Web Application Development
Module 5 QB Solutions

Part A

- **Explain the pillars of Redux?**

Redux is a predictable state container for JavaScript applications. It helps you write apps that behave consistently, run in different environments (client, server, and native), and are easy to test. Redux manages an application's state with a single global object called Store.

Pillars of Redux
These are Redux's main pillars:

Store
A store is an object that holds the application's state tree. There should only be a single store in a Redux app, as the composition happens at the reducer level.

- getState() returns the current state of the store.
- dispatch() dispatches an action. It is the only way to update the application state.
- subscribe() subscribes a change listener to the state.
- unsubscribe() is useful when you no longer want to call your listener method when the state changes.

Action
An action is a plain object that represents an intention to change the state. They must have a property to indicate the type of action to be carried out.

- Actions are payloads of information that send data from your application to your store.
- Any data, whether from UI events or network callbacks, needs to eventually be dispatched as actions.
- Actions must have a type field, indicating the type of action being performed.

Reducers
Reducers are pure functions that specify how the application's state changes in response to actions sent to the store.

- Actions only describe what happened, not how the application's state changes.
- A reducer is a function that accepts the current state and action, and returns a new state with the action performed.
- combineReducers() utility can be used to combine all the reducers in the app into a single index reducer which makes maintainability much easier.

- **Explain React + Redux with a suitable diagram.**

React Redux is the official React binding for Redux. It enables React components to read data

from a Redux store and send updates to the store. Redux helps apps scale by providing a sensible way to manage state through a unidirectional data flow model. React Redux is conceptually straightforward. It subscribes to the Redux store, checks to see if the data that the component wants has changed, and re-renders the component.

- **Give a code example to demonstrate embedding two or more components into one.**

Child1.js

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

class Child1 extends Component {
  render() {
    return (
      <li>
        This is a child component 1.
      </li>
    );
  }
}

export default Child1;
```

Child2.js

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

class Child2 extends Component {
  render() {
    return (
      <li>
        This is a Child component 2.
      </li>
    );
  }
}

export default Child2;
```

App.js

```
import React, { Component } from 'react';
import Child1 from './components/child1';
import Child2 from './components/child2';

class App extends Component {
  render() {
    return (
      <div>
        <div>This is a parent component</div>
        <Child1 />
        <Child2 />
      </div>
    );
  }
}

export default App;
```

- **Give a code example to modularize code in react.**

  Filename: index.js

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
// Importing default export
import File from "./DefaultExport";
// Importing named exports
import { NamedExport } from "./NamedExport";
ReactDOM.render(
    <React.StrictMode>
        <File />
        <NamedExport />
    </React.StrictMode>,
document.getElementById("root")
);
```

Filename:DefaultExport.js

```
import React from "react";
const DefaultExport = () => {
return (
    <div>
        <h1>This is from default export</h1>
        <h2>Hello Coders</h2>
    </div>
);
};
// Default export
export default DefaultExport;
```

Filename:NamedExport.js

```
import React from "react";
const NamedExport = () => {
   return (
      <div>
         <h1>This is from named export</h1>
         <h2>Nice to see you</h2>
      </div>
   );
};
// Named Export
export { NamedExport };
```

- **Apply the concept of redux with real time examples.**
  Redux **allows you to manage your app's state in a single place and keep changes in your app more predictable and traceable**. It makes it easier to reason about changes occurring in your app.
  A few real life examples are as follows:
- User Login:
  In this situation the program should be able to remember who has logged in. In these cases we can use so that the whole program will know the state of user login with other required details.
- Switch from light mode to dark mode:
  If a user wants to change from light mode to dark mode then the whole application will be notified that the user wants the website in dark mode thus changing the state will change the whole website.
- **How do you implement React routing (The extension isn't working work on the code later)**
  We can implement react routing with the help of two tags according to the latest version of React:
- Routes
- Route

```
import React, { Component } from "react";
import { Routes, Route } from 'react-router-dom';
class App extends Component {

 render() {
   return (
              <div>
              <Routes>
            <Route path = "/" element = {<HomePage />} />
            <Route path = "/cars" element = {<CarPage />} />
           <Route path = "/show" element = {<ShowPage />} />
```

```
        <Route path = "/signin" element = {<Auth />} />
    </Routes>
        </div>
          );
        }
      }
```

- **Web application without redux**
  Yes, you can build web applications without using Redux. Redux has nothing to do with React. They can work together, but it's not a rule of thumb. Although Redux is really helpful when managing states on large React applications, you can choose to use it or not.
- **Web Application with redux (Using Store and reducers)**
  Redux is a state management library that can be used to manage the state of any application, not just React. When using Redux, you will see that we may need to write more code to get the same things done. That is by design, but your application state is more manageable, and testing becomes easy.
  Three Pillars of Redux
- Store
- Action
- Reducer
- **List some of the cases when you should use Refs.**
- Managing focus, text selection, or media playback.
- Triggering imperative animations.
- Integrating with third-party DOM libraries.

Part B

- **What is an event in React?**
  An event is an action that could be triggered as a result of a user action or a system-generated event. For example, a mouse click, the loading of a web page, pressing a key, window resizing, and other interactions are called events. React has its own event handling system, which is very similar to handling events on DOM elements.

  The React event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event. Handling events with React has some syntactic differences from handling events on the DOM. These are:
  1. React events are named in camel case instead of lowercase.
  2. With JSX, a function is passed as the event handler instead of a string.
  3. In react, we cannot return false to prevent the default behavior. We must call preventDefault event explicitly to prevent the default behavior.
- **Draw a diagram showing how data flows through Redux.**

- **How will you distinguish Redux from Flux?**

Redux: Redux is a predictable state container for JavaScript apps. Redux is a library that can be used with any UI layer or framework, including React, Angular, Ember, and vanilla JS. Redux can be used with React; both are independent of each other. Redux is a state-managed library used in JavaScript apps. It simply manages the state of your application, or, in other words, it is used to manage the data of the application. It is used with a library like React.

Flux: Flux is the application architecture, or we can say JavaScript architecture, that is used for building client-side web applications or UI for client applications. You can begin using Flux without having to write a lot of new code. Flux overcomes the drawbacks of MVC, such as instability and complexity.

- **What are the advantages of using React?**
  1. A centralized state management system, i.e., store
  2. Performance Optimizations
  3. Pure reducer functions
  4. Storing long-term data
  5. Time-travel Debugging
  6. Great supportive community

- **Where would you put AJAX calls in your React code?**
  It is totally recommended that API calls be made in the componentDidMount() life cycle method.
- **You must've heard that "In React, everything is a component". What do you understand from that statement?**
  The building blocks of a React application's UI are called components. Any app UI created using React is divisible into a number of small, independent, and reusable pieces known as components. React renders each of the components independent of each other. Hence, there is no effect of rendering a component on the rest of the app's UI.
- **Can a browser read JSX?**
  Browsers can't read JSX because there is no inherent implementation for the browser engines to read and understand it. JSX is not intended to be implemented by the engines or browsers; it is intended to be used by various transpilers to transform this JSX into valid JavaScript code.
- **Define HOC in React. What are the benefits of HOC?**
  A higher-order component (HOC) is an advanced technique in React for reusing component logic. HOCs are not part of the React API per se. They are a pattern that emerges from React's compositional nature.
  Concretely, a higher-order component is a function that takes a component and returns a new component.

- **What is a store in Redux?**
  A store is an immutable object tree in Redux. A store is a state container that holds the application's state. Redux can have only a single store in your application. Whenever a store is created in Redux, you need to specify the reducer.
  The only way to change the state inside it is to dispatch an action on it.
  A store is not a class. It's just an object with a few methods on it. To create it, pass your root-reducing function to createStore.

- **What is an arrow function and how is it used in React?**
  With arrow functions, there is no binding for this. In regular functions, this keyword represented the object that called the function, which could be the window, the document, a button, or whatever. With arrow functions, this keyword always represents the object that defined the arrow function.
- **How is React different from React Native?**

- **How is React different from Angular?**
  React is a declarative, efficient, and flexible JavaScript library for building user interfaces. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook. React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It
  Create simple views for each state of your application, and React will efficiently update and render the appropriate component whenever your data changes. The declarative view makes your code more predictable and easier to debug. Angular is a popular open-source JavaScript framework created by Google for developing web applications. Front-end developers use frameworks like Angular or React for presenting and manipulating data efficiently. Updated Angular is much more efficient compared to the older version of Angular, especially since the core functionality was moved to different modules. That's why it becomes so much faster and smoother compared to the older one. newly added angular CLI. With that package, you can create a scaffolding for your Angular project.

- **What are the components of Redux?**
  STORE: A store is a place where the entire state of your application is listed. It manages the status of the application and has a dispatch (action) function. It is like a brain responsible for all the moving parts in Redux. ACTION: Action is sent or dispatched from the view, which are payloads that can be read by reducers. It is a pure object created to store the information of the user's event. It includes information such as the type of action, time of occurrence, location of occurrence, its coordinates, and which state it aims to change.
  REDUCER: Reducer reads the payloads from the actions and then updates the store via the state accordingly. It is a pure function to return a new state from the initial state.
- **What are pure components?**
  Pure components limit re-rendering when the component's re-rendering is unnecessary. Pure components are React-extended class components. PureComponent.
- **How would you create a form in React?**
  In React forms are created with the help of form tag from HTML.
  But in react we can use the state to store the data and use it to either send / push the data to the database or just to display even when the user refreshes the page.
  Example code:
  Import React from "react";

```
class NameForm extends React.Component {
  constructor(        ) {
    super(        );

this.       = {value: ''};

    this.                    = this.handleChange.bind(this);
```

```
        this.                    = this.handleSubmit.bind(this);
      }


handleChange(       ) {
  this.setState({value:         .       .       });
}

      handleSubmit(       ) {
        alert('A name was submitted: ' + this.       .       );
            .preventDefault();
      }

      render() {
        return (

  <form onSubmit={this.handleSubmit}>
          <label>


    <input type="text" value={this.state.value} onChange={this.handleChange} />
          </label>
           <input type="submit" value="Submit" />
         </form>
       );
      }
    }
```

- **What are the advantages of using Redux?**
  1. Centralized state management system i.e. Store 2. Performance Optimizations 3. Pure reducer functions
  4. Storing long-term data 5. Time-travel Debugging 6. PoGreat supportive community

- **What is Redux?**
  Redux is a predictable state container for JavaScript apps. As the application grows, it becomes difficult to keep it organized and maintain data flow. Redux solves this problem by managing the application's state with a single global object called Store. Redux fundamental principles help in maintaining consistency throughout your application, which makes debugging and testing easier.
- **How are forms created in React? (Repeated)**
- **What are the three principles that Redux follows?**
  Single source of truth
  The global state of your application is stored in an object tree within a single store.
  State is read-only
  The only way to change the state is to emit an action, an object describing what happened.
  Changes are made with pure functions
  To specify how the state tree is transformed by actions, you write "pure reducers."

- **What do you understand by “Single source of truth”?**
  Single source of truth

The global state of your application is stored in an object tree within a single store. Because the state from your server can be serialized and hydrated into the client with no additional coding, it is simple to build universal apps. A single state tree also makes it easier to debug or inspect an application; it also enables you to persist your app's state in development for a faster development cycle. Some functionality that has been traditionally difficult to implement—undo and redo, for example—can suddenly become trivial to implement if all of your states are stored in a single tree.

Part C

- **How are actions defined in Redux?**
- **Explain the role of Reducer.**
- **What are the advantages of Redux?**
- **What is a React Router?**
- **Why is the switch keyword used in React Router v4?**
- **Why do we need a Router in React?**
- **List down the advantages of React Router.**
- **What do you understand about Refs in React?**
- **How is React Router different from conventional routing?**
- **What is NPM?**
- **What is Prop Drilling and How can you avoid it?**
- **What is React Context?**
- **What is a React Map?**
- **What is React conditional rendering?**
- **How is React Router different from conventional routing? (repeated)**
- **Why Switch keyword used in React Router V4?**
- **How many ways can we style the React Component?**
- **What is the significance of stores in Redux?**
- **What are Redux Devtools?**
- **Are there any similarities between Redux and RxJs?**