Mobile Application Development

Week13　Desing a Music App by Scaffold composable (2)

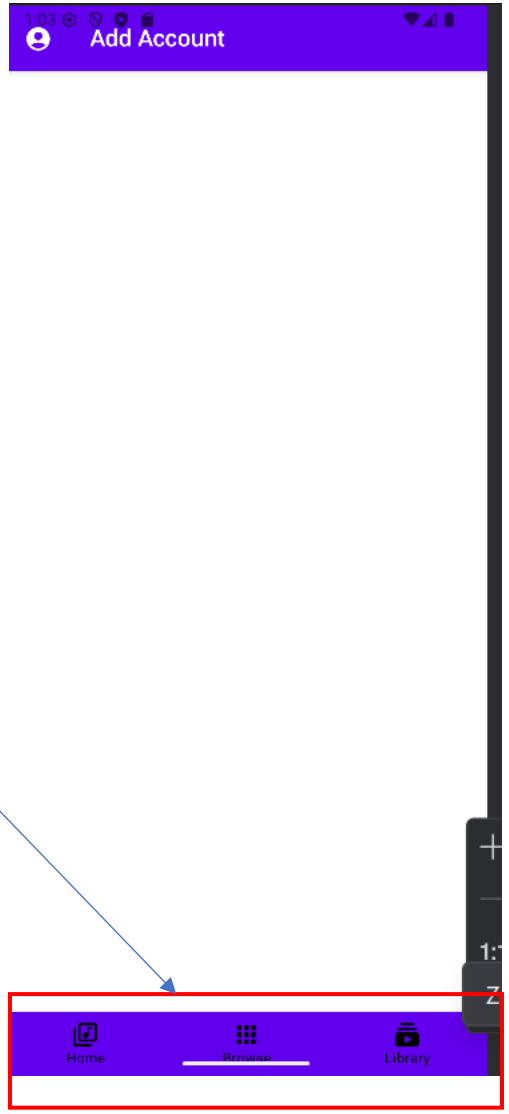
Yi SUN
Kobe Institute of Computing

Add the BottomBar to Screen (1)

- Add the BottomScreen class at Screen.kt

Screen.kt

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

    sealed class BottomScreen(
        val bTitle:String, val bRoute: String, @DrawableRes val icon: Int
    ):Screen(bTitle,bRoute){
        object  Home: BottomScreen(bTitle = "Home", bRoute = "home", R.drawable.outline_library_music_24)
        object Library: BottomScreen(bTitle = "Library", bRoute = "library", R.drawable.baseline_subscriptions_24)
        object Browse: BottomScreen(bTitle = "Browse", bRoute = "browse", R.drawable.baseline_apps_24)


    }

    sealed class DrawerScreen(val dTitle:String, val dRoute:String, @DrawableRes val icon:Int)
        :Screen(dTitle,dRoute){
```



- Add the list of bottom items at Screen.kt for mainview to control.

Screen.kt

```
val screenInBottom = listOf(
    Screen.BottomScreen.Home,
    Screen.BottomScreen.Browse,
    Screen.BottomScreen.Library
)
```

Add the BottomBar to Screen (2)

- Add the BottomBar in Scaffold

MainView.kt

```
val bottomBar: @Composable () -> Unit ={
    if(currentScreen is Screen.DrawerScreen || currentScreen == Screen.BottomScreen.Home){
        BottomNavigation(Modifier.wrapContentSize()) {
            screenInBottom.forEach{
                item->
                BottomNavigationItem(
                    selected = currentRoute == item.bRoute,
                    onClick = {controller.navigate(item.bRoute)
                                title.value = item.bTitle},
                    icon = { Icon(contentDescription = item.bTitle, painter = painterResource(id = item.icon))},
                    label = {Text(text = item.bTitle)},
                    selectedContentColor = Color.White,
                    unselectedContentColor = Color.Black
                )
            }
        }
    }
}

Scaffold(
    bottomBar = bottomBar,
    topBar = {
        TopAppBar(title = {Text(title.value)},
            navigationIcon = { IconButton(onClick = {
```

Add the BottomBar Navigation Routes

- Add the composable an Navigation class in MainView.kt

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.BottomScreen.Home.bRoute){

        }
        composable(Screen.BottomScreen.Browse.bRoute){

        }
        composable(Screen.BottomScreen.Library.bRoute){

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

        }
```

Add the HomeView

Create a HomeView.kt file at ui.theme package

HomeView.kt

```
@Composable
fun Home(){
    val categories = listOf("Hits", "Rock", "Pop", "R&B", "Country")
    val grouped = listOf<String>("New Release", "Favorites", "Top
Rated", "Editor Selection").groupBy { it[0] }
    LazyColumn {
        grouped.forEach{
            val sit = it
            stickyHeader {
                Text(text=sit.value[0], modifier = Modifier.padding(16.dp))
                LazyRow {
                    items(categories){
                        cate->
                        BrowserItem(cate= cate,
R.drawable.baseline_apps_24)
                    }
                }
            }
        }
    }
}
```

HomeView.kt

```
@Composable
fun BrowserItem(cate:String, drawable:Int){
    Card(modifier = Modifier.padding(16.dp).size(200.dp),
        border = BorderStroke(3.dp, color = Color.DarkGray)) {
        Column(verticalArrangement = Arrangement.Center,
            horizontalAlignment = Alignment.CenterHorizontally) {
            Text(text = cate)
            Image(painter = painterResource(id=drawable),
contentDescription = cate)
        }
    }
}
```

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.BottomScreen.Home.bRoute){
            Home()
        }
        composable(Screen.BottomScreen.Browse.bRoute){

        }
```

Add the Browse View

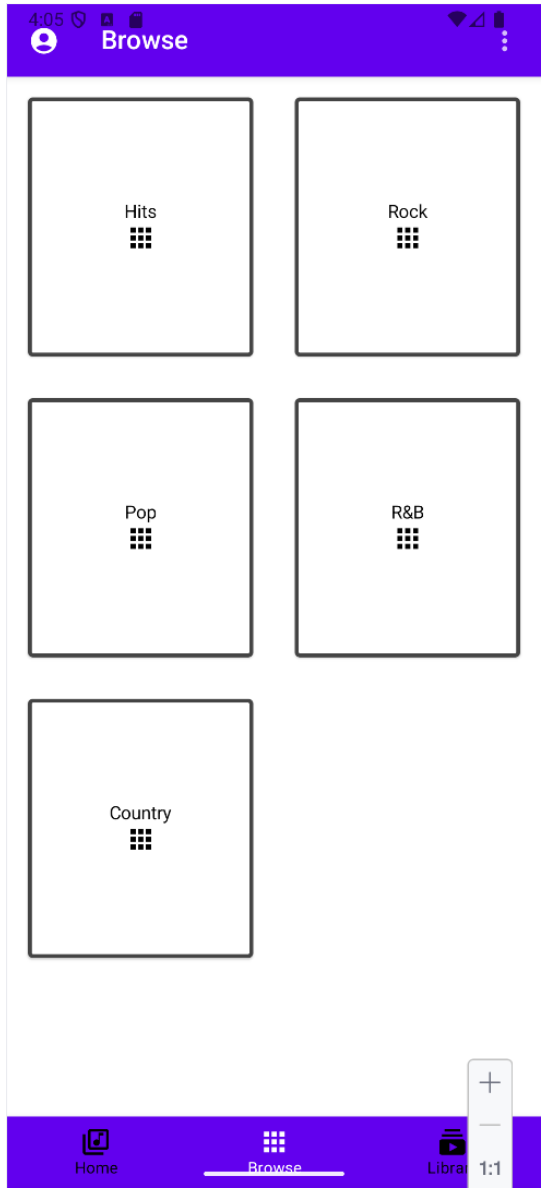
Create a BrowseView.kt file at ui.theme package

BrowseView.kt

```
@Composable
fun Browse(){
    val categories = listOf("Hits", "Rock", "Pop", "R&B", "Country")
    LazyVerticalGrid(GridCells.Fixed(2)) {
        items(categories){ cate ->
            BrowserItem(cate = cate, drawable = R.drawable.baseline_apps_24)
        }
    }
}
```

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.BottomScreen.Home.bRoute){
            Home()
        }
        composable(Screen.BottomScreen.Browse.bRoute){
            Browse()
        }
        composable(Screen.BottomScreen.Library.bRoute){
```



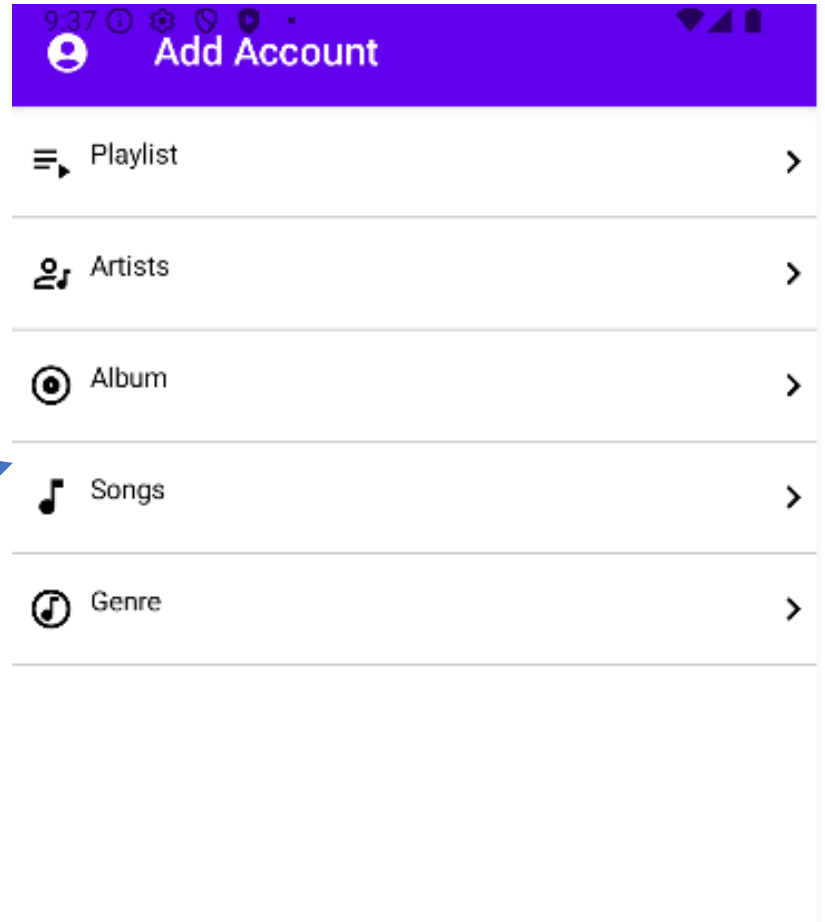
6

Add the Library View(1)

- Prepare the Icon Resource data list
- Create a DataResource.kt file

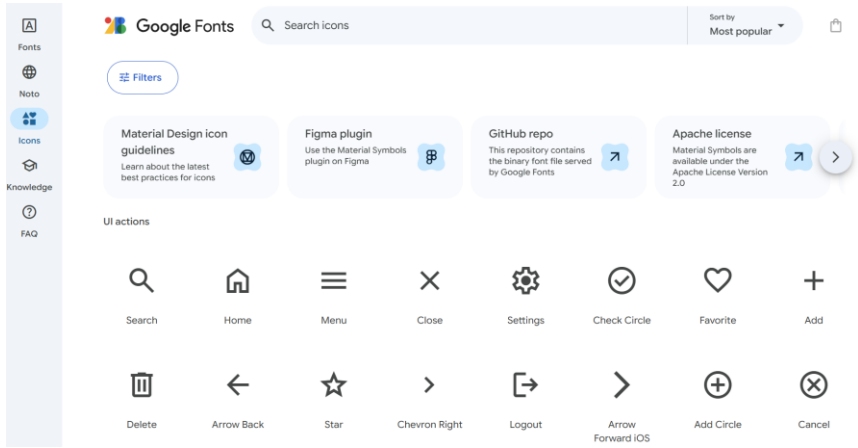
DataResource.kt

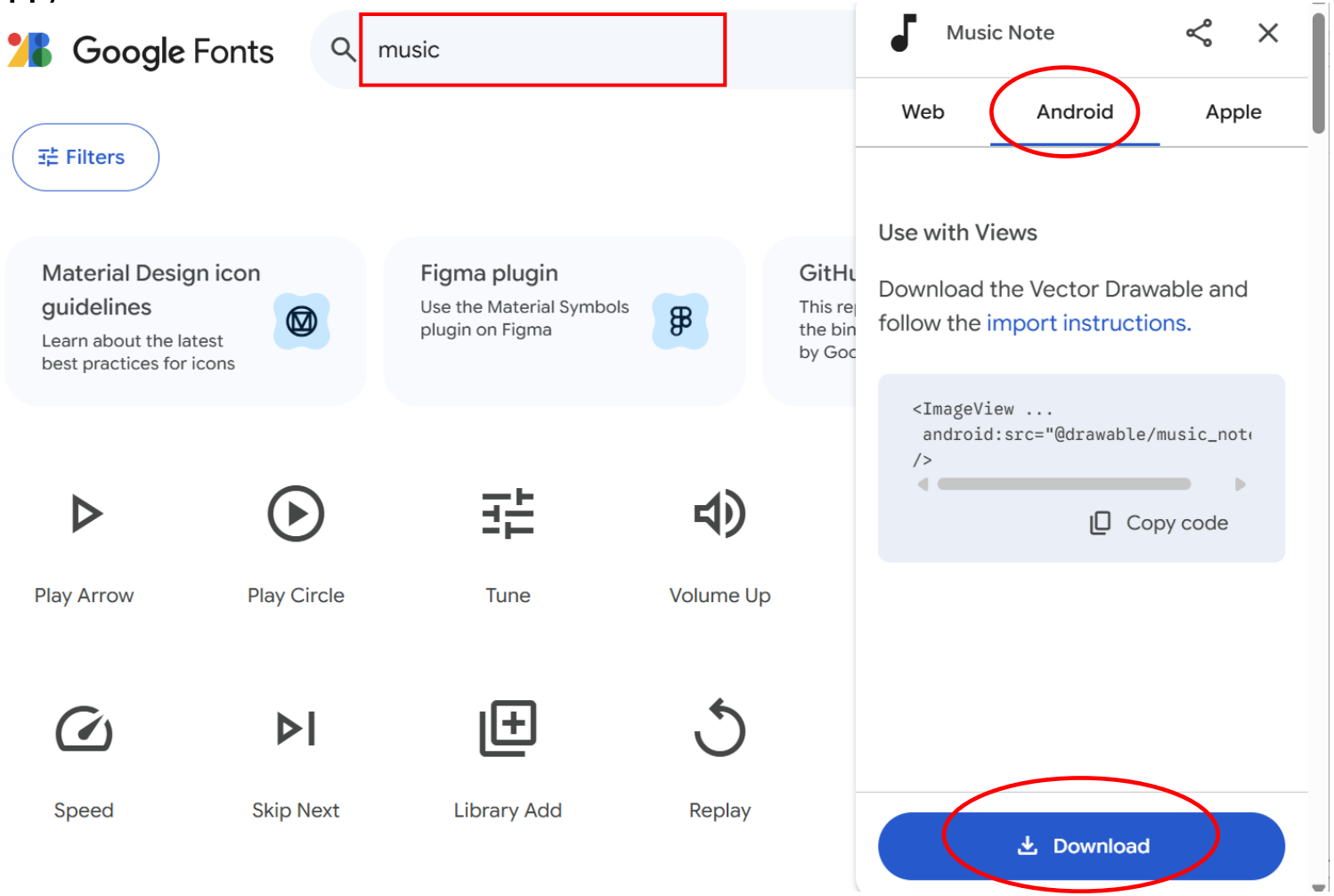
```
data class Lib(@DrawableRes val icon: Int, val name:String)

val libraries = listOf<Lib>(
    Lib(R.drawable.playlist_play_24px,"Playlist"),
    Lib(R.drawable.artist_24px,"Artists"),
    Lib(R.drawable.album_24px,"Album"),
    Lib(R.drawable.music_note_24px,"Songs"),
    Lib(R.drawable.genres_24px,"Genre")
)
```



Add the Library View(2)

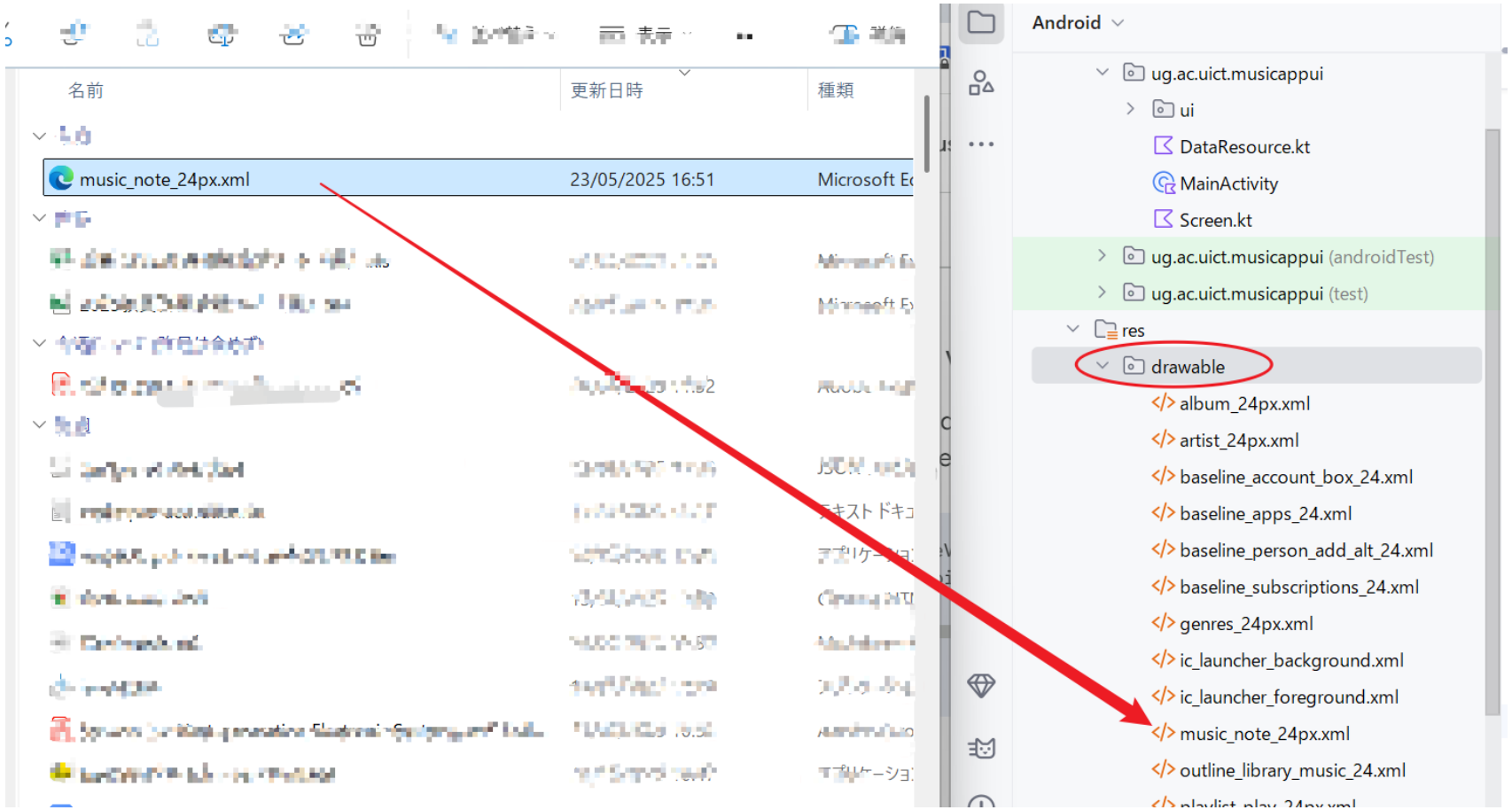
- Import the Icon resource from Google Fonts
- https://fonts.google.com/





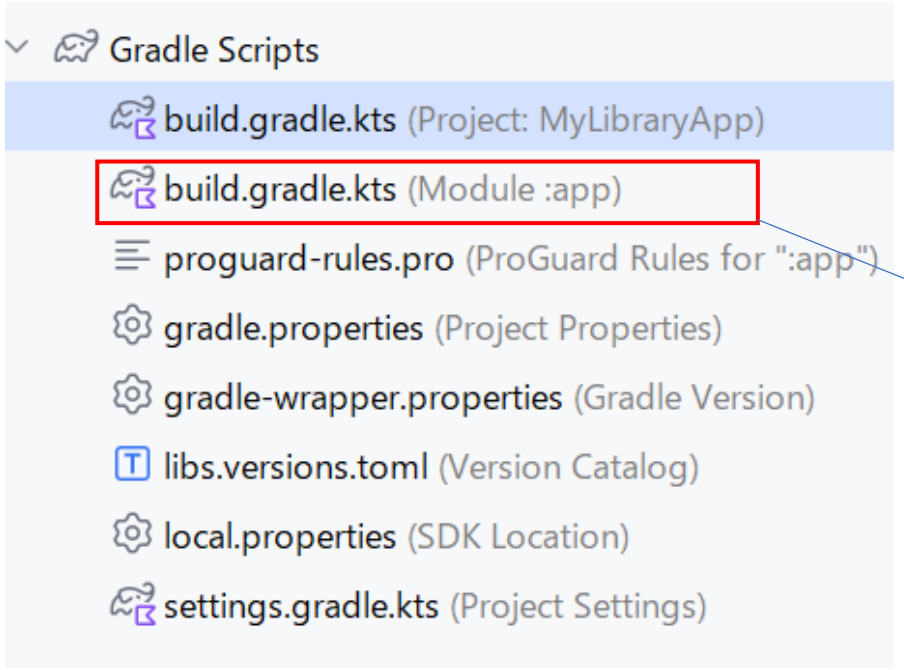
Add the Library View(3)

- Copy the downloaded xml file to res->drawable package



Add the Library View(4)

- Add the dependency in your app-level build.gradle files
- The Appcompat allows access to new APIs on older API versions of the platform (many using Material Design).

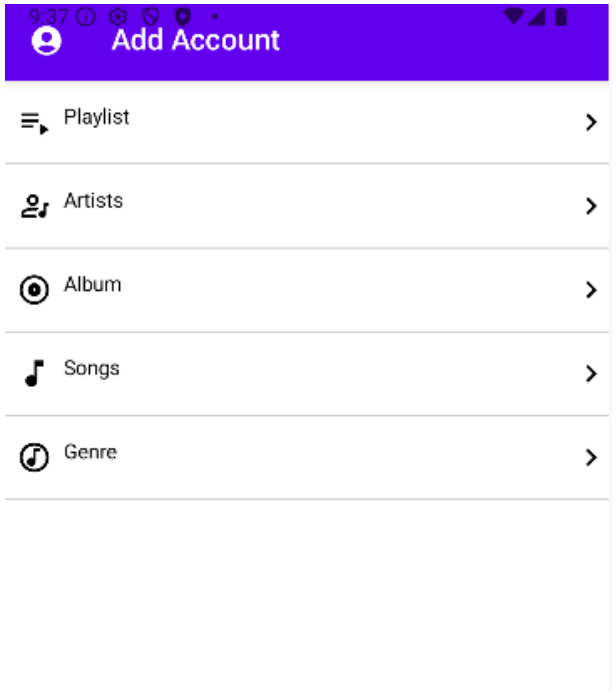



```
implementation("androidx.compose.material:material:$compose_version")
implementation("androidx.compose.ui:ui-tooling-preview:$compose_version")

val appcompat_version = "1.7.0"

implementation("androidx.appcompat:appcompat:$appcompat_version")
// For loading and tinting drawables on older versions of the platform
implementation("androidx.appcompat:appcompat-resources:$appcompat_version")

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

https://developer.android.com/jetpack/androidx/releases/appcompat

Add the Library View(5)

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



LibraryView.kt

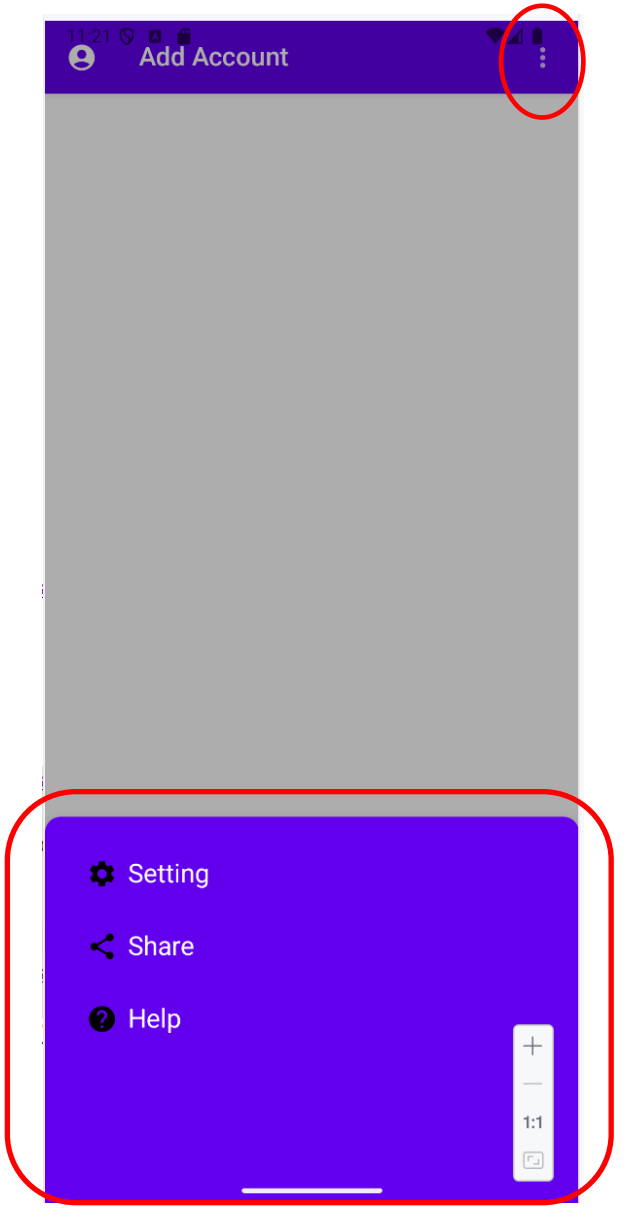
```
@Composable
fun Library(){
    LazyColumn() {
        items(libraries){
            lib ->
            LibItem(lib = lib)
        }

    }
}
@Composable
fun LibItem(lib: Lib){
    Column {
        Row(modifier = Modifier.fillMaxWidth().padding(vertical = 16.dp),
            horizontalArrangement = Arrangement.SpaceBetween){
            Row {
                Icon(painter = painterResource(id=lib.icon), modifier =
                Modifier.padding(horizontal = 8.dp), contentDescription = lib.name)
                Text(text=lib.name)
            }
            Icon(imageVector = Icons.Default.KeyboardArrowRight, contentDescription = "Arrow Right")
        }
        Divider(color = Color.LightGray)
    }
}
```

MainView.kt

```
composable(Screen.BottomScreen.Home.bRoute){
    Home()
}
composable(Screen.BottomScreen.Browse.bRoute){
    Browse()
}
composable(Screen.BottomScreen.Library.bRoute){
    Library()
}
composable(Screen.DrawerScreen.AddAccount.route)
{
```

Bottom sheets

- Bottom sheets show secondary content anchored to the bottom of the screen
- https://m2.material.io/components/sheets-bottom



Add a Bottom sheets to the Scaffold(1)

MainView.kt

```
val modalSheetState = androidx.compose.material.rememberModalBottomSheetState(
    initialValue = ModalBottomSheetValue.Hidden,
    confirmValueChange = { it != ModalBottomSheetValue.HalfExpanded}
)

ModalBottomSheetLayout(
    sheetState = modalSheetState,
    sheetShape = RoundedCornerShape(topStart = 12.dp, topEnd = 12.dp),
    sheetContent = { MoreBottomSheet(modifier = Modifier.fillMaxWidth()) }
) {
    Scaffold(
        bottomBar = bottomBar,
        topBar = {
            TopAppBar(title = {Text(title.value)},
                actions = {
                    IconButton( onClick = {
                        scope.launch {
                            if(modalSheetState.isVisible)
                                modalSheetState.hide()
                            else
                                modalSheetState.show()
                        }
                    }) {
                        Icon(imageVector = Icons.Default.MoreVert, contentDescription = "more")
                    }
                },
                navigationIcon = { IconButton(onClick = {
                    scope.launch {
                        scaffoldState.drawerState.open()
```

Creating and remembering the state for a ModalBottomSheetLayout (a modal bottom sheet component). This state object allows you to control the sheet's visibility (showing/hiding it) and react to its state changes.

ModalBottomSheetLayout is a component from the Jetpack Compose Material library that allows you to implement a modal bottom sheet. This is a panel that slides up from the bottom of the screen, typically to present options or additional information related to the current context. When the sheet is visible, it overlays the main screen content.

Include the all Scaffold() to the ModalBottomSheetLayout()

You want your entire application interface (built with Scaffold) to act as the background, and the MoreBottomSheet that slides up from the bottom should be a temporary, overlaying modal view. ModalBottomSheetLayout is designed precisely for this layout pattern. You put the Scaffold inside to tell ModalBottomSheetLayout: "This is my main screen; please display your sheet on top of it."

Add a Bottom sheets to the Scaffold(2)

MainView.kt

```
val modalSheetState = androidx.compose.material.rememberModalBottomSheetState(
    initialValue = ModalBottomSheetValue.Hidden,
    confirmValueChange = { it != ModalBottomSheetValue.HalfExpanded}
)

ModalBottomSheetLayout(
    sheetState = modalSheetState,
    sheetShape = RoundedCornerShape(topStart = 12.dp, topEnd = 12.dp),
    sheetContent = { MoreBottomSheet(modifier = Modifier.fillMaxWidth()) }
) {
    Scaffold(
        bottomBar = bottomBar,
        topBar = {
            TopAppBar(title = {Text(title.value)},
                actions = {
                    IconButton( onClick = {
                        scope.launch {
                            if(modalSheetState.isVisible)
                                modalSheetState.hide()
                            else
                                modalSheetState.show()
                        }
                    }) {
                        Icon(imageVector = Icons.Default.MoreVert, contentDescription = "more")
                    }
                },
                navigationIcon = { IconButton(onClick = {
                    scope.launch {
                        scaffoldState.drawerState.open()
```

This action item is an IconButton that displays the "more vertical" (three dots) icon.When this icon button is clicked:
A coroutine is launched.
Inside the coroutine, it checks the current visibility of a modal bottom sheet (controlled by modalSheetState).
If the bottom sheet is currently visible, the hide() function is called to animate it closed.
If the bottom sheet is currently hidden, the show() function is called to animate it open.

Add a Bottom sheets to the Scaffold(3)

The UI Block of the Bottom sheets

MainView.kt

```
@Composable
fun MoreBottomSheet(modifier: Modifier){
    Box(Modifier.fillMaxWidth().height(300.dp).background(
        MaterialTheme.colors.primarySurface
    )){
        Column(modifier = modifier.padding(16.dp),
            verticalArrangement = Arrangement.SpaceBetween){
            Row(modifier = modifier.padding(16.dp)) {
                Icon(modifier = Modifier.padding(end = 8.dp),
                    painter = painterResource(id = R.drawable.baseline_settings_24), contentDescription = "Settings"
                )
                Text(text = "Setting", fontSize = 20.sp, color = Color.White)
            }
            Row(modifier = modifier.padding(16.dp)) {
                Icon(modifier = Modifier.padding(end = 8.dp),
                    painter = painterResource(id = R.drawable.baseline_share_24), contentDescription = "Share"
                )
                Text(text = "Share", fontSize = 20.sp, color = Color.White)
            }
            Row(modifier = modifier.padding(16.dp)) {
                Icon(modifier = Modifier.padding(end = 8.dp),
                    painter = painterResource(id = R.drawable.baseline_help_24), contentDescription = "Help"
                )
                Text(text = "Help", fontSize = 20.sp, color = Color.White)
            }
        }
    }
}
```