MERN Stack Assignment 1

1. **What exactly do you mean when you say "prop drilling," and how do you avoid it?**

   Prop Drilling is the process by which you pass data from one part of the React Component tree to another by going through other parts that do not need the data but only help in passing it around.

   React Context *provides us with a way to pass data through to components in our tree, without manually having to pass props down at every level.*

   There are 3 primary concepts behind React Context:

   - **Creating a context:** You can create a new type of context using the `React.createContext` API. For each value which you would like to be accessed by multiple components, you should make a new context. For example, you can create a `ThemeContext` for the application's theme:

     ```
     // Pass in the default value 'dark'
     const ThemeContext = React.createContext('dark');
     ```

   - **Providers:** Providers are used to pass a context value to the tree below it. Any component rendered inside the `Provider` can read it's value, no matter how deep it is in the tree. Here's an example:

     ```
     // Provide the theme value `dark` to components rendered inside
     // `MyComponent`.
     <ThemeContext.Provider value="dark">
       <MyComponent>
     </ThemeContext.Provider>
     ```

   - **Consumers:** Consumers are used to read the value of a context. These need a function as a child, which receives the current context value and returns a React node:

     ```
     <MyContext.Consumer>
       {theme => /* JSX here can use the theme context value */}
     </MyContext.Consumer>
     ```

2. **In React JS, how do you add validation to props?**

   Need of Validating Props in React JS: Props are used to passing the read-only attributes to React components. For the proper functioning of components and to avoid future bugs and

glitches it is necessary that props are passed correctly. Hence, it is required to use props validation for improving react component's performance.
React JS has an inbuilt feature for validating props data type to make sure that values passed through props are valid. React components have a property called *propTypes* which is used to setup data type validation.

Syntax: The syntax to use *propTypes* is shown below.

```
class Component extends React.Component {

  render() {}

}

Component.propTypes = {/* definition goes here*/};
```

3. Is it possible to use classes in NodeJS?

Yes, it is possible to use classes in Node JS. Classes are templates for creating objects. We can create classes in Node JS by using JavaScript prototype and ECMA Script.

4. What is the purpose of super(props)?

- **Super():** It is used to call the constructor of its parent class. This is required when we need to access some variables of its parent class.
- **Props:** It is a special keyword that is used in react stands for properties. Used for passing data from one component to another. Props data is read-only, which means that data coming from the parent should not be changed by child components.

  If we want to use *this* in the constructor, we need to pass it to super. If we want to use *this.props* inside the constructor we need to pass it with the super() function. Otherwise, we don't want to pass props to super() because we see *this.Props* are available inside the render function.

5. Why are the Express app and server separated?

Express 'app' and 'server' must be kept separate as by doing this, you will be separating the API declaration from the network related configuration which benefits in the below listed ways:

- It allows testing the API in-process without having to perform the network calls
- Faster testing execution
- Getting wider coverage metrics of the code
- Allows deploying the same API under flexible and different network conditions
- Better separation of concerns and cleaner code