Hands-on Activity 14.1 Your First Animation using Expo	
Course Code: CPE026	**Program:** Computer Engineering
Course Title: Emerging Technologies 3 in CpE	**Date Performed:** November 16, 2024
Section: CPE41S8	**Date Submitted:** November 30, 2024
Name: Alferos, Joshua L. Efa, Christian Ed B.	**Instructor:** Engr. Roman Richard

1. Objectives

After this module, the students should be able to:

- Demonstrate the inclusion of animation in different aspects of the mobile application.

2. Intended Learning Outcome

After this module, the students should be able to:

- Demonstrate the use of geolocation on a mobile application built in React Native through Expo location.

3. Discussion

The discussion for this activity can be found here: 8.1 Animation using Expo

4. Materials and Equipment

- Nodejs LTS
- Visual Studio Code
- Emulator/Simular for Android/iOS

5. Procedure

Part 1: Using an Animated Component

First, Import the necessary library to create an Animated object.

```
import Animated from 'react-native-reanimated';
```

This Animated object is implemented on React Native built-in components that we've used before such as View, ScrollView or FlatList. For this part, we are still using the same components, but we have them wrapped by the Animated object (Animated.View).

Sample Code:

```
import Animated from 'react-native-reanimated';

export default function App() {
  return (
    <Animated.View
      style={{
        width: 100,
```

```
      height: 100,
      backgroundColor: 'violet',
    }}
  />
);
}
```

Task:

1. Create an application with the sample code. What happens? Provide a screenshot and observation.

Part 2: Defining and Using a Shared Value

A shared valueLinks to an external site. is a driving factor of all your animations. Basically, when you use a shared value, the data stored in it is automatically synchronized between the JavaScript thread and the UI thread. This is used through the useSharedValue hook. You can think of it as a React state which is automagically kept in sync between the “JavaScript” and the “native” side of your app.

Example:

```
import { useSharedValue } from 'react-native-reanimated';
```

As with any other React hookLinks to an external site., you need to define it in your component's body (it can be a value of any type). In a shared value, you can store any JS value like number, string or boolean but also data structures such as array and object.

In this code sample, use 100 as the default value of the useSharedValue hook and pass the returned value as an inline style of the Animated.View

```
import Animated, { useSharedValue } from 'react-native-reanimated';

export default function App() {
  const width = useSharedValue(100);

  return (
    <Animated.View
      style={{
        width,
        height: 100,
        backgroundColor: 'violet',
      }}
    />
  );
}
```

Tasks:

1. Run the code by modifying the initial application from part 1. What changed? Change the value of the animation hook for the Animation.View component. Note your observations.
2. For this section, a very simple animtion will be implemented to modify the width of an element (expanding by 50px on each press). To do this, we must access and modify the shared value

through the .value prop. We just do this over and over for each operation.

We will create a function inside called handlePress that will modify this shared value.

```
import { Button, View } from 'react-native';
import Animated, { useSharedValue } from 'react-native-reanimated';

export default function App() {
  const width = useSharedValue(100);

  const handlePress = () => {
    width.value = width.value + 50;
  };

  return (
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Animated.View
        style={{
          width,
          height: 100,
          backgroundColor: 'violet',
        }}
      />
      <Button onPress={handlePress} title="Click me" />
    </View>
  );
}
```

Tasks:

1. Modify the code so that you start with a width of 500, and every press will decrease the shared value by 10. Show screenshots.
2. Modify the code so that you start with 100 default value, and every press will increase the shared value by 25. Show screenshots.

Part 3: Using an Animation Function

Finally, import withSpring function and wrap around width.value + 50 in the handlePress function so that the value which withSpring returns modifies the shared value. This will create a bouncy spring animation that transitions the width of the element from its current value (here width.value) to the new one (here width.value + 50).

```
import { Button, View } from 'react-native';
import Animated, { useSharedValue, withSpring } from 'react-native-reanimated';

export default function App() {
  const width = useSharedValue(100);

  const handlePress = () => {
    width.value = withSpring(width.value + 50);
```

```
  };

  return (
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Animated.View
        style={{
          width,
          height: 100,
          backgroundColor: 'violet',
        }}
      />
      <Button onPress={handlePress} title="Click me" />
    </View>
  );
}
```

Tasks:

1. Show your output so far. Note all your observations.

To Summarize

We learned about Animated components, shared values and how to use them to create a simple animation. To sum up:

Animated components are used to define animatable elements.
Shared values are a driving factor of all animations and we define them using a useSharedValue hook.
Shared values are always accessed and modified by their .value property (eg. sv.value = 100;).
To create smooth animations modify shared values using animation functions like withTiming

7. Output

ALFEROS
Part 1: Using an Animated Component

Task:

1. **Create an application with the sample code. What happens? Provide a screenshot and observation.**

Code	Output

```
animation > animation > JS App.js > App
import Animated from 'react-native-reanimated';

export default function App() {
  return (
    <Animated.View
      style={{
        width: 100,
        height: 100,
        backgroundColor: 'violet',
      }}
    />
  );
}
```

Observation: The output is a single square entity that is color violet.

Part 2: Defining and Using a Shared Value
Tasks:

1. **Modify the code so that you start with a width of 500, and every press will decrease the shared value by 10. Show screenshots.**
2. **Modify the code so that you start with 100 default value, and every press will increase the shared value by 25. Show screenshots.**

Code	Output

1.

animation > animation > App.js > App > justifyContent

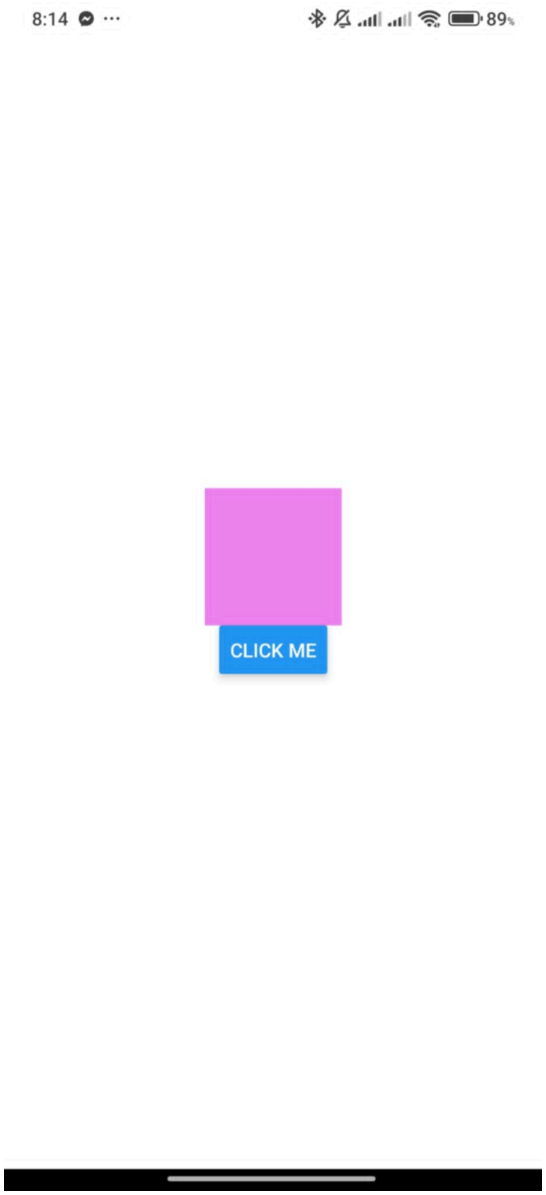
```
import Animated, { useSharedValue, useAnimatedStyle } from 'react
import { Button, View } from 'react-native';

export default function App() {
  const width = useSharedValue(500);

  const handlePress = () => {
    width.value = width.value - 10;
  };

  const animatedStyle = useAnimatedStyle(() => {
    return {
      width: width.value,
      height: 100,
      backgroundColor: 'violet',
    };
  });

  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems:
      <Animated.View style={animatedStyle} />
      <Button onPress={handlePress} title="CLICK ME" />
    </View>
  );
}
```

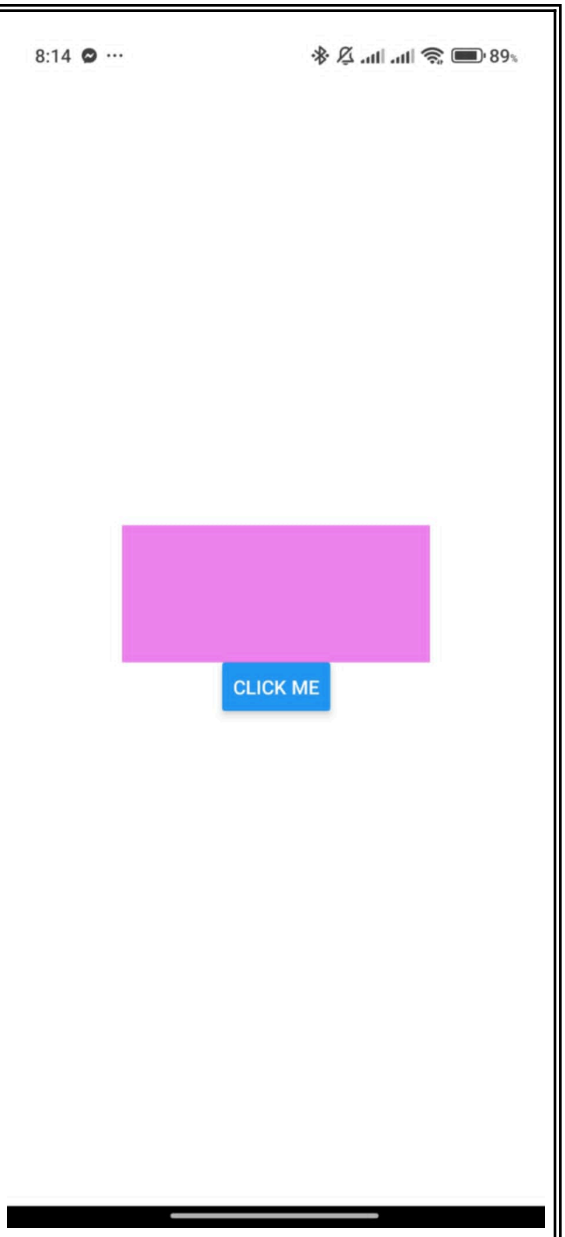

8:13
89%
CLICK ME

2.

animation > animation > JS App.js > ...

```
1  import Animated, { useSharedValue, useAnimatedStyle } from 'react
2  import { Button, View } from 'react-native';
3
4  export default function App() {
5    const width = useSharedValue(100);
6
7    const handlePress = () => {
8      width.value = width.value + 25;
9    };
10
11   const animatedStyle = useAnimatedStyle(() => {
12     return {
13       width: width.value,
14       height: 100,
15       backgroundColor: 'violet',
16     };
17   });
18
19   return (
20     <View style={{ flex: 1, justifyContent: 'center', alignItems:
21       <Animated.View style={animatedStyle} />
22       <Button onPress={handlePress} title="CLICK ME" />
23     </View>
24   );
25 }
```



Part 3: Using an Animation Function
Tasks:

1. Show your output so far. Note all your observation

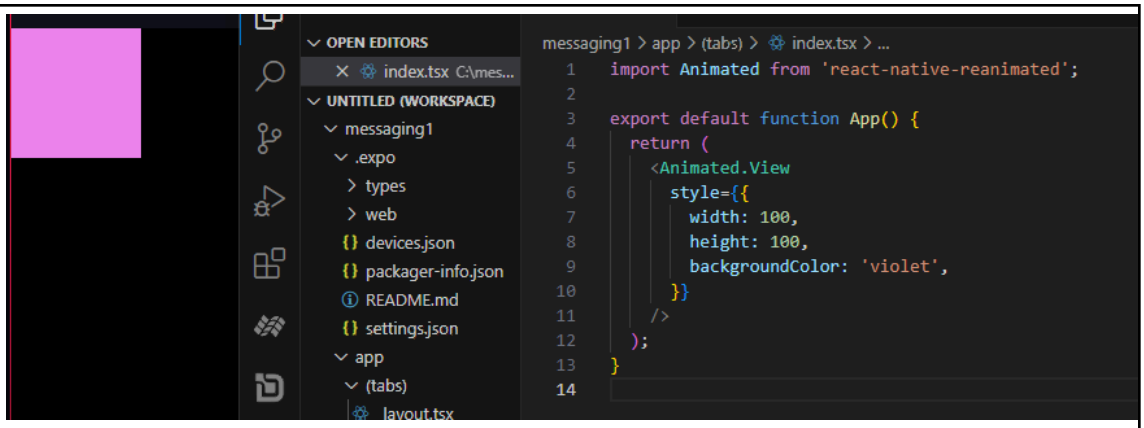
Code	Output

```
animation > animation > JS App.js > App
import { Button, View } from 'react-native';
import Animated, { useSharedValue, withSpring } from 'react-nativ

export default function App() {
  const width = useSharedValue(100);

  const handlePress = () => {
    width.value = withSpring(width.value + 50);
  };

  return (
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Animated.View
        style={{
          width,
          height: 100,
          backgroundColor: 'violet',
        }}
      />
      <Button onPress={handlePress} title="Click me" />
    </View>
  );
}
```

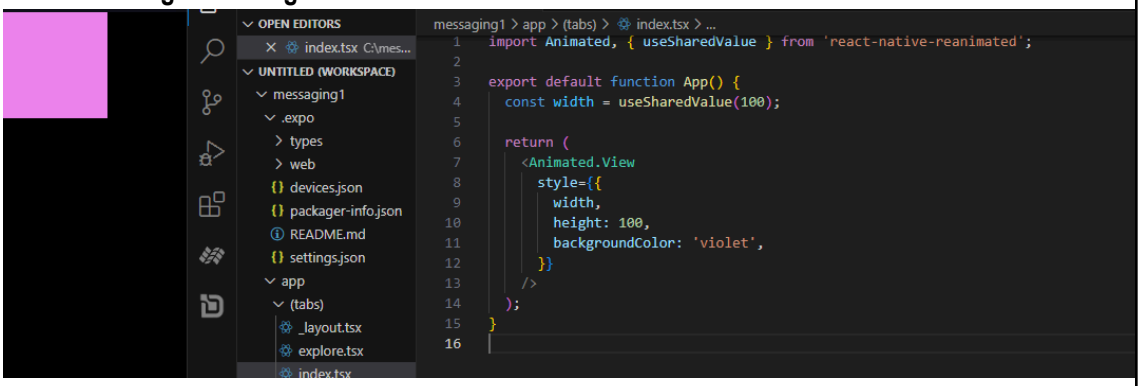
Observation: After adding withSpring, the animation on the box is much smoother compared to previous procedures.

EFA

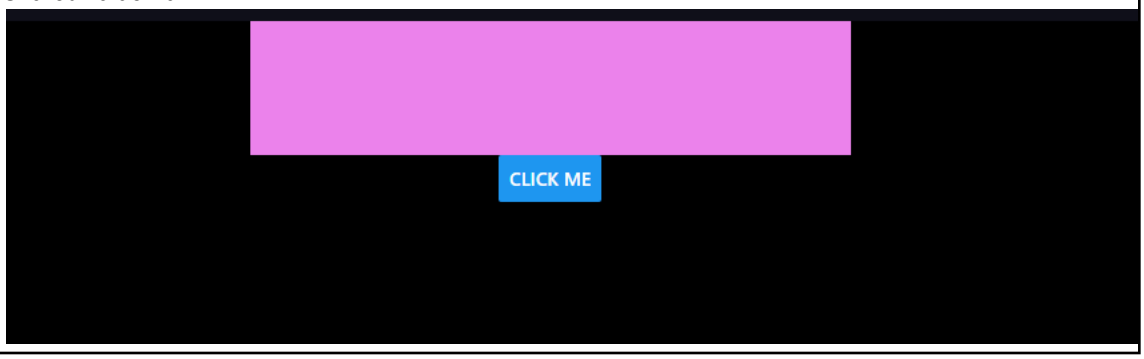
Part 1 Using an Animated Component



Part 2 Defining and Using a Shared Value

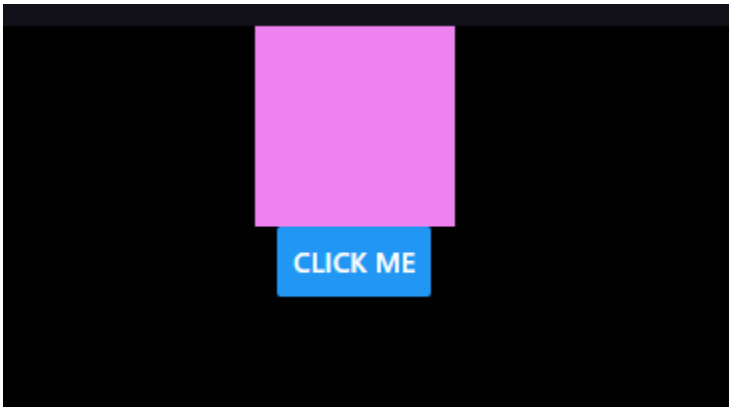


Task:
500 width
Shared value 10



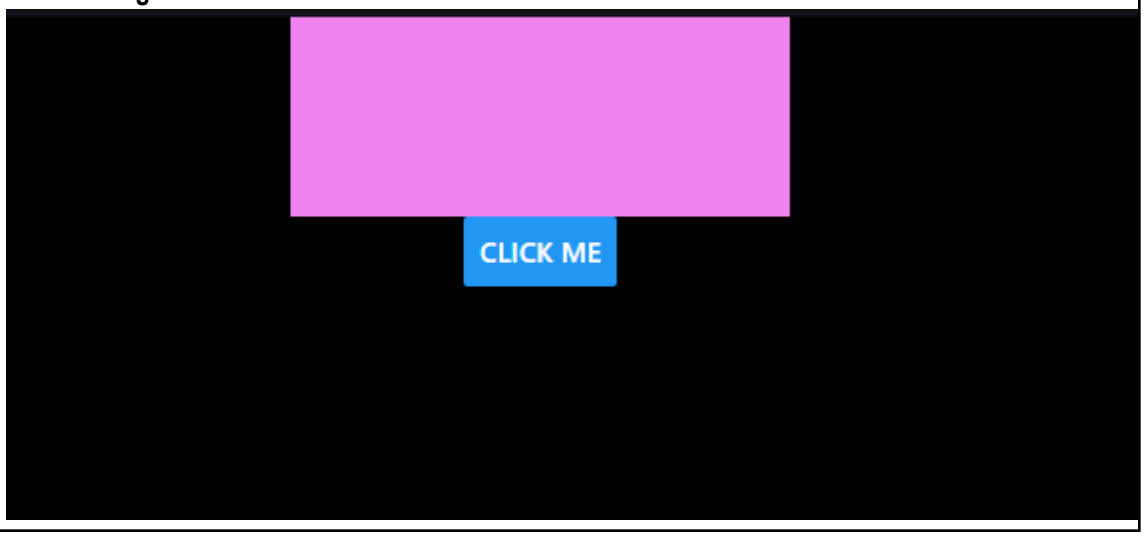
```
saging1 > app > (tabs) > index.tsx > App
import { Button, View } from 'react-native';
import Animated, { useSharedValue } from 'react-native-reanimated';

export default function App() {
  const width = useSharedValue(500);

  const handlePress = () => {
    width.value = width.value - 10; // Decrease by 10 on each press
  };


  return (
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Animated.View
        style={{
          width,
          height: 100,
          backgroundColor: 'violet',
        }}
      />
      <Button onPress={handlePress} title="Click me" />
```

100 width
25 shared value



```
import { Button, View } from 'react-native';
import Animated, { useSharedValue } from 'react-native-reanimated';

export default function App() {
  const width = useSharedValue(100);

  const handlePress = () => {
    width.value = width.value - 25; // Decrease by 10 on each press
  };


  return (
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Animated.View
        style={{
          width,
          height: 100,
          backgroundColor: 'violet',
        }}
      />
      <Button onPress={handlePress} title="Click me" />
```

Part 3: Using an Animation Function



```
import { Button, View } from 'react-native';
import Animated, { useSharedValue, withSpring } from 'react-native-reanimated';

export default function App() {
  const width = useSharedValue(100);

  const handlePress = () => {
    width.value = withSpring(width.value + 50); // Adds a spring animation effect
  };

  return (
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Animated.View
        style={{
          width,
          height: 100,
          backgroundColor: 'violet',
        }}
      />
      <Button onPress={handlePress} title="Click me" />
    </View>
  );
}
```

OBSERVATION :

- **The each different parts has different kinds of animation. When the button is pressed, the width will change smoothly with a bouncy animation due to withSpring. It will feel like an elastic effect, making the UI look more interactive.**

8. Supplementary Activity

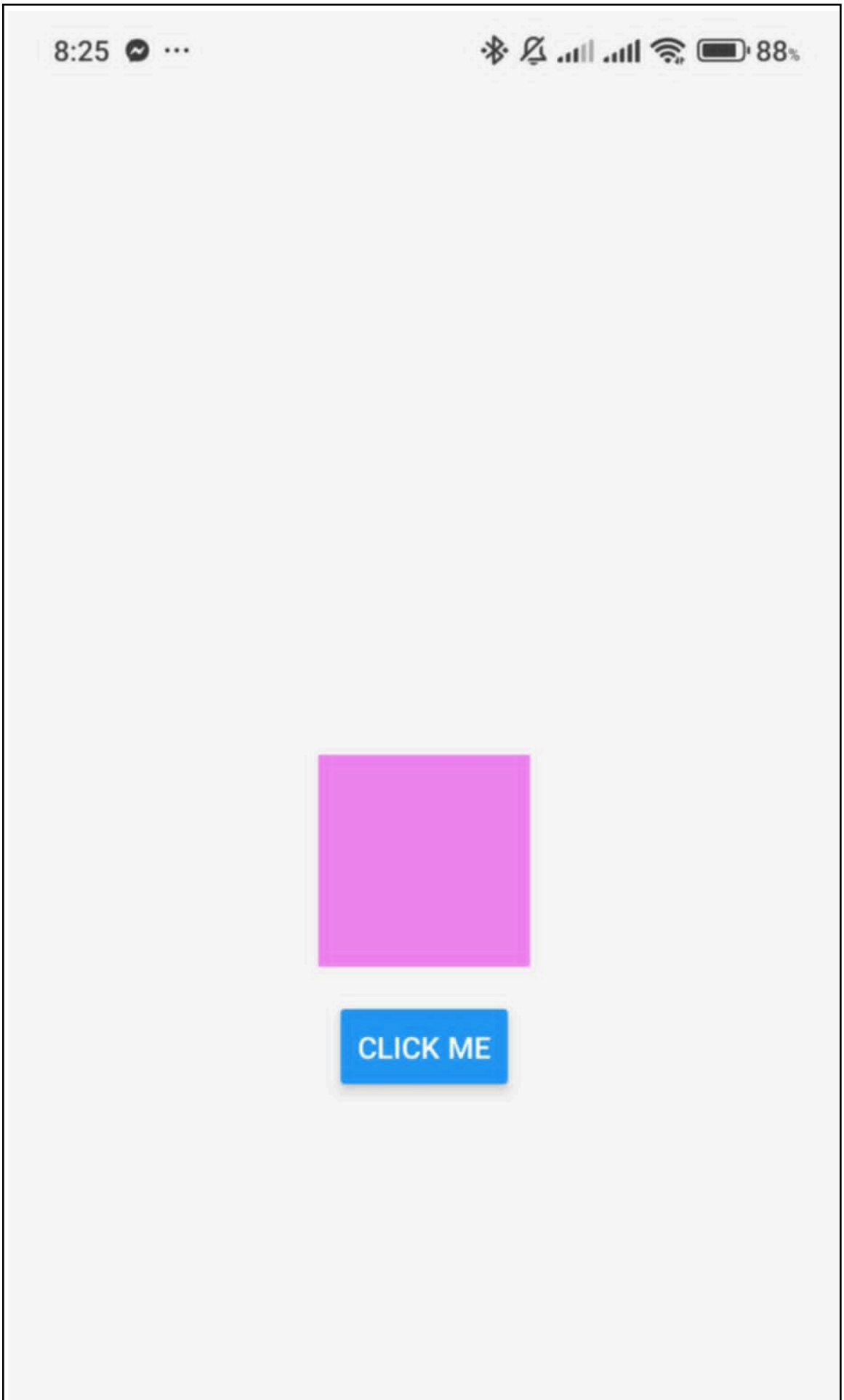
For this activity, you must include different animation styles to your mobile application.

1. Choose any component you have in your mobile application and use the reanimated expo package to add any animation.
2. Include screenshots of the code and the application to clearly demonstrate the implementation of the animation.
3. Screen record a short 15 second video and include a link in this submission.

ALFEROS

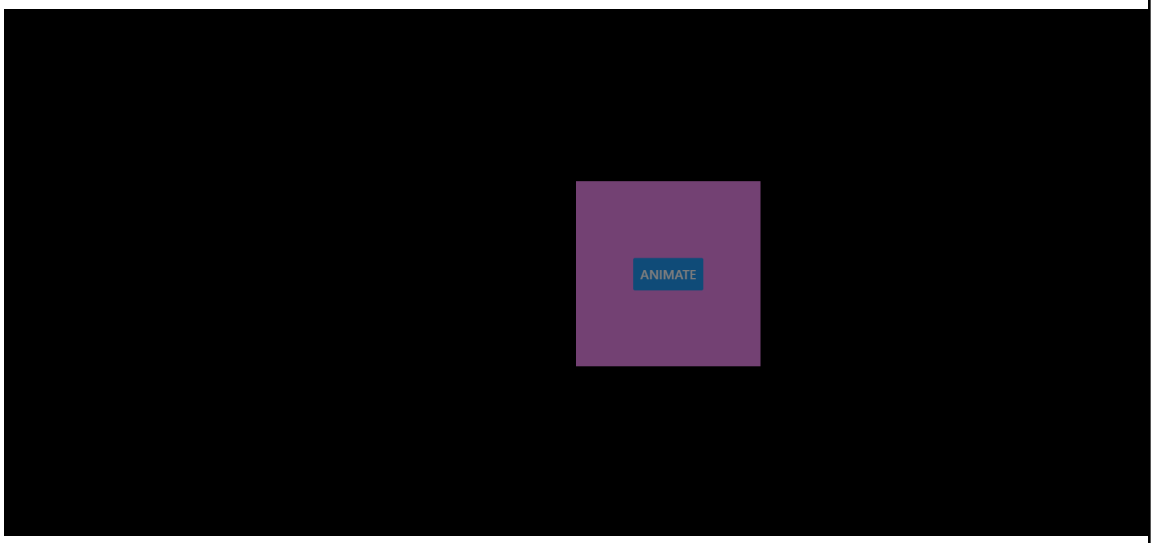
imation > animation > JS App.js > styles > container > flex

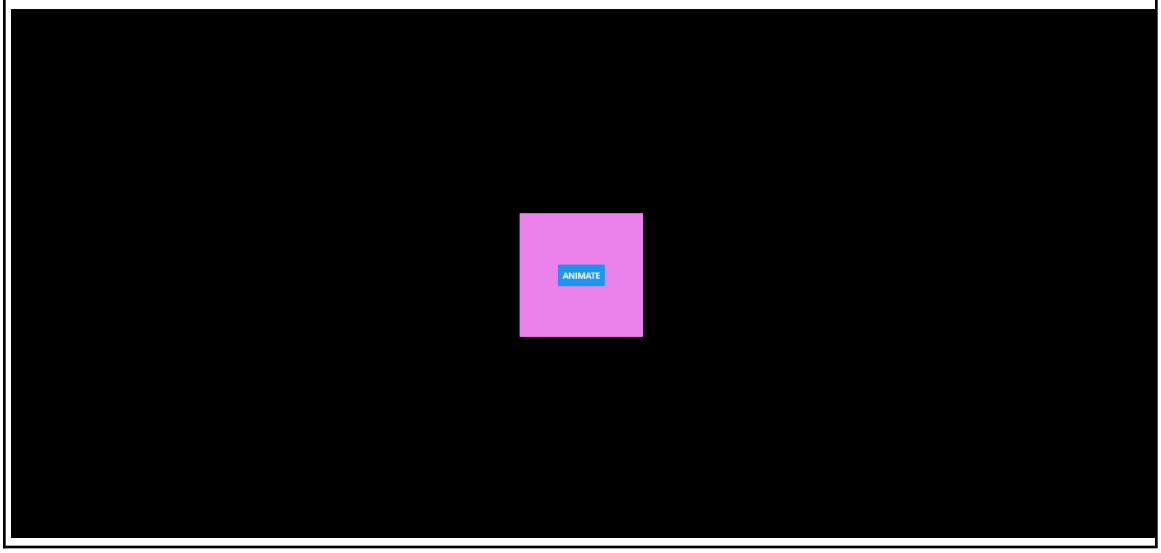
```
import { Button, View, StyleSheet } from 'react-native';
import Animated, { useSharedValue, useAnimatedStyle, withSpring }

export default function App() {
  const width = useSharedValue(100);

  const handlePress = () => {
    width.value = withSpring(width.value + 50, { damping: 8, stif
  };

  const animatedStyle = useAnimatedStyle(() => {
    return {
      width: width.value,
      height: 100,
      backgroundColor: 'violet',
    };
  });

  return (
    <View style={styles.container}>
      <Animated.View style={[animatedStyle, styles.animatedBox]}
      <Button onPress={handlePress} title="Click me" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#f5f5f5',
  },
  animatedBox: {
    marginBottom: 20,
  },
});
```


8:25
88%
CLICK ME

https://drive.google.com/drive/folders/17xwrDPbFltGCqh9R0no6MUTp-YchjrYT?usp=sharing

EFA





```
import React, { useState } from 'react';
import { Button, View } from 'react-native';
import Animated, { useSharedValue, withSpring, withTiming, withDelay } from 'react-native-reanimated';

export default function App() {
  const [animated, setAnimated] = useState(false);
  const scale = useSharedValue(1);
  const opacity = useSharedValue(1);
  const translateX = useSharedValue(0);

  const handlePress = () => {
    // Change the animation state on button press
    setAnimated(!animated);

    // Apply animations
    scale.value = withSpring(animated ? 1 : 1.5);  // Scale animation
    opacity.value = withTiming(animated ? 1 : 0.5, { duration: 500 });  // Fade animation
    translateX.value = withSpring(animated ? 0 : 100);  // Move horizontally
  };

  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Animated.View
        style={{
          width: 200,
          height: 200,
          backgroundColor: 'violet',
          justifyContent: 'center',
          alignItems: 'center',
          opacity: opacity,
          transform: [{ scale: scale }, { translateX: translateX }],
        }}
      >
        <Button onPress={handlePress} title="Animate" />
      </Animated.View>
    </View>
```

[VIDEO LINK]

6. Assessment Rubric

Lab Activity Rubric

Criteria	Ratings						Pts
SO 7 PI 1 **Student Outcome 7.1** Acquire and apply new knowledge from outside sources. threshold: 4.8 pts	6 pts Excellent | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently and applies knowledge learned into practice	5 pts Good | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently	4 pts Satisfactory | Look beyond classroom requirements, showing interest in pursuing knowledge independently	3 pts Unsatisfactory | Begins to look beyond classroom requirements, showing interest in pursuing knowledge independently	2 pts Poor | Relies on classroom instruction only	1 pts Very Poor | No initiative or interest in acquiring new knowledge	6 pts
SO 7 PI 2 **Student Outcome 7.2** Learn independently threshold: 4.8 pts	6 pts Excellent | Completes an assigned task independently and practices continuous improvement	5 pts Good | Completes an assigned task without supervision or guidance	4 pts Satisfactory | Requires minimal guidance to complete an assigned task	3 pts Unsatisfactory | Requires detailed or step-by-step instructions to complete a task	2 pts Poor | Shows little interest to complete a task independently	1 pts Very Poor | No interest to complete a task independently	6 pts
SO 7 PI 3 **Student Outcome 7.3** Critical thinking in the broadest context of technological change threshold: 4.8 pts	6 pts Excellent | Synthesizes and integrates information from a variety of sources; formulates a clear and precise perspective; draws appropriate conclusions	5 pts Good | Evaluate information from a variety of sources; formulates a clear and precise perspective.	4 pts Satisfactory | Analyze information from a variety of sources; formulates a clear and precise perspective.	3 pts Unsatisfactory | Apply the gathered information to formulate the problem	2 pts Poor | Gather and summarized the information from a variety of sources but failed to formulate the problem	1 pts Very Poor | Gather information from a variety of sources	6 pts
SO 7 PI 4 **Student Outcome 7.4** Creativity and adaptability to new and emerging technologies threshold: 4.8 pts	6 pts Excellent | Ideas are combined in original and creative ways in line with the new and emerging technology trends to solve a problem or address an issue.	5 pts Good | Ideas are creative and adapt the new knowledge to solve a problem or address an issue	4 pts Satisfactory | Ideas are creative in solving a problem, or address an issue	3 pts Unsatisfactory | Shows some creative ways to solve the problem	2 pts Poor | Shows initiative and attempt to develop creative ideas to solve the problem	1 pts Very Poor | Ideas are copied or restated from the sources consulted	6 pts
							Total Points: 24