Assignment

Page No.
Date

1. what is difference between Named Export, Default export and * as export

Named Export	Default Export	as Export

- multiples code can be exported
- For import we have to use { } bracket
ex-
import {Head c} from "./con/c"
Syntax for export
export const variable = "value";
export function fnaa() {
}
for ex-
export const variable = "value";
export function myfun() {
	only Single code export is possible
	- For import we use directly no bracket needed
	ex-
	import anything from "./"
	Syntax.
	Const variable = "value"
	export default variable;
		- allow to importing all exported entities as properties of single object
		it is used when we want to import everything from module without specifying each name individually.
		import * as model from "./.."
		Now we can access export identities using dot notation
		Console.log (model.variable)
		model.myfun()

SAGAR

Page No.

Date

Q what is the importance of Config.js file.

it is a file that contains ~~different~~ various setting, parameters and options used by an application or system.

- Environment - specific Configuration → The config.js file allow developers to store environment specific setting such as database credentials. By having separate configuration for different environments, it become easier to manage and deploy application in various setting.

Security:- Storing sensitive information such as passwords or access token in the Config.JS file help protect them from being exposed in source code or.

- the term "Config-JS" refer to a Javascript file that typically contains configuration setting and variable for application or system. The specific name & Config.js is not a standardized convention and can vary depend on developer or project.

In general, a config.js file is used to store various parameter and setting that control behaviour of application.

Page No.
Date

These setting can include environment specific configuration such as database, API keys that may affect application functionality.

- The content & structure of a config.js can vary based on specific needs of application & preference of the development team. Some projects may opt for a single config.js file that contain all configuration while others split configuration into multiple files or use different format such as JSON or YAML.

2) what are React Hooks?

React hook are functions that allow developers to use state and other React features in functional components. introduced in 16.8

React hook is the normal function at the end of day.
we get hook by from React by importing from react.
Every hook has specific function for it.

Some common hooks
(i) useState — it allow functional components to have their own internal state. it return a state variable and function to update that variable. with useState, we can handle & update states values

within function component:

useState return = [variable, function to be update]

or

```
Const searchbar = useState()
Const [searchtext, searchtextbar] = searchbar
```

or

```
Const [searchText, setSearchtext] = useStat("")
```

default value set

→ why do we need a useState Hook?

- useState is React Hook that let's u to add a state variable to your component

```
Const [state, setState] = useState(initialState)
```

useState - it enables functional components to have state allowing them to manage and update data over time.

parameters

- 'initialState' - The u value you want the state to be initially. it can be value of any type, but there is a special behaviour for function. This argument is ignored after the initial render.

SAGAR
Page No.
Date

- if you pass function as initial state, it will be treated as an initializer function. it should be pure, should take no arguments. React will call initializer function when initializing component & store its return value as initial state.

- Returns
  useState returns an array with exactly two values
  (1) the current state. During the first render it will match the initial state you have passed.

  (2) The set function that let's you update state to different value & trigger re-render

- Caveats
  - useState is a Hook, so we can call only it at top level of your component or your own Hooks. you can't call it inside loops or conditions. if we need that, extract a new component & move state into it