LAB 5

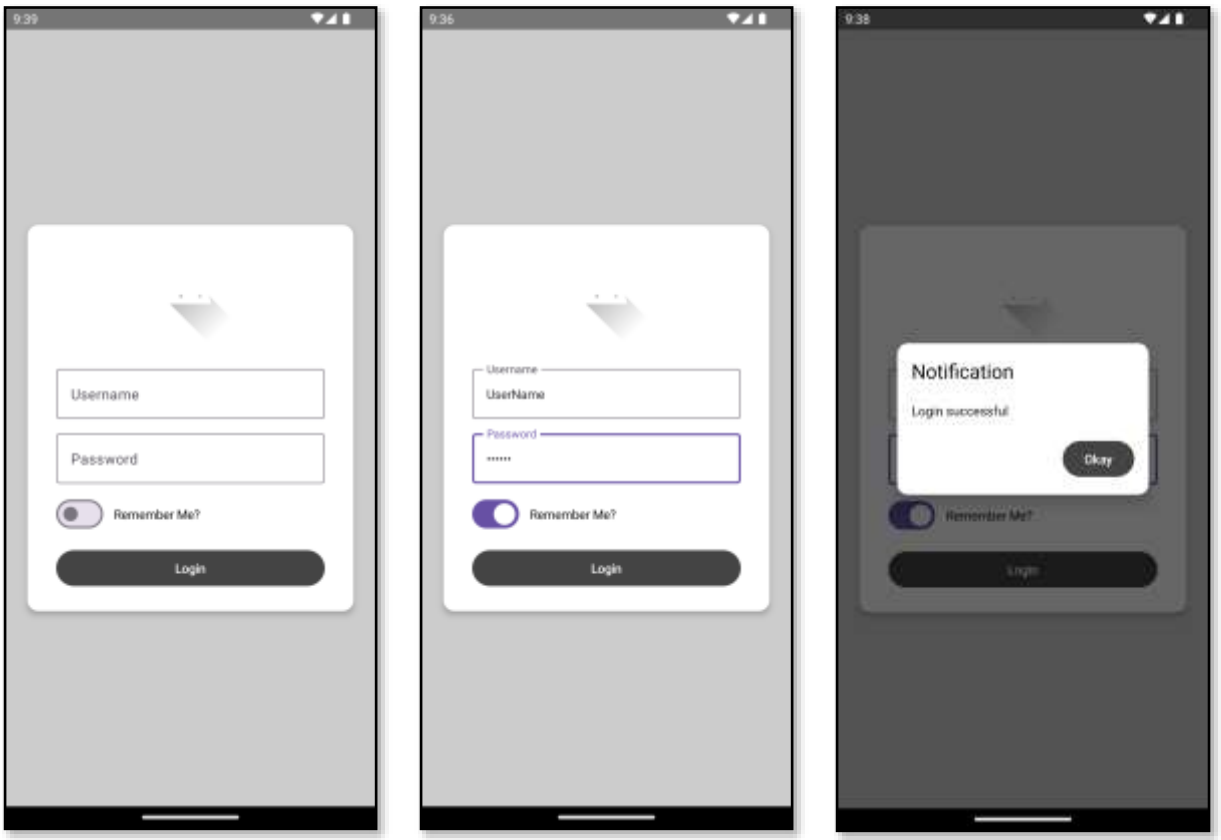
MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Biết cách sử dụng các hàm và tham số của Card.
- ✓ Biết cách sử dụng Switch và cách tuỳ chỉnh màu sắc của Switch.
- ✓ Biết cách sử dụng Chip và hiển thị Chip trong 1 danh sách có thể cuộn.

NỘI DUNG

BÀI 1: Sử dụng lại giao diện Login Screen ở bài 1 Lab 4 hãy thực hiện các yêu cầu sau:



❖ Yêu cầu giao diện:

- Đưa Logo và các trường user name and password vào trong Card, Card sẽ nằm ở chính giữa màn hình.
- Thêm 1 Switch component để lưu thông tin đăng nhập.
- Sử dụng Dialog thay cho Toast để hiển thị thông báo khi nhấn nút Login.

❖ Hướng dẫn và code tham khảo:

- Sử dụng lại project của bài 1 lab 4 hoặc tạo project mới để tiếp tục các bước sau.
- Để đưa Card và các thành phần khác ở giữa màng hình ta cần tạo 1 hàm LoginApp và sử dụng Box.

```
@Composable
fun LoginApp() {
    Box(
        modifier = Modifier.fillMaxSize().background(Color.LightGray),
        contentAlignment = Alignment.Center,
    ) {
        Card(
            modifier = Modifier.fillMaxWidth().padding(24.dp),
            colors = CardDefaults.cardColors(containerColor =
Color.White),
            elevation = CardDefaults.cardElevation(defaultElevation =
4.dp)
        ) {
            LoginScreen()
        }
    }
}
```

- **cardColors** và **cardElevation** là hai hàm từ **CardDefaults** cung cấp cách để định nghĩa giá trị mặc định cho màu sắc và độ cao của các thành phần **Card**.
    - **cardColors**: Đây là hàm trả về một đối tượng **CardColors** được dùng để định nghĩa màu sắc cho các phần khác nhau của một **Card**. Nó có thể nhận các tham số để tùy chỉnh màu sắc:
        - **backgroundColor**: Màu nền của **Card**.
        - **contentColor**: Màu của nội dung bên trong **Card**, bao gồm văn bản và các thành phần UI khác.

- **cardElevation**: Đây là hàm cung cấp giá trị độ cao cho **Card**, tạo hiệu ứng nổi hoặc bóng đổ. Hàm này có thể nhận các tham số để tùy chỉnh độ cao trong các trạng thái khác nhau:
  - **defaultElevation**: Độ cao mặc định của **Card**.
  - **pressedElevation**: Độ cao của **Card** khi được người dùng nhấn vào.
  - **disabledElevation**: Độ cao khi **Card** bị vô hiệu hóa.
  - **hoveredElevation**: Độ cao khi có con trỏ chuột di chuyển qua **Card**.
  - **focusedElevation**: Độ cao khi **Card** đang được focus.

- Điều chỉnh lại hàm LoginScreen và đưa nó vào trong Card:

```
@Composable
fun LoginScreen() {
    val context = LocalContext.current
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxWidth().padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        // Code như ở bài 3 lab 4
    }
}
```

- Tạo 1 DialogComponent để hiển thị thông báo thay cho Toast, Dialog chứa Title, message và button Okay.

```
@Composable
fun DialogComponent(
    onConfirmation: () -> Unit,
    dialogTitle: String,
    dialogMessage: String,
) {
    Dialog(onDismissRequest = {}) {
        Card(
            colors = CardDefaults.cardColors(
                containerColor = Color.White,
            ),
            modifier = Modifier.padding(20.dp),
```

```
            elevation = CardDefaults.cardElevation(
                defaultElevation = 10.dp
            )
        ) {
            Column(
                modifier = Modifier.padding(16.dp),
                horizontalAlignment = Alignment.Start,
            ) {
                Text(dialogTitle, style =
MaterialTheme.typography.titleLarge)
                Spacer(modifier = Modifier.height(20.dp))
                Text(dialogMessage, style =
MaterialTheme.typography.bodyMedium)
                Spacer(modifier = Modifier.height(20.dp))
                Button(
                    onClick = onConfirmation,
                    modifier = Modifier.align(Alignment.End),
                    colors = ButtonDefaults.buttonColors(
                        containerColor = Color.DarkGray,
                        contentColor = Color.White
                    )
                ) {
                    Text("Okay")
                }
            }
        }
    }
}
```

- Để hiển thị Dialog ta cần sử dụng 1 state là showDialog và điều chỉnh sự kiện onClick của nút Login trong hàm LoginSceen như sau:

```
@Composable
fun LoginScreen() {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var showDialog by remember { mutableStateOf(false) }
    var dialogMessage by remember { mutableStateOf("") }

    if (showDialog) {
        DialogComponent(
            onConfirmation = { showDialog = false },
            dialogTitle = "Notification",
            dialogMessage = dialogMessage
        )
    }
       // Các đoạn code còn lại
       Button(
           onClick = {
              if (username.isNotBlank() && password.isNotBlank()) {
                  dialogMessage = "Login successful"
              } else {
```

```
                    dialogMessage = "Please enter username and password"
                }
                showDialog = true
            },
            colors = ButtonDefaults.buttonColors(
                containerColor = Color.DarkGray,
                contentColor = Color.White
            )
        ) {
            Text("Login")
        }
    }
```

- Tạo hàm RememberMe để hiển thị Switch lưu thông tin đăng nhập:

```
@Composable
fun RememberMeSwitch() {
    var isChecked by remember { mutableStateOf(false) }

    Row(
        Modifier.fillMaxWidth(),
        verticalAlignment = Alignment.CenterVertically,
    ) {
        Switch(
            checked = isChecked,
            onCheckedChange = { isChecked = it }
        )
        Text("Remember Me?", modifier = Modifier.padding(start =
12.dp))
    }
}
```

- Gọi hàm RememberMe trong hàm LoginScreen và điều chỉnh 1 số style cho hợp lý:

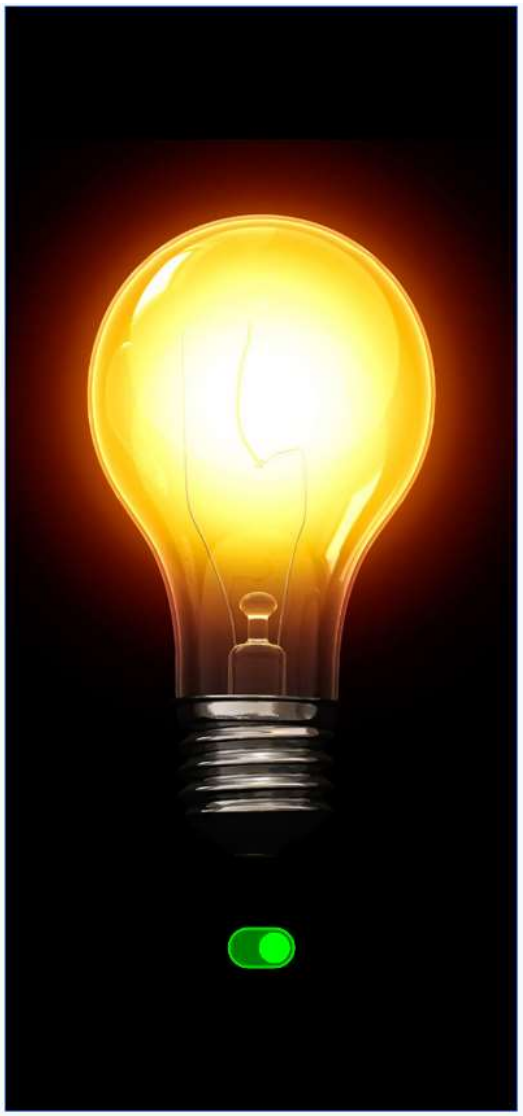
```
@Composable
fun LoginScreen() {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxWidth()
            .padding(32.dp, 24.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        // Logo
        Image(
            painter = painterResource(id =
R.drawable.ic_launcher_foreground),
            contentDescription = "Logo",
        )
```

```
        Spacer(modifier = Modifier.height(20.dp))

        // Username TextField
        OutlinedTextField( modifier = Modifier.fillMaxWidth(),
            value = username,
            onValueChange = { username = it },
            label = { Text("Username") },
        )

        Spacer(modifier = Modifier.height(8.dp))

        // Password TextField
        OutlinedTextField( modifier = Modifier.fillMaxWidth(),
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            visualTransformation = PasswordVisualTransformation()
        )

        Spacer(modifier = Modifier.height(10.dp))

        RememberMeSwitch()

        Spacer(modifier = Modifier.height(12.dp))

        // Login Button
        Button(
            onClick = {
                if (username.isNotBlank() && password.isNotBlank()) {
                    Toast.makeText(context, "Login successful",
Toast.LENGTH_LONG).show()
                } else {
                    Toast.makeText(
                        context,
                        "Please enter username and password",
                        Toast.LENGTH_LONG
                    ).show()
                }
            },
            colors = ButtonDefaults.buttonColors(
                containerColor = Color.DarkGray,
                contentColor = Color.White
            ),
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Login")
        }
    }
}
```

- Chạy ứng dụng và quan xác kết quả thực tế:

BÀI 2: Xây dựng giao diện ứng dụng điều khiển đèn.

Khi đèn tắt



Khi đèn bật

❖ Yêu cầu giao diện:

- Sử dụng **Column** để căn giữa các thành phần trong màn hình.
- Đặt màu nền cho **Column** là màu đen.
- Hiển thị hình ảnh đèn bật (**bulb_on**) hoặc đèn tắt (**bulb_off**) tùy thuộc vào trạng thái của **switch**.
- Hiển thị **Switch** dưới hình ảnh đèn và cho phép người dùng tương tác để thay đổi trạng thái của đèn.

❖ Logic Xử Lý:

- Sử dụng **remember { mutableStateOf(false) }** để theo dõi trạng thái của switch (bật/tắt).
- Trạng thái của switch sẽ kiểm soát hình ảnh đèn hiển thị (bật hoặc tắt).
- Cập nhật hình ảnh đèn tương ứng khi trạng thái của switch thay đổi.

❖ Tùy Chỉnh Switch:

- Tùy chỉnh màu sắc của switch khi bật và tắt.
- Đảm bảo switch có màu xanh khi bật và màu xám khi tắt.
- Tùy chỉnh màu của viền switch khi bật.

❖ Hướng dẫn và code tham khảo:

- Tạo dự án Kotlin mới với tên dự án là Lab5_MSSV_2
- Chọn template là EmptyActivity và nhấn **Finish**.
- Sau khi quá trình đồng bộ hoàn tất mở class MainActivity.kt và thực hiện tạo hàm LightSwitch:

```
@Composable
fun LightSwitch() {
    Column(
        modifier = Modifier.fillMaxSize().background(Color.Black),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        val isChecked = remember { mutableStateOf(false) }

        if (isChecked.value) {
            Image(
                painter = painterResource(id = R.drawable.bulb_on),
                contentDescription = "Light is On",
                modifier = Modifier.fillMaxWidth(),
                contentScale = ContentScale.FillWidth
            )
        } else {
            Image(
                painter = painterResource(id =
R.drawable.bulb_off),
                contentDescription = "Light is Off",
                modifier = Modifier.fillMaxWidth(),
                contentScale = ContentScale.FillWidth
            )
        }
```

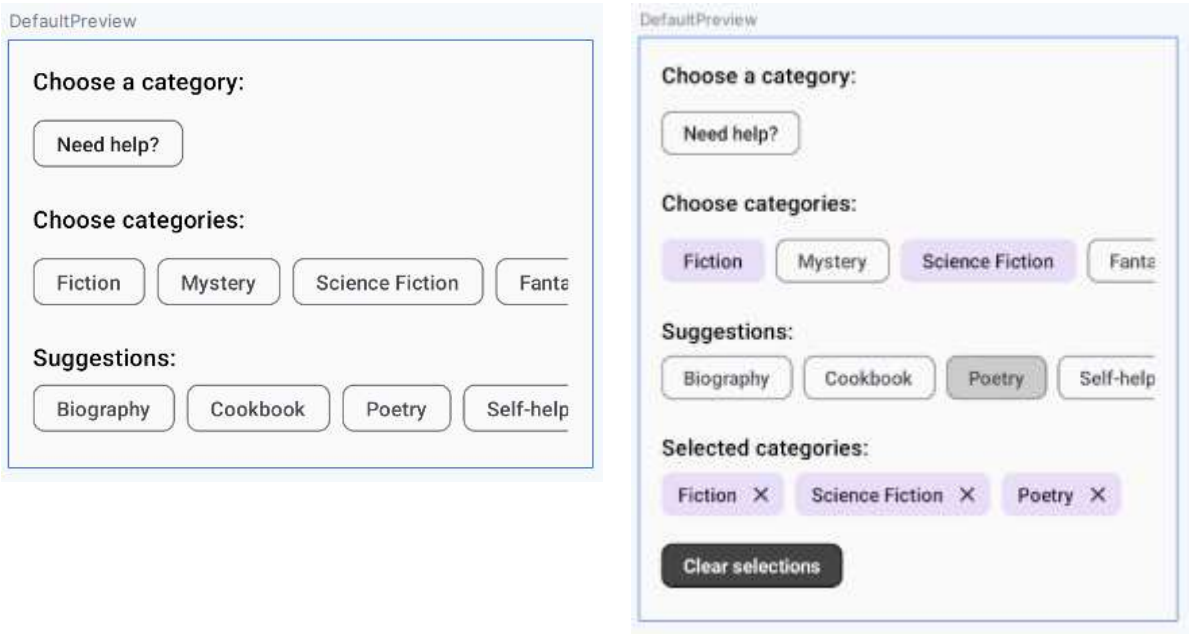
```
        Spacer(modifier = Modifier.height(24.dp))

        Switch(
            checked = isChecked.value,
            onCheckedChange = { isChecked.value = it },
            colors = SwitchDefaults.colors(
                checkedThumbColor = Color.Green,
                uncheckedThumbColor = Color.Gray,
                checkedTrackColor = Color.Green.copy(alpha = 0.5f),
                uncheckedTrackColor = Color.Gray.copy(alpha =
0.5f),
                checkedBorderColor = Color.Green.copy(alpha =
0.75f),
            )
        )
    }
}
```

Các tham số của **SwitchDefaults**:

- **checkedThumbColor**: Màu của nút trượt khi Switch được chọn (bật).
- **uncheckedThumbColor**: Màu của nút trượt khi Switch không được chọn (tắt).
- **checkedTrackColor**: Màu của đường trượt khi Switch được chọn, thường có độ trong suốt.
- **uncheckedTrackColor**: Màu của đường trượt khi Switch không được chọn, cũng có độ trong suốt.
- **checkedBorderColor**: Màu viền của Switch khi ở trạng thái checked (chỉ có từ phiên bản nhất định của Jetpack Compose).
- **disabledCheckedThumbColor**, disabledUncheckedThumbColor, **disabledCheckedTrackColor**, **disabledUncheckedTrackColor**: Màu sắc khi Switch ở trạng thái vô hiệu hóa.

❖ Thêm hàm LightSwitch vào setContent và chạy ứng dụng, quan xác kết quả.

Bài 3: Xây Dựng Ứng Dụng Lựa Chọn Thể Loại Sách Sử Dụng Chips.



❖ Yêu Cầu Giao diện:

- Hiển thị Danh Sách Thể Loại:
  - Hiển thị danh sách các thể loại sách cho người dùng lựa chọn sử dụng FilterChip.
  - Danh sách các thể loại gồm: "Fiction", "Mystery", "Science Fiction", "Fantasy", "Adventure", "Historical", "Horror", "Romance".
- Hiển Thị và Xử Lý Gợi Ý:
  - Dưới danh sách thể loại, hiển thị một danh sách gợi ý sử dụng SuggestionChip.
  - Danh sách gợi ý gồm: "Biography", "Cookbook", "Poetry", "Self-help", "Thriller".
  - Khi một gợi ý được chọn, nó sẽ được thêm vào danh sách thể loại đã chọn.
- Hiển Thị Thể Loại Đã Chọn:
  - Dưới danh sách gợi ý, hiển thị các thể loại đã chọn sử dụng InputChip.

- Mỗi InputChip sẽ có một biểu tượng để cho phép người dùng bỏ chọn thể loại đó.
- Cung Cấp Chức Năng Hỗ Trợ:
- Sử dụng AssistChip ở trên cùng của màn hình để cung cấp hướng dẫn hoặc trợ giúp cho người dùng.
- Khi có thể loại nào được chọn, hiển thị AssistChip "Clear selections" để cho phép người dùng bỏ chọn tất cả các thể loại đã chọn.

❖ Gợi ý:

- Sử dụng Column để chứa tất cả các components và sắp xếp chúng theo chiều dọc.
- Dùng Row kết hợp với horizontalScroll để cho phép người dùng cuộn danh sách các chips.
- Lưu trữ thể loại đã chọn trong một biến trạng thái để cập nhật giao diện khi có sự thay đổi.

❖ Hướng dẫn và code tham khảo:

- Tạo dự án Kotlin mới với tên dự án là Lab5_MSSV_3
- Chọn template là EmptyActivity và nhấn **Finish**.
- Sau khi quá trình đồng bộ hoàn tất mở class MainActivity.kt và thực hiện tạo hàm **CategoryApp** và 2 danh sách **categoris** và **suggestions**:

```
@Composable
fun CategoryApp() {
    val categories = listOf("Fiction", "Mystery", "Science
Fiction", "Fantasy", "Adventure", "Historical", "Horror",
"Romance")
    val suggestions = listOf("Biography", "Cookbook", "Poetry",
"Self-help", "Thriller")
var selectedCategories by remember {mutableStateOf(setOf<String>())
}
}
```

- Sử dụng state để lưu danh sách các thể loại được chọn.
- Sử dụng Column để các thành phần sắp sếp từ trên xuống.

❖ Chia nhỏ các thành phần thành các hàm nhỏ để giảm bớt độ phức tạp của hàm **CategoryApp** là: **CategoryChips**, **SuggestionChips**, **SelectedCategoriesChips**:

- Tạo hàm **CategoryChips:**
  - Sử dụng Text để hiển thị section title.
  - Danh sách sẽ được hiển thị trong 1 **Row** với **horizontalScroll** (cho phép cuộn ngang).

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun CategoryChips(
    categories: List<String>,
    selectedCategories: Set<String>,
    onCategorySelected: (String) -> Unit
) {
    Text("Choose categories:", style = MaterialTheme.typography.titleMedium)
    Spacer(modifier = Modifier.height(8.dp))
    Row(modifier = Modifier.horizontalScroll(rememberScrollState())) {
        categories.forEach { category ->
            FilterChip(
                selected = selectedCategories.contains(category),
                onClick = { onCategorySelected(category) },
                label = { Text(category) },
                modifier = Modifier.padding(end = 8.dp)
            )
        }
    }
}
```

- Tạo hàm SuggestionChips:

```
@Composable
fun SuggestionChips(
    suggestions: List<String>,
    selectedCategories: Set<String>,
    onSuggestionSelected: (String) -> Unit
) {
    Text("Suggestions:", style = MaterialTheme.typography.titleMedium)
    Row(modifier = Modifier.horizontalScroll(rememberScrollState())) {
        suggestions.forEach { suggestion ->
            val isSelected = selectedCategories.contains(suggestion)
            val chipColors = if (isSelected) {
                SuggestionChipDefaults.suggestionChipColors(
                    containerColor = Color.LightGray
                )
            } else {
                SuggestionChipDefaults.suggestionChipColors()
            }
```

```
            SuggestionChip(
                onClick = { onSuggestionSelected(suggestion) },
                label = { Text(suggestion) },
                colors = chipColors,
                modifier = Modifier.padding(end = 8.dp)
            )
        }
    }
}
```

- Tạo hàm **SelectedCategoriesChips:**

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SelectedCategoriesChips(selectedCategories: Set<String>,
onCategoryRemoved: (String) -> Unit) {
    Text("Selected categories:", style =
MaterialTheme.typography.titleMedium)
    Row(modifier = Modifier.horizontalScroll(rememberScrollState())) {
        selectedCategories.forEach { selectedCategory ->
            InputChip(
                selected = true,
                onClick = { },
                label = { Text(selectedCategory) },
                trailingIcon = {
                    Icon(
                        imageVector = Icons.Filled.Close,
                        contentDescription = "Deselect",
                        modifier = Modifier.clickable {
onCategoryRemoved(selectedCategory) }.size(18.dp)
                    )
                },
                modifier = Modifier.padding(end = 8.dp),
            )
        }
    }
}
```

- Thêm các hàm trên vào trong CategoryApp và điểu chỉnh giao diện cho giống với yêu cầu:

```
@Composable
fun CategoryApp() {
    val categories = listOf("Fiction", "Mystery", "Science Fiction",
"Fantasy", "Adventure", "Historical", "Horror", "Romance")
    val suggestions = listOf("Biography", "Cookbook", "Poetry", "Self-help",
"Thriller")

    var selectedCategories by remember { mutableStateOf(setOf<String>()) }
    Column(modifier = Modifier.padding(16.dp)) {
```

```
        Text("Choose a category:", style =
MaterialTheme.typography.titleMedium)
        Spacer(modifier = Modifier.height(8.dp))

        AssistChip(
            onClick = { /* Do something */ },
            label = { Text("Need help?") }
        )

        Spacer(modifier = Modifier.height(16.dp))

        CategoryChips(categories, selectedCategories, onCategorySelected = {
category ->
            selectedCategories = if (selectedCategories.contains(category))
                selectedCategories - category
            else
                selectedCategories + category
        })

        Spacer(modifier = Modifier.height(16.dp))

        SuggestionChips(suggestions, selectedCategories,
onSuggestionSelected = { suggestion ->
            selectedCategories = selectedCategories + suggestion
        })

        if (selectedCategories.isNotEmpty()) {

            Spacer(modifier = Modifier.height(16.dp))

            SelectedCategoriesChips(selectedCategories, onCategoryRemoved =
{ category ->
                selectedCategories = selectedCategories - category
            })

            Spacer(modifier = Modifier.height(4.dp))

            AssistChip(
                onClick = { selectedCategories = setOf() },
                label = {
                    Text(
                        "Clear selections",
                        style = TextStyle(color = Color.White, fontWeight =
FontWeight.Bold)
                    )
                },
                colors = AssistChipDefaults.assistChipColors(containerColor
= Color.DarkGray),
                border = AssistChipDefaults.assistChipBorder(borderColor =
Color.Black)
            )
        }
```

```
        }
}
```

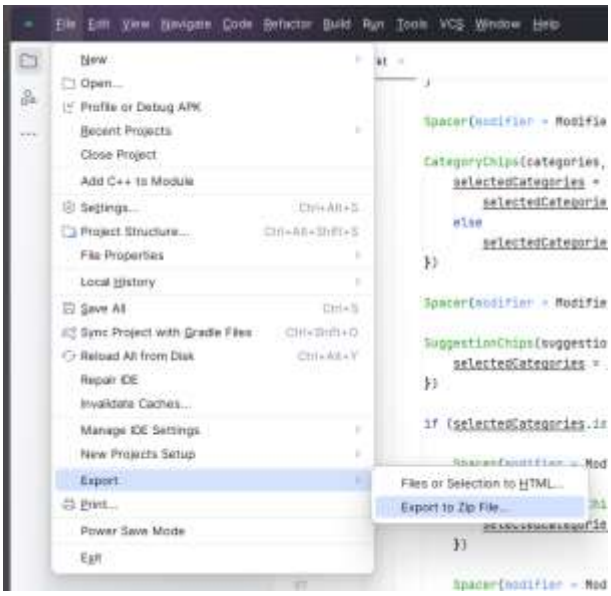
- Chạy ứng dụng và quan xác kết quả.

BÀI 4: GV CHO THÊM

*** YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

Hướng dẫn nén project: **File** > **Export** > **Export to Zip File**.



--- Hết ---