Mobile Application Development

Week10　Use Scaffold Layout Template

Yi SUN
Kobe Institute of Computing

Understand the Scaffold in Jetpack Compose

Scaffold is a composable that serves as a layout foundation, simplifying the implementation of a screen's structure, including common UI elements like top bars, bottom navigation, floating action buttons, and snackbars. By using Scaffold, you can create a well-organized and consistent layout with ease.

Key Components of Scaffold:

- TopBar: Displays a TopAppBar for navigation, titles, or actions.
- BottomBar: Typically used to display a navigation bar for different sections of the app.
- FloatingActionButton (FAB): A button used to execute primary actions.
- SnackbarHost: A container to display brief messages (snack bars).
- Content: The main body of the UI.

An example of Scaffold

```
@Composable
fun Scaffold(
    modifier: Modifier = Modifier,
    topBar: @Composable () -> Unit = {},
    bottomBar: @Composable () -> Unit = {},
    snackbarHost: @Composable () -> Unit = {},
    floatingActionButton: @Composable () -> Unit = {},
    floatingActionButtonPosition: FabPosition = FabPosition.End,
    containerColor: Color = MaterialTheme.colorScheme.background,
    contentColor: Color = contentColorFor(containerColor),
    contentWindowInsets: WindowInsets = ScaffoldDefaults.contentWindowInsets,
    content: @Composable (PaddingValues) -> Unit
)
```

Ref: https://medium.com/@ramadan123sayed/understanding-scaffold-in-jetpack-compose-a-comprehensive-guide-e248c2406412

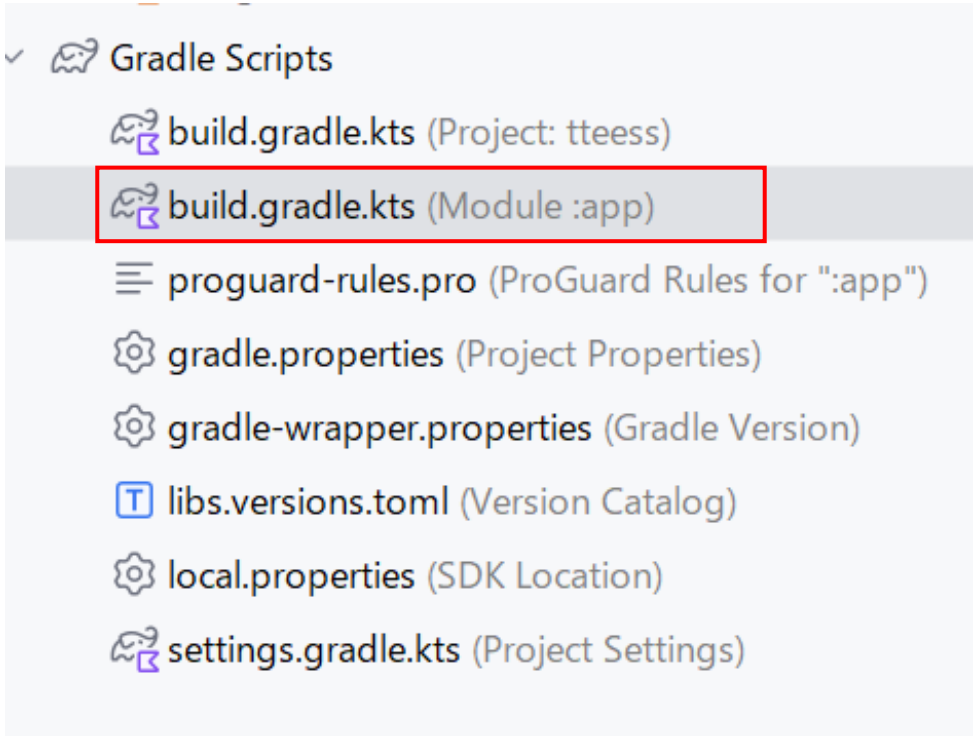
Start from a Sampe application
Personal library management

- Function
  - List up all books in your bookshelf
  - Add the new book to your App
- What you learn
  - Use the TopBar and FloatingActionButton
  - Link the Data with ROOM database for data permanent repository (next week)

1. Open AndroidStudio
2. Select `File > New > NewProject`
3. Choose `Empty Activity`
4. Name: MyLibraryApp

Add the navigation dependencies

- Make sure you've added the correct dependency in your build.gradle (app-level) file:




```
dependencies {
    val nav_version = "2.8.8"

    implementation("androidx.navigation:navigation-compose:$nav_version")
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
```

- Sync the Project:After making these changes, sync your project in Android Studio to download the dependency.
- Cache/Index Issues:If everything seems correct but the dependency still isn't found, try cleaning the project and rebuilding it. You can also invalidate caches via File > Invalidate Caches / Restart... in Android Studio.

Prepare the Topbar view

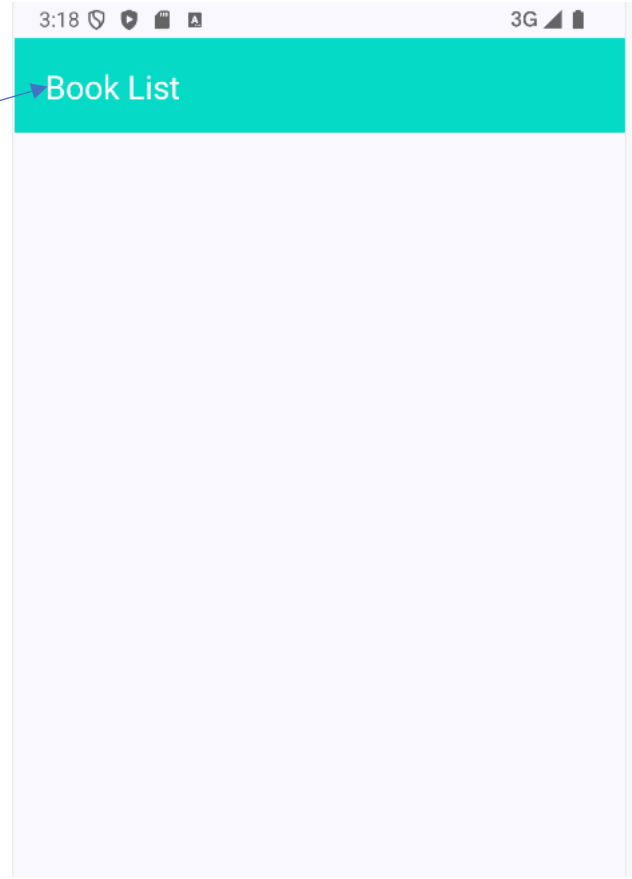
Add a new Kotlin file ***AppBar.kt***

```
@Composable
fun AppBarView(
    title: String,
    onBackNavClicked: ()-> Unit = {}
){
    TopAppBar(title = {
        Text(text= title,
            color= colorResource(id=R.color.white),
            modifier = Modifier.padding(start = 4.dp).heightIn(max=24.dp)
        )},
        modifier = Modifier.statusBarsPadding(),
        colors = TopAppBarDefaults.topAppBarColors(
            containerColor = colorResource(id=R.color.teal_200)
        )
    )
}
```

Prepare the Home view

Add a new Kotlin file ***HomeView.kt***

```
@Composable
fun HomeView(modifier: Modifier = Modifier){
    Scaffold(
        topBar ={ AppBarView(title="Book List") }
    ) {
        LazyColumn(modifier = Modifier.fillMaxSize().padding(it)) {  }
    }
}
```



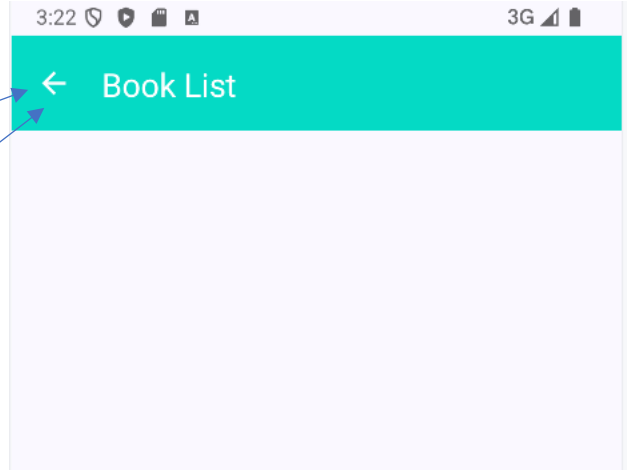
Modify the ***MainActivity.kt*** file

```
MyLibraryAppTheme {
            Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                HomeView(modifier = Modifier.padding(innerPadding))
            }
        }
    }
}
```

Add a NavigationIcon to AppBar

AppBar.kt

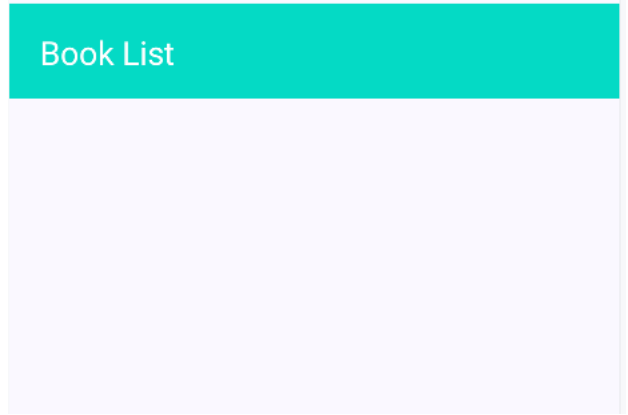
```
fun AppBarView(
    title: String,
    onBackNavClicked: ()-> Unit = {}
){
    val navigationIcon : @Composable (()->Unit) ={
        IconButton(onClick = {onBackNavClicked()}) {
            Icon(
                imageVector = Icons.AutoMirrored.Filled.ArrowBack,
                tint= Color.White,
                contentDescription = "Back icon"
            )
        }
    }
    TopAppBar(title = {
        Text(text= title,
            color= colorResource(id=R.color.white),
            modifier = Modifier.padding(start = 4.dp).heightIn(max=24.dp)
        )},
        modifier = Modifier.statusBarsPadding(),
        navigationIcon=navigationIcon,
        colors = TopAppBarDefaults.topAppBarColors(
            containerColor = colorResource(id=R.color.teal_200)
```



Only show the BackIcon when not on the home screen

AppBar.kt

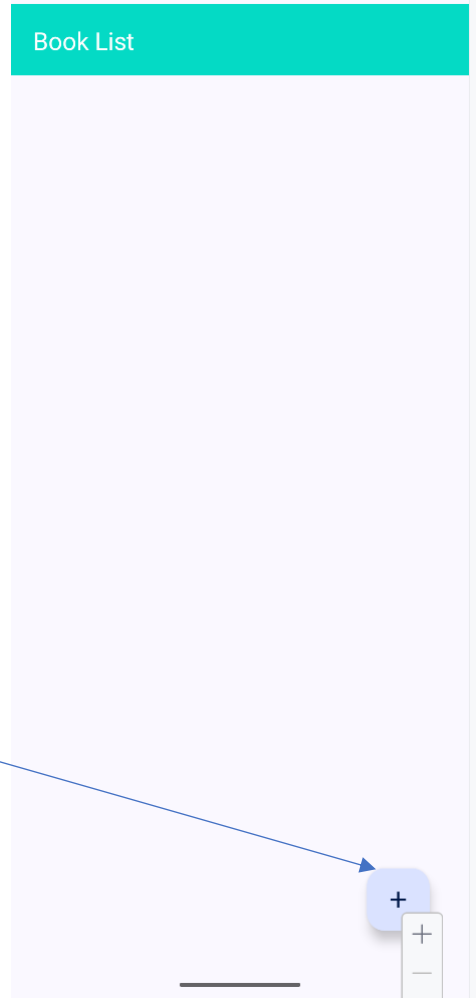
```
fun AppBarView(
    title: String,
    onBackNavClicked: ()-> Unit = {}
){
    val navigationIcon : @Composable (()->Unit)? =
        if(!title.contains("Book List")){
        {
            IconButton(onClick = {onBackNavClicked()}) {
                Icon(
                    imageVector = Icons.AutoMirrored.Filled.ArrowBack,
                    tint= Color.White,
                    contentDescription = "Back icon"
                )
            }
        }}else{
            null
        }
    TopAppBar(title = {
        Text(text= title,
            color= colorResource(id=R.color.white),
            modifier = Modifier.padding(start = 4.dp).heightIn(max=24.dp)
        )},
        modifier = Modifier.statusBarsPadding(),
        navigationIcon=navigationIcon ?: {},
        colors = TopAppBarDefaults.topAppBarColors(
            containerColor = colorResource(id=R.color.teal_200)
        )
```



Add a FloatingActionButton at Home view

Homeview.kt

```
fun HomeView(modifier: Modifier){
    Scaffold(
        topBar ={ AppBarView(title="Book List") },
        floatingActionButton = {
            FloatingActionButton(
                onClick = {},
                modifier = Modifier.padding(all = 20.dp),
            ) {
                Icon(
                    imageVector = Icons.Default.Add, contentDescription = "add button"
                )
            }
        }
    ) {
        LazyColumn(modifier = Modifier.fillMaxSize().padding(it)) {  }
    }
}
```



Prepare the data class of books

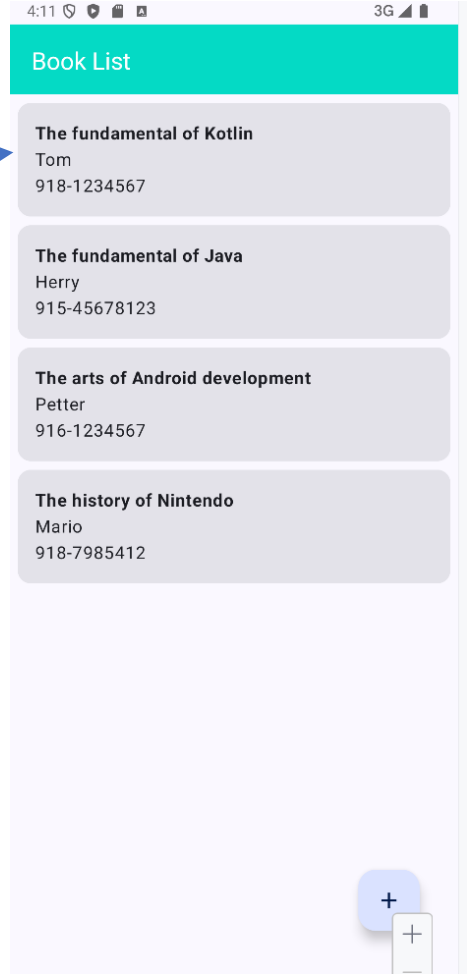
Add a new file ***Book.kt***

```
data class Book(
    val id: Long = 0L,
    val title: String="",
    val isbn: String="",
    val author: String="",
)
```

Add a card view for each book item

Homeview.kt

```
@Composable
fun BookItem(book:Book, onClick: ()-> Unit){
    Card(modifier=Modifier.fillMaxSize()
        .padding(top = 8.dp, start = 8.dp, end= 8.dp)
        .clickable { onClick() }
    ){
        Column(modifier = Modifier.padding(16.dp)) {
            Text(text= book.title, fontWeight = FontWeight.Bold)
            Text(text=book.author)
            Text(text=book.isbn)
        }
    }
}
```



prepare the Dummy book data

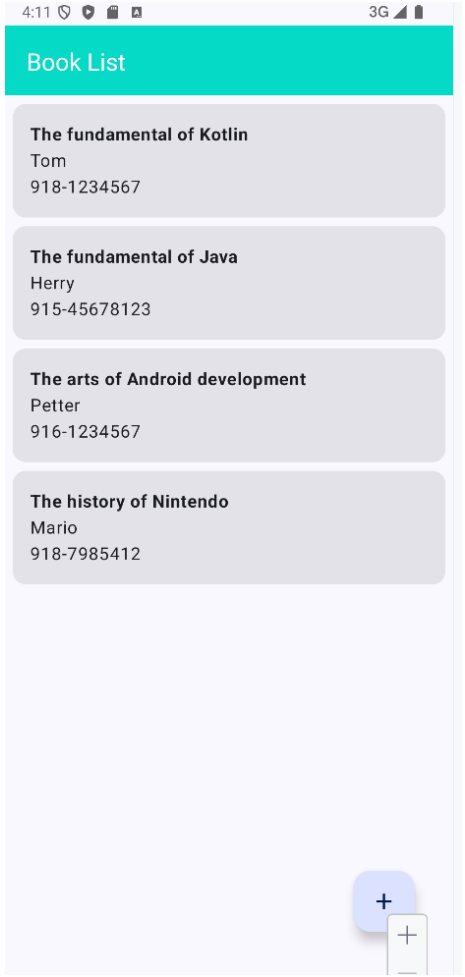
Book.kt

```
object Dummybook{
    val bookList = listOf(
        Book(title="The fundamental of Kotlin", isbn="918-1234567", author="Tom"),
        Book(title="The fundamental of Java", isbn="915-45678123", author="Herry"),
        Book(title="The arts of Android development", isbn="916-1234567", author="Petter"),
        Book(title="The history of Nintendo", isbn="918-7985412", author="Mario"),
    )
}
```

Display the Dummy data in home view

Homeview.kt

```
@Composable
fun HomeView(modifier: Modifier){
    Scaffold(
        topBar ={ AppBarView(title="Book List") },
        floatingActionButton = {
            FloatingActionButton(
                onClick = {},
                modifier = Modifier.padding(all = 20.dp),
            ) {
                Icon(
                    imageVector = Icons.Default.Add, contentDescription = "add button"
                )
            }
        }
    ) {
        LazyColumn(modifier = Modifier.fillMaxSize().padding(it)) {
            items(Dummybook.bookList){
                book -> BookItem(book = book) { }
            }
        }
    }
}
```



Prepare the Navigation

Add a ***Screen.kt*** file to store the route of Screens

```
sealed class Screen(val route:String) {
    object HomeScreen: Screen("home_screen")
    object AddScreen: Screen("add_screen")
}
```

Add a BookViewModel class ***BookViewModel.kt*** file to present the BookView

```
class BookViewModel:ViewModel() {
}
```

Add a ***Navigation.kt*** file to manage the screen navi

```
@Composable
fun Navigation(
    viewModel: BookViewModel = viewModel(),
    navController: NavHostController = rememberNavController(),
    modifier: Modifier
) {
    NavHost(
        navController = navController,
        startDestination = Screen.HomeScreen.route
    ){
        composable(Screen.HomeScreen.route){ HomeView(modifier)}
    }
}
```

Modify the MainActivity and HomeView

MainActivity.kt

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            MyLibraryAppTheme {
                MyLibraryAppTheme {
                    Navigation(modifier = Modifier.fillMaxSize())
                }
            }
        }
    }
}
```

Homeview.kt

```
@Composable
fun HomeView(modifier: Modifier){
    Scaffold(
        modifier = modifier,
        topBar ={ AppBarView(title="Book List") },
        floatingActionButton = {
            FloatingActionButton(
                onClick = {},
                modifier = Modifier.padding(all = 20.dp),
            ) {
```

Add a Book info edit(add) view

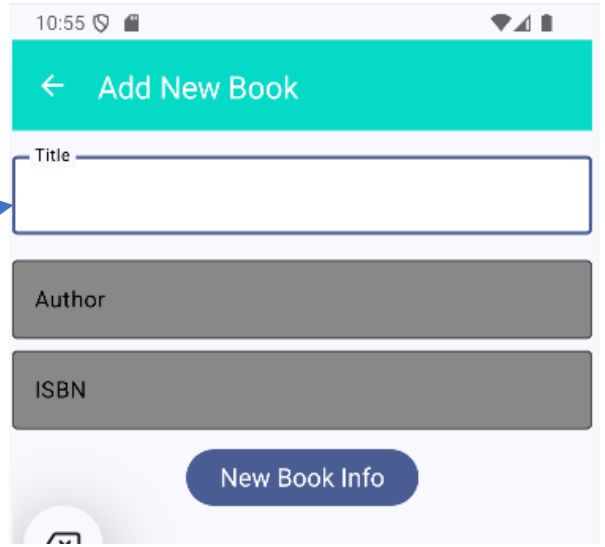
Add a ***AddEditDetailView.kt*** file to present the edit book info view

```
@Composable
fun AddEditDetailView(
    id: Long,
    viewModel: BookViewModel,
    navController: NavController
){
    Scaffold(
        topBar = {AppBarView(title = if(id != 0L) "Update Book info" else "Add New Book")}
    )
        {
        Column(modifier = Modifier.padding(it).wrapContentSize(),
            horizontalAlignment =  Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center)
        {
            Spacer(modifier = Modifier.height(10.dp))
        }
    }
}
```

Design the OutlinedTextField

AddEditDetailView.kt

```
@Composable
fun BookTextField(
    label:String,
    value:String,
    onValueChanged: (String) -> Unit
){
    OutlinedTextField(
        value = value,
        onValueChange = onValueChanged,
        label = { Text(text = label, color = Color.Black)},
        modifier = Modifier.fillMaxWidth(),
        keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Text),
        colors = TextFieldDefaults.colors(
            focusedTextColor = Color.Black,
                    unfocusedTextColor = Color.Black,
                    cursorColor = Color.Blue,
            focusedContainerColor = Color.White,
            unfocusedContainerColor = Color.White
        )
    )
}
```



Modify the BookViewModel

BookViewModel.kt

```
class BookViewModel:ViewModel() {
    var bookTitleState by mutableStateOf("")
    var bookAuthorState by mutableStateOf("")
    var bookIsbnState by mutableStateOf("")

    fun onBookTitleChanged(newString:String){
        bookTitleState = newString
    }
    fun onBookAuthorChanged(newString:String){
        bookAuthorState = newString
    }
    fun onBookIsbnChanged(newString:String){
        bookIsbnState = newString
    }

}
```

Modify the AddEditDetailView to design the screen

AddEditDetailView.kt

```
{
    Spacer(modifier = Modifier.height(10.dp))
    BookTextField(
        label = "Title",
        value = viewModel.bookTitleState,
        onValueChanged = { viewModel.onBookTitleChanged(it) }
    )
    Spacer(modifier = Modifier.height(10.dp))
    BookTextField(
        label = "Author",
        value = viewModel.bookAuthorState,
        onValueChanged = { viewModel.onBookAuthorChanged(it) }
    )
    Spacer(modifier = Modifier.height(10.dp))
    BookTextField(
        label = "ISBN",
        value = viewModel.bookIsbnState,
        onValueChanged = { viewModel.onBookIsbnChanged(it) }
    )
    Spacer(modifier = Modifier.height(10.dp))
}
...
```

```
 Spacer(modifier = Modifier.height(10.dp))
    Button(onClick = {
        if(viewModel.bookTitleState.isNotEmpty()&&
            viewModel.bookAuthorState.isNotEmpty()&&
            viewModel.bookIsbnState.isNotEmpty()){

        }else{}
    }){
        Text(
            text = if (id != 0L) "Update Book Info" else "New Book Info",
            style = TextStyle(fontSize = 18.sp)
            )
    }
}
```

Modify the Navigation

Navigation.kt

```
@Composable
fun Navigation(
    viewModel: BookViewModel = viewModel(),
    navController: NavHostController = rememberNavController(),
    modifier: Modifier
) {
    NavHost(
        navController = navController,
        startDestination = Screen.HomeScreen.route
    ){
        composable(Screen.HomeScreen.route){ HomeView(modifier,navController,viewModel)}
        composable(Screen.AddScreen.route){
            AddEditDetailView(id = 0L, viewModel = viewModel, navController=navController )
        }
    }
}
```

Add the onClick event in HomeView

Homeview.kt

```
@Composable
fun HomeView(modifier: Modifier,
             navController: NavController,
             viewModel: BookViewModel
){
    Scaffold(
        modifier = modifier,
        topBar ={ AppBarView(title="Book List") },
        floatingActionButton = {
            FloatingActionButton(
                onClick = {navController.navigate(Screen.AddScreen.route)},
                modifier = Modifier.padding(all = 20.dp),
            ) {
                Icon(
                    imageVector = Icons.Default.Add, contentDescription = "add button"
                )
```

Add the Back Icon event by navigatUp()

AddEditDetailView.kt

```
fun AddEditDetailView(
    id: Long,
    viewModel: BookViewModel,
    navController: NavController
){
    Scaffold(
        topBar = {
            AppBarView(title = if(id != 0L) "Update Book info" else "Add New Book")
            {navController.navigateUp()}
        }
    )
        {
        Column(modifier = Modifier.padding(it).wrapContentSize(),
```