**MATHEW CLINT M. BRAZAL**

1. Create a sample code snippet that demonstrates passing a callback function from a parent component down to its child(ren) component.

const { Component } = React;

class Parent extends Component {

constructor(props) {

super(props);

this.child = React.createRef();

}

onClick = () => {

this.child.current.getAlert();

};

render() {

return (

<div>

<Child ref={this.child} />

<button onClick={this.onClick}>Click</button>

</div>

);

}

}

class Child extends Component {

getAlert() {

alert('getAlert from Child');

}

render() {

return <h1>Hello</h1>;

}

}

ReactDOM.render(<Parent />, document.getElementById('root'));

2. Explain props-drilling.

Props are the data we pass or can access from the top-level components to any number of child components on our website. When working with small applications, prop drilling can serve as a fast and easy method of data transfer between components. Unlike other common methods of data transfer, prop drilling is relatively easy to learn and implement. In addition to this, data passed as props can easily be updated on state change to reflect the new changes. Prop drilling does have its downsides, and in some cases it isn’t worth it. As your codebase increases, prop drilling can make your code overly complicated, and this can only get worse with more additions. In addition to this, props can be passed down to components that don’t necessarily need it just for the data to get to the child component, leading to an unnecessary increase in the code base.