Assignment 5
Owner - Mohit R

1. What is the difference between named export and default export and * as export?

- Import & Export Modules are ES6 features that allow us to write reusable js code (modules), which when exported, allows other modules to use it by importing.
- **Named Export:**
    - When want to export multiple things from a file we make use of Named Exports
    - We place the keyword export before the component/variable/object/module anything that we want to export.
    - There can be more than one named component in a file. While importing the Component name must be enclosed within {}.
    - Example: In config.js file, there are two named exports IMG_CDN and restaurantList.
    - While importing IMG_CDN in ReastaurantCard.js file, import {IMG_CDN} from "../config"; In Body.js file, while importing import {restaurantList} from "../config"; is used.
- **Default Export:**
    - Each component file can have only one export default. While importing just the Component name can be used (without any {})
    - Example: Check Body.js file inside components in this chapter. Body component is exported using export default Body;
    - In the app.js file, while importing import Body from "./components/Body";
- **Import Maps:**
    - While importing, the component can be given an alias name and then in that file, that component is referred to only using that alias name.
    - Example: Check Footer.js file inside components in this chapter. Footer Component is exported as named export using export const Footer = () => {}
    - In the app.js file, while importing use import {Footer as MainFooter} from "./components/Footer";. is the component name now, error is thrown if For using import map in default exported component, just use import NewComponentName from './components/Header';
- **import * as**
    - It is useful when we import all the components/modules functions as namespace object which contains all exports as properties.
    - Those exported components can be accessed by Module.component name
    - Example: Check Header.js file inside components in this chapter. There are three components exported.
    - In app.js, while importing use import * as MainHeader from '. /components/Header'. <MainHeader.Header></MainHeader.Header> is used to render Header Component in AppLayout Component.

2. What is the importance of config.js file

- config.js file can be used to store the hardcoded values in one file, so that when the value needs to be modified, it can be easy to do the modification in one file.
- Example: All API Base URLs, CDN links, config data from backend, default values needed in the app could be placed in config.js file.

3. What are React Hooks?

- React Hooks are a new addition to React from React 16.8 version.
- Earlier, state and other component features could be handled only using Class Components. But with version 16.8, React introduced a new pattern called Hooks.
- With React Hooks, we can use state, and other React features, in a functional component empowering functional programming in React.
- Hooks are JavaScript functions that manage the state's behaviour and side effects by isolating them from a component.
- React provides a bunch of standard in-built hooks like useState(), useEffect(), usecontext(), useReducer(), usecallback, useMemo(), useRef(), useLayoutEffect(), useDebugValue() and other additional hooks.

4. Why do we need useState() hook?

- useState() is one of the basic hooks functions which creates a state and assigns the initialState value passed in the parameter.
- UseState() creates a state variable and maintains the state of the component.
- It also provides a setState function, the state can be updated only using this function.
- const [state, setState] = useState(initialState);
- The setState function is used to update the state. It accepts a new state value and enqueues a re-render of the component.
- setState(newState)
- During subsequent re-renders, the first value returned by useState will always be the most recent state after applying updates.
- If we want to use the prev state value instead of the first value, we can pass a function to setState, it receives previous state and returns updated state.