FROM XML TO COMPOSE

MY JOURNEY OF TRANSFORMING AN EXISTING LARGE APP TO JETPACK COMPOSE

AHMED TIKIWA | SENIOR SOFTWARE ENGINEER - ANDROID @ LUNO | @AKITIKKX

WHY COMPOSE?

WHAT ARE OTHER COMPANIES SAYING?

WHAT ARE OTHER COMPANIES SAYING?

"It's much easier to trace through code when it's all written in the same language [Kotlin] and often the same file, rather than jumping back and forth between Kotlin and XML"

- Monzo

WHAT ARE OTHER COMPANIES SAYING?

"Our theming layer is vastly more intuitive and legible. We've been able to accomplish within a single Kotlin file what otherwise extended across multiple XML files that were responsible for attribute definitions and assignments via multiple layered theme overlays."

- Twitter

WHEN WAS JETPACK COMPOSE INTRODUCED

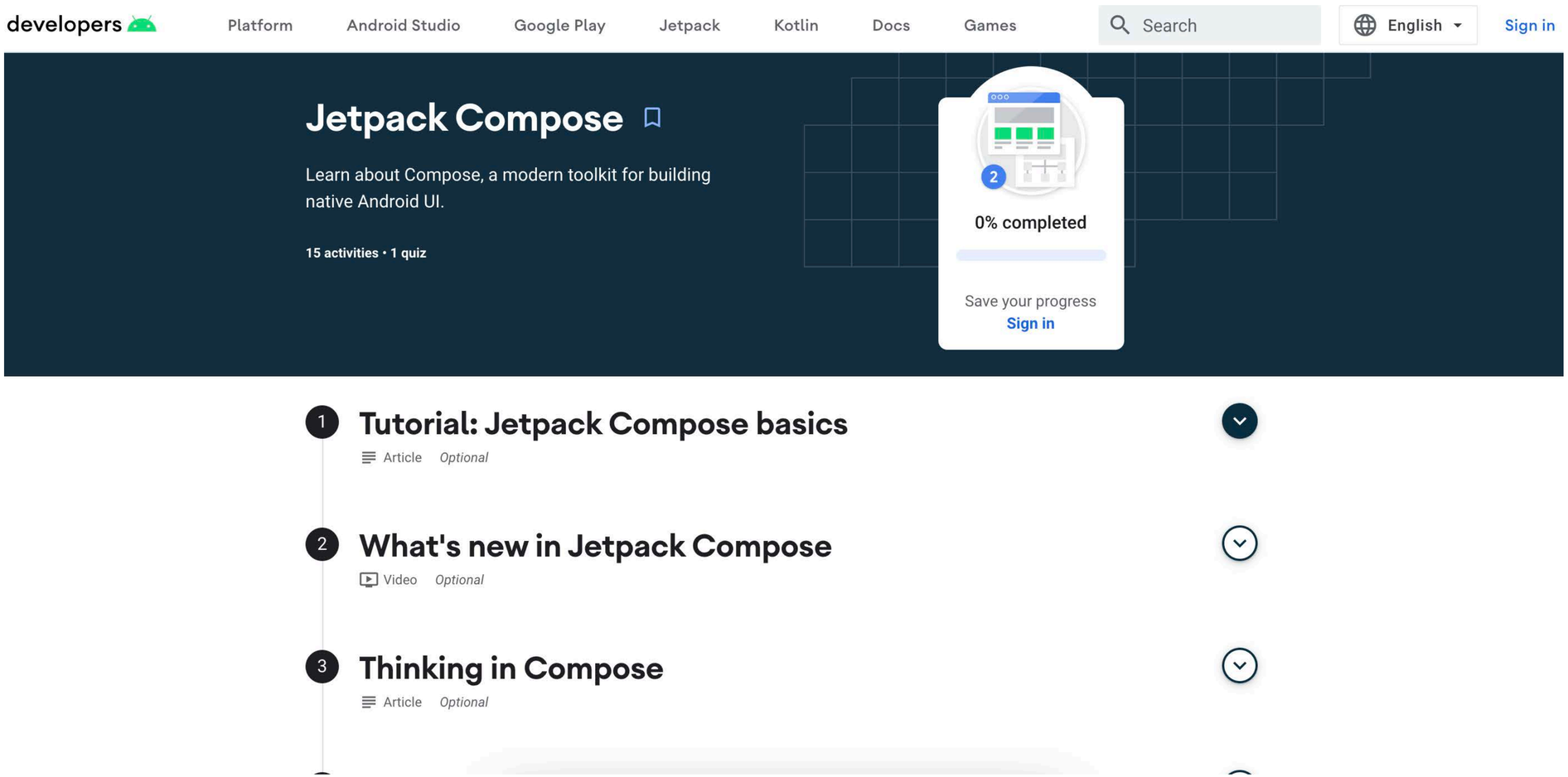
WHEN WAS JETPACK COMPOSE INTRODUCED

- Announced at Google I/O 2019 as a preview
- *“One of the areas we never solved was UI. We really wanted to look at how could you make it super simple to develop UI.”*
  - Karen Ng, Group Product Manager at Google
- *“What I think is once people start seeing Compose in action, it really becomes a delightful thing to program”*
  - Leland Richardson, Software Engineer at Google
- Excited about new library

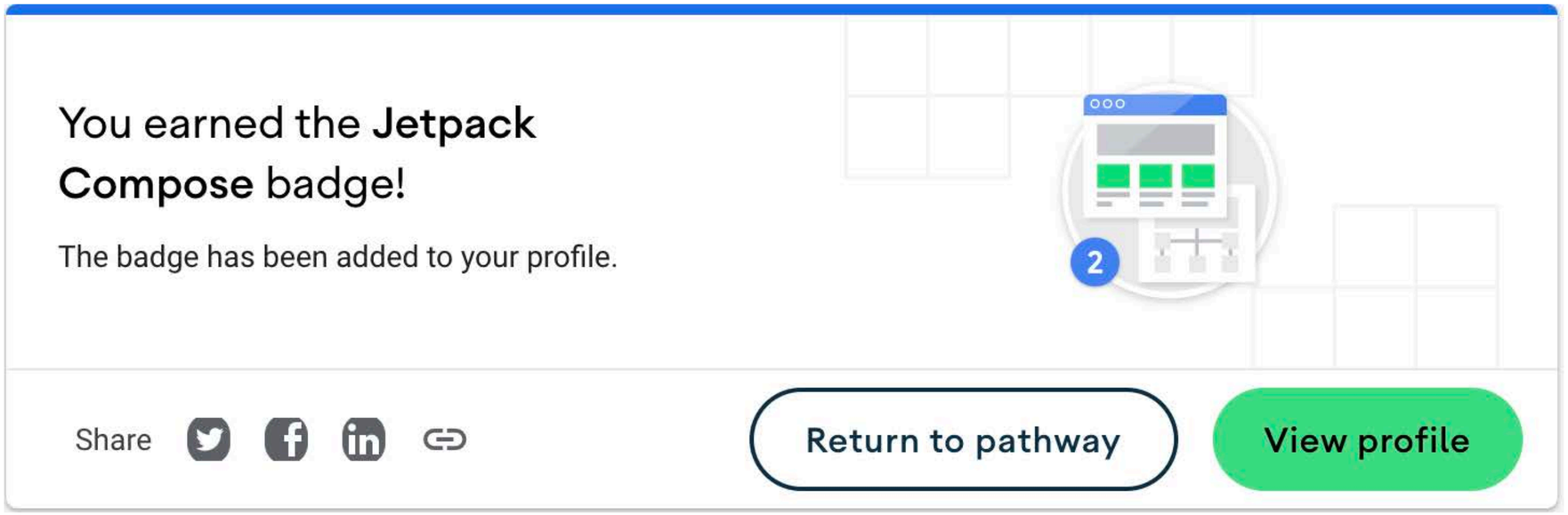
SOURCE: HTTPS://EVENTS.GOOGLE.COM/IO2019/RECAP/

LEARNING COMPOSE

LEARNING COMPOSE



LEARNING COMPOSE



LEARNING COMPOSE

Jetpack Compose Crash course for Android with Kotlin

Modern Android apps with **Jetpack Compose** and integrations: MVVM, Coroutines, ViewModel, LiveData, Retrofit, Navigation

Catalin Ghita

4.6 ★★★★½ (829)

11 total hours • 94 lectures • All Levels

Android Jetpack Compose: The Comprehensive Bootcamp [2022]

Kotlin **Jetpack Compose**: Firebase Firestore, Hilt & Dagger, ROOM DB, ViewModel, Retrofit, Navigation & Clean Architecture

Paulo Dichone | Android, Java, Flutter Developer and Teacher

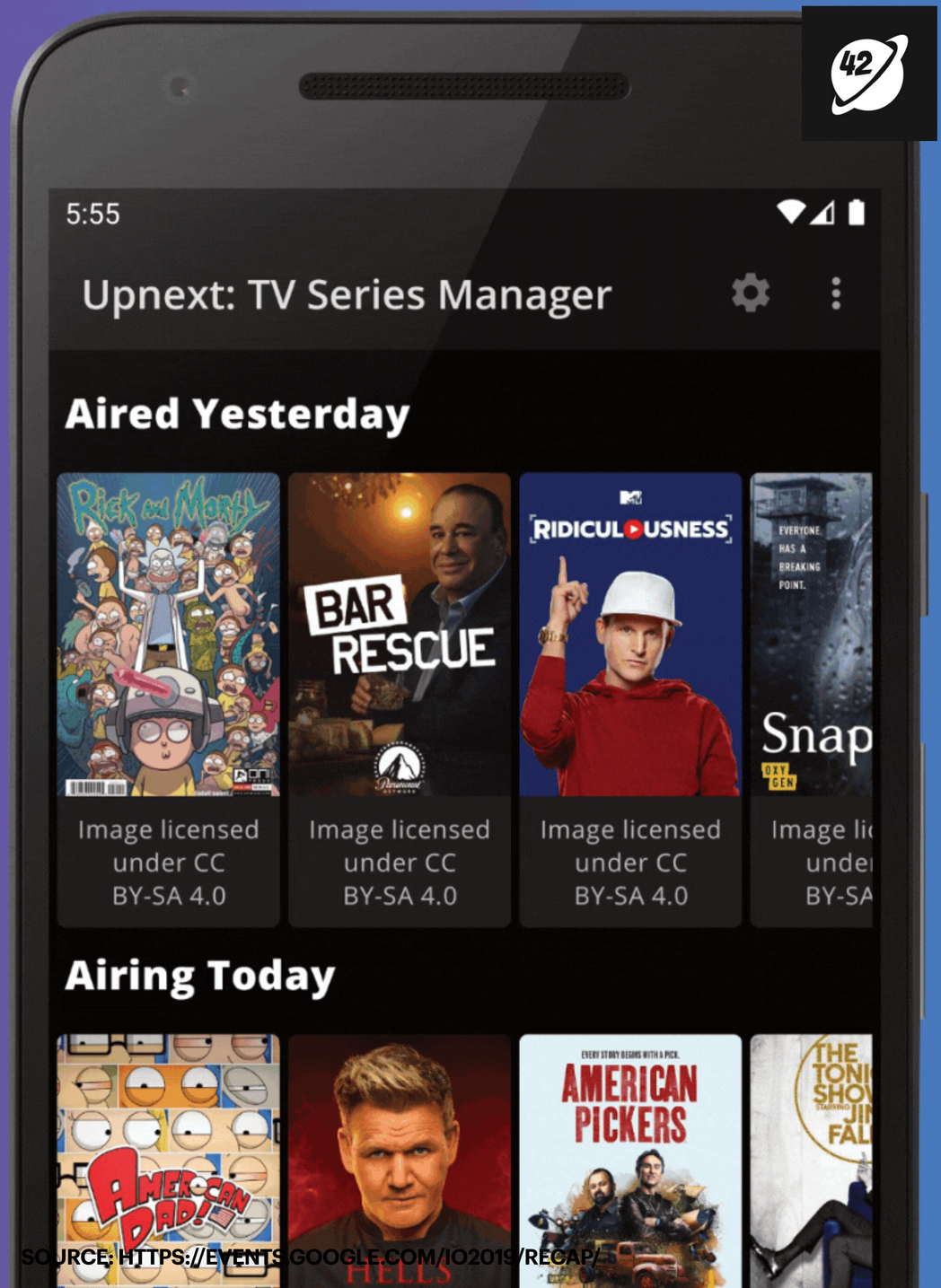
4.7 ★★★★½ (201)

36.5 total hours • 284 lectures • All Levels

Bestseller

ABOUT THE APP

- Created in 2015, available on Google Play Store
- Dashboard screen
- Search screen
- Explore Screen
- Show Detail Screen
- Seasons list
- Episodes list
- Trakt account screen



SOURCE: HTTPS://EVENTS.GOOGLE.COM/IO2019/RECAP/

ADOPTING COMPOSE: CHOOSING THE APPROACH

ADOPTING COMPOSE: CHOOSING THE APPROACH

- **The bottom-up approach** starts migrating the smaller UI elements on the screen, like a Button or a TextView, followed by its ViewGroup elements until everything is converted to composable functions.

- **The top-down approach** starts migrating the fragments or view containers, like a FrameLayout, ConstraintLayout, or RecyclerView, followed by the smaller UI elements on the screen.

WHAT DOES INTEROPERABILITY LOOK LIKE

WHAT DOES INTEROPERABILITY LOOK LIKE

- Each fragment has an associated XML layout
- Gradual migration
- Use of ComposeView
  - Introduce Compose UI content into XML layout
  - A container

WHAT DOES INTEROPERABILITY LOOK LIKE

- Removed
  - TextViews
  - RecyclerViews
  - NestedScrollView
  - ConstraintLayout
  - LinearProgressIndicator
- Added
  - ComposeView

```
<layout ...>

    <androidx.compose.ui.platform.ComposeView
        android:id="@+id/compose_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</layout>
```

WHAT DOES INTEROPERABILITY LOOK LIKE

- Referenced ComposeView with data-binding
- setViewCompositionStrategy()
- setContent()

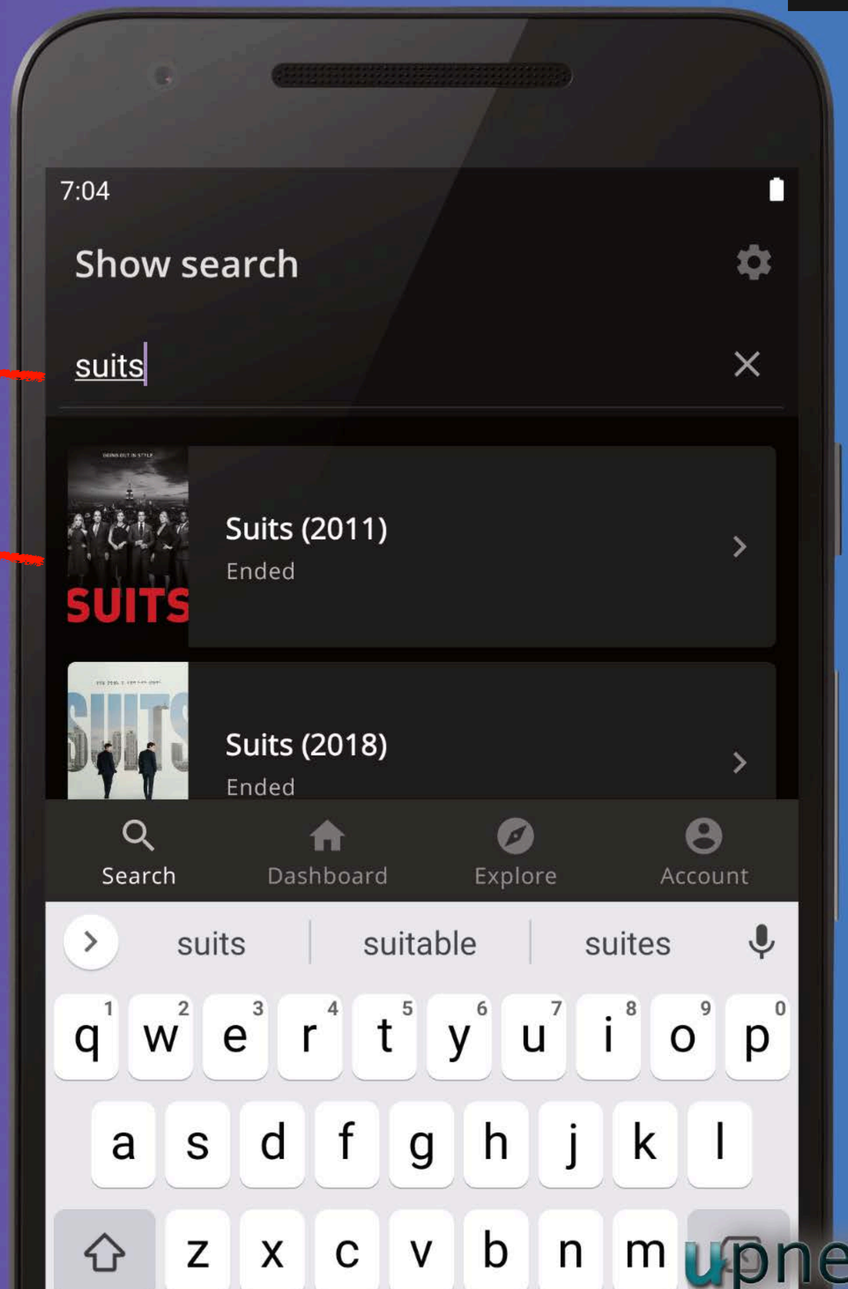
```
@OptIn(
    ExperimentalMaterialApi::class,
    ExperimentalComposeUiApi::class
)
override fun onCreateView(
    ...
): View {
    ...

    binding.composeContainer.apply {
        setViewCompositionStrategy(
            ViewCompositionStrategy.DisposeOnViewTreeLifecycleDestroyed
        )
        setContent {
            MdcTheme {
                SearchScreen(navController = findNavController())
            }
        }
    }

    return binding.root
}
```

BREAKING DOWN THE SCREEN CHANGES

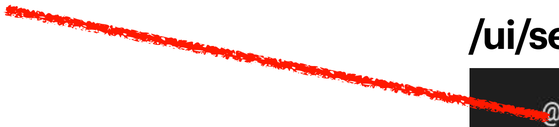
SEARCH SCREEN

BREAKING DOWN THE CHANGES: SEARCH SCREEN

- Focus on:
  - Search bar
  - Search results list
    - The search result item
    - Displaying the list
  - Removing RecyclerView adapter, ViewHolder and item layout



BREAKING DOWN THE CHANGES: SEARCH SCREEN



/ui/search/SearchScreen.kt

```
@Composable
fun SearchScreen(
    viewModel: SearchViewModel = hiltViewModel(),
    navController: NavController
) {
    val searchResultsList = viewModel.searchResponse.observeAsState()

    val isLoading = viewModel.isLoading.observeAsState()

    Surface {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(8.dp)
        ) {


        }
    }
}
```

WHAT IS A COMPOSABLE?

- Built around composable functions
- Define app’s UI
- Provide data to be displayed
- No more focus on UI construction process

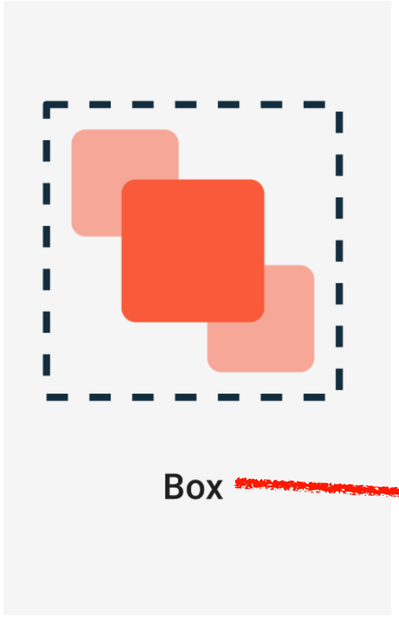
BREAKING DOWN THE CHANGES: SEARCH SCREEN

/ui/search/SearchScreen.kt

```
@Composable
fun SearchScreen(
    viewModel: SearchViewModel = hiltViewModel(),
    navController: NavController
) {
    val searchResultsList = viewModel.searchResponse.observeAsState()

    val isLoading = viewModel.isLoading.observeAsState()

    Surface(modifier = Modifier.fillMaxSize()) {
        Column(
            modifier = Modifier.padding(8.dp)
        ) {



        }
    }
}
```

Column

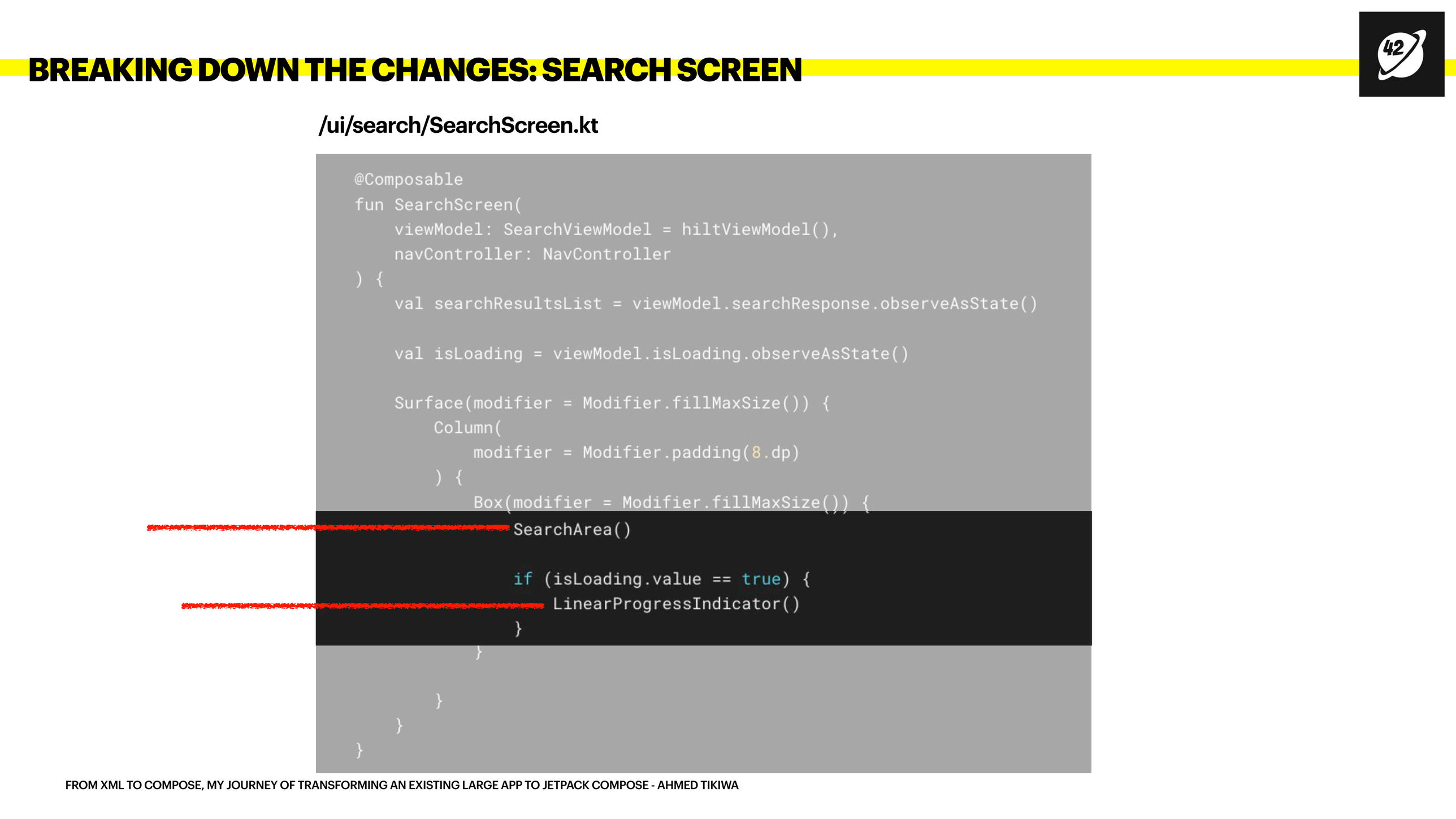
BREAKING DOWN THE CHANGES: SEARCH SCREEN



/ui/search/SearchScreen.kt

```
@Composable
fun SearchScreen(
    viewModel: SearchViewModel = hiltViewModel(),
    navController: NavController
) {
    val searchResultsList = viewModel.searchResponse.observeAsState()

    val isLoading = viewModel.isLoading.observeAsState()

    Surface(modifier = Modifier.fillMaxSize()) {
        Column(
            modifier = Modifier.padding(8.dp)
        ) {
            Box(modifier = Modifier.fillMaxSize()) {

            }
        }
    }
}
```

BREAKING DOWN THE CHANGES: SEARCH SCREEN



/ui/search/SearchScreen.kt

```
@Composable
fun SearchScreen(
    viewModel: SearchViewModel = hiltViewModel(),
    navController: NavController
) {
    val searchResultsList = viewModel.searchResponse.observeAsState()

    val isLoading = viewModel.isLoading.observeAsState()

    Surface(modifier = Modifier.fillMaxSize()) {
        Column(
            modifier = Modifier.padding(8.dp)
        ) {
            Box(modifier = Modifier.fillMaxSize()) {
                SearchArea()

                if (isLoading.value == true) {
                    LinearProgressIndicator()
                }
            }

        }
    }
}
```

BREAKING DOWN THE CHANGES: SEARCH SCREEN

/ui/search/SearchScreen.kt

```
@OptIn(ExperimentalComposeUiApi::class, ExperimentalMaterialApi::class)
@Composable
fun SearchArea(
    searchResultsList: List<ShowSearch>?,
    onTextSubmit: (query: String) -> Unit,
    onResultClick: (item: ShowSearch) -> Unit
) {
    Column(modifier = Modifier.padding(top = 8.dp)) {
        SearchForm {
            onTextSubmit(it)
        }

        searchResultsList?.let { results ->
            SearchResultsList(list = results) {
                onResultClick(it)
            }
        }
    }
}
```

BREAKING DOWN THE CHANGES: SEARCH SCREEN

/ui/search/SearchScreen.kt

```kotlin
@OptIn(ExperimentalComposeUiApi::class, ExperimentalMaterialApi::class)
@Composable
fun SearchArea(
    searchResultsList: List<ShowSearch>?,
    onTextSubmit: (query: String) -> Unit,
    onResultClick: (item: ShowSearch) -> Unit
) {
    Column(modifier = Modifier.padding(top = 8.dp)) {
        SearchForm {
            onTextSubmit(it)
        }

        searchResultsList?.let { results ->
            SearchResultsList(list = results) {
                onResultClick(it)
            }
        }
    }
}
```

```kotlin
@ExperimentalComposeUiApi
@Composable
fun SearchForm(
    onSearch: (String) -> Unit
) {
    val searchQueryState = rememberSaveable { mutableStateOf("") }

    SearchInputField(
        inputLabel = stringResource(id = R.string.search_input_hint),
        valueState = searchQueryState,
        onValueChange = {
            onSearch(searchQueryState.value.trim())
        }
    )

}
```

BREAKING DOWN THE CHANGES: SEARCH SCREEN

/ui/search/SearchScreen.kt

```
@ExperimentalComposeUiApi
@Composable
fun SearchForm(
    onSearch: (String) -> Unit
) {
    val searchQueryState = rememberSaveable { mutableStateOf("") }

    SearchInputField(
        inputLabel = stringResource(id = R.string.search_input_hint),
        valueState = searchQueryState,
        onValueChange = {
            onSearch(searchQueryState.value.trim())
        }
    )

}
```

```
@Composable
fun SearchInputField(
    modifier: Modifier = Modifier,
    inputLabel: String,
    valueState: MutableState<String>,
    onValueChange: (value: String) -> Unit
) {
    OutlinedTextField(
        value = valueState.value,
        onValueChange = {
            valueState.value = it
            onValueChange(valueState.value)
        },
        label = { Text(inputLabel) },
        singleLine = true,
        modifier = modifier
            .padding(8.dp)
            .fillMaxWidth()
    )
}
```

BREAKING DOWN THE CHANGES: SEARCH SCREEN

/ui/search/SearchScreen.kt

```kotlin
@OptIn(ExperimentalComposeUiApi::class, ExperimentalMaterialApi::class)
@Composable
fun SearchArea(
    searchResultsList: List<ShowSearch>?,
    onTextSubmit: (query: String) -> Unit,
    onResultClick: (item: ShowSearch) -> Unit
) {
    Column(modifier = Modifier.padding(top = 8.dp)) {
        SearchForm {
            onTextSubmit(it)
        }

        searchResultsList?.let { results ->
            SearchResultsList(list = results) {
                onResultClick(it)
            }
        }
    }
}
```

```kotlin
@ExperimentalMaterialApi
@Composable
fun SearchResultsList(
    list: List<ShowSearch>,
    onClick: (item: ShowSearch) -> Unit
) {
    LazyColumn {
        items(list) {
            SearchListCard(item = it) {
                onClick(it)
            }
        }
    }
}
```

BREAKING DOWN THE CHANGES: SEARCH SCREEN

/ui/search/SearchScreen.kt

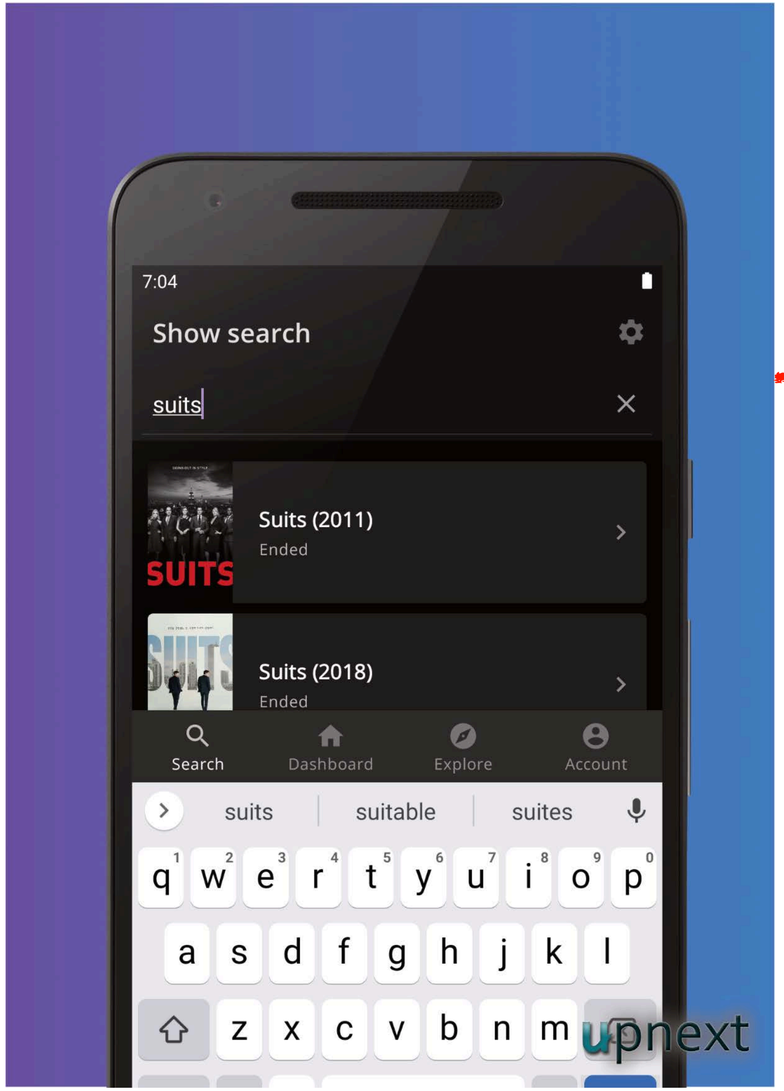
```
@Composable
fun SearchScreen(
    viewModel: SearchViewModel = hiltViewModel(),
    navController: NavController
) {
    val searchResultsList = viewModel.searchResponse.observeAsState()

    val isLoading = viewModel.isLoading.observeAsState()

    Surface(modifier = Modifier.fillMaxSize()) {
        Column(
            modifier = Modifier.padding(8.dp)
        ) {
            Box(modifier = Modifier.fillMaxSize()) {
                SearchArea()

                if (isLoading.value == true) {
                    LinearProgressIndicator()
                }
            }

        }
    }
}
```

```
SearchArea(
    searchResultsList = searchResultsList.value,
    onResultClick = {
        val directions =
            SearchFragmentDirections.actionSearchFragmentToShowDetailFragment(
                ShowDetailArg(
                    source = "search",
                    showId = it.id,
                    showTitle = it.name,
                    showImageUrl = it.originalImageUrl,
                    showBackgroundUrl = it.mediumImageUrl
                )
            )
        navController.navigate(directions)
    },
    onTextSubmit = {
        viewModel.onQueryTextSubmit(it)
    }
)
```

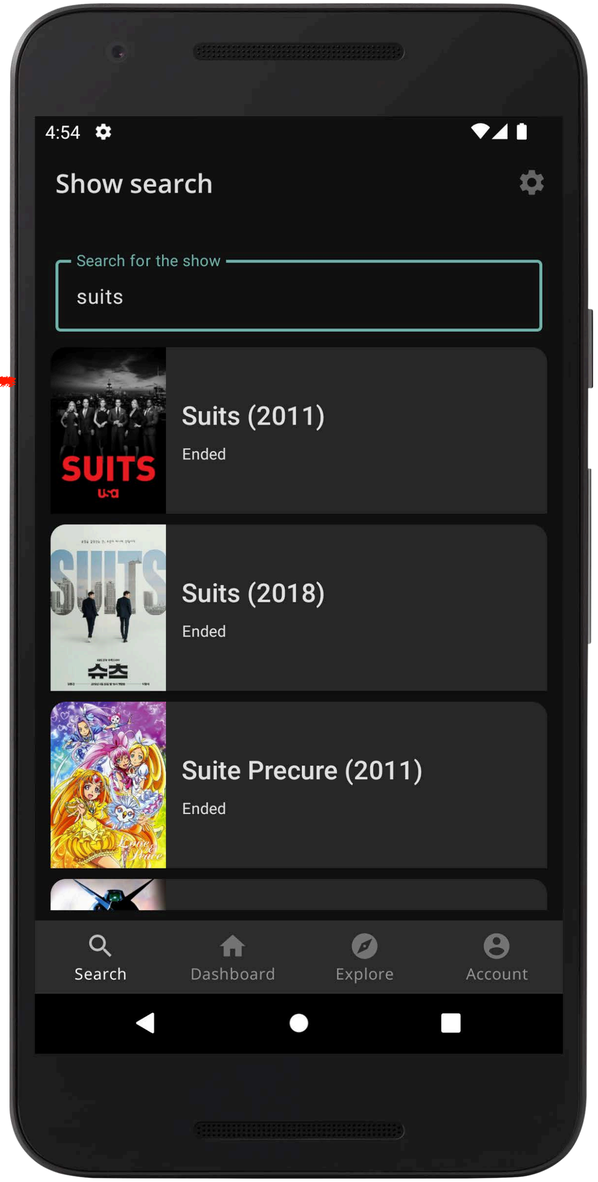
BEFORE AND AFTER'S

BEFORE AND AFTER: SEARCH SCREEN

SearchFragment.kt



@Composable SearchScreen()

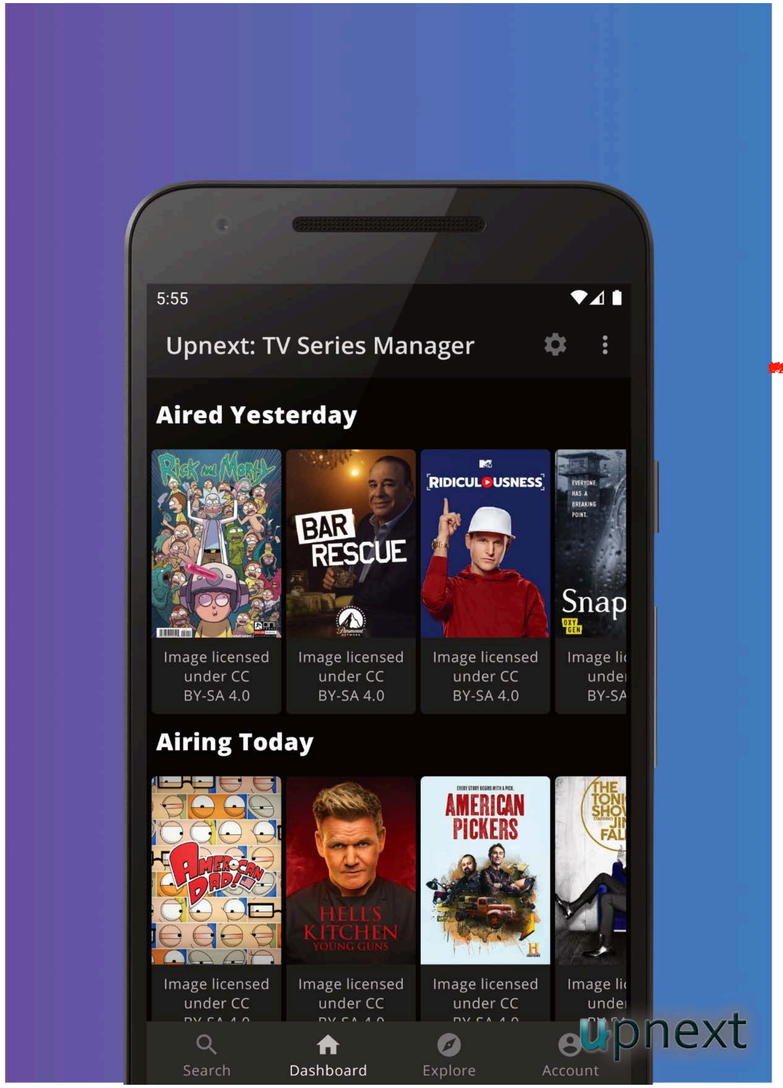


@Composable SearchForm()

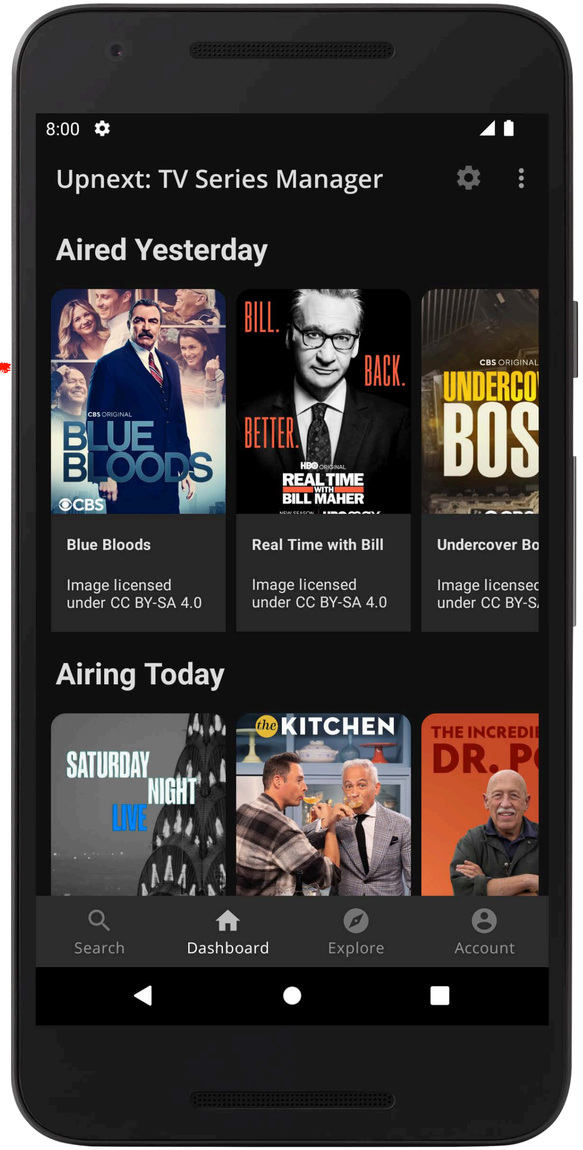
@Composable SearchResultsList()

BEFORE AND AFTER: DASHBOARD SCREEN

DashboardFragment.kt



@Composable DashboardScreen()

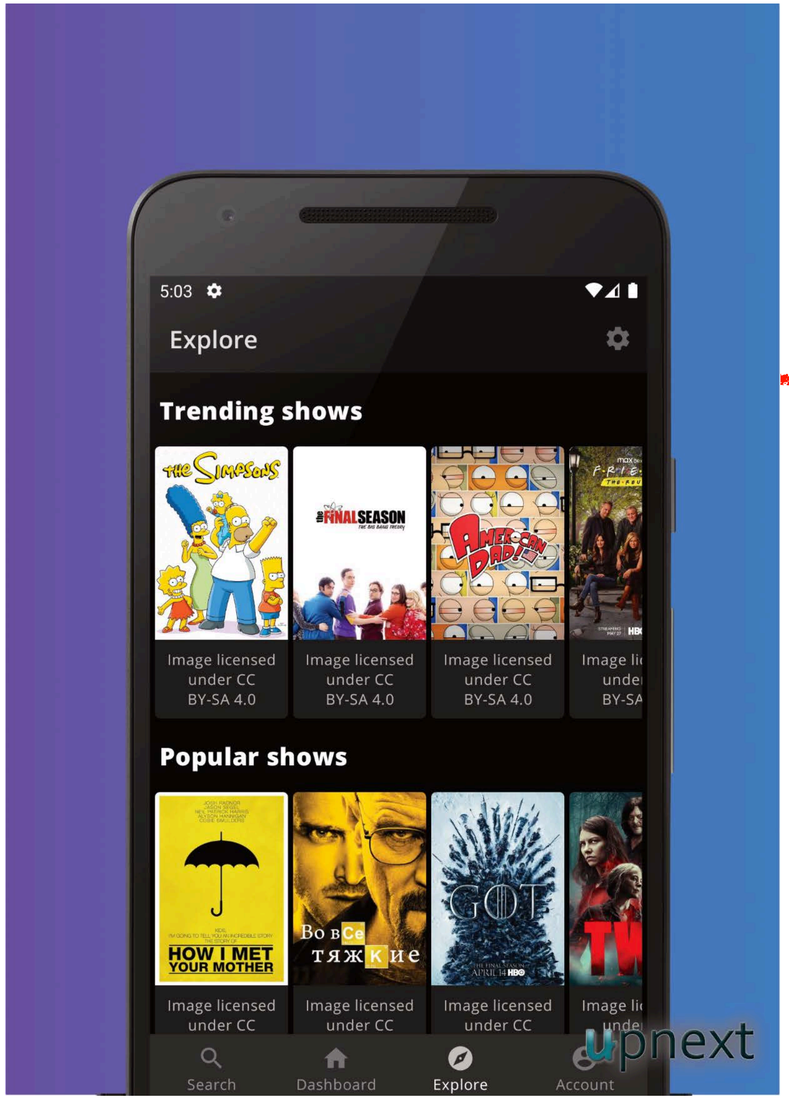


@Composable ShowsRow() & uses LazyRow()

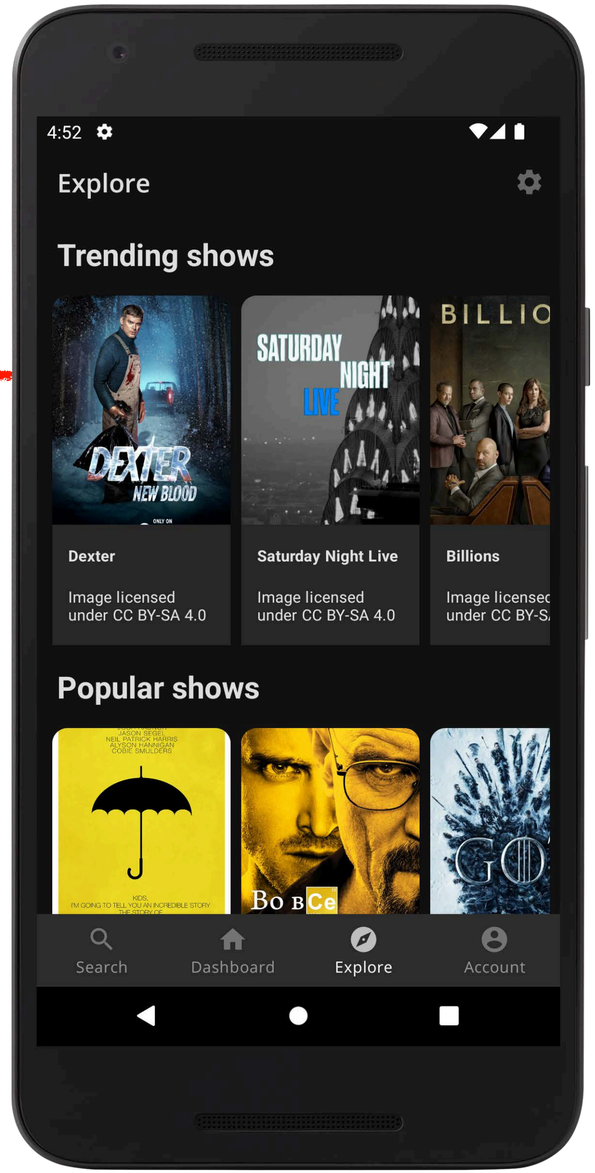
@Composable ShowsRow() & uses LazyRow()

BEFORE AND AFTER: EXPLORE SCREEN

ExploreFragment.kt



@Composable ExploreScreen



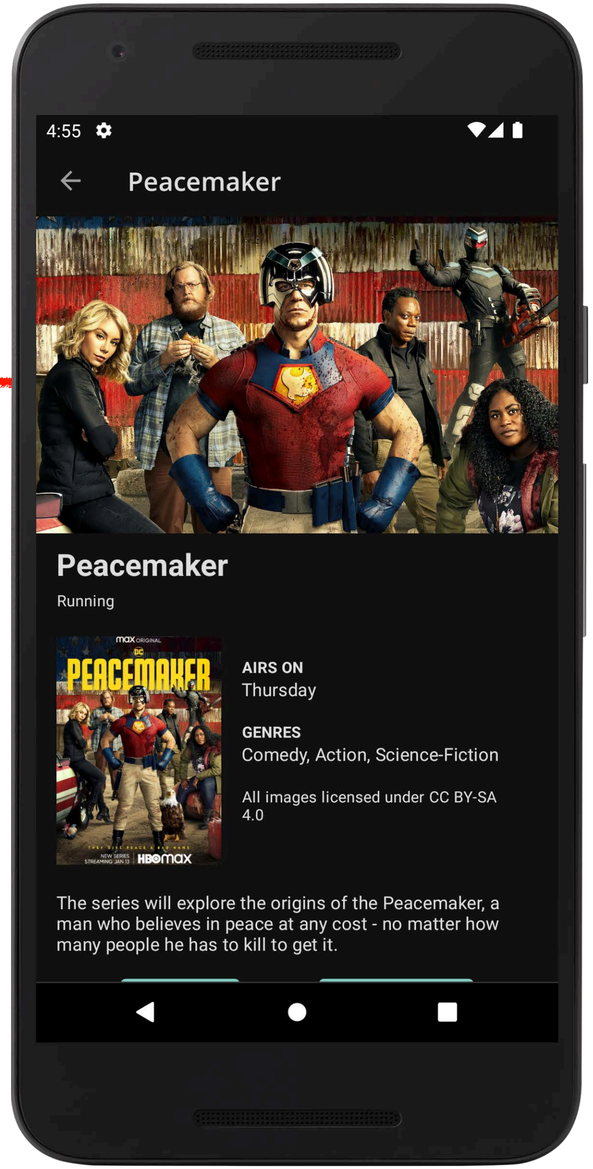
@Composable TrendingShowsRow() & uses LazyRow()

@Composable PopularShowsRow() & uses LazyRow

BEFORE AND AFTER: SHOW DETAIL SCREEN

ShowDetailFragment.kt

@Composable ShowDetailScreen



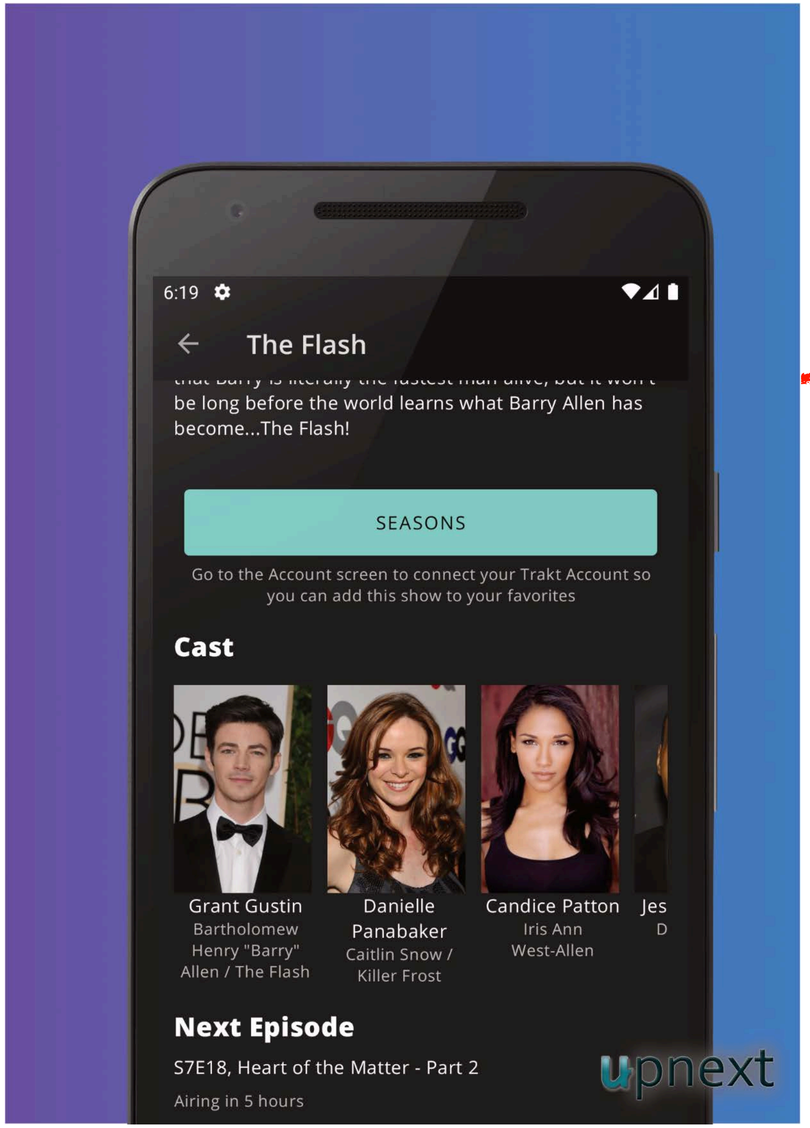
@Composable BackdropAndTitle()

@Composable PosterAndMetadata()

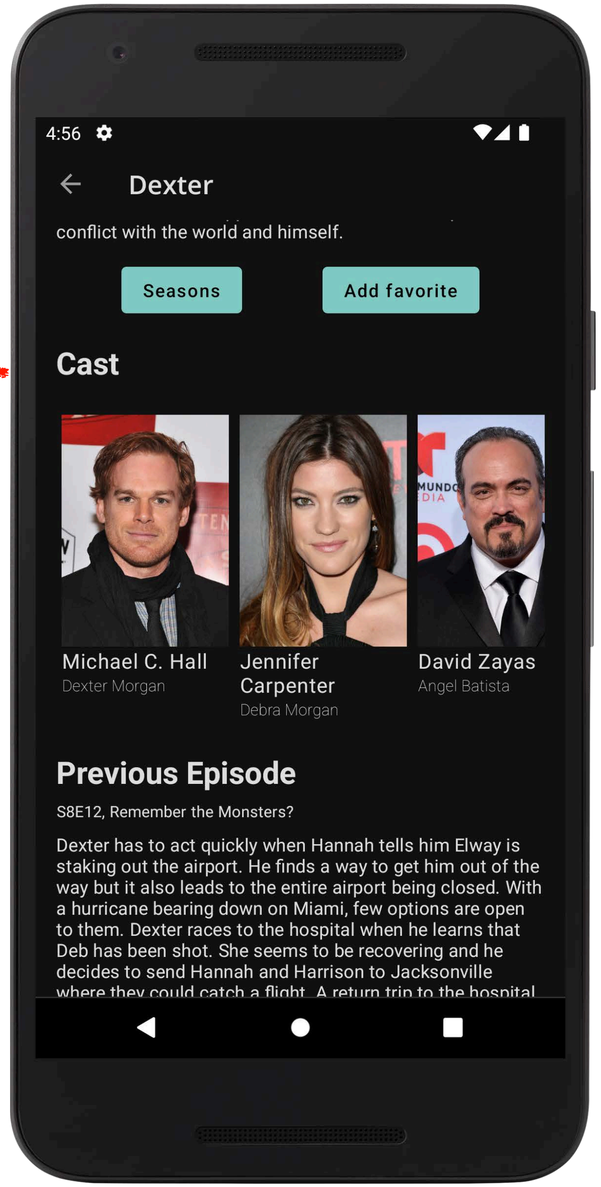
@Composable Text()

BEFORE AND AFTER: SHOW DETAIL SCREEN

ShowDetailFragment.kt



@Composable ShowDetailScreen



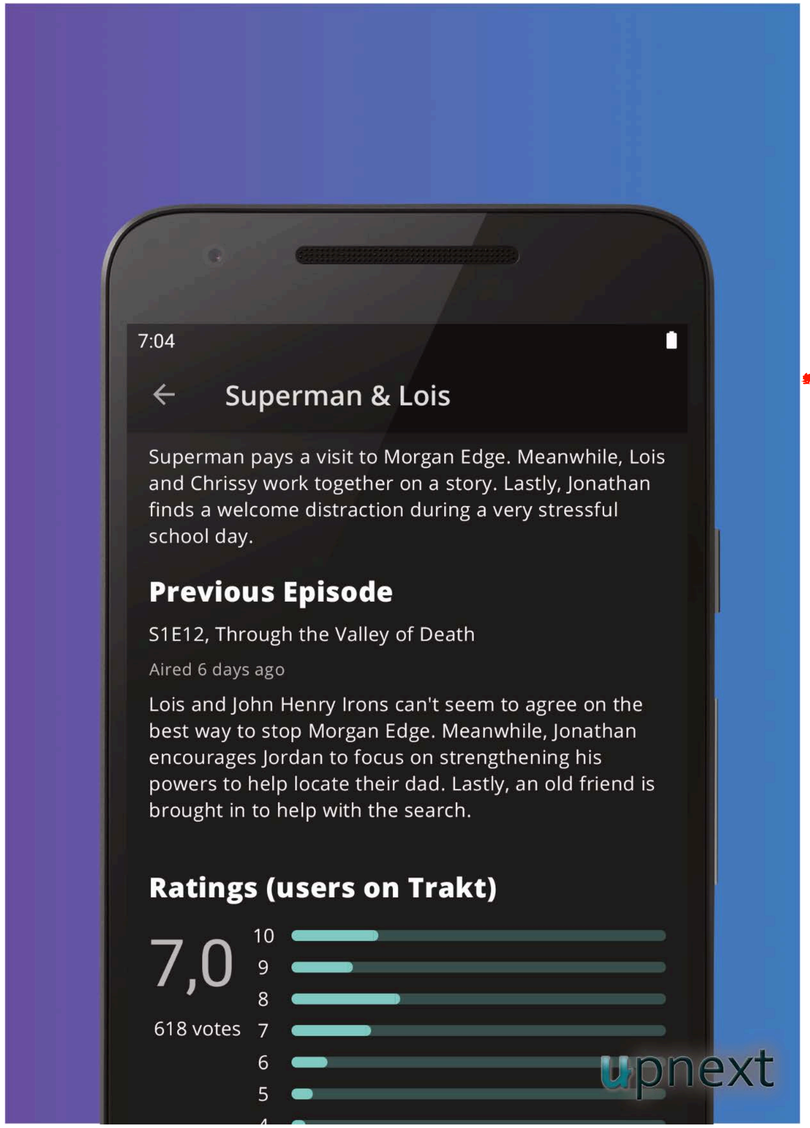
@Composable ShowDetailButtons

@Composable ShowCastList() & uses LazyRow()

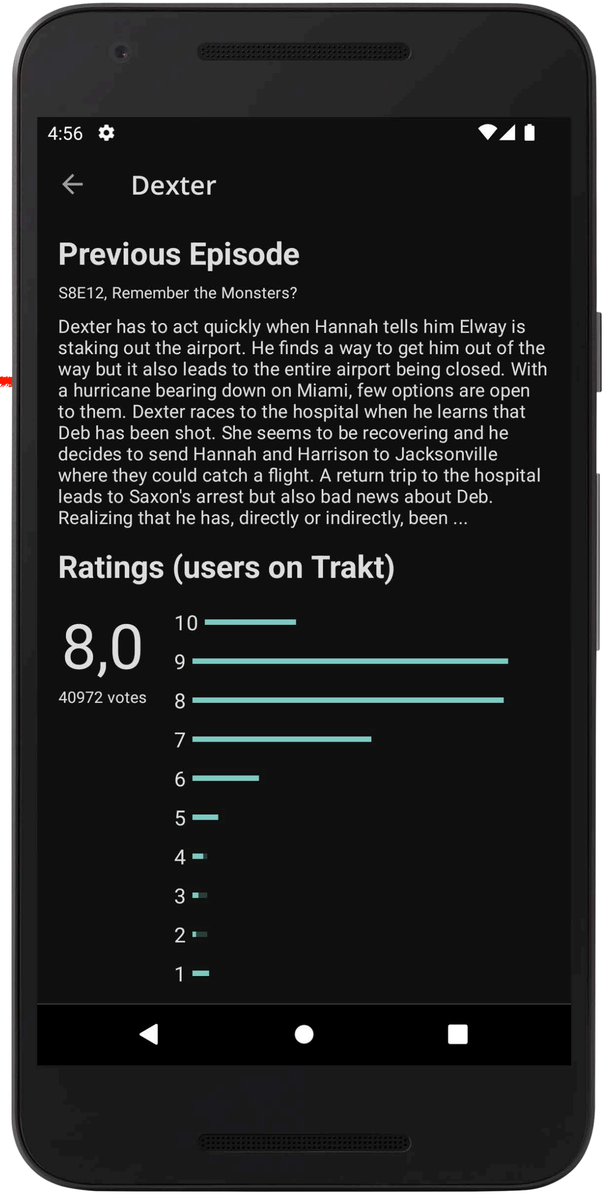
@Composable PreviousEpisode()

BEFORE AND AFTER: SHOW DETAIL SCREEN

ShowDetailFragment.kt



@Composable ShowDetailScreen

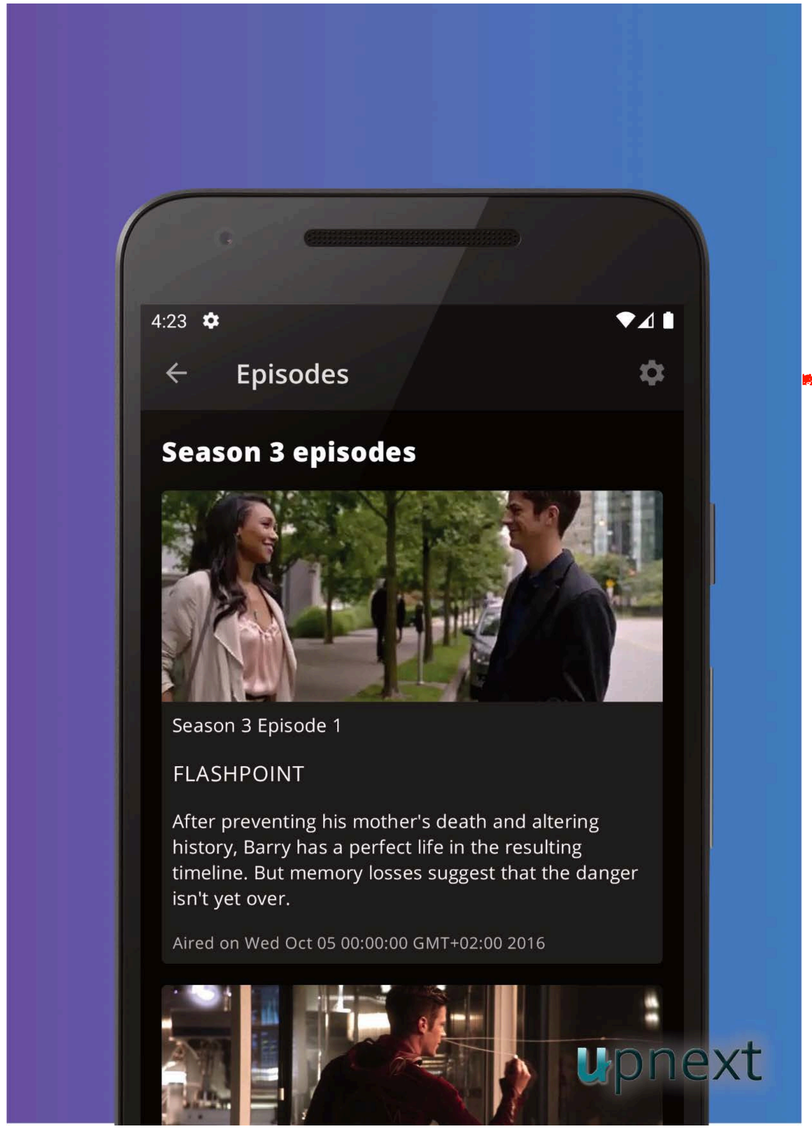


@Composable PreviousEpisode()

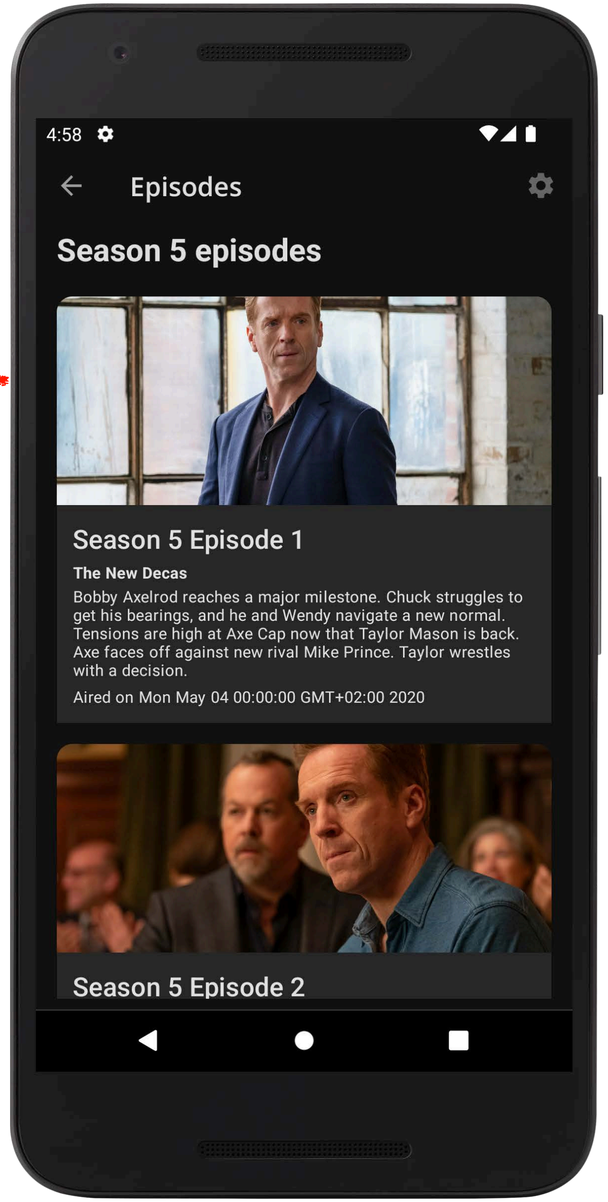
@Composable TraktRatingSummary()

BREAKING DOWN THE CHANGES: SEARCH SCREEN

ShowSeasonsEpisodesFragment.kt



@Composable ShowSeasonEpisodesScreen



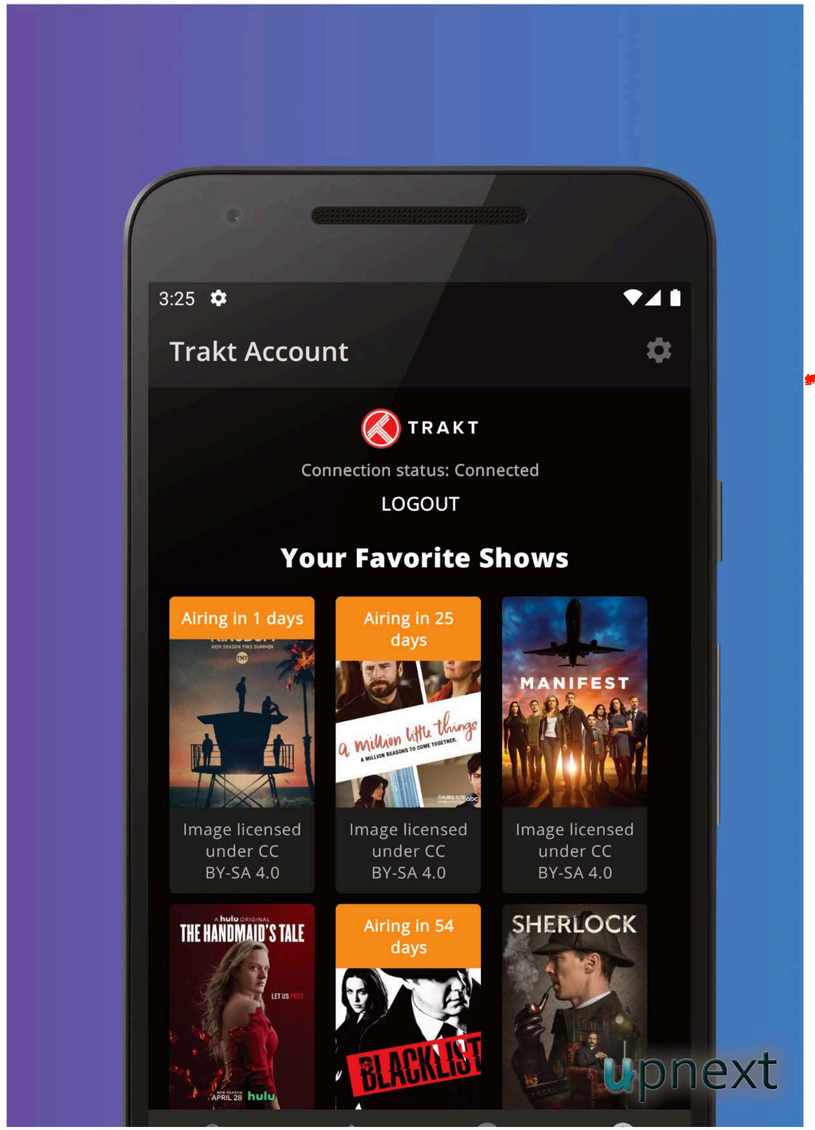
@Composable
SectionHeadingText()

@Composable
ShowSeasonEpisodes()
& uses LazyColumn()

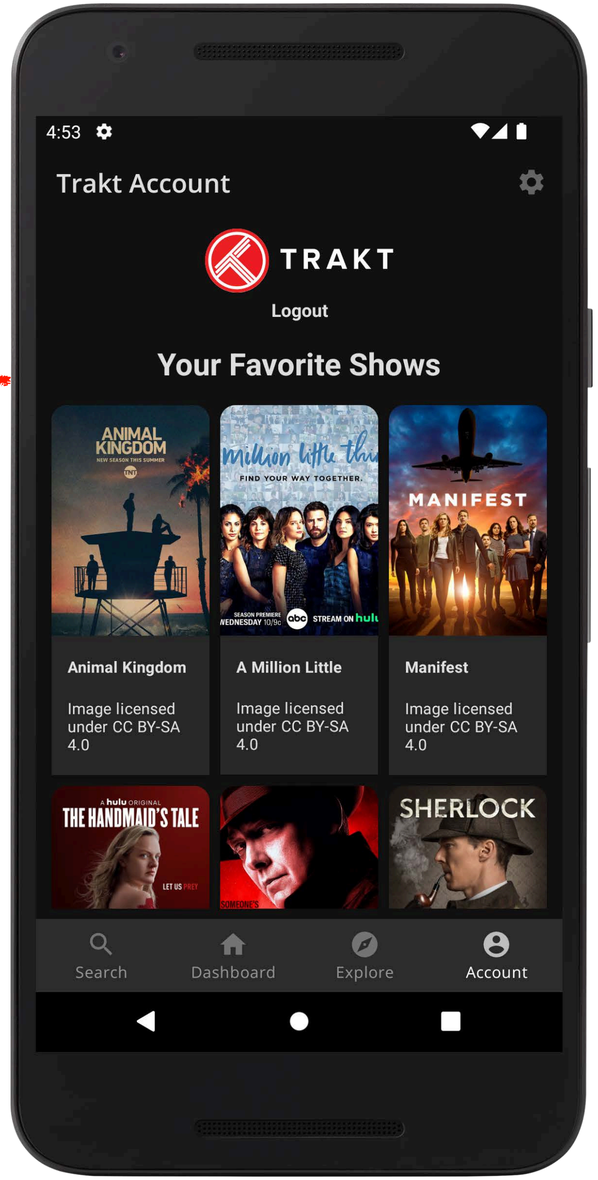
@Composable
ShowSeasonEpisodeCard()

BREAKING DOWN THE CHANGES: SEARCH SCREEN

TraktAccountFragment.kt



@Composable TraktAccountScreen()



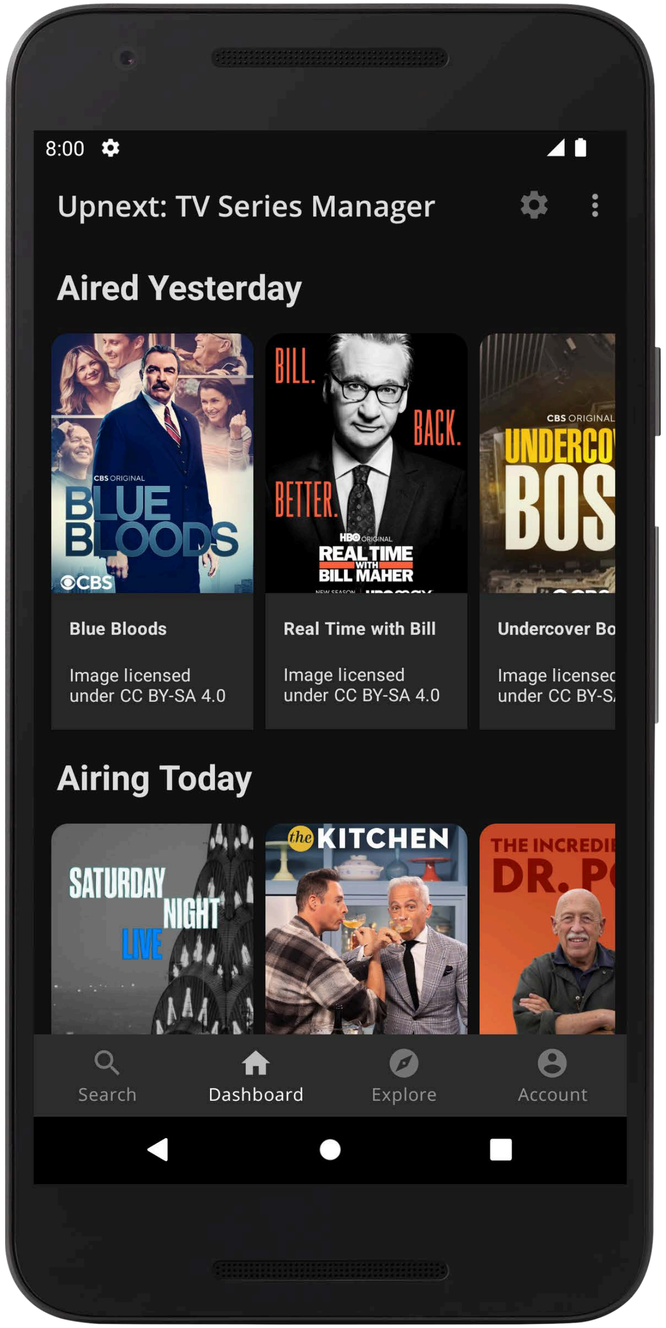
@Composable SectionHeadingText()

@Composable FavoritesList() & uses LazyVerticalGrid()

@Composable ListPosterCard()

WHAT STILL NEEDS TO BE UPDATED OR CONVERTED

- Toolbar
- Bottom navigation bar
- Migration to Compose Navigation
- Removal of all fragment files
- Replace Surface with Scaffold
- Replacement of observeAsState with MutableState observation
- Add animations
- Add tests for Composables



RESOURCES

- Official Compose Documentation
  https://developer.android.com/jetpack/compose
- Official Compose course
  https://developer.android.com/courses/pathways/compose
- Compose Layout Basics https://developer.android.com/jetpack/compose/layouts/basics
- State Hoisting https://developer.android.com/jetpack/compose/state#state-hoisting
- UpNext TV Series Manager code is available now as an open source project soon. You can view the code I mentioned and more here https://github.com/akitikkx/upnext in the branch “**feature/add-compose**”. Contributions welcome from the community! Please read my Readme and contribution Guidelines for more information

AHMED TIKIWA

SENIOR SOFTWARE ENGINEER - ANDROID @ LUNO

@ahmed_tikiwa

FROM XML TO COMPOSE, MY JOURNEY OF TRANSFORMING AN EXISTING LARGE APP TO JETPACK COMPOSE