Mobile Application Development

Week12　Desing a Music App by Scaffold composable

Yi SUN
Kobe Institute of Computing

The Scaffold composable

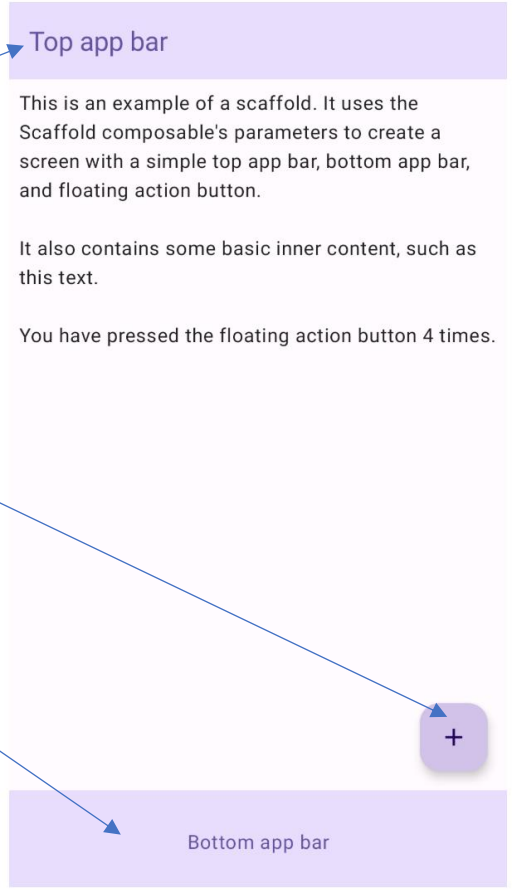
The Scaffold composable provides a straightforward API you can use to quickly assemble your app's structure according to Material Design guidelines. Scaffold accepts several composables as parameters. Among these are the following:

topBar: The app bar across the top of the screen.

bottomBar: The app bar across the bottom of the screen.

floatingActionButton: A button that hovers over the bottom-right corner of the screen that you can use to expose key actions.

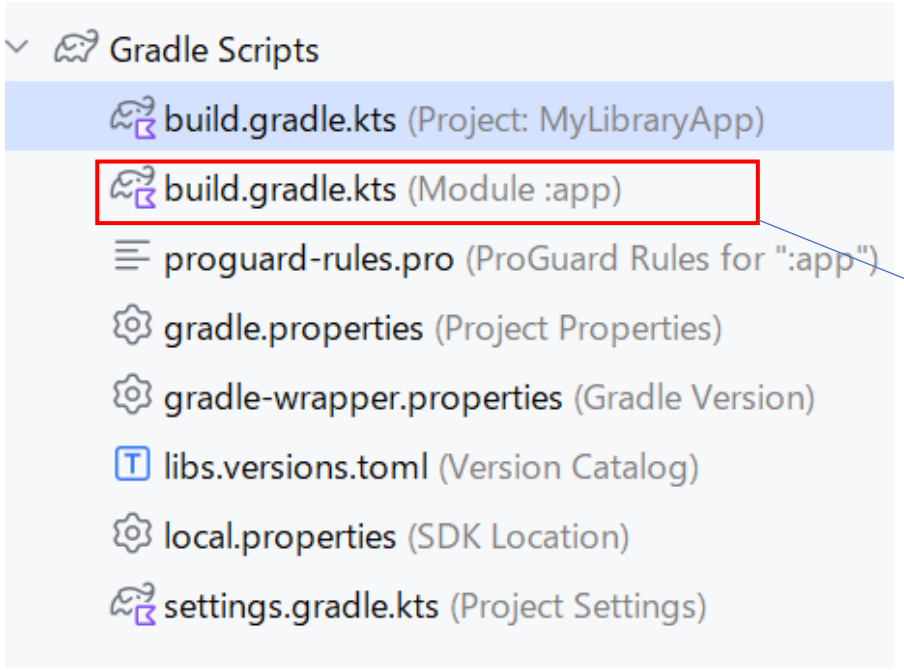
You can also pass Scaffold content as you would to other containers. It passes an innerPadding value to the content lambda that you can then use in child composables.



https://developer.android.com/develop/ui/compose/components/scaffold

Add the navigation dependencies

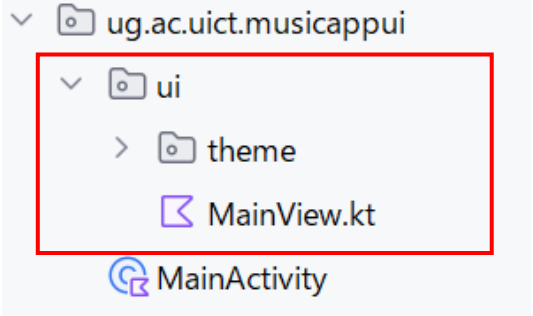
- Add the dependency in your app-level build.gradle files




```
dependencies {

    val nav_version = "2.9.0"
    implementation("androidx.navigation:navigation-compose:$nav_version")

    val compose_version ="1.8.1"
    implementation("androidx.compose.ui:ui:$compose_version")
    implementation("androidx.compose.material:material:$compose_version")
    implementation("androidx.compose.ui:ui-tooling-preview:$compose_version")

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
```

Design the main page

- Create a MainView.kt file at ui package



MainView.kt

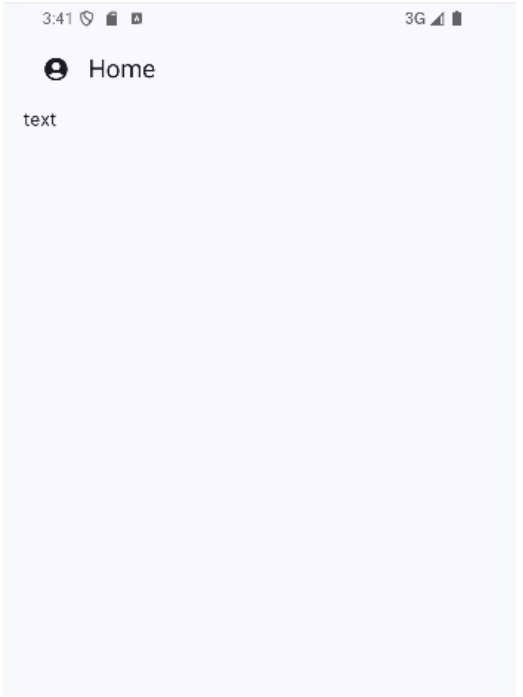
```
fun MainView(){
    val scaffoldState: ScaffoldState = rememberScaffoldState()
    val scope: CoroutineScope = rememberCoroutineScope()

    Scaffold(
        topBar = {
            TopAppBar(title = { Text("Home")},
                navigationIcon = { IconButton(onClick = {
                    scope.launch {
                        scaffoldState.drawerState.open()
                    }
                }) {
                    Icon(imageVector = Icons.Default.AccountCircle, contentDescription = "Menu")
                }})
        }
    ) { paddingValues ->
        Text("text", modifier = Modifier.padding(paddingValues))
    }
}
```

Change the MainActivity.kt as an Entry point

- Use the Surface to load the MainView() as the main view.

MainActivity.kt

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            MusicAppUITheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MainView()
                }
            }
        }
    }
}
```



The main view

Define the Screen object

- Add a Screen.kt file to make the definition of the object at the screen.

Screen.kt

```
sealed class Screen(val title:String, val route: String) {

    sealed class DrawerScreen(val dTitle:String, val dRoute:String, @DrawableRes val icon: Int)
        : Screen(dTitle,dRoute){
            object Account: DrawerScreen(
                "Account",
                "account",
                R.drawable.baseline_account_box_24
            )
        object Subscription: DrawerScreen(
            "Subscription",
            "subscribe",
            R.drawable.baseline_library_music_24
        )
        object  AddAccount: DrawerScreen(
            "Add Account",
            "add_account",
            R.drawable.baseline_person_add_alt_1_24
        )
        }
}
```

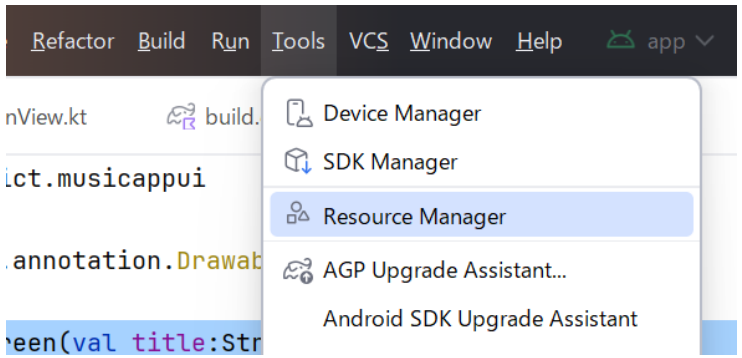
sealed class ScreenDeclares a closed hierarchy of “screens” in your app.
Each subclass must be defined in the same file.
Carries two properties:
title: String – human-readable name
route: String – navigation route/key

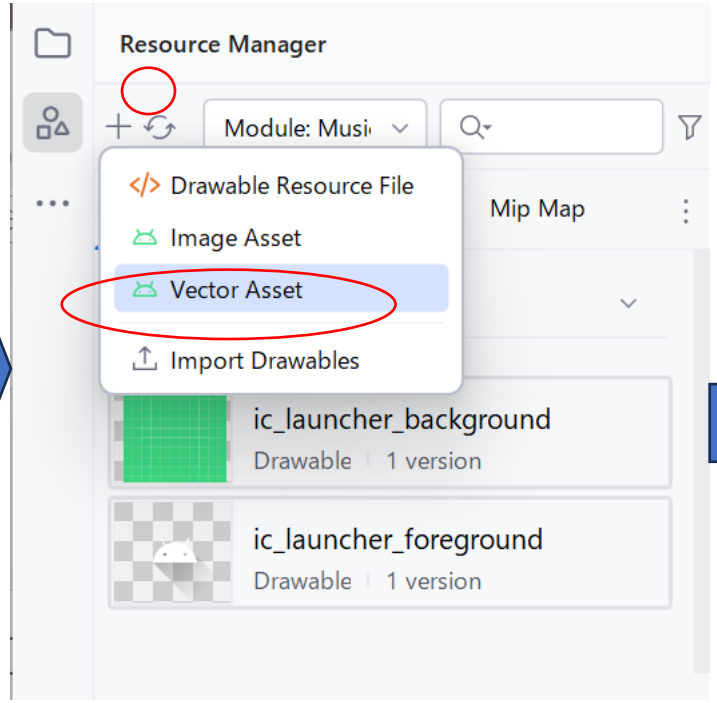
Nested sealed class DrawerScreen
Inherits from Screen by calling Screen(dTitle, dRoute).
Adds one more property:
@DrawableRes val icon: Int – a compile-time-checked drawable resource ID
Used to represent items in a navigation drawer.

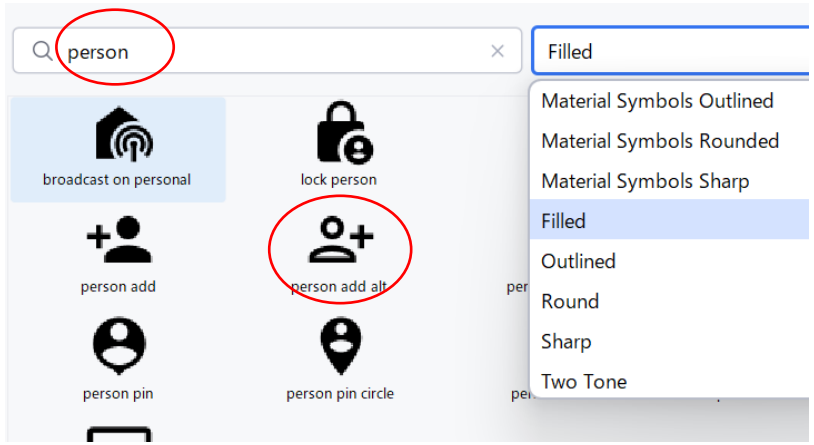
object declarations
Account, Subscription, AddAccount are singleton instances of DrawerScreen.
Each provides its own dTitle, dRoute, and icon.
Being object, they’re instantiated once and can be compared by identity.

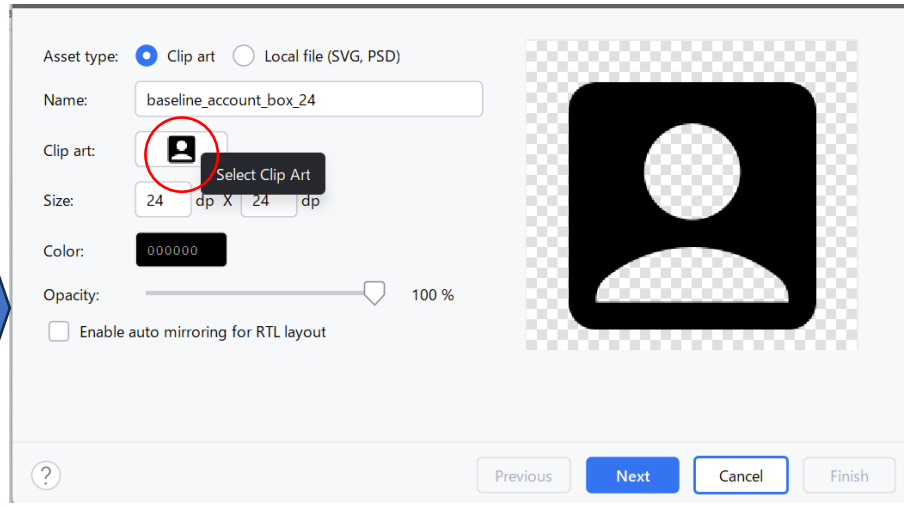
Add the Drawable Resource

- Click the Menu->Tools->Resource Manager
- Click the + -> Vector Asset
- Click the Clip art: Icon
- Find the Clip you want use by keyword.









Add the reuseable compose of DrawerItem in MainView.kt

MainView.kt

```
@Composable
fun DrawerItem(
    selected: Boolean,
    item: Screen.DrawerScreen,
    onDrawerItemClicked : () -> Unit
){
    val background = if (selected) Color.DarkGray else Color.White
    Row(
        Modifier
            .fillMaxWidth()
            .padding(horizontal = 8.dp, vertical = 16.dp)
            .background(background)
            .clickable {
                onDrawerItemClicked()
            }) {
        Icon(
            painter = painterResource(id = item.icon),
            contentDescription = item.dTitle,
            Modifier.padding(end = 8.dp, top = 4.dp)
        )
        Text(
            text = item.dTitle,
            style = MaterialTheme.typography.titleMedium,
        )
    }
}
```

selected: Whether this item is currently active/highlighted.
item: A Screen.DrawerScreen object carrying title, route, and icon.
onDrawerItemClicked: Lambda to invoke when the row is tapped.

A horizontal row with an icon on the left, the title next to it, and a background that changes color when selected == true

Making the drawer appear

MainView.kt

```
fun MainView(){
    val scaffoldState: ScaffoldState = rememberScaffoldState()
    val scope: CoroutineScope = rememberCoroutineScope()

    val controller: NavController = rememberNavController()
    val navBackStackEntry by controller.currentBackStackEntryAsState()
    val currentRoute = navBackStackEntry?.destination?.route

    val title = remember{
        mutableStateOf("")
    }

    Scaffold(
        topBar = {
            TopAppBar(title = { Text("Home")},
                                .....
                {
                    Icon(imageVector = Icons.Default.AccountCircle, contentDescription = "Menu")
                })
    }, scaffoldState = scaffoldState,
    drawerContent = {
        LazyColumn(Modifier.padding(16.dp)) {
            items(screensInDrawer){
                item ->
                DrawerItem(selected = currentRoute == item.dRoute, item = item) {
                    scope.launch {
                        scaffoldState.drawerState.close()
                    }
                    if(item.dRoute == "add_account"){

                    }else{
                        controller.navigate(item.dRoute)
                        title.value = item.dTitle
                    }
```

Sets up Jetpack Navigation to switch between screens.

Observes the current back-stack entry so you can know which screen (route) is active.

Holds the current screen title; updated when the user selects a drawer item.

selected becomes true when the drawer route matches the current route.
Click behavior:
Close the drawer.
If it's not the special "add_account" route, navigate and update the title.

Screen.kt

```
...
val screensInDrawer = listOf(
    Screen.DrawerScreen.Account,
    Screen.DrawerScreen.Subscription,
    Screen.DrawerScreen.AddAccount
)
```

Setting up the MainViewModel

- Create a MainViewModel.kt file at ui package

```
class MainViewModel:ViewModel() {
    private val _currentScreen: MutableState<Screen> = mutableStateOf(Screen.DrawerScreen.AddAccount)
    val currentScreen: MutableState<Screen> get() = _currentScreen
    fun setCurrentScreen(screen:Screen){
        _currentScreen.value = screen
    }
}
```

Setting up the Navigation in MainView

MainView.kt

```
fun MainView(){
    val scaffoldState: ScaffoldState = rememberScaffoldState()
    val scope: CoroutineScope = rememberCoroutineScope()
    val viewModel: MainViewModel = viewModel()

    val controller: NavController = rememberNavController()
    val navBackStackEntry by controller.currentBackStackEntryAsState()
    val currentRoute = navBackStackEntry?.destination?.route

    val currentScreen = remember{
        viewModel.currentScreen.value
    }
```

```
        }

    ) { paddingValues ->
        Navigation(navController = controller,viewModel = viewModel, pd = paddingValues)
    }
}
```

When you use Scaffold, Compose will calculate inner insets (to account for your TopAppBar, any bottom bars, or the open drawer) and pass them into this lambda as a PaddingValues object.

```
@Composable
fun Navigation(navController: NavController, viewModel: MainViewModel, pd:PaddingValues){
    NavHost(navController = navController as NavHostController,
        startDestination = Screen.DrawerScreen.AddAccount.route,
        modifier = Modifier.padding(pd)) {
        composable(Screen.DrawerScreen.AddAccount.route){

        }
        composable(Screen.DrawerScreen.Subscription.route){

        }
    }
}
```

Obtains (and remembers) an instance of your MainViewModel scoped to this composable's lifecycle. You can now read or observe its data.

viewModel.currentScreen is presumably a MutableState<Screen> (or StateFlow<Screen>). Wrapping viewModel.currentScreen.value in remember { … } means currentScreen is set once when MainView first runs, and won't update thereafter.

composable(route)Defines a destination in the navigation graph keyed by its route string.
Lambda bodyThe UI that should render when navController.navigate(route) brings you to this screen.

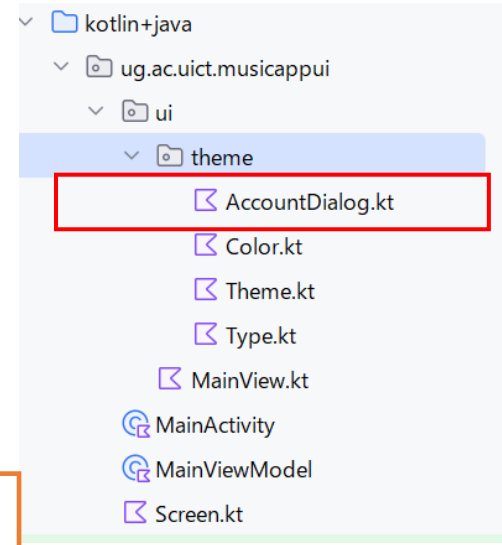
Add Account Dialog

- Create a AccountDailog.kt file at ui.theme package

AccountDialog.kt

```
@Composable
fun AccountDialog(dialogOpen: MutableState<Boolean>){
    if(dialogOpen.value){
        AlertDialog(
            onDismissRequest = {
                dialogOpen.value = false
            },
            confirmButton = {
                TextButton(onClick = {
                    dialogOpen.value = false
                }) {
                    Text("Confirm")
                }
            },
            dismissButton = {
                TextButton(onClick = {
                    dialogOpen.value = false
                }) {
                    Text("Dismiss")
                }
            },
            title = {
                Text("Add Account")
            },
```

```
text = {
    Column(modifier = Modifier.wrapContentHeight().padding(top = 16.dp),
        verticalArrangement = Arrangement.Center
    ) {
        TextField(value = "", onValueChange = {
            // Todo
        }, modifier = Modifier.padding(top = 16.dp),
            label = {Text(text="Email")})
        TextField(value = "", onValueChange = {

        }, modifier = Modifier.padding(top = 8.dp),
            label = { Text(text = "Password")})
    }
},
modifier = Modifier.fillMaxWidth()
    .background(MaterialTheme.colors.primarySurface)
    .padding(8.dp),
shape = RoundedCornerShape(5.dp),
backgroundColor = Color.White,
properties = DialogProperties(
    dismissOnBackPress = true,
    dismissOnClickOutside = true
)
```



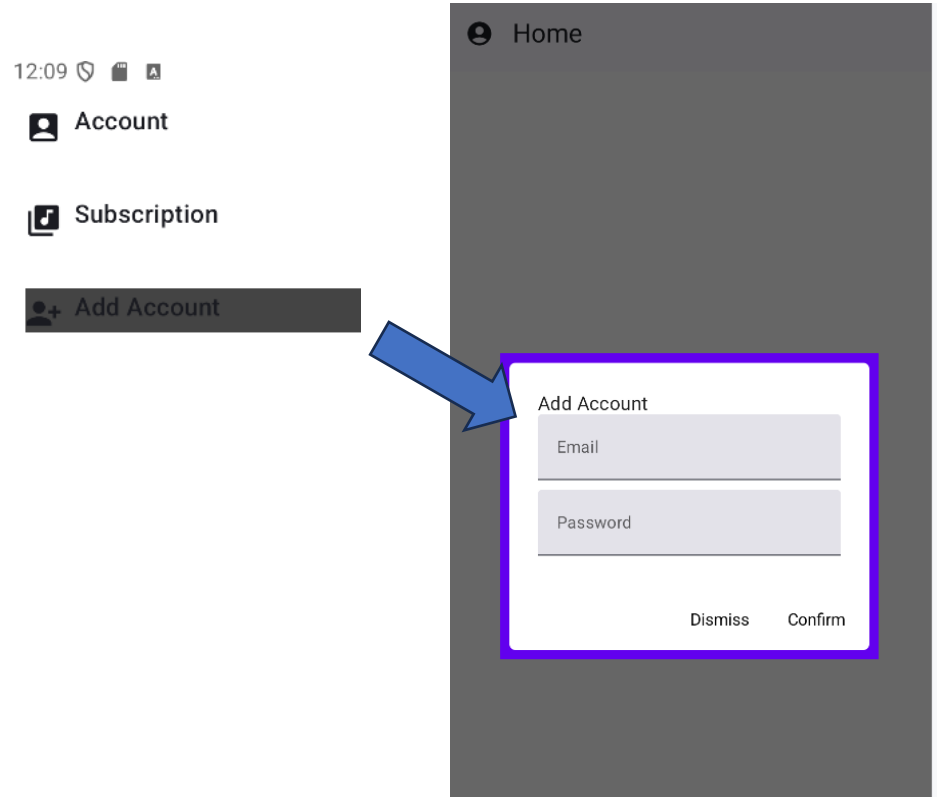
Add the openDialog event at MainView

MainView.kt

```
val controller: NavController = rememberNavController()
val navBackStackEntry by controller.currentBackStackEntryAsState()
val currentRoute = navBackStackEntry?.destination?.route

val dialogOpen = remember{
    mutableStateOf(false)
}
```

```
DrawerItem(selected = currentRoute==item.dRoute, item = item){
    scope.launch {
        scaffoldState.drawerState.close()
    }
    if(item.dRoute == "add_account"){
        dialogOpen.value = true
    }else{
        controller.navigate(item.dRoute)
        title.value = item.dTitle
    }
}
```

```
    ) { paddingValues ->
        Navigation(navController = controller,viewModel = viewModel, pd = paddingValues)
        AccountDialog(dialogOpen = dialogOpen)
    }
}
```



Add a AccountView

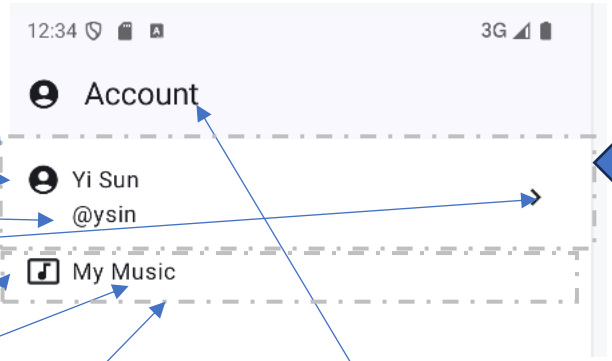
Create a AccountView.kt file at ui.theme package

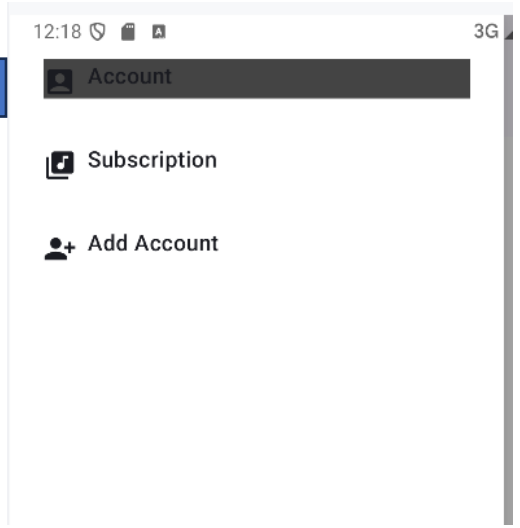
AccountView.kt

```
fun AccountView(){
    Column(
        modifier = Modifier.fillMaxSize().padding(16.dp)
    ){
        Row (
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween
        ){
            Row() {
                Icon(imageVector = Icons.Default.AccountCircle,
                    contentDescription = "Account",
                    modifier = Modifier.padding(end = 8.dp)
                )
                Column {
                    Text("Yi Sun")
                    Text("@ysin")
                }
            }
            IconButton(onClick = {}) {
                Icon(imageVector = Icons.AutoMirrored.Filled.KeyboardArrowRight,
                    contentDescription = null)
            }
        }
        Row(modifier = Modifier.padding(top = 16.dp)){
            Icon(
                painter = painterResource(id = R.drawable.baseline_music_video_24),
                contentDescription = "My Music",
                modifier = Modifier.padding(end = 8.dp)
            )
            Text(text = "My Music")
        }
    }
}
```

MainView.kt

```
@Composable
fun Navigation(navController: NavController, viewModel: MainViewModel, pd:PaddingValues){
    NavHost(navController = navController as NavHostController,
        startDestination = Screen.DrawerScreen.AddAccount.route,
        modifier = Modifier.padding(pd)) {
        composable(Screen.DrawerScreen.AddAccount.route){

        }
        composable(Screen.DrawerScreen.Subscription.route){

        }
        composable(Screen.DrawerScreen.Account.route){
            AccountView()
        }
    }
}
```





MainView.kt

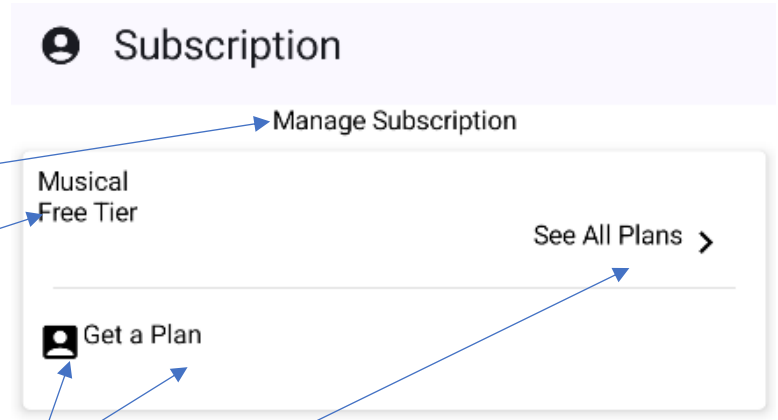
```
Scaffold(
    topBar = {
        TopAppBar(title = { Text(title.value)},
            navigationIcon = { IconButton(onClick = {
                scope.launch {
                    scaffoldState.drawerState.open()
```

Add a Subscription View

Create a Subscription.kt file at ui.theme package

Subscription.kt

```
fun Subscription(){
    Column(
        modifier = Modifier.height(200.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text = "Manage Subscription")
        Card(modifier = Modifier.padding(8.dp), elevation = 4.dp) {
            Column(modifier = Modifier.padding(8.dp)) {
                Column() {
                    Text(text = "Musical")
                    Row(Modifier.fillMaxWidth(), horizontalArrangement = Arrangement.SpaceBetween){
                        Text(text = "Free Tier")
                        TextButton(onClick = { }) {
                            Row {
                                Text(text = "See All Plans")
                                Icon(imageVector = Icons.AutoMirrored.Filled.KeyboardArrowRight,
                                    contentDescription = "See All Plans"
                                )
                            }
                        }
                    }
                }
                Divider(thickness = 1.dp, modifier = Modifier.padding(horizontal = 8.dp))
                Row(Modifier.padding(vertical = 16.dp)) {
                    Icon(imageVector = Icons.Default.AccountBox, contentDescription = "Get a Plan")
                    Text(text = "Get a Plan")
                }
```



MainView.kt

```
fun Navigation(navController: NavController, viewModel:
MainViewModel, pd:PaddingValues){
    NavHost(navController = navController as NavHostController,
        startDestination = Screen.DrawerScreen.AddAccount.route,
        modifier = Modifier.padding(pd)) {
        composable(Screen.DrawerScreen.AddAccount.route){

        }
        composable(Screen.DrawerScreen.Subscription.route){
            Subscription()
        }
        composable(Screen.DrawerScreen.Account.route){
            AccountView()
        }
```