LAB 4

MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Biết cách sử dụng các Component và bố trí nội dung theo mong muốn
- ✓ Biết cách phối hợp các Component với nhau

NỘI DUNG

Bài 1: Tạo Giao Diện Đăng Nhập Với Jetpack Compose

❖ Yêu cầu giao diện:

- Sử dụng ImageView để hiển thị logo.
- Có 2 trường nhập liệu là Username và Password và 1 nút Login.
- Sử dụng trường nhập liệu cho mật khẩu với kiểu nhập mật khẩu, nghĩa là các ký tự nhập vào sẽ được ẩn đi để đảm bảo bảo mật.
- Khi người dùng nhấn vào nút "Login", cần hiển thị một Toast thông báo.
    - Nếu cả username và password đã được nhập, hiển thị thông báo "Login successful".
    - Nếu một trong hai hoặc cả hai trường thông tin còn trống, hiển thị thông báo "Please enter username and password".

❖ Hướng dẫn và code tham khảo:

- Tạo dự án Kotlin mới với tên dự án là Lab4_MSSV
- Chọn template là EmptyActivity và nhấn **Finish**.
- Sau khi quá trình đồng bộ hoàn tất mở class MainActivity.kt và thực hiện 1 số thay đổi sau:
    - Loại bỏ các hàm mẫu có sẵn chỉ để lại class MainActivity như sau:

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
        }
    }
}
```

- Tạo 1 lớp là LoginScreen() có annotation với @Composable

```
@Composable
fun LoginScreen() {
}
```

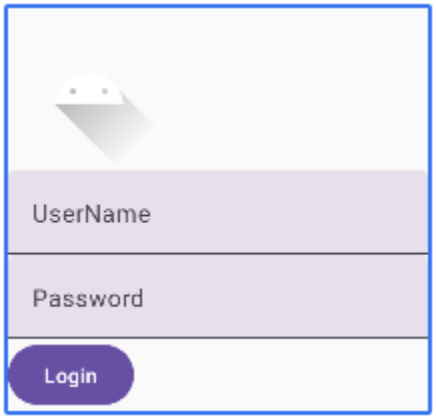
- Để có thể hiển thị Logo và các trường theo chiều dọc cần sử dụng 1 component là Column.

```
@Composable
fun LoginScreen() {
    Column {

    }
}
```

- Tiếp tục thêm các thành phần còn lại vào Column.

```
@Composable
fun LoginScreen(){
    val context = LocalContext.current
    var userName by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    Column {
        Image(painter = painterResource(id =
R.drawable.ic_launcher_foreground), contentDescription =
"Logo")
        TextField(value = userName, onValueChange = {
userName = it }, label = { Text(text = "UserName") })
        TextField(value = password, onValueChange = {
password = it }, label = { Text(text = "Password") })
        Button(onClick = {
            Toast.makeText(context, "Enter userName $userName
password $password", Toast.LENGTH_LONG).show()
        }) {
            Text(text = "Login")
        }
    }
}
```

- `remember` giữ giá trị khi UI được vẽ lại, ngăn không cho giá trị reset về ban đầu.
- `mutableStateOf` tạo một biến có thể thay đổi, và khi nó thay đổi, UI sẽ tự động cập nhật để phản ánh sự thay đổi đó.



Hình ảnh xem trước của Login Screen

- Gọi hàm LoginScreen trong setContent của MainActivity

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            LoginScreen()
        }
    }
}
```

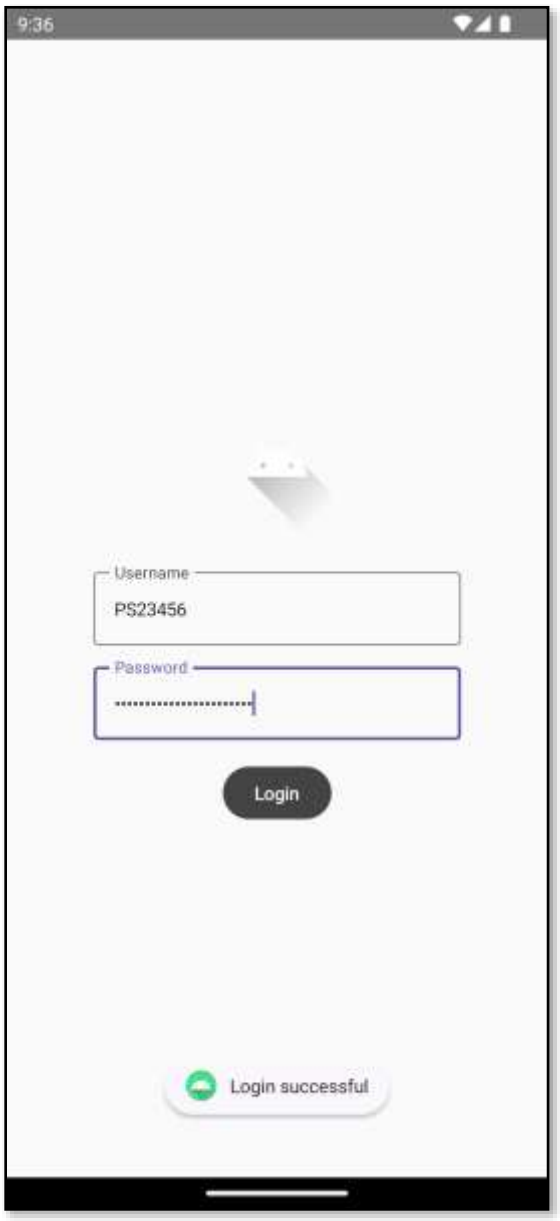
❖ Chạy ứng dụng nhập thông tin username và password và nhấn nút Login quan xác kết quả.

❖ Tiếp tục thực hiện các yêu cầu còn lại và thêm style cho giao diện.

Dưới đây là code tham khảo khi thêm style cho giao diện:

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

    Column(
        modifier = Modifier.fillMaxSize().padding(16.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(
            value = username,
            onValueChange = { username = it },
            label = { Text("Username") },
        )

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

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

        Spacer(modifier = Modifier.height(16.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)) {
            Text("Login")
        }
    }
}
```


9:36
Username
PS23456
Password
Login
Login successful

Bài 2: Tạo Giao Diện Hiển Thị Danh Sách Hình Ảnh

❖ Yêu Cầu giao diện:

- Hiển Thị Icon:
  - Hiển thị một Icon ở phần đầu của màn hình. Sử dụng Icon composable với hình ảnh lấy từ R.drawable.ic_launcher_foreground làm resource.
- Danh Sách Hình Ảnh Ngang:
  - Tạo một danh sách hình ảnh có thể cuộn ngang (HorizontalImageList) sử dụng Row và horizontalScroll.
  - Mỗi hình ảnh trong danh sách nên có kích thước 200dp và được làm tròn các góc với RoundedCornerShape(12.dp).
  - Hãy đảm bảo có padding phù hợp giữa các hình ảnh.
- Danh Sách Hình Ảnh Dọc:
  - Tạo một danh sách hình ảnh có thể cuộn dọc (VerticalImageList) sử dụng Column và verticalScroll.
  - Mỗi hình ảnh nên được căn chỉnh chiều rộng để lấp đầy khả năng hiển thị và có tỷ lệ chiều cao phù hợp, làm tròn các góc với RoundedCornerShape(12.dp).
  - Hình ảnh nên có padding phù hợp để tách biệt chúng ra khỏi nhau.

❖ Hướng dẫn và code tham khảo:

- Tạo dự án Kotlin mới với tên dự án là Lab4_MSSV
- Chọn template là EmptyActivity và nhấn **Finish**.
- Sau khi quá trình đồng bộ hoàn tất mở class MainActivity.kt và thực hiện 1 số thay đổi sau:
  - Loại bỏ các hàm mẫu có sẵn chỉ để lại class MainActivity như sau:

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
        }
    }
}
```

- Để có thể hiển thị Icon và các composable function khác theo chiều dọc ta cần sử dụng component Column.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Column {

            }
        }
    }
}
```

- Thêm Icon vào bên trong component Column.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Column {
                Icon(
                    painter = painterResource(id =
R.drawable.ic_launcher_foreground),
                    contentDescription = "Logo"
                )
            }
        }
    }
}
```

❖ Để hiển thị danh sách hình ảnh ngang

- Tạo hàm HorizontalImageList với annotation @Composable với đầu vào là 1 danh sánh image recourses.

```
@Composable
fun HorizontalImageList(imageList: List<Int>) {
    val scrollState = rememberScrollState()
```

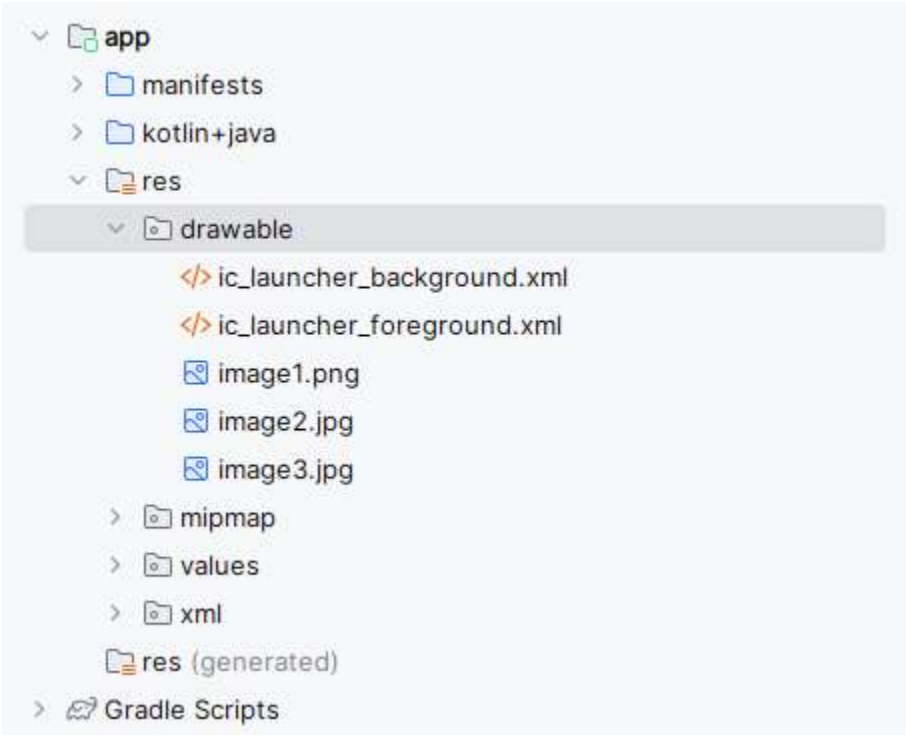
```
    Row(modifier = Modifier
        .horizontalScroll(scrollState)
        .padding(16.dp)) {
        imageList.forEachIndexed { index, image ->
            Image(
                painter = painterResource(id = image),
                contentDescription = "Image Description",
                contentScale = ContentScale.FillHeight,
                modifier = Modifier
                    .size(200.dp)
                    .clip(RoundedCornerShape(12.dp))
                    .padding(
                        start = if (index == 0) 0.dp else 8.dp,
                        end = 8.dp
                    )
            )
        }
    }
}
```

❖ Để hiển thị danh sách hình ảnh dọc

- Tạo hàm VerticalImageList với annotation @Composable với đầu vào là 1 danh sánh image recourses.

```
@Composable
fun VerticalImageList(imageList: List<Int>) {
    val scrollState = rememberScrollState()

    Column(modifier = Modifier
        .verticalScroll(scrollState)
        .padding(16.dp)) {
        imageList.forEachIndexed { index, image ->
            Image(
                painter = painterResource(id = image),
                contentDescription = "Image Description",
                contentScale = ContentScale.FillWidth,
                modifier = Modifier
                    .clip(RoundedCornerShape(12.dp))
                    .padding(
                        top = if (index == 0) 0.dp else 8.dp,
                        bottom = 8.dp
                    )
            )
        }
    }
}
```

❖ Thêm các hình ảnh cần hiển thị vào đường dẫn: **app/res/drawable**



❖ Tạo danh sách chứa các hình đã thêm bên trên vào trong setContent.

```
val images = listOf(R.drawable.image1, R.drawable.image2,
R.drawable.image3)
```

❖ Tạo 2 hàm để xem trước giao diện của 2 hàm 2 hàm HorizontalImageList và VerticalImageList

```
@Preview(showBackground = true)
@Composable
fun PreviewHorizontalImageList() {
    HorizontalImageList(listOf(R.drawable.image1, R.drawable.image2,
R.drawable.image3))
}

@Preview(showBackground = true)
@Composable
fun PreviewVerticalImageList() {
    VerticalImageList(listOf(R.drawable.image1, R.drawable.image2,
R.drawable.image3))
}
```

❖ Gọi 2 hàm HorizontalImageList và VerticalImageList trong setContent.

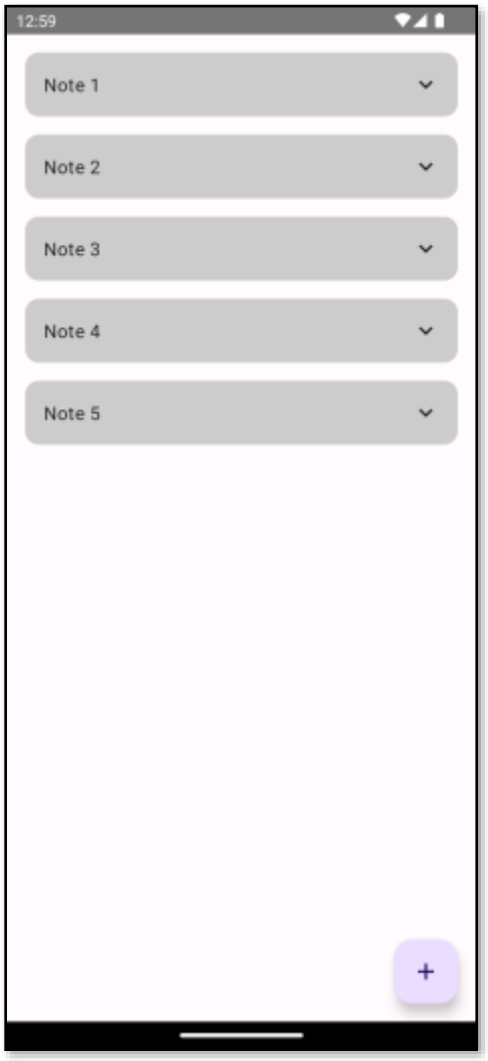
```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContent {
            val images = listOf(R.drawable.image1, R.drawable.image2,
R.drawable.image3)
            Column {
                Icon(
                    painter = painterResource(id =
R.drawable.ic_launcher_foreground),
                    contentDescription = "Logo"
                )
                HorizontalImageList(images)
```

```
                    VerticalImageList(images)
                }
            }
        }
    }
```

❖ Chạy ứng dụng và quan xác kết quả, đảm bảo danh sách hình ảnh có thể cuộn ngang và danh sách hình ảnh có thể cuộn dọc.

Bài 3: Tạo giao diện ứng dụng ghi chú sử dụng Jetpack Compose.

❖ Dựa vào các bài tập ở trên hãy tạo giao diện như sau.



❖ Yêu cầu giao diện:

- Hiển thị danh sách các ghi chú.
- Mỗi ghi chú có Text hiển thị tiêu đề và Icon dropdown.
- Sử dụng FAB (Floating Action Button) ở góc phải với Icon Add.

❖ Hướng dẫn và code tham khảo:

- Tạo dự án Kotlin mới với tên dự án là Lab4_MSSV
- Chọn template là EmptyActivity và nhấn **Finish**.
- Sau khi quá trình đồng bộ hoàn tất mở class MainActivity.kt và thực hiện 1 số thay đổi sau:
  - Loại bỏ các hàm mẫu có sẵn chỉ để lại class MainActivity như sau:

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
        }
    }
}
```

❖ Tạo hàm NoteApp

```
@Composable
fun NoteApp(paddingValues: PaddingValues) {
    val notes = listOf("Note 1", "Note 2", "Note 3", "Note 4",
"Note 5")

    Column(modifier =
Modifier.padding(paddingValues).padding(8.dp)) {
        notes.forEach { note ->
            NoteCard(noteText = note)
        }
    }
}
```

❖ Tạo hàm NoteApp

```
@Composable
fun NoteCard(noteText: String) {
    Box(
        modifier = Modifier.fillMaxWidth().padding(8.dp)
            .background(color = Color.LightGray, shape =
MaterialTheme.shapes.medium)
    ) {
        Row(verticalAlignment = Alignment.CenterVertically) {
            Text(
                text = noteText,
                modifier = Modifier.weight(1f).padding(16.dp),
                style = MaterialTheme.typography.bodyLarge
            )
```

```
            Icon(
                imageVector = Icons.Filled.KeyboardArrowDown,
                contentDescription = "Expand Note",
                modifier =
Modifier.padding(16.dp).align(Alignment.CenterVertically)
            )
        }
    }
}
```

❖ Thêm FloatingActionButton vào Scaffold, để cho phép định vị nó một cách linh hoạt trong giao diện người dùng.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Scaffold(
                floatingActionButton = {
                    FloatingActionButton(onClick = { /* Handle
FAB click action here */ }) {
                        Icon(Icons.Filled.Add,
contentDescription = "Add")
                    }
                }
            ) { innerPadding ->
                NoteApp(innerPadding)
            }
        }
    }
}
```

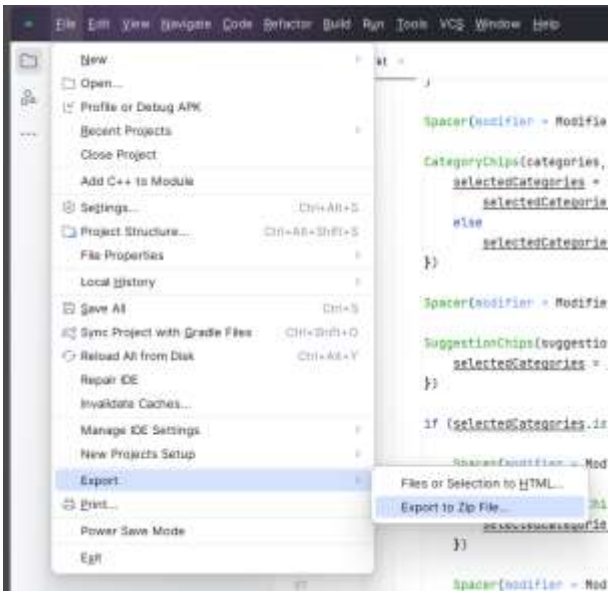
❖ Chạy ứng dụng và quan xác kết quả.

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 ---