.image} />          )   ) : (  *// Nếu không có ảnh thì hiển thị placeholder*       <View style={styles.imagePlaceholder}>            <Text style={styles.imageText}>TRỐNG</Text>       </View>   )}   <TouchableOpacity style={styles.button} onPress={pickImage}>       <Text style={styles.buttonText}>Chọn ảnh</Text>   </TouchableOpacity> |

# **Đọc, thêm, xóa, sửa dữ liệu với redux toolkit**

## **B1: Tạo Slice.js**

|  |
| --- |
| import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';  import axios from 'axios';  const apiURL = 'https://mockapi.io/api/v1/products'; // Thay bằng link Mock API của bạn  // Async actions  export const fetchProducts = createAsyncThunk('products/fetchProducts', async () => {    const response = await axios.get(apiURL);    return response.data;  });  export const addProduct = createAsyncThunk('products/addProduct', async (product) => {    const response = await axios.post(apiURL, product);    return response.data;  });  export const deleteProduct = createAsyncThunk('products/deleteProduct', async (id) => {    await axios.delete(`${apiURL}/${id}`);    return id;  });  export const updateProduct = createAsyncThunk('products/updateProduct', async ({ id, updatedProduct }) => {    const response = await axios.put(`${apiURL}/${id}`, updatedProduct);    return response.data;  });  // Slice  const productSlice = createSlice({    name: 'products',    initialState: {      products: [],      loading: false,      error: null,    },    reducers: {},    extraReducers: (builder) => {      builder        // Fetch products        .addCase(fetchProducts.pending, (state) => {          state.loading = true;        })        .addCase(fetchProducts.fulfilled, (state, action) => {          state.loading = false;          state.products = action.payload;        })        .addCase(fetchProducts.rejected, (state, action) => {          state.loading = false;          state.error = action.error.message;        })        // Add product        .addCase(addProduct.fulfilled, (state, action) => {          state.products.push(action.payload);        })        // Delete product        .addCase(deleteProduct.fulfilled, (state, action) => {          state.products = state.products.filter((product) => product.id !== action.payload);        })        // Update product        .addCase(updateProduct.fulfilled, (state, action) => {          const index = state.products.findIndex((product) => product.id === action.payload.id);          if (index !== -1) {            state.products[index] = action.payload;          }        });    },  });  export default productSlice.reducer; |

## **B2: Tạo file store.js**

|  |
| --- |
| import { configureStore } from '@reduxjs/toolkit';  import productReducer from './productSlice';  const store = configureStore({    reducer: {      products: productReducer,    },  });  export default store; |

## **B3: Bỏ vào trang muốn đọc dữ liệu**

|  |
| --- |
| import React, { useEffect, useState } from 'react';  import { useSelector, useDispatch } from 'react-redux';  import { fetchProducts, addProduct, deleteProduct, updateProduct } from './productSlice';  const ProductScreen = () => {    const [name, setName] = useState('');    const [price, setPrice] = useState('');    const [editId, setEditId] = useState(null);    const { products, loading, error } = useSelector((state) => state.products);    const dispatch = useDispatch();    useEffect(() => {      dispatch(fetchProducts());    }, [dispatch]);    const handleAddOrUpdate = () => {      const product = { name, price };      if (editId) {        dispatch(updateProduct({ id: editId, updatedProduct: product }));        setEditId(null);      } else {        dispatch(addProduct(product));      }      setName('');      setPrice('');    };    const handleEdit = (product) => {      setEditId(product.id);      setName(product.name);      setPrice(product.price);    };    if (loading) {      return <ActivityIndicator size="large" color="#0000ff" />;    }    if (error) {      return <Text>Error: {error}</Text>;    }    return (      <View style={styles.container}>        <Button title={editId ? 'Cập nhật' : 'Thêm'} onPress={handleAddOrUpdate} />      </View>    );  };  export default ProductScreen;  <FlatList          data={filteredDonuts} //danh sách data           keyExtractor={(item) => item.id.toString()}           renderItem={hienThi} //hamHienThi  **numColumns={2}**  **columnWrapperStyle= {styles.row}**   /> |

## **B4: Sửa app.js**

|  |
| --- |
| import store from './redux\_toolkit/store';  import { Provider } from 'react-redux';  Sau đó bao toàn bộ App.js bởi  <Provider store={store}>  </Provider> |

# **Spanner**

|  |
| --- |
| import React, { useRef } from 'react';  import Carousel from 'react-native-snap-carousel';  const { width } = Dimensions.get('window'); // Chiều rộng màn hình  const ImageSlider = () => {    const carouselRef = useRef(null);    const data = [      require('../assets/header1.jpg'),      require('../assets/header2.jpg'),      require('../assets/header3.jpg'),    ];    const renderItem = ({ item }) => (      <View style={styles.bannerImg}>        <Image source={item} style={styles.bannerForImg} />      </View>    );    return (      <Carousel        ref={carouselRef}        data={data}        renderItem={renderItem}        sliderWidth={width} // Chiều rộng của slider        itemWidth={width} // Chiều rộng từng item (mỗi ảnh)        autoplay={true} // Tự động cuộn        autoplayInterval={3000} // Thời gian giữa mỗi lần cuộn (ms)        loop={true} // Lặp lại slider        dotStyle={styles.dot} // Style cho dot không active        activeDotStyle={styles.activeDot} // Style cho dot active      />    );  }; |

# **Giỏ Hàng ReduxToolkit**

|  |
| --- |
| import { createSlice } from '@reduxjs/toolkit';  const cartSlice = createSlice({    name: 'cart',    initialState: {      items: [], *// Danh sách sản phẩm trong giỏ hàng*      total: 0, *// Tổng tiền*    },    reducers: {      addItem: (state, action) => {        const product = action.payload;        const existingItem = state.items.find((item) => item.id === product.id);        if (existingItem) {          existingItem.quantity += 1; *// Tăng số lượng nếu sản phẩm đã có*        } else {  *// Thêm sản phẩm mới với quantity = 1*          state.items.push({ ...product, quantity: 1 });        }  *// Cập nhật tổng tiền*        state.total = state.items.reduce(          (sum, item) => sum + item.price \* item.quantity,          0        );      },      removeItem: (state, action) => {        const productId = action.payload;        state.items = state.items.filter((item) => item.id !== productId);        state.total = state.items.reduce((sum, item) => {          const price = Number(item.price) || 0;          const quantity = Number(item.quantity) || 0;          return sum + price \* quantity;        }, 0);        console.log('Total after removeItem:', state.total);      },      incrementQuantity: (state, action) => {        const productId = action.payload;        const existingItem = state.items.find((item) => item.id === productId);        if (existingItem) {          existingItem.quantity += 1;        }        state.total = state.items.reduce((sum, item) => {          const price = Number(item.price) || 0;          const quantity = Number(item.quantity) || 0;          return sum + price \* quantity;        }, 0);        console.log('Total after incrementQuantity:', state.total);      },      decrementQuantity: (state, action) => {        const productId = action.payload;        const existingItem = state.items.find((item) => item.id === productId);        if (existingItem && existingItem.quantity > 1) {          existingItem.quantity -= 1;        } else if (existingItem) {          state.items = state.items.filter((item) => item.id !== productId);        }        state.total = state.items.reduce((sum, item) => {          const price = Number(item.price) || 0;          const quantity = Number(item.quantity) || 0;          return sum + price \* quantity;        }, 0);        console.log('Total after decrementQuantity:', state.total);      },    },  });  export const { addItem, removeItem, incrementQuantity, decrementQuantity } =    cartSlice.actions;  export default cartSlice.reducer; |

# **==================== ÔN TỚI ĐÂY**

# **Giỏ hàng hooks (không cần)**

## **B1: Custom hook .Đặt tên class là useCart (giữ nguyên tất cả)**

|  |
| --- |
| import React, { createContext, useContext, useState, useMemo } from 'react';  *// Chia se gio hang giua cac component*  const CartContext = createContext();  *// Custom hook để sử dụng context dễ dàng hơn*  export const useCart = () => useContext(CartContext);  *// CartProvider bao bọc toàn bộ ứng dụng và cung cấp giỏ hàng*  export const CartProvider = ({ children }) => {    const [cart, setCart] = useState([]);  *// Hàm thêm sản phẩm vào giỏ hàng*    const addToCart = (product) => {      setCart((prevCart) => {        const existingProduct = prevCart.find((item) => item.id === product.id);        if (existingProduct) {          return prevCart.map((item) =>            item.id === product.id              ? { ...item, quantity: item.quantity + 1 }              : item          );        } else {          return [...prevCart, { ...product, quantity: 1 }];        }      });    };  *// Hàm giảm số lượng sản phẩm*    const decreaseQuantity = (productId) => {      setCart((prevCart) =>        prevCart          .map((item) =>            item.id === productId              ? { ...item, quantity: item.quantity - 1 }              : item          )          .filter((item) => item.quantity > 0) *// Loại bỏ sản phẩm nếu số lượng bằng 0*      );    };  *// Hàm xóa sản phẩm khỏi giỏ hàng*    const removeFromCart = (productId) => {      setCart((prevCart) => prevCart.filter((item) => item.id !== productId));    };  *//Tính tổng tiền giỏ hàng*  *//useMemo giúp ghi nhớ các giá trị đã tính toán*    const totalAmount = useMemo(      () =>        cart.reduce((total, item) => total + item.price \* item.quantity, 0),      [cart]    );    return (      <CartContext.Provider        value={{ cart, addToCart, decreaseQuantity, removeFromCart, totalAmount }}      >        {children}      </CartContext.Provider>    );  }; |

## **B2: Import trong APP.js và bao bọc code**

|  |
| --- |
| import {CartProvider } from './hook/useCart'  const chuyeManHinh= () => {    return (      <CartProvider>      <NavigationContainer>  //      </NavigationContainer>      </CartProvider>    );}  export default chuyeManHinh; |

## **B3 : Trang có nút thêm giỏ hàng**

|  |
| --- |
| import {useCart} from '../hook/useCart'  const {addToCart } = useCart();  const themSP = () =>{         addToCart(bike);         navigation.navigate('Card');  }  **//Gắn sự kiện**  <TouchableOpacity style={styles.button} onPress={themSP}>         <Text style={styles.textButton}>Add to Card</Text>  </TouchableOpacity> |

## **B4: Trang giỏ hàng**

|  |
| --- |
| import {useCart} from '../hook/useCart'  const {cart, addToCart, decreaseQuantity, removeFromCart, totalAmount} = useCart();  <FlatList       data={cart}       keyExtractor={(item) => item.id.toString()}       renderItem={hienThi}  />  **//Tong tien**  <Text > Tổng tiền : {totalAmount} </Text>  **//Tang**  <TouchableOpacity style={styles.button}                              onPress= {() =>addToCart(item)}  >              <Text style={styles.textButon}>+</Text>  </TouchableOpacity>  **//Giam**  <TouchableOpacity style={styles.button}            onPress= {() =>decreaseQuantity(item.id)}  >        <Text style={styles.textButon}>-</Text>  </TouchableOpacity> |

# **Đăng nhập mảng**

|  |
| --- |
| const LoginScreen = ({ navigation }) => {    // Mảng dữ liệu người dùng    const users = [      { email: 'user1@example.com', password: 'password1' },      { email: 'user2@example.com', password: 'password2' },    ];    // State để lưu email và password nhập vào    const [email, setEmail] = useState('');    const [password, setPassword] = useState('');    // Hàm xử lý khi nhấn "Continue"    const handleContinue = () => {      // Tìm người dùng trong mảng      const user = users.find((u) => u.email === email && u.password === password);      if (user) {        // Nếu thông tin đúng, chuyển sang Screen 02        navigation.navigate('Screen02');      } else {        // Nếu sai, hiển thị thông báo        Alert.alert('Error', 'Invalid email or password');      }    };    return (        <TextInput          placeholder="Email"          style={styles.input}          value={email}          onChangeText={(text) => setEmail(text)}          keyboardType="email-address"          autoCapitalize="none"        />        <TextInput          placeholder="Password"          style={styles.input}          value={password}          onChangeText={(text) => setPassword(text)}          secureTextEntry        />        <Button title="Continue" onPress={handleContinue} />    );  }; |

# **Đăng nhập và đăng kí với ReduxToolkit (mảng)**

|  |
| --- |
| // userSlice.js  import { createSlice } from '@reduxjs/toolkit';  const userSlice = createSlice({    name: 'users',    initialState: {      users: [        { email: 'user1@example.com', password: 'password1' },        { email: 'user2@example.com', password: 'password2' },      ], // Mảng người dùng giả lập      currentUser: null, // Lưu trữ người dùng đang đăng nhập      isAuthenticated: false, // Trạng thái đăng nhập      error: null, // Lỗi nếu có    },    reducers: {      registerUser: (state, action) => {        const newUser = action.payload;        // Kiểm tra xem email đã tồn tại chưa        const existingUser = state.users.find((user) => user.email === newUser.email);        if (!existingUser) {          state.users.push(newUser); // Thêm người dùng mới vào mảng        } else {          state.error = 'Email đã được đăng ký!';        }      },      loginUser: (state, action) => {        const { email, password } = action.payload;        const user = state.users.find((user) => user.email === email && user.password === password);        if (user) {          state.currentUser = user; // Cập nhật người dùng hiện tại          state.isAuthenticated = true; // Đánh dấu đã đăng nhập          state.error = null;        } else {          state.error = 'Thông tin đăng nhập không đúng!';        }      },      logoutUser: (state) => {        state.currentUser = null; // Xóa người dùng hiện tại        state.isAuthenticated = false; // Đánh dấu đăng xuất      },    },  });  export const { registerUser, loginUser, logoutUser } = userSlice.actions;  export default userSlice.reducer; |

**Lưu ý : store và app giống các phần trên**

## **Trang đăng kí**

|  |
| --- |
| import { useDispatch, useSelector } from 'react-redux';  import { registerUser } from './userSlice';  const dispatch = useDispatch();    const users = useSelector((state) => state.users.users);    const handleRegister = () => {      // Kiểm tra xem email đã tồn tại chưa      const existingUser = users.find((user) => user.email === email);      if (existingUser) {        setError('Email đã được đăng ký!');      } else {        // Đăng ký người dùng mới        dispatch(registerUser({ email, password }));        setEmail('');        setPassword('');        setError('');        alert('Đăng ký thành công!');        navigation.navigate('Login'); // Chuyển sang màn hình đăng nhập      }    }; |

## **Trang đăng nhập**

|  |
| --- |
| import { useDispatch, useSelector } from 'react-redux';  import { loginUser } from './userSlice';  const dispatch = useDispatch();    const isAuthenticated = useSelector((state) => state.users.isAuthenticated);    const handleLogin = () => {      dispatch(loginUser({ email, password }));      if (isAuthenticated) {        alert('Đăng nhập thành công!');        navigation.navigate('Home'); // Chuyển sang màn hình chính sau khi đăng nhập thành công      } else {        setError('Thông tin đăng nhập không đúng!');      }  }; |

# **Đăng nhập và đăng kí với ReduxToolkit (api)**