Kotlin Flows, StateFlow, and SharedFlow: When to Use What in Android?

Hardik Trivedi

@MrHardikTrivedi

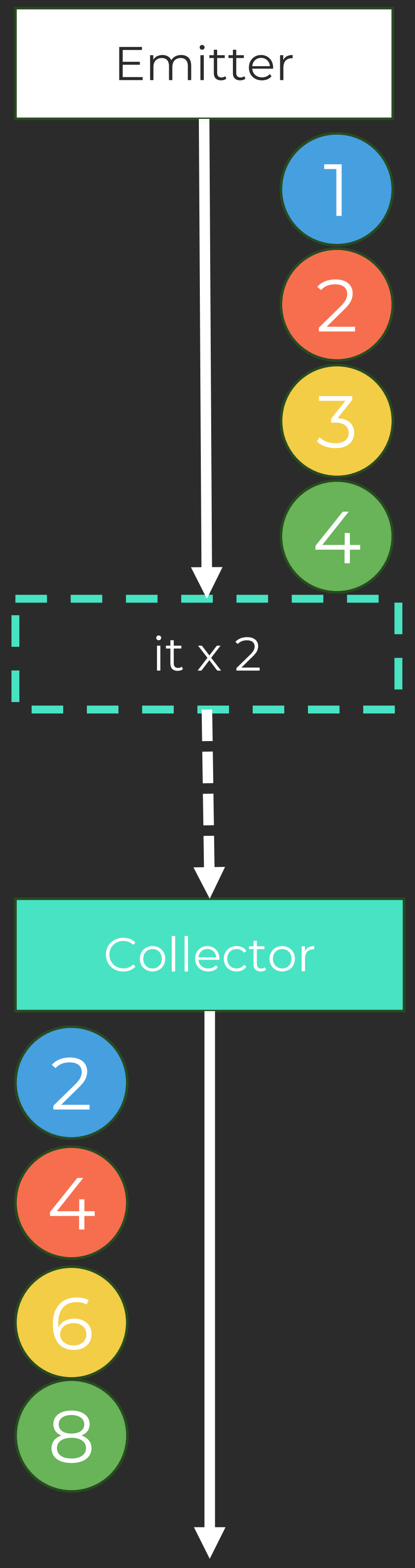
in/mrhardiktrivedi

Course rundown

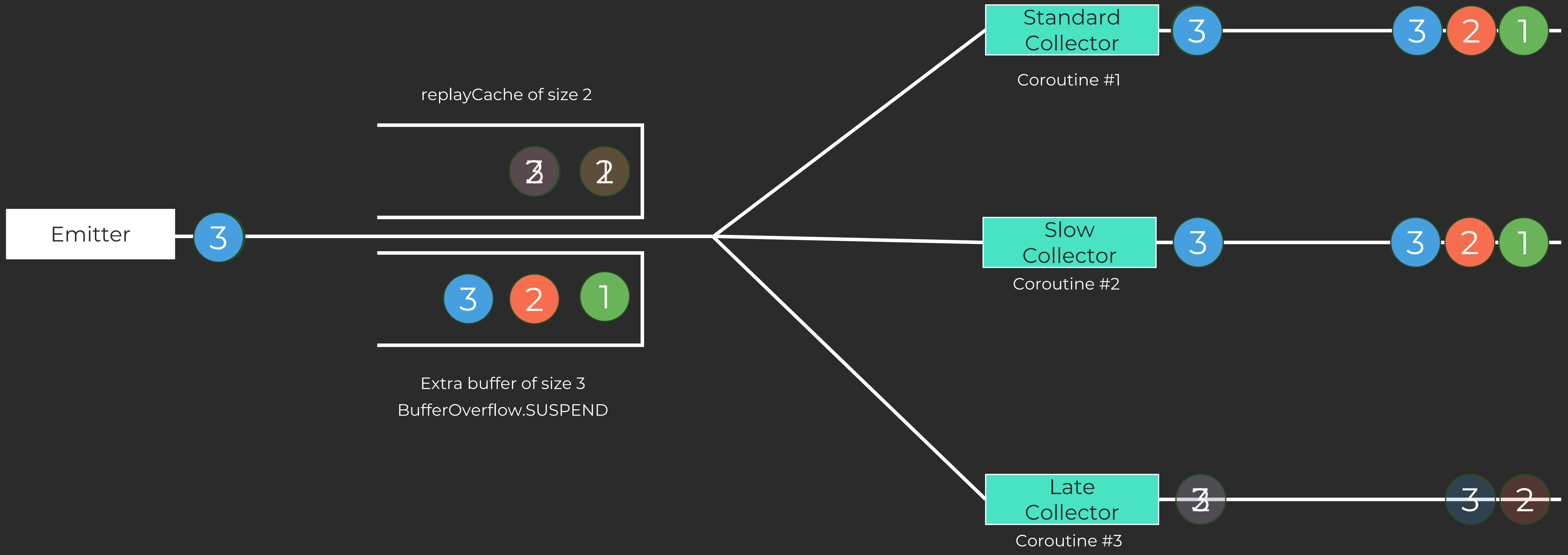
- What are the different types of Flow in Kotlin?
- Understanding a demo app which uses different Flows and Jetpack compose.
- Starter project – The Event App
- Keeping track of attendees using Flow
- Emitting events using SharedFlow – replay cache and extra buffer
- Update UI using a StateFlow
- Coding challenge and solution

What is Flow?

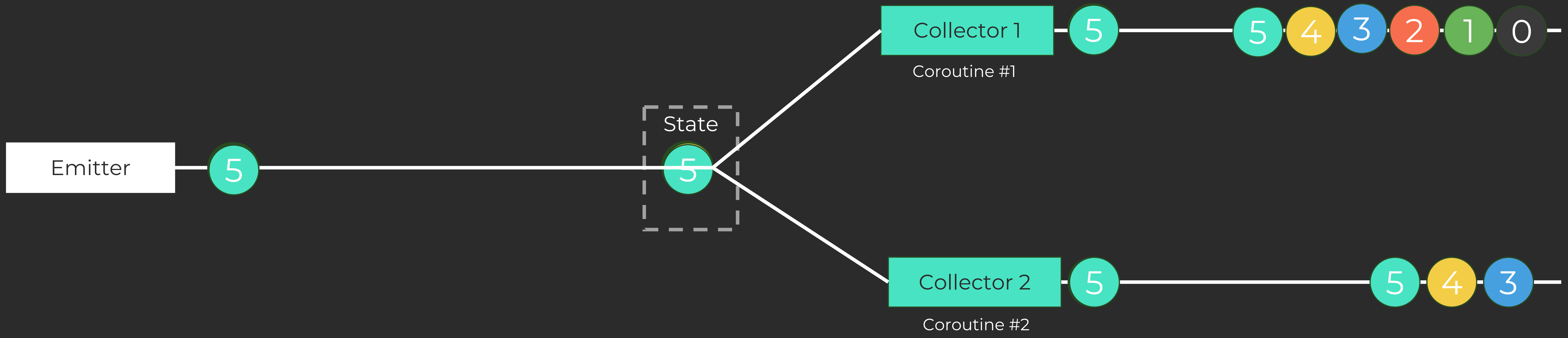
- An asynchronous data stream that sequentially emits values and completes normally or with an exception.
- Flow offers operators like map, filter, zip, combine, take etc to either transform or process the upstream and return a downstream flow which can be collected.
- To collect a flow, you require terminal operators like collect, reduce, toList or launchIn, that start collection of the flow in the given scope.



How SharedFlow works?

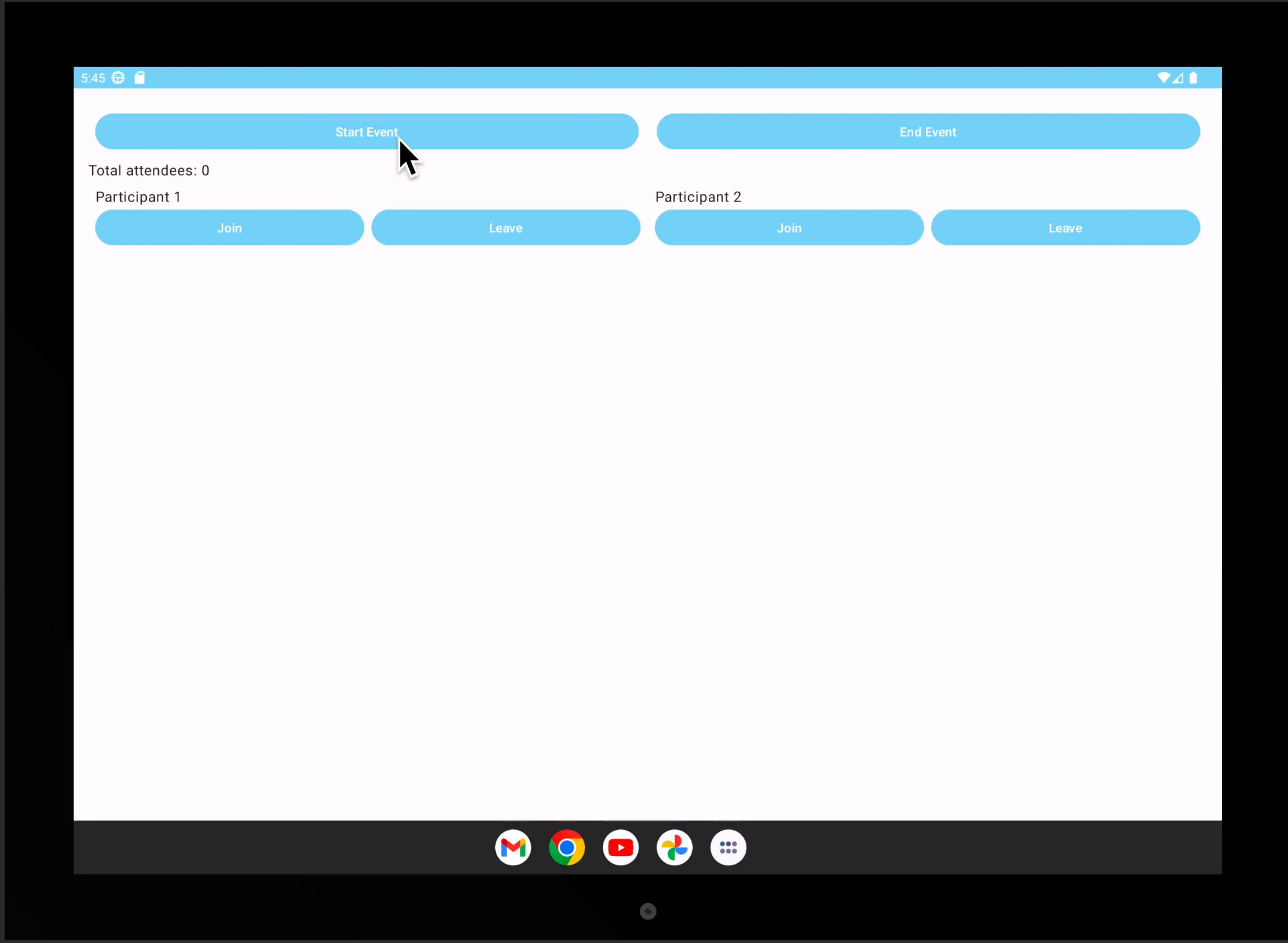


How StateFlow works?



Course Demo App

- Build an app which shows various events.
- There will be two buttons which respectively starts and ends the event
- Participants can join and leave an event at any time
- Information about the event will be published as and when they start
- For the demo purpose an app is demonstrated on a tablet, so that all content is visible straight away.

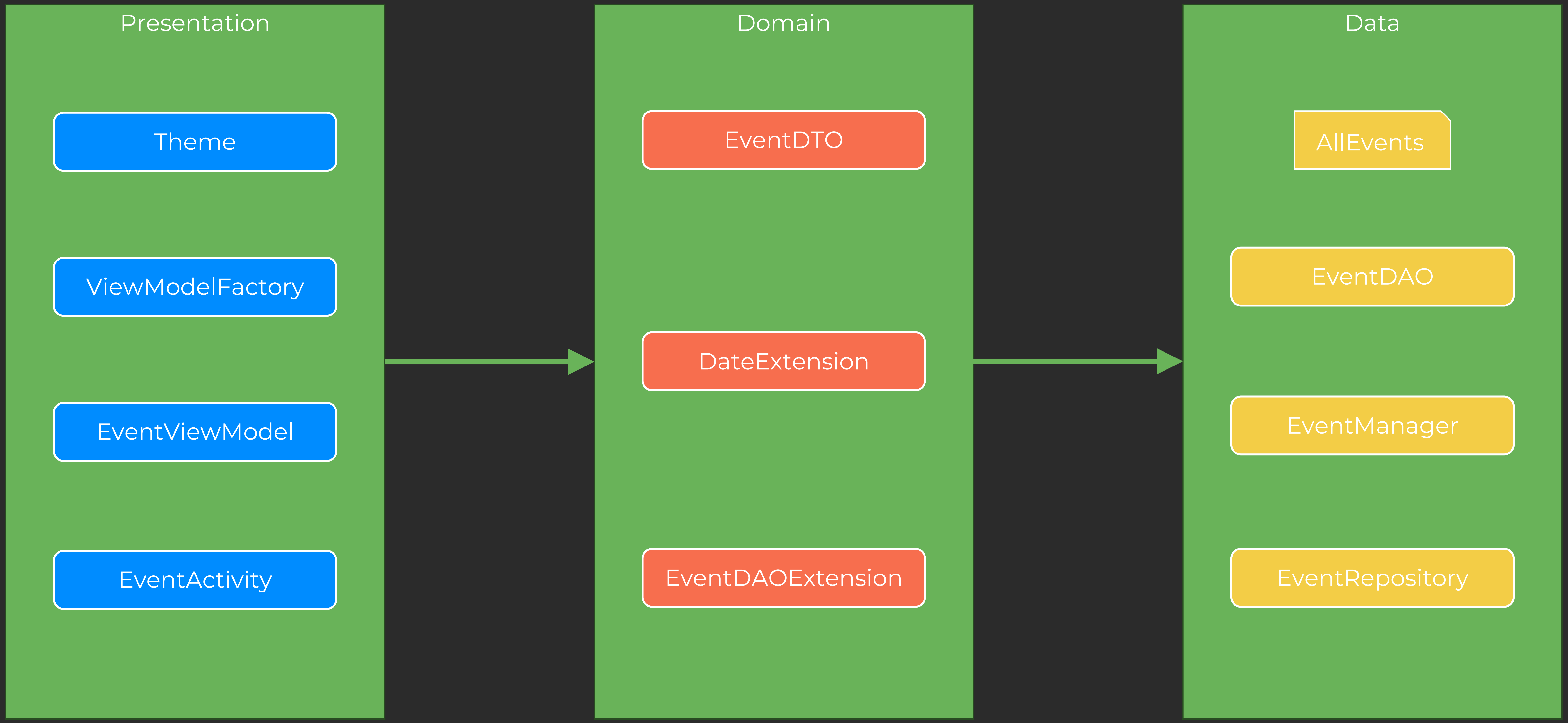


Total attendees: 0

Application architecture

Up Next

Application architecture



Starter project

Up Next

Create EventManager

Up Next

Complete repository layer

Up Next

Start and end event

Up Next

Join and leave event per participant

Up Next

Showing event data

Up Next

Coding challenge

Up Next

Coding challenge solution

Up Next

Closure

- Flow, AbstractFlow, StateFlow, SharedFlow
- How to leverage concept of replayCache and extraBufferCapacity for SharedFlow
- Initial state and listening to further changes in StateFlow

Thank you

Hardik Trivedi

@MrHardikTrivedi

in/mrhardiktrivedi