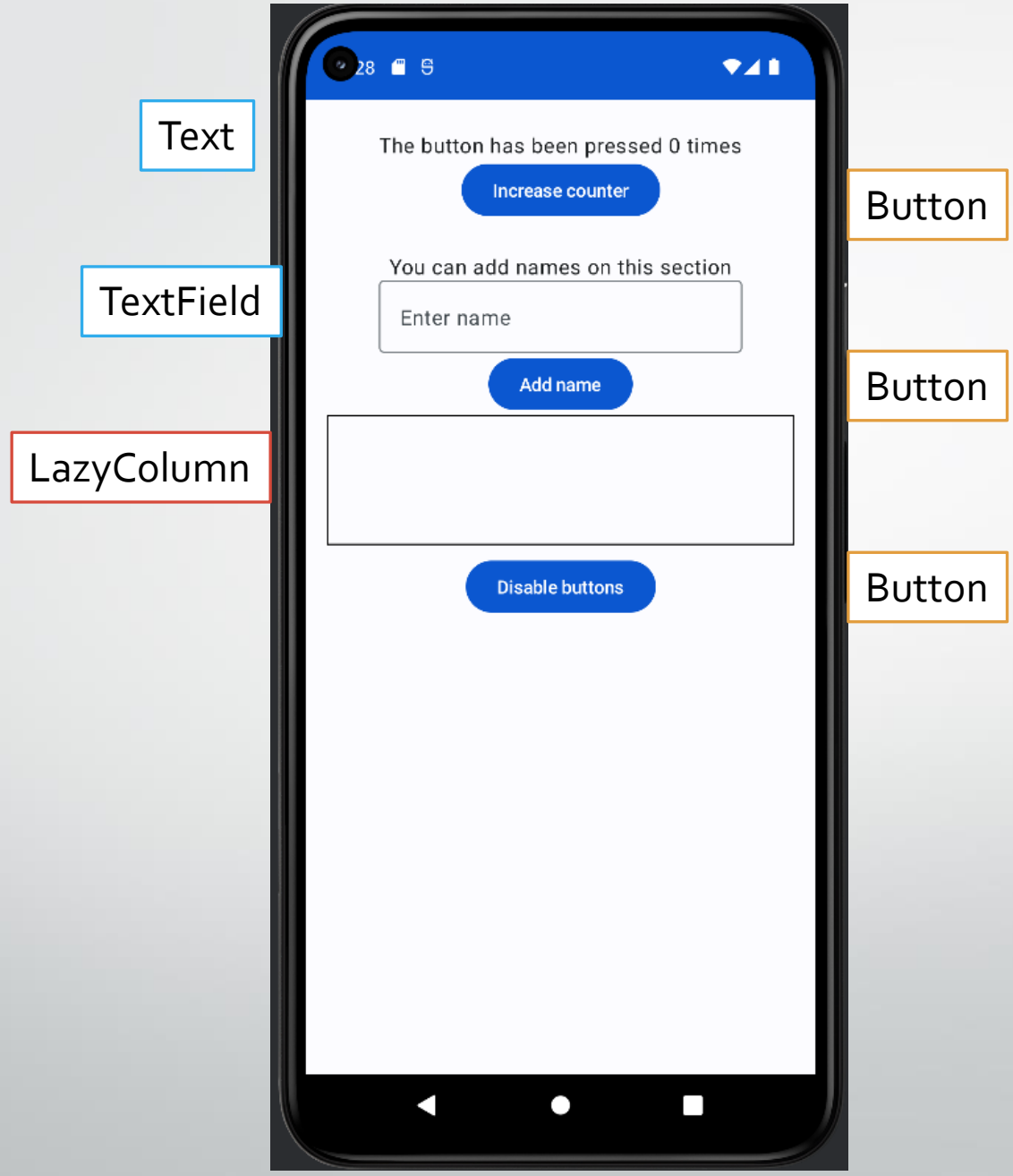
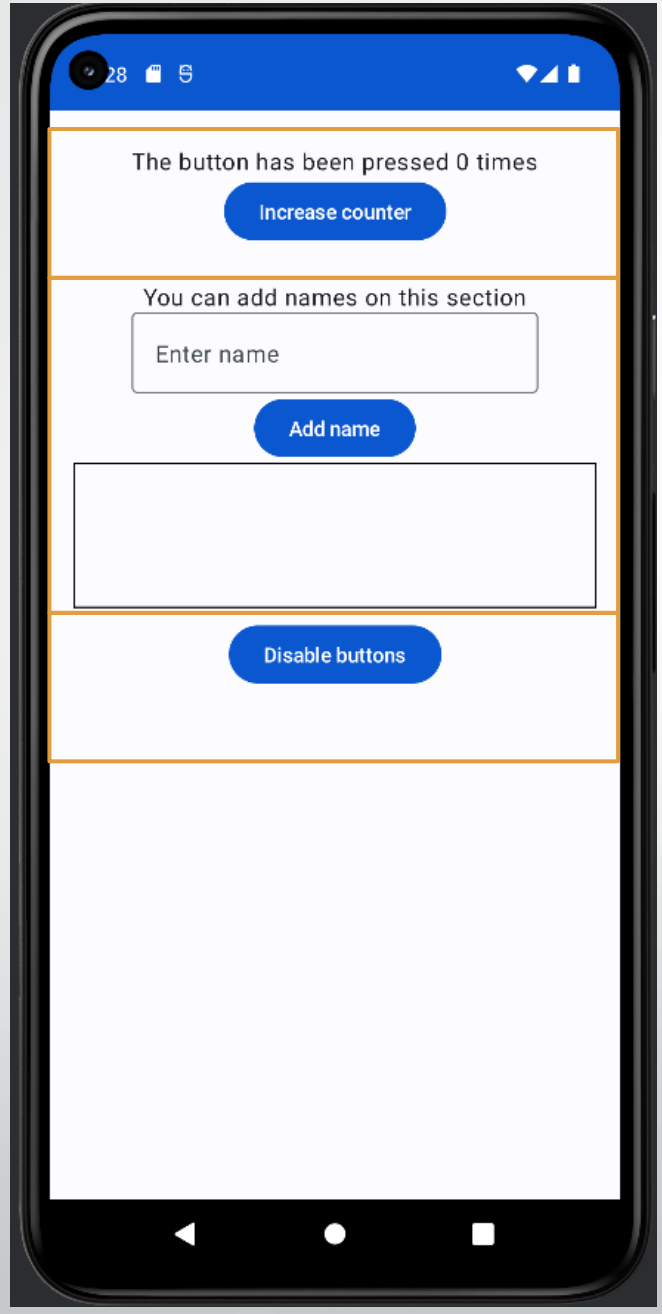
Jetpack Compose

Buttons, LazyColumn, AlertDialog andTextField


Text
TextField
LazyColumn
The button has been pressed 0 times
Increase counter
You can add names on this section
Enter name
Add name
Disable buttons
Button
Button
Button


The button has been pressed 0 times
Increase counter
You can add names on this section
Enter name
Add name
Disable buttons

1- Contador

2- Función que aumente el contador

3- Funcionalidad del botón

Remember

- remember can be used to store both mutable and immutable objects. Note: remember stores objects in the Composition, and forgets the object when the composable that called remember is removed from the Composition.

```
val pressedCounter: MutableState<Int> = remember {
    mutableStateOf( value: 0)
}
```

```
val buttonsScope = remember {
    buttonsStatus
}
```

```
val nameListScope = remember { nameList }
```

```
val name: MutableState<String> = remember { mutableStateOf( value: "") }
```

Dummy Data

```
val nameList: MutableList<String> = mutableListOf()
```

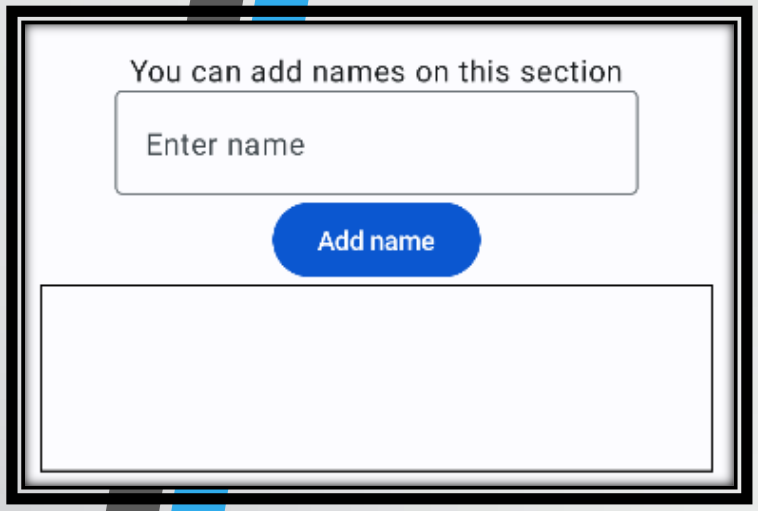
```
var buttonsStatus: MutableState<Boolean> = mutableStateOf( value: true)
```


```
Column(
    modifier = Modifier.padding(16.dp),
    horizontalAlignment = Alignment.CenterHorizontally
)
```

```
Text(text = "The button has been pressed ${pressedCounter.value} times")
```

```
Button(
    enabled = buttonsScope.value,
    onClick = {
        pressedCounter.value++
    }

) {
    Text(text = "Increase counter")
}
```

```
@Composable
fun ModifyTextComponent() {


    val pressedCounter: MutableState<Int> = remember {
        mutableStateOf( value: 0)
    }

    val buttonsScope = remember {
        buttonsStatus
    }
    Column(
        modifier = Modifier.padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    )
    {
        Text(text = "The button has been pressed ${pressedCounter.value} times")
        Button(
            enabled = buttonsScope.value,
            onClick = {
                pressedCounter.value++
            }

        ) {
            Text(text = "Increase counter")
        }
    }
}

 AlexisFlop *
@Preview(showSystemUi = false)
@Composable
private fun ModifyTextComponentPreview() {
    ModifyTextComponent()
}
```




```
Column(
    modifier = Modifier
        .fillMaxWidth()
        .padding(8.dp),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {
```

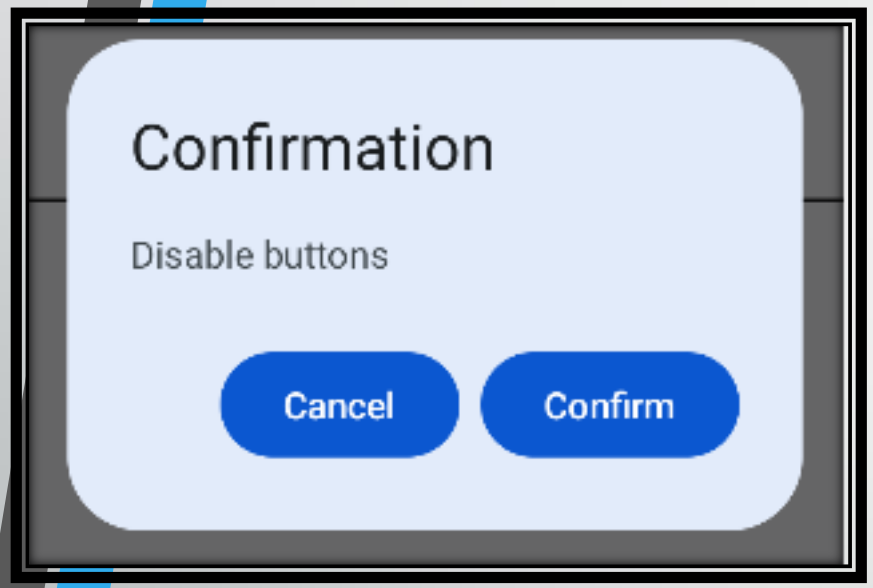
```
Text(text = "You can add names on this section")
```

```
val nameListScope = remember { nameList }
val name: MutableState<String> = remember { mutableStateOf( value: "") }
val buttonsScope = remember {
    buttonsStatus
}
```

```
OutlinedTextField(
    value = name.value, onValueChange = { name.value = it },
    singleLine = true,
    placeholder = { Text(text = "Enter name") },
    keyboardOptions = KeyboardOptions(
        keyboardType = KeyboardType.Text,
        imeAction = ImeAction.Done,
        capitalization = KeyboardCapitalization.Sentences,
        autoCorrect = true
    )
)
```

```
Button(
    enabled = buttonsScope.value,
    onClick = {
        nameList.add(name.value)
        name.value = ""
        Log.d( tag: "Name List", nameListScope.toString())
    }
) {

    Text(text = "Add name")
}
```

```
LazyColumn(
    modifier = Modifier
        .fillMaxWidth()
        .border(1.dp, Color.Black)
        .height(100.dp)
        .padding(horizontal = 4.dp)
) {
    itemsIndexed(nameListScope) { index, nameToIndexed ->
        Text(text = "Item #$index: $nameToIndexed")
    }
}
```

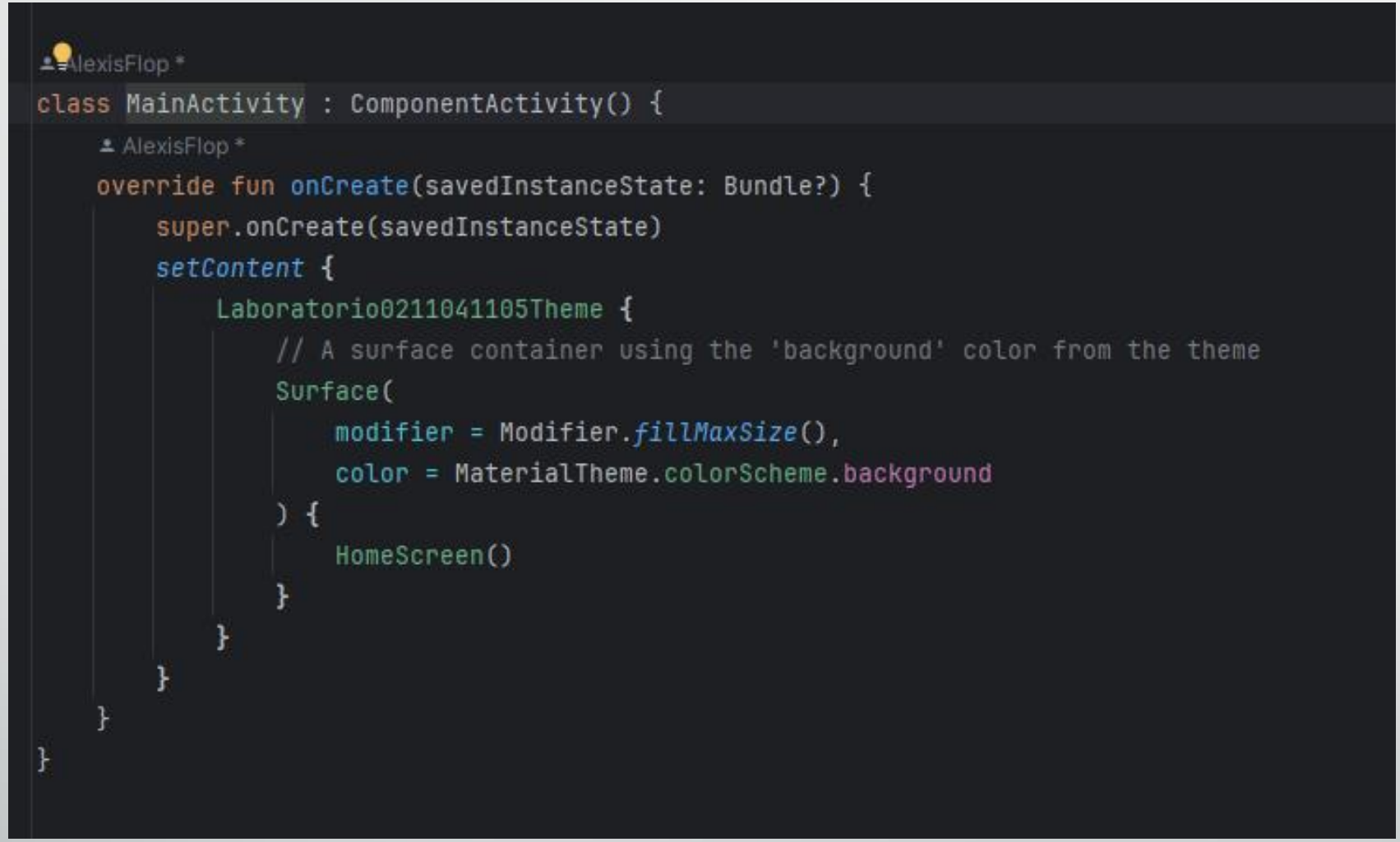


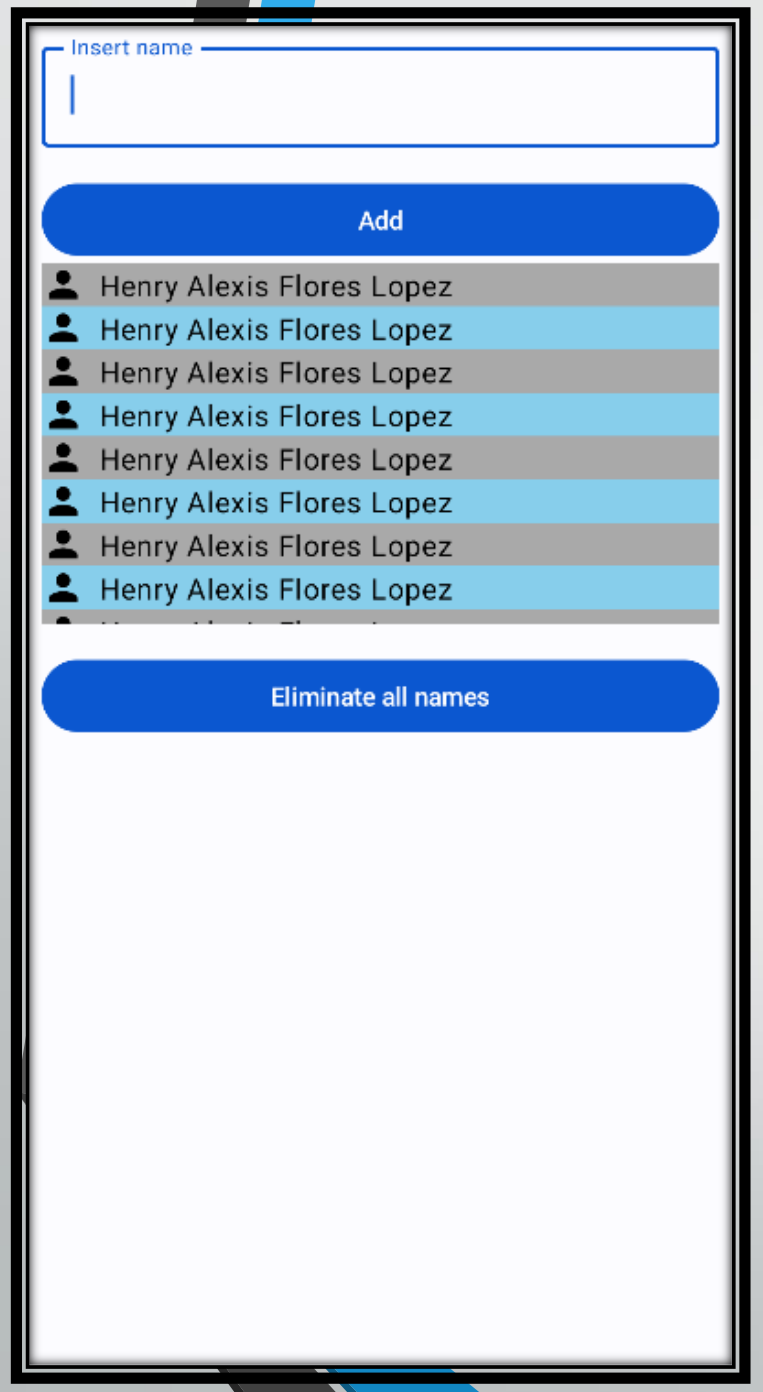

```
Button(onClick = {
    dialogStatus.value = !dialogStatus.value
}) {
    Text(text = if (buttonsScope.value) "Disable buttons" else "Enable buttons")
}
```

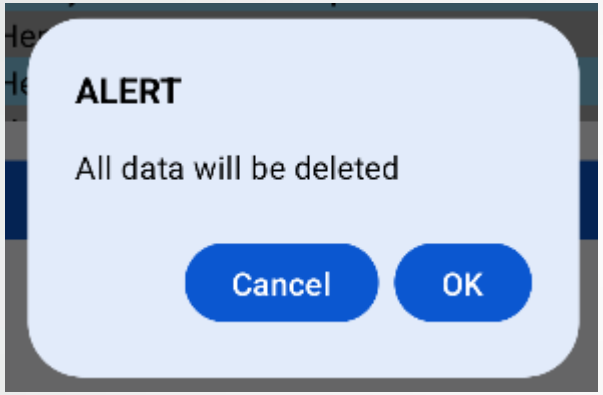
```
val dialogStatus: MutableState<Boolean> = remember { mutableStateOf( value: false) }
val buttonsScope = remember {
    buttonsStatus
}
```

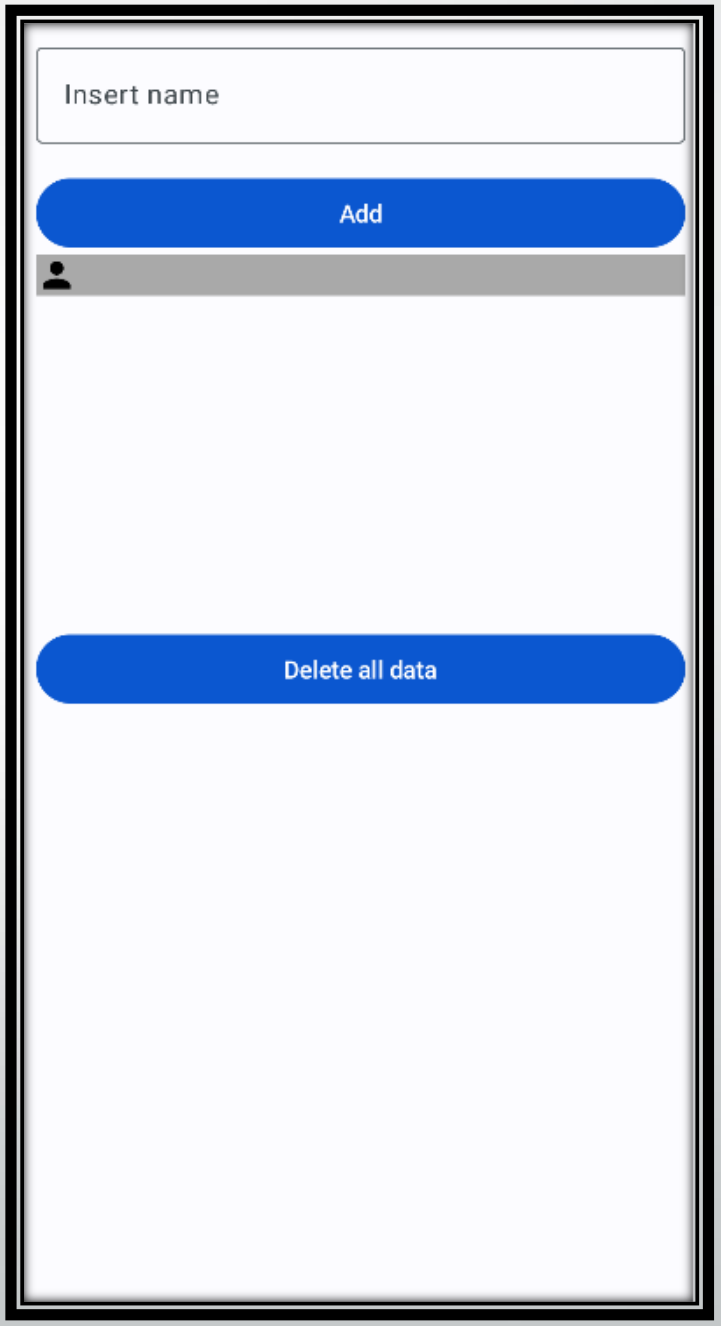
```
if (dialogStatus.value) {
    AlertDialog(onDismissRequest = { dialogStatus.value = false },
        properties = DialogProperties(
            dismissOnBackPress = true,
            dismissOnClickOutside = false
        ),
        confirmButton = {
            Button(onClick = {
                Log.d( tag: "BEFORE",  msg: "$buttonsStatus" + "$buttonsScope")
                buttonsStatus.value = !buttonsStatus.value
                Log.d( tag: "AFTER",  msg: "$buttonsStatus" + "$buttonsScope")
                dialogStatus.value = false
            }) {
                Text(text = "Confirm")

            }
        },
        dismissButton = {
            Button(onClick = {
                dialogStatus.value = false
            }) {
                Text(text = "Cancel")
            }
        },
        text = { Text(text = if (buttonsScope.value) "Disable buttons" else "Enable buttons") },
        title = { Text(text = "Confirmation") }
    )
}
```

```
@Composable
fun HomeScreen() {
    Column(
        modifier = Modifier.padding(8.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        ModifyTextComponent()
        AddNamesComponent()
        DisableOptionButton()
    }
}

@Preview(showSystemUi = true)
@Composable
private fun HomeScreenPreview() {
    HomeScreen()
}
```


AlexisFlop *
class MainActivity : ComponentActivity() {
AlexisFlop *
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContent {
Laboratorio0211041105Theme {
// A surface container using the 'background' color from the theme
Surface(
modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colorScheme.background
) {
HomeScreen()
}
}
}
}
}


Insert name
Add
Henry Alexis Flores Lopez
Henry Alexis Flores Lopez
Henry Alexis Flores Lopez
Henry Alexis Flores Lopez
Henry Alexis Flores Lopez
Henry Alexis Flores Lopez
Henry Alexis Flores Lopez
Henry Alexis Flores Lopez
Eliminate all names


ALERT
All data will be deleted
Cancel
OK


Insert name
Add
Delete all data