📘 Worksheet 2: React Native Components

🎯 Learning Outcomes

After completing this worksheet, students will be able to:

1. Create **Dumb (Presentational) Components** in React Native
2. Create **Smart Components** that manage data and pass props
3. Use **props** to send data between components
4. Use **useState** to manage component state
5. Detect **button press events** with `onPress`

Step 0 – Create a New Project

Open your Terminal and run:

```
npx create-expo-app rn-components
cd rn-components
npx expo start
```

Step 1 – Clean App.js and Set Up the Basic Structure

Delete everything in `App.js`, then insert the following code (Fill in each blank according to its number):

```
import { ___[1]___ , ___[2]___ } from 'react-native';

export default function App() {
  return (
    <   [3]    >
      <   [4]    >Hello</___[4]___>
    </___[3]___>
  );
}
```

Step 2 – Create a Dumb Component

Add the following code **above** the App component. Fill the blanks as numbered:

```
function DisplayMessage({ ___[5]___ }) {
  return (
    <Text>
      Message: ___[6]___
    </Text>
  );
}
```

Now call this component from inside App that the value is "Hello WU"

```
<DisplayMessage message="___[7]___" />
```

Blanks to fill:

Step 3 – Add a Smart Component (Using useState)

Import useState at the top:

```
import { useState } from 'react';
```

Inside the App component, add the state declaration:

```
const [message, setMessage] = useState(___[8]___);
```

Send the state value to the DisplayMessage component:

```
<DisplayMessage message={___[9]___} />
```

Step 4 – Add Buttons to Detect Press Events

Insert two buttons and fill the missing parts:

```
<Button
  title="SHOW A"
  onPress={() => setMessage(___[10]___)}
/>

<Button
  title="SHOW B"
  onPress={() => setMessage(___[11]___)}
/>
```

Step 5 – Full Code Structure with Numbered Blanks

Below is the full App.js with all blanks included for students to fill:

```
import { useState } from 'react';
import { ___[1]___ , ___[2]___ , Button } from 'react-native';

// Dumb Component
function DisplayMessage({ ___[5]___ }) {
  return <Text>Message: ___[6]___</Text>;
}

// Smart Component
export default function App() {
  const [message, setMessage] = useState(___[8]___);

  return (
    <___[3]___ style={{ padding: 40 }}>
      <DisplayMessage message={___[9]___} />

      <Button
        title="SHOW A"
        onPress={() => setMessage(___[10]___)}
      />

      <Button
        title="SHOW B"
        onPress={() => setMessage(___[11]___)}
      />
    </___[3]___>
  );
}
```

Step 6 – Expected Output

Your application should behave as follows:

- Initially, the displayed message is empty (`""`) or the chosen default
- Press **SHOW A** → Display: *Message: A*
- Press **SHOW B** → Display: *Message: B*

Reflection Questions

1. What is the main difference between a Dumb Component and a Smart Component?
   Answer: ____
2. Why do we need `useState` in a Smart Component? **Answer:**
   ____
3. Explain how `onPress={() => setMessage("A")}` works. **Answer:**
   ____