# Software Engineering Summit

## Android Workshop


Instructor: Yashar Atajan
TA: Nick Capurso

# About Yashar

Yashar Atajan
@yaxarat

Undergrad - Virginia Commonwealth University (VCU)

Software Engineer working on Android SDKs.

# About Nick

Nick Capurso
@nickcapurso

Undergrad - Texas Christian Univ (TCU)
Grad - George Washington Univ (GWU)
I also teach Android at GWU!

Capital One Wallet, Capital One Mobile, SDKs

e i ough e intern & TDP programs :)

# Before we get started...

- We'll be using this GitHub repo as a base.
- The workshop assumes you completed the pre-reqs and have a working Android development environment.
  - Ability to run the app on the Android emulator or a physical device.
- Import the "Start" folder into Android Studio and run it on an emulator / device.

# If you need help...

Short questions:

- Use the “Raise Hand” action in Zoom. Or feel free to interrupt me!
- You can also post short questions in the Zoom chat - we’ll be monitoring it!

Debugging help:

- Post your issue in the #ses-may21help-xg Slack channel with any relevant code / screenshots.
- During short breaks in the workshop, we will use Zoom breakout rooms for any in-depth debugging with screen share.
- Slack: @yashar (Instructor), @nickcapurso (TA)

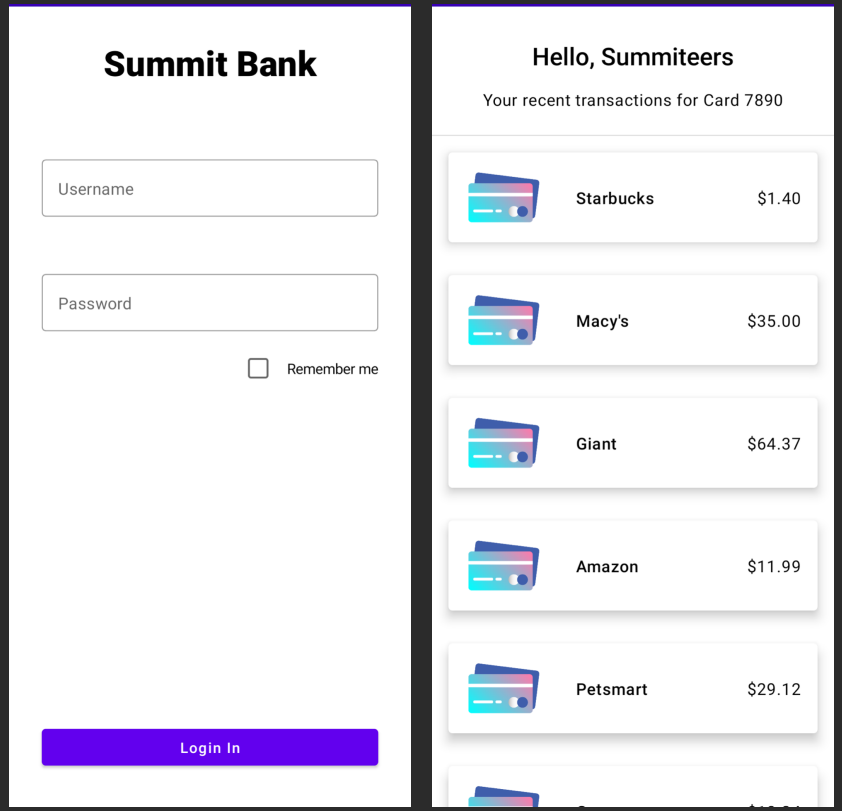
Step-by-step instructions:

- Full text instructions for this workshop are available on GitHub.

# android Workshop

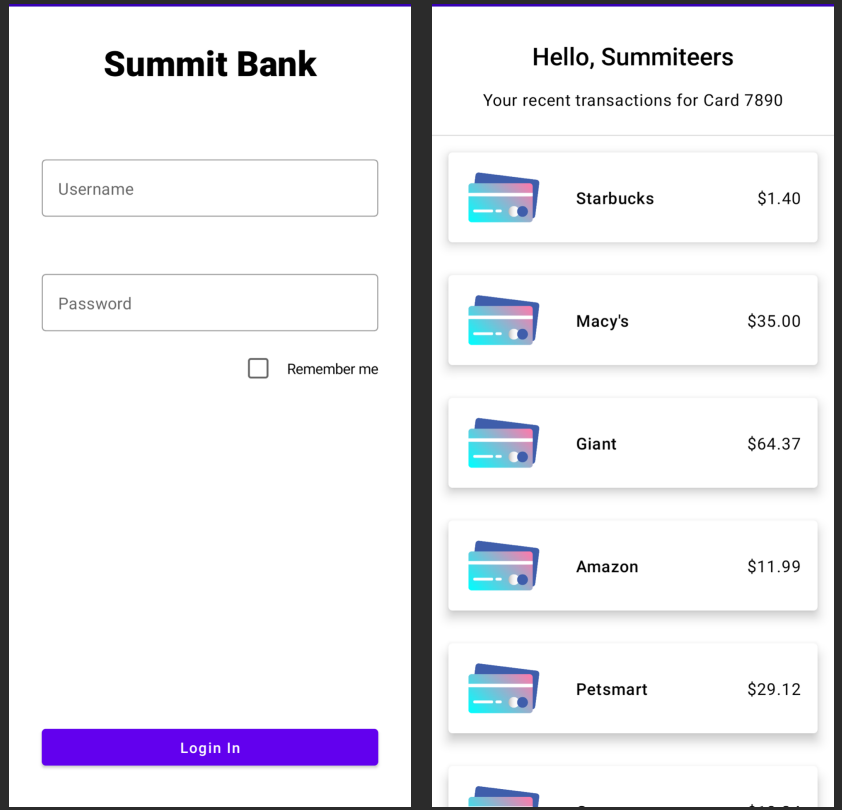
# What’re we building?

- A simple app that allows the user to login & view a list of recent transactions.



# What’re we building?

- Topics
  - UI Design
  - Launching new screens
  - Networking (mocked)
  - Rendering a list
  - Data storage



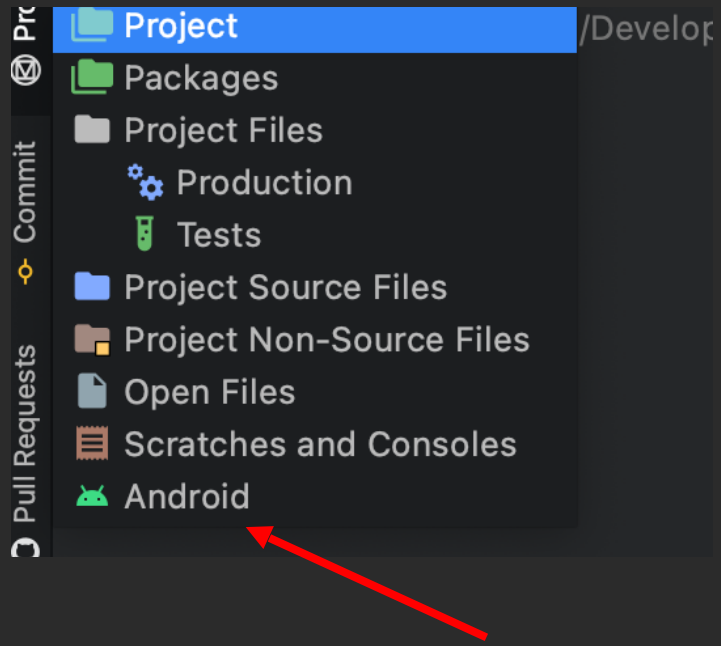
# Android Studio

Two Android Studio configurations to make the workshop go more smoothly...

# Android Studio

In the left-panel, switch to the “Android” view.

- I prefer this view of the project files as I think it is more concise and easy to navigate.




Click Android if not pre-set.

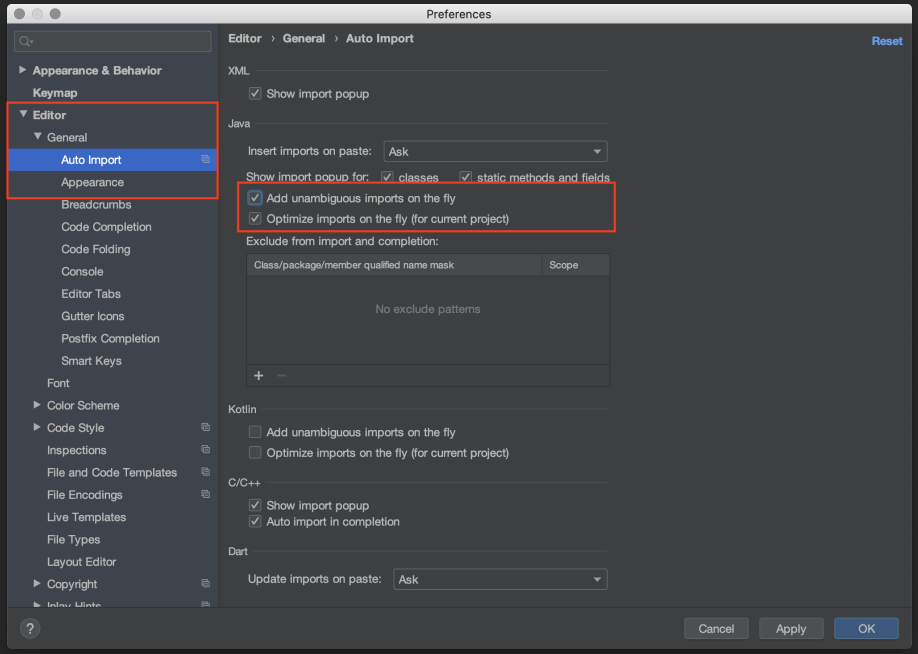
# Android Studio

Open Settings / Preferences

- (Windows) File ➜ Settings
- (OS X) Android Studio ➜ Preferences

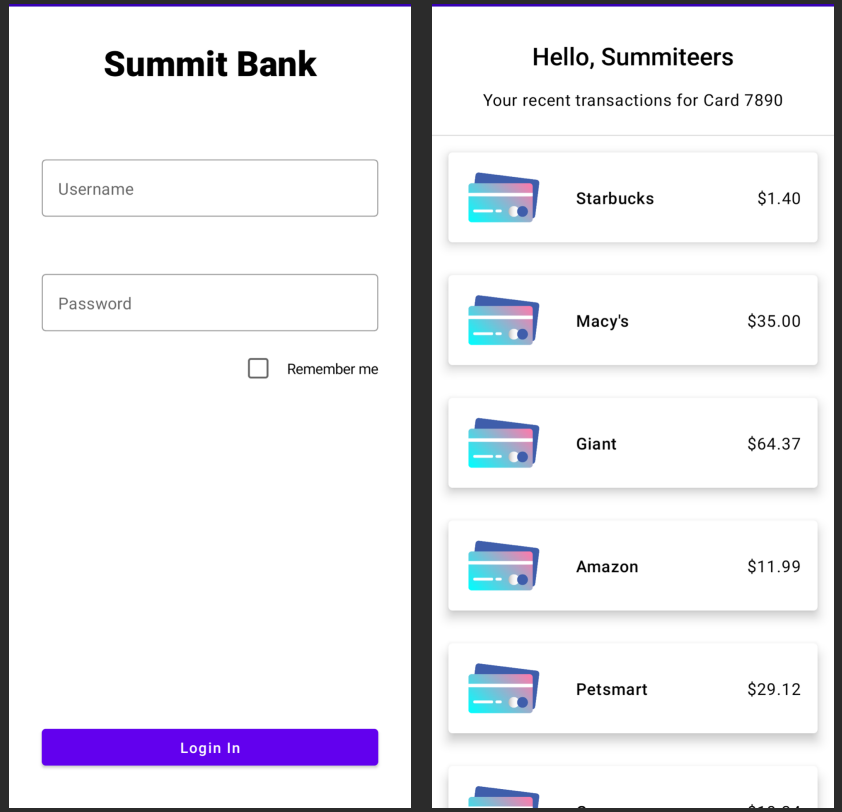
Turn on auto-imports!

- Under: Editor ➜ General ➜ Auto Import
  - Enable “Add unambiguous imports...”
  - Enable “Optimize imports...”

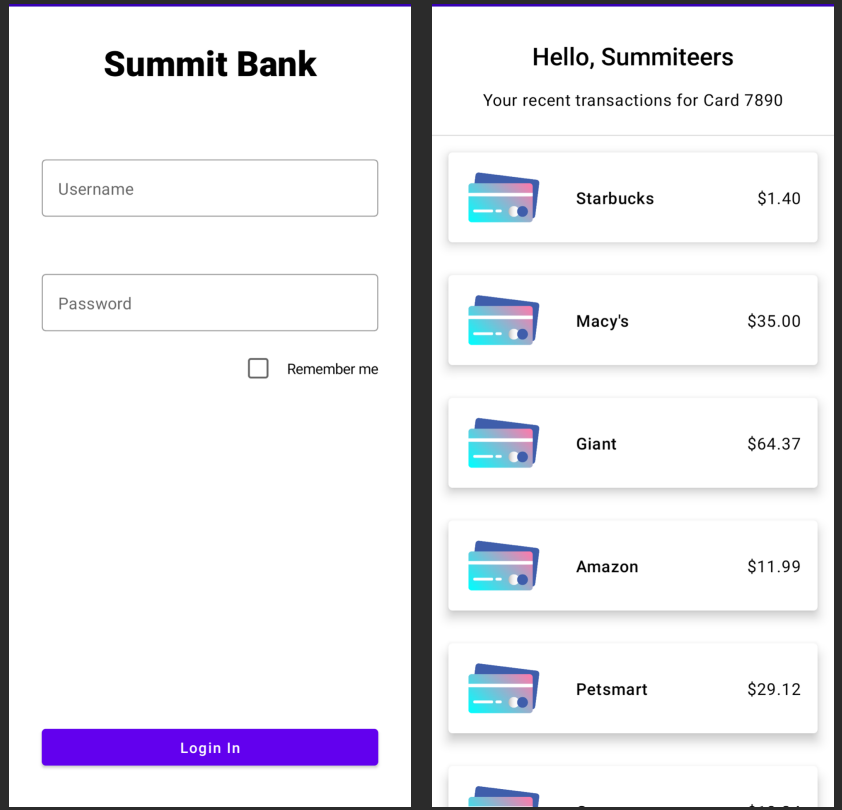


# Fragments

- Apps are comprised of “Fragments” that live inside an “Activity”.
- You can think of these as the individual screens.



- Two main components:
  - UI design
    - What does the UI look like.
  - UI behaviors
    - How does the UI react to user input.



# UI Design

# UI Design

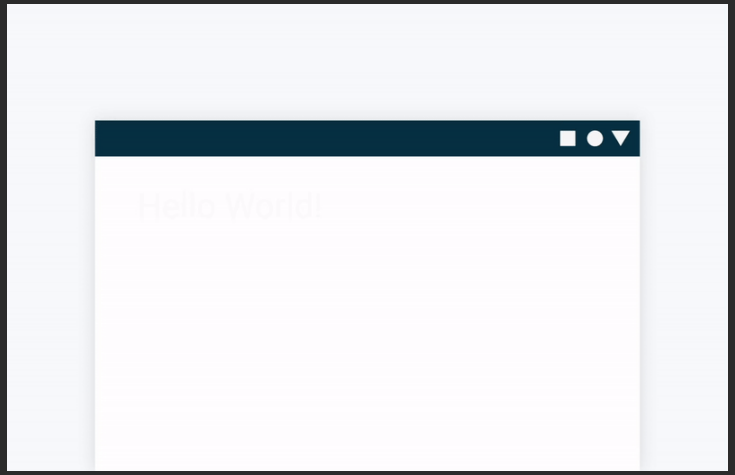
UIs are built with a Jetpack Compose framework.

```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.h2,
                )
            }
        }
    }
}
```

# UI Design

Jetpack Compose is built around composable functions.

These functions let you define your app's UI programmatically.

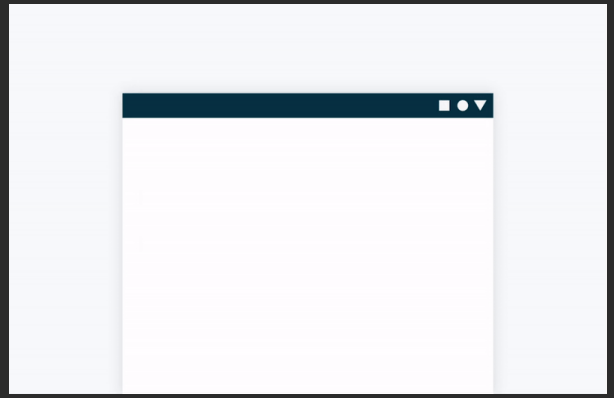



```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Text("Hello world!")
        }
    }
}
```

# UI Design

UI elements are hierarchical, with elements contained in other elements.

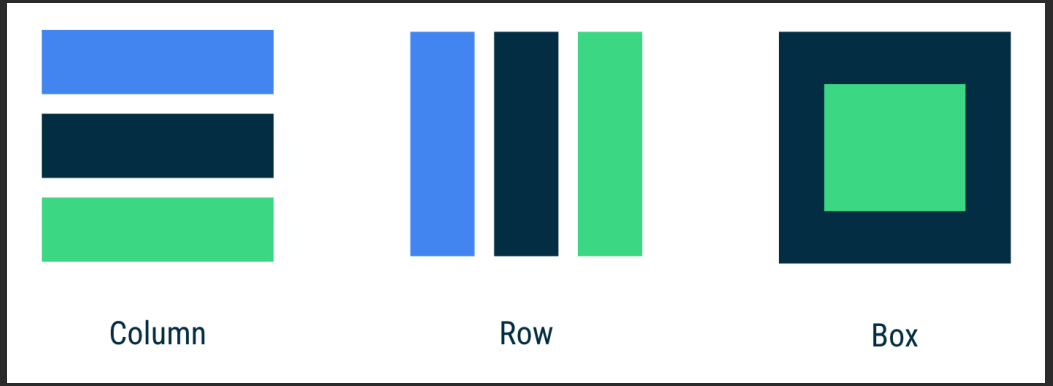
You build a UI hierarchy by calling composable functions from other composable functions.



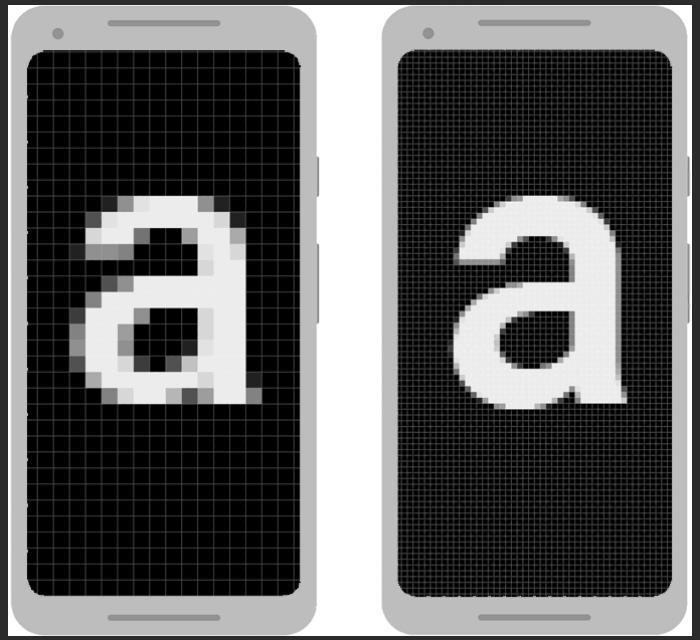
```
@Composable
fun NewsStory() {
    Column(
        modifier = Modifier.padding(16.dp)
    ) {
        Image(
            painter = painterResource(R.drawable.header),
            contentDescription = null
        )

        Text("A day in Shark Fin Cove")
        Text("Davenport, California")
        Text("December 2018")
    }
}
```

# UI Layouts

- Column: to place items vertically on the screen.
- Row: to place items horizontally on the screen.
- Box: to put one element on top of another.

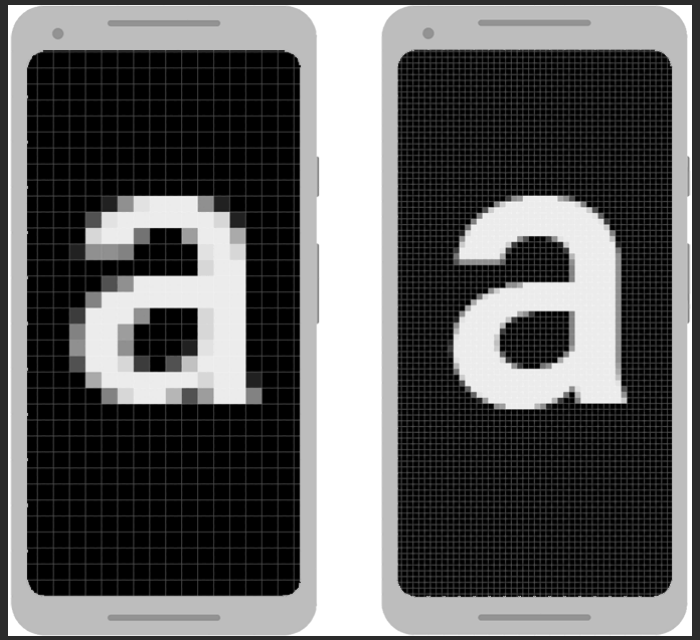


# UI Layouts



- On Android, we don’t measure in pixels.
  - Two screens of the *same size* can have *different densities*.
- Android provides “dp” or *density-independent pixels* as a measure.

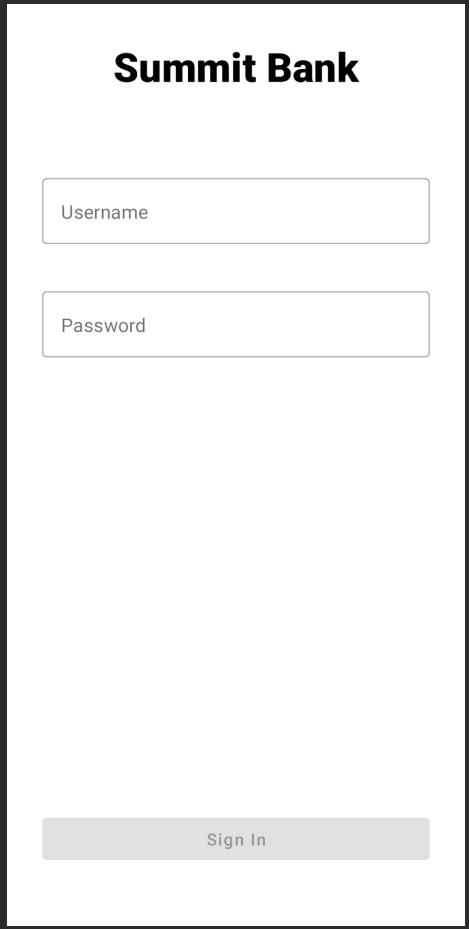
# UI Layouts



- There’s also “sp” or *scalable pixels*, which are used with text.

# Login Screen UI

Let’s build a simple login form, without a “Remember Me” checkbox.

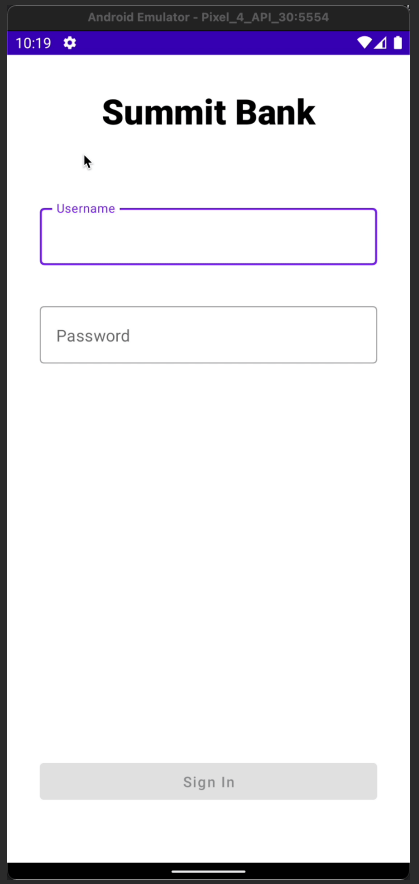


# UI Behaviors

# Login Screen Behaviors

Now we can actually write some code to add behaviors to the UI.

- Disable the “Sign In” button unless text was inputted.
- Show loading indicator while login



# Login Screen Behaviors

Behaviors and logic for our UI are written Kotlin in an ViewModel class.

# Login Screen Behaviors

Android apps use an “event-driven” programming style.

Rather than having a main() function and executing sequentially, we write code that responds to events.

- e.g.  when the screen loads, when the user performs some input, etc.

# Login Screen Behaviors

As an Android developer, you may want to react when:

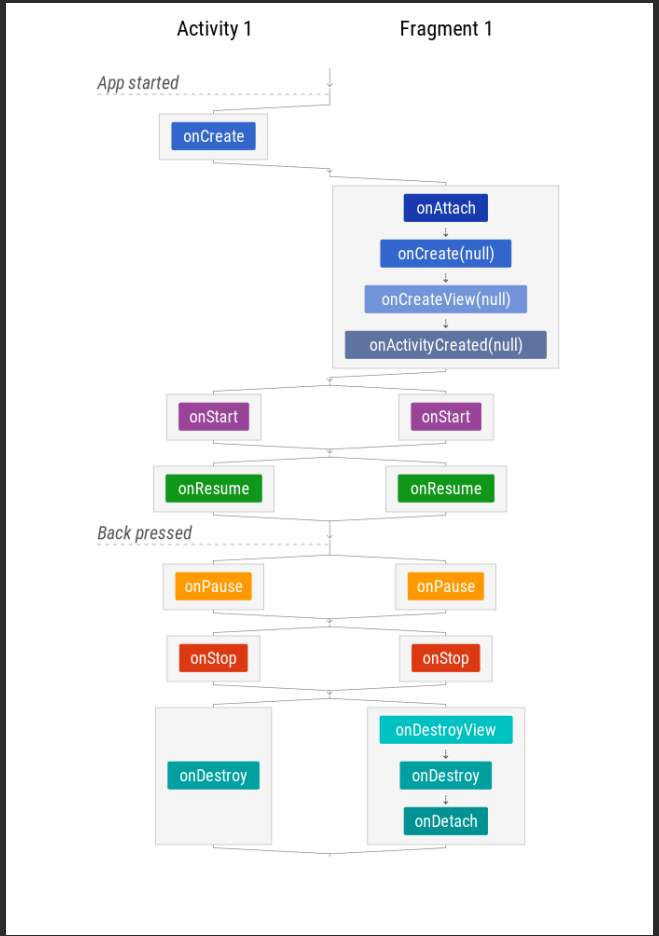
- When the screen is created
- When the user puts the app in the background
- When a popup is obscuring your screen
- etc.

# Login Screen Behaviors

Android notifies your Activity and Fragment of these events via callbacks:

- onCreate
- onStart
- onStop
- etc.

This is called the “lifecycle”!



# Networking

# Networking

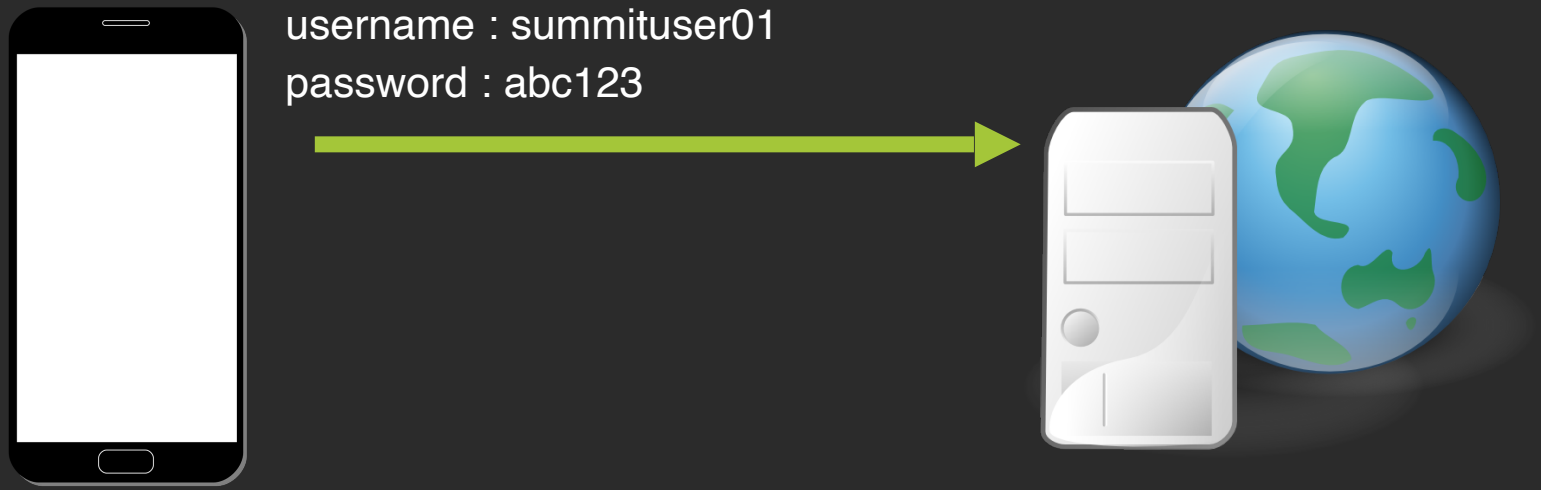
Usually, signing in involves making a network call.

- Server will validate credentials and return user data.

# Networking

Usually, signing in involves making a network call.

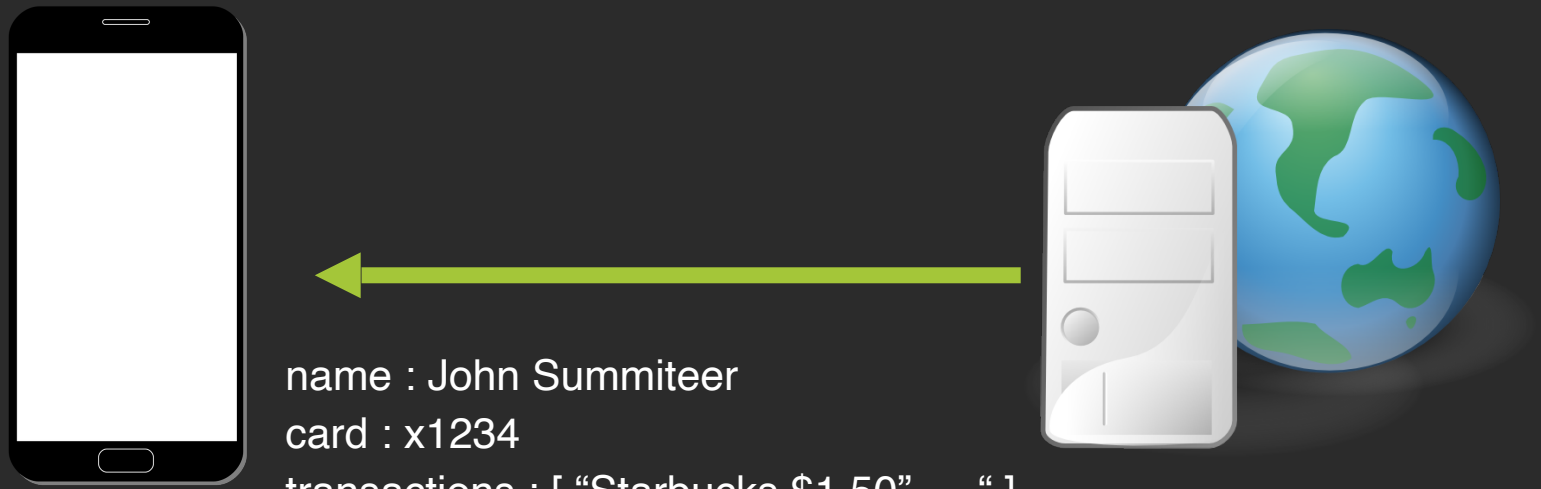
- Server will validate credentials and return user data.



# Networking

Usually, signing in involves making a network call.

- Server will validate credentials and return user data.



# Networking

Networking can be a complex subject in Android.

- Need to be on a background thread.
- Handling connection errors.
- Creating a network request.
- Parsing a network response.

# Networking

The server sends back data using some standardized format.

JavaScript Object Notation (JSON) is fairly common.

- You'd also need to code to parse the data you need from the server response.

```
{
  "name":"Summiteers",
  "cardLastFour":"7890",
  "transactions":[
    {
      "merchant":"Starbucks",
      "amount":"$1.40"
    },
    {
      "merchant":"Macy's",
      "amount":"$35.00"
    },
    {
      "merchant":"Giant",
      "amount":"$64.37"
    },
    {
```

# Networking

I’d recommend using a library here:

- OkHttp
- The “asynchronous” example here is good too.

# Networking

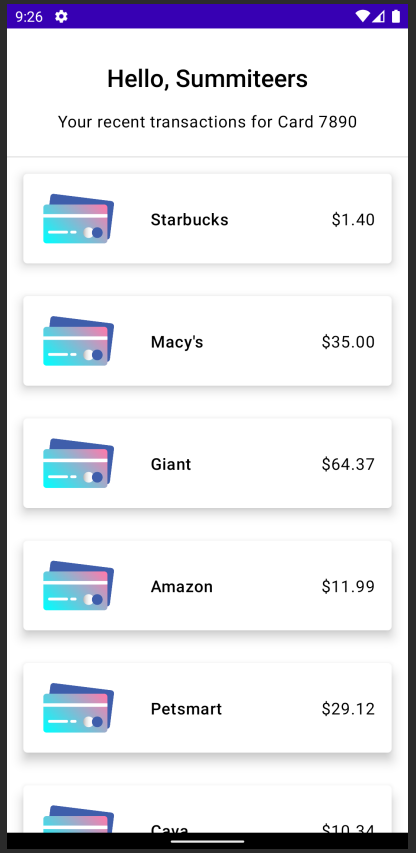
For the purpose of the workshop let’s use a prebuilt “LoginService” class to retrieve user details after they click the “Sign In” button...

- User’s name (“Summiteers”)
- User’s card last 4 (“7890”)
- List of transactions (“Starbucks $1.50, Amazon $32.10, ...)

# Second Fragment

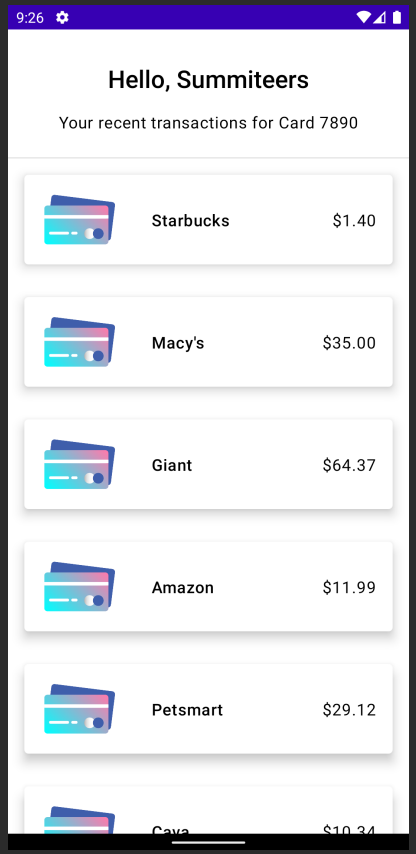
We'll display the user's data on a separate Fragment called the "SummaryFragment".

We need to create this new Fragment and launch it from our LoginFragment (passing it the data returned by the server).



On login success, we want to start the “SummaryFragment”

We also want to pass the user’s details to that fragment to display.



# Fragment Transaction

New fragments are launched using “FragmentManager”.

With fragmentManager, you:

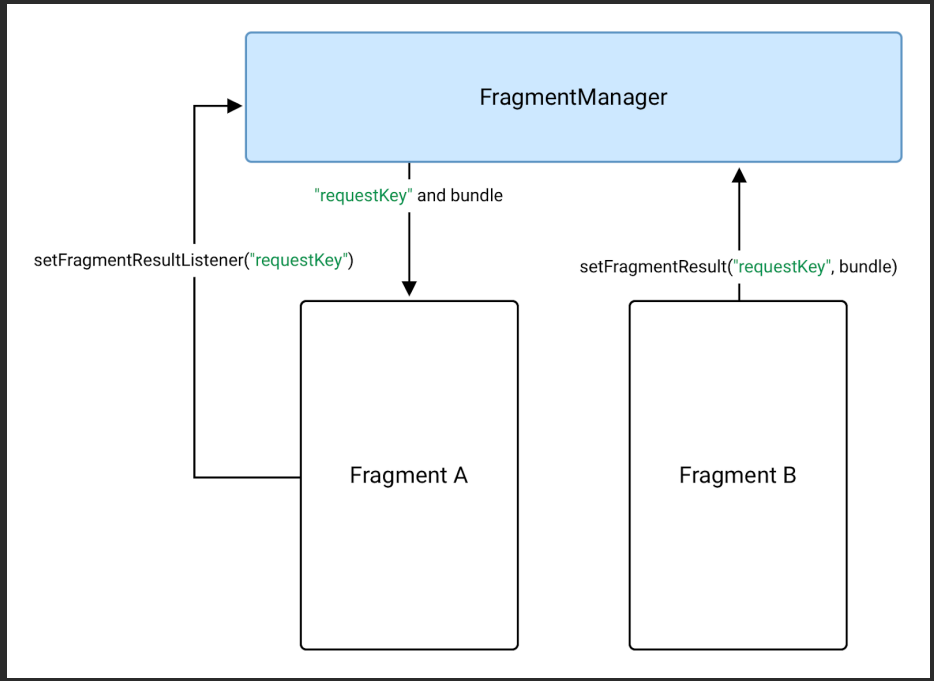
- Specify which fragment should be launched
- How the fragment should be launched
- Where the fragment should be launched

```
parentFragmentManager
    .beginTransaction()
    .replace(R.id.main_container, SummaryFragment())
    .commit()
```

# Fragment Result

You can pass information between fragments using FragmentResult.

It’s a storage for fragments where multiple fragments can use a key to store specific datas.

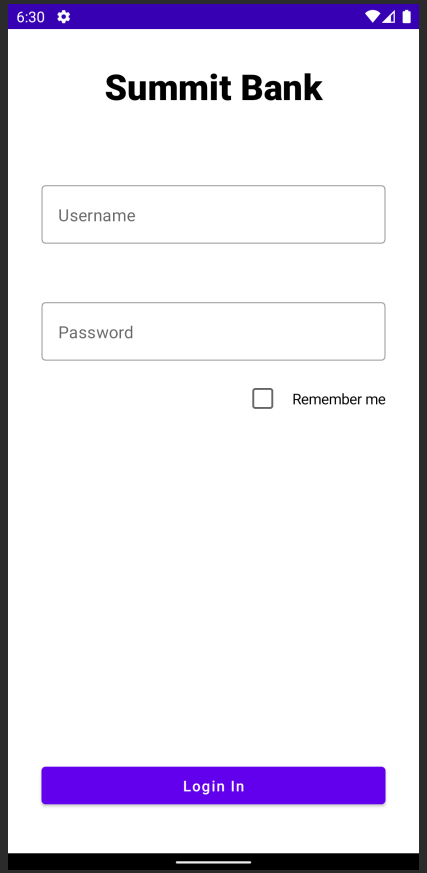


# Extras

# Extra features!

Let’s add on a “Remember Me” feature to remember user credentials.

And also an ability to select each individual transactions.



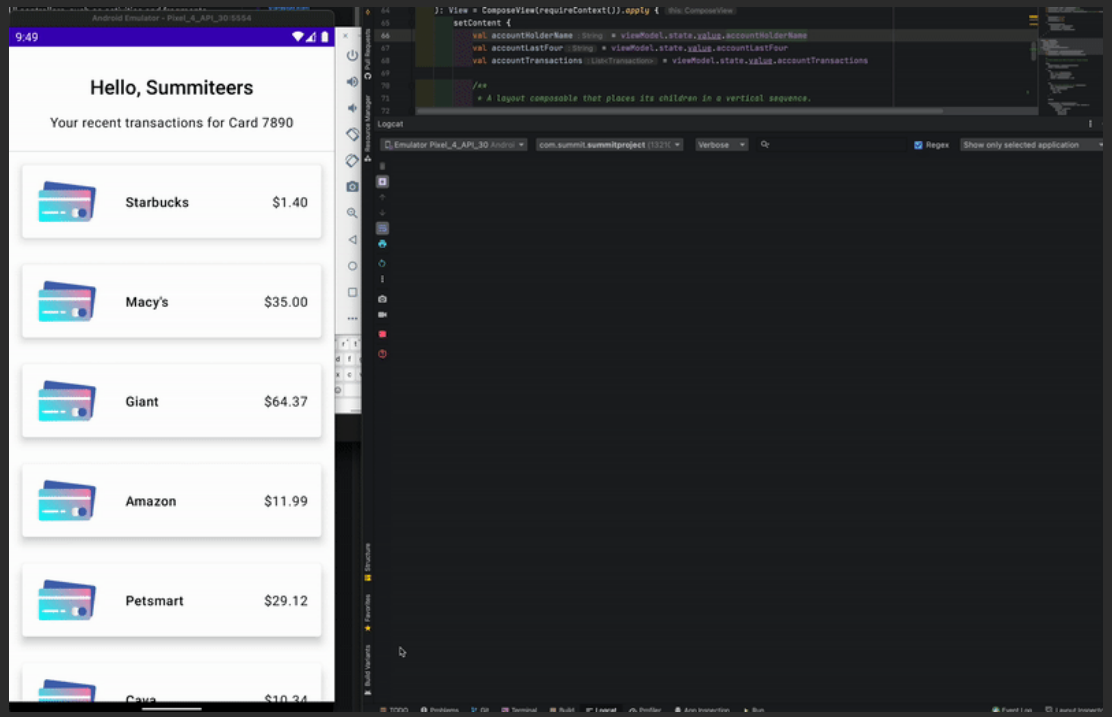
# Simple data storage

You can use “SharedPreferences” to store simple key-value pairs to the app’s private storage.

```
/**
 * The key under which the **username** is stored in sharedPreference.
 */
const val PREF_USERNAME = "username"

/**
 * The key under which the **remember me** is stored in sharedPreference.
 */
const val PREF_REMEMBER_ME = "remember_me"
```

# Reacting to list interactions

Let’s also add a simple behavior when the user taps on a transaction.

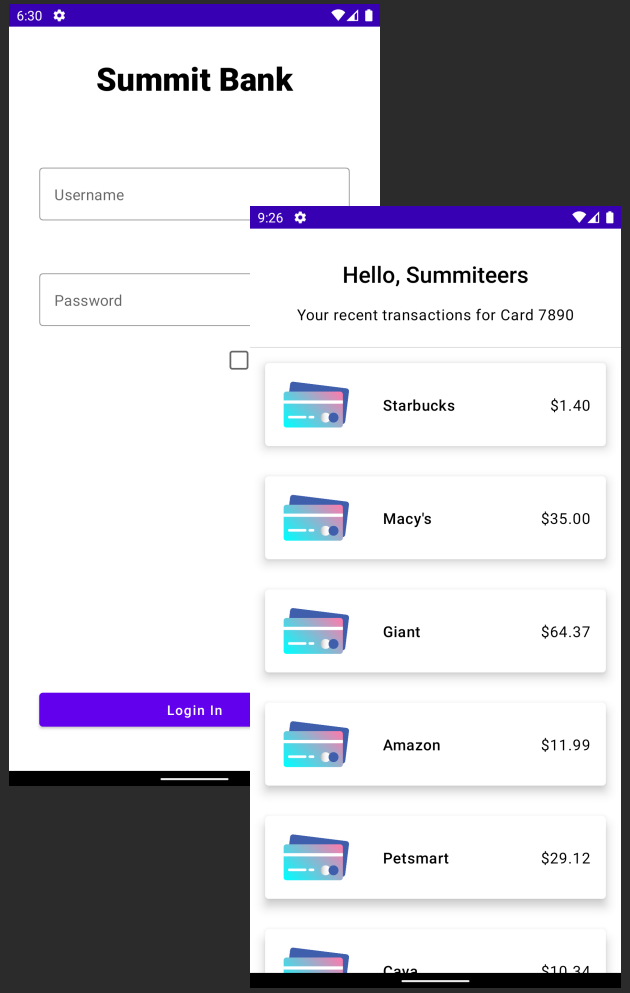


# Wrapping Up

# What did we build?

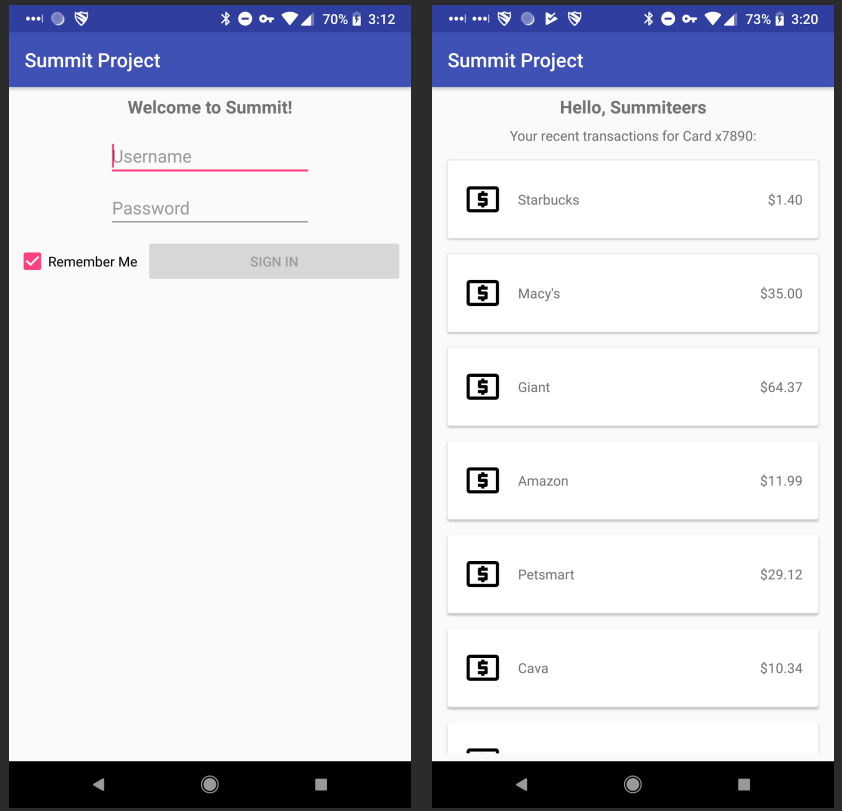
A simple app that allows the user to login & view a list of recent transactions:

- Two screens
- UI with Jetpack Compose
- Basic networking and data storage



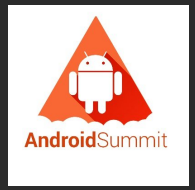
# What did we build?

If you want the completed project with comments, checkout the “Completed” or “Extra” folder.



# Android at Capital One

- Over 200 Android (and iOS) devs work on the main Capital One Mobile app alone!
  - There are other apps built in-house too - like our CreditWise app.
  - A few devs have Flutter side-projects. Our conference app and website for the Android Summit conference is built in Flutter.

# Want to continue learning Android?

- Start learning modern Android using Google Codelabs
  - Android Kotlin Fundamentals

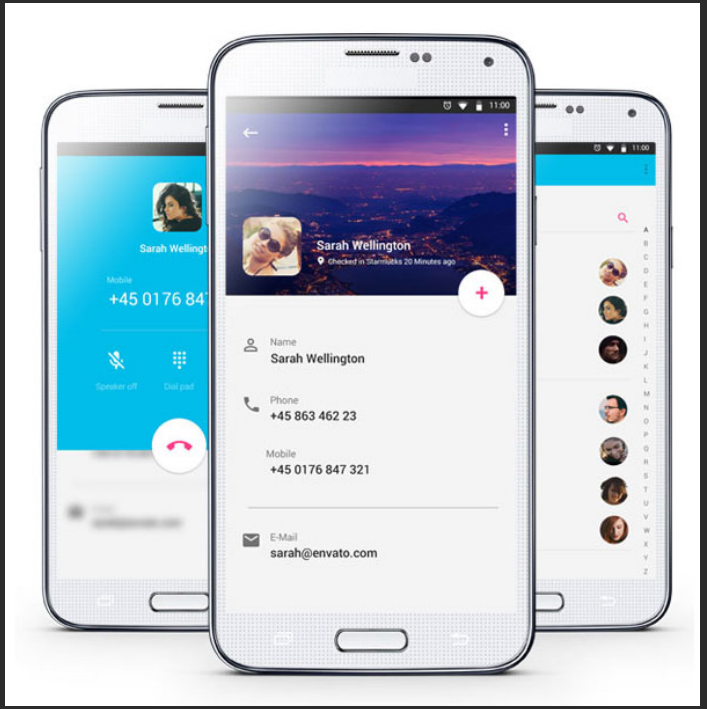
- Check out this article in the C1 Tech Blog about helpful resources for learning Android
  - The Most Helpful Resources From My First Year in Android

# Other Resources

“I want a nicer UI”

- https://developer.android.com/guide/topics/ui/look-and-feel

- https://material.io/develop/android



Other Resources

# “I want to learn more about networking”

- Basics:
  https://developer.android.com/training/basics/network-ops/connecting
- Popular libraries for making network calls:
  http://square.github.io/okhttp/
  https://square.github.io/retrofit/
- Threading:
  https://developer.android.com/guide/components/processes-and-threads
- JSON parsing:
  https://www.tutorialspoint.com/android/android_json_parser.htm

Other Resources

# “I want to interface with an API”

- Capital One’s Hackathon API (Nessie)
  http://api.reimaginebanking.com/

- “Host Your Own” Fake API:
  https://www.mocky.io/

# “I want to use some hardware features”

- Camera:
  https://developer.android.com/guide/topics/media/camera
- Sensors:
  https://developer.android.com/guide/topics/sensors/sensors_overview
- Bluetooth:
  https://developer.android.com/guide/topics/connectivity/bluetooth
- NFC:
  https://developer.android.com/guide/topics/connectivity/nfc/

# “I want a backend for user credentials and data”

- Firebase Authentication (free): https://firebase.google.com/docs/auth/
- Firebase Realtime DB (also free): https://firebase.google.com/docs/database/

Thanks for attending!

Find us on Slack and LinkedIn:

- Slack: @yashar
  - https://www.linkedin.com/in/yaxarat/
- Slack: @nickcapurso
  - https://www.linkedin.com/in/nickcapurso