Day 1 Session 1:

Step 1:

App.js

Delete all the existing code

```
import React from 'react';
import './App.css';
function App() {
return <h1>Hello, World</h1>;
}
export default App;
```

Step 2:

```
import React from 'react';
import './App.css';
function App() {
return
(
<h1>Hello, World</h1>
<p>Welcome to VIVA Institute of Technology</p>
)
}
export default App;
```

This will give an error as we are adding multiple lines in return.

Since the JSX spans multiple lines, you'll need to wrap the expression in parentheses.

When you return JSX from a function or statement, you must return a single element. That element may have nested children, but there must be a single top-level element. In this case, you are returning two elements.

The fix is a small code change. Surround the code with an empty tag. An empty tag is an HTML element without any words. It looks like this: <> .

```
import React from 'react';
import './App.css';
function App() {
```

```
return (
 <>
<h1>Hello, World</h1>
<p>Welcome to VIVA Institute of Technology</p>
</>
)
}
export default App;
```

Another Fix for the problem can be specifying the div tag.

```
import React from 'react';
import './App.css';
function App() {
return (
 <div>
<h1>Hello, World</h1>
<p>Welcome to VIVA Institute of Technology</p>
</div>
)
}
export default App;
```

Step 3:

Create a new App1.js file and relink everything

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App1 from './App1';
import reportWebVitals from './reportWebVitals';
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <React.StrictMode>
   <App1 />
 </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

App1.js

```
import React from 'react';
import './App.css';
function App1() {
return (
 <div>
<h1>Hello, World</h1>
<p>Welcome to VIVA Institute of Technology and MET</p>
</div>
);
}
export default App1;
```

Step 4: Adding Styling to an Element with Attributes

Change the contents of App.css

```
.container {
 display: flex;
 flex-direction: column;
 align-items: center;
 }
```

Change the contents ofApp1.js

```
import React from 'react';
import './App.css';
function App1() {
return (
 <div className='container'>
<h1>Hello, World</h1>
<p>Welcome to VIVA Institute of Technology and MET</p>
</div>
);
}
export default App1;
```

Points to Remember:

1) When you return JSX from a function or statement, you must return a single element. That element may have nested children, but there must be a single top-level element.
2) since JSX is JavaScript, it has a few limitations. One of the limitations is that JavaScript has reserved keywords.
3) That means you can't use certain words in any JavaScript code. For example, you can't make a variable called null because that word is already reserved.
4) One of the reserved words is class . React gets around this reserved word by changing it slightly. Instead of adding the attribute class , you will add the attribute className .
5) JSX looks like standard markup, but the advantage of JSX is that even though it looks like HTML, it has the power of JavaScript
6) That means you can assign variables and reference them in your attributes. To reference an attribute, wrap it with curly braces—{}—instead of quotes

Step 5: Adding Variables to your code

```
import React from 'react';
import './App.css';
function App1() {
   const gret="Greetings"
return (
 <div className='container'>
<h1 id={gret}>Hello, World</h1>
```

```
<p>Welcome to VIVA Institute of Technology and MET</p>
</div>
);
}
export default App1;
```