Learning Diary

Tankopedia App
A Jetpack Compose journey

Day 1: Getting Started

I decided to start building an app called *Tankopedia* using Jetpack Compose. I've been curious about how modern Android development works, and this seemed like a fun project to dive into.

My Goals for Today:

- Set up the project with Jetpack Compose.
- Think about what information each tank should have.
- Create a main screen that lists all tanks.
- Add images and some basic info for each tank.
- Make something happen when a tank is tapped – even something simple.

What Actually Happened:

I opened Android Studio and started a new project using Jetpack Compose. It felt quite different from the XML layout style I was used to, so I was a little confused at first, but I got it working.

I created a Tank data class and decided it should include things like the tank's name, an image, a short description, its country, a flag image, and whether it's marked as a favourite.

Originally, I thought of starting with a welcome screen, but then I figured: why make users tap extra buttons? So, I made the tank list the very first thing you see when opening the app. I used a `LazyColumn` for the list. It's great for longer lists and it started off by just showing the tank's name and main image.

Adding the images took a bit more effort than I expected. I had to collect pictures for both the tanks and their country flags and place them in the `res/drawable` folder. That's when I ran into Android's picky rules about filenames, everything must be lowercase with underscores, no spaces. I messed that up a few times before it stuck.

Later, I improved the list to include the country flag alongside the tank image and name.

I also added a basic detail screen – nothing fancy, just showing the name of the tapped tank. I used a `selectedTank` variable in my `AppContent` composable to switch between the list and the detail screen.

What I Learned Today:

- Android is super strict about resource file names, the lowercase + underscore rule is burned into my brain now.
- It's worth thinking about the user experience early on. Showing the list right away felt like the right choice.
- Jetpack Compose is starting to make sense. I'm beginning to see how composable fit together and how state is managed.

Tomorrow's Plan:

- Add actual tank details to the detail screen.
- Make the favorite button functional.
- Try to improve the app's visual design – maybe a cool background?

Day 2: Making it Look Better and Adding Features

Time to make the app look more polished and add some interactive stuff.

Goals for Today:

- Make the favourite button work.
- Add a nice, blurred background.
- Fix layout issues caused by phone notches/cameras.
- Embed YouTube videos in the detail screen to show tanks in action.

Here's What Happened:

I started by updating the `Tank` data class to include an `isFavorite` property. Then I added a star icon on both the list and detail screens. Tapping it now updates the favourite status of that tank.

Getting this to work consistently was a bit tricky. I had to manage the app state carefully, especially to make sure that when a tank is favourited on the detail screen, it reflects on the list screen too. So, I kept the tank list and the logic for updating favourites inside `AppContent`, and passed functions down to the child composable as needed.

I also added a filter button that lets you view just your favourite tanks. It was really satisfying to see that work!

Blurred Background:
This part was a pain in the ass with a bigger screen. I used an `Image` composable and added a `.blur()` modifier. Then I learned about `WindowCompat.setDecorFitsSystemWindows(window, false)`, this lets the background go behind the status and nav bars. It didn't work perfectly right away, I had to add padding to the content to avoid white bars, but after a few tries, I got it working.

Handling Notches and Cameras:
On my phone, the top buttons were getting a bit too close to the selfie camera. Luckily, the padding I added for the system bars helped a lot. I also used `Modifier.offset` to fine-tune the button placement. It was a good reminder to test layouts on more modern phones.

YouTube Integration:
I wanted the app to show more than just text and images, so I decided to embed YouTube videos for each tank. I added a `youtubeURL` to the `Tank` data. Then I found a library (`androidyoutubeplayer`) that works with Compose. I created a `YoutubePlayer` composable using that library and added it to the detail screen. It wasn't too hard once I figured out how to extract the video ID from the URL.

Things I Learned Today:

- **State Management Matters:** I see now why it's important to have a single source of truth. Sharing state across composables can get messy if you're not careful.
- **Designing for Real Devices:** Not all phones are the same – notches and cameras can mess up your layout if you're not paying attention.
- **Using Libraries in Compose:** It's possible to use existing Android libraries with Compose, you just need to wrap them in your own composables.
- **Edge-to-Edge Layouts:** Making the UI use the full screen looks modern, but it takes some effort to manage insets and padding correctly.

How I'm Feeling So Far:
Honestly, I'm proud of how much progress I've made in just two days. Day 1 was all about setting the foundation, and Day 2 made the app feel much more real and fun. The blurred background and YouTube videos are my favourite features so far. I still have a lot to learn, but I'm enjoying the process!