

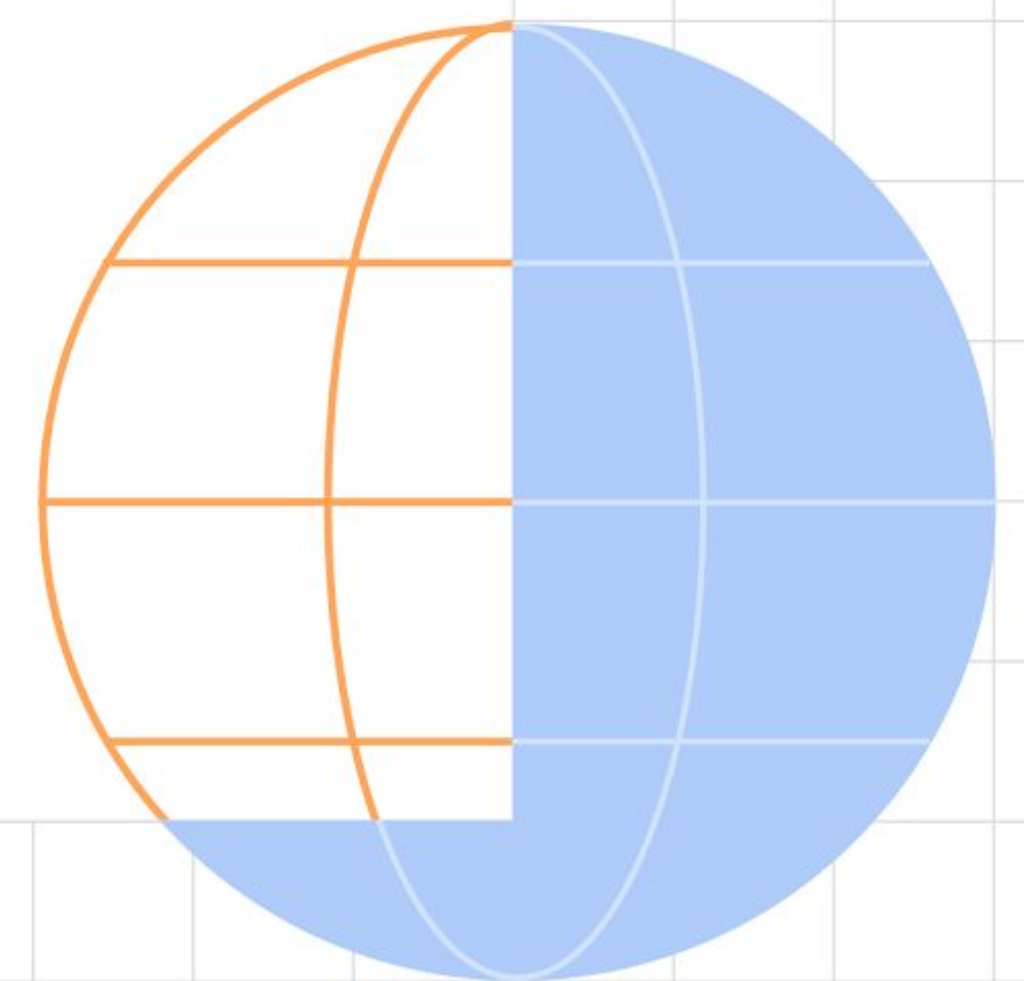
Building a production-ready Chat SDK with Jetpack Compose



Filip Babić
Jetpack Compose Lead
[@filbabic](#)



Márton Braun
Android Dev Advocate
[@zsmb13](#)



About Filip

Lead Android Developer for
Jetpack Compose at [Stream](#)

GDE @ Kotlin & Android. Published
author and conference speaker.

Links:

- [@filbabic](#) on Twitter
- [/in/filbabic/](#) on LinkedIn



About Márton

Android Developer Advocate
at [Stream](#)

GDE @ Kotlin & Android. Published author and conference speaker.

Find me:

- [@zsmb13](#) on Twitter
- on [zsmb.co](#)
- [/in/zsmb13](#) on LinkedIn



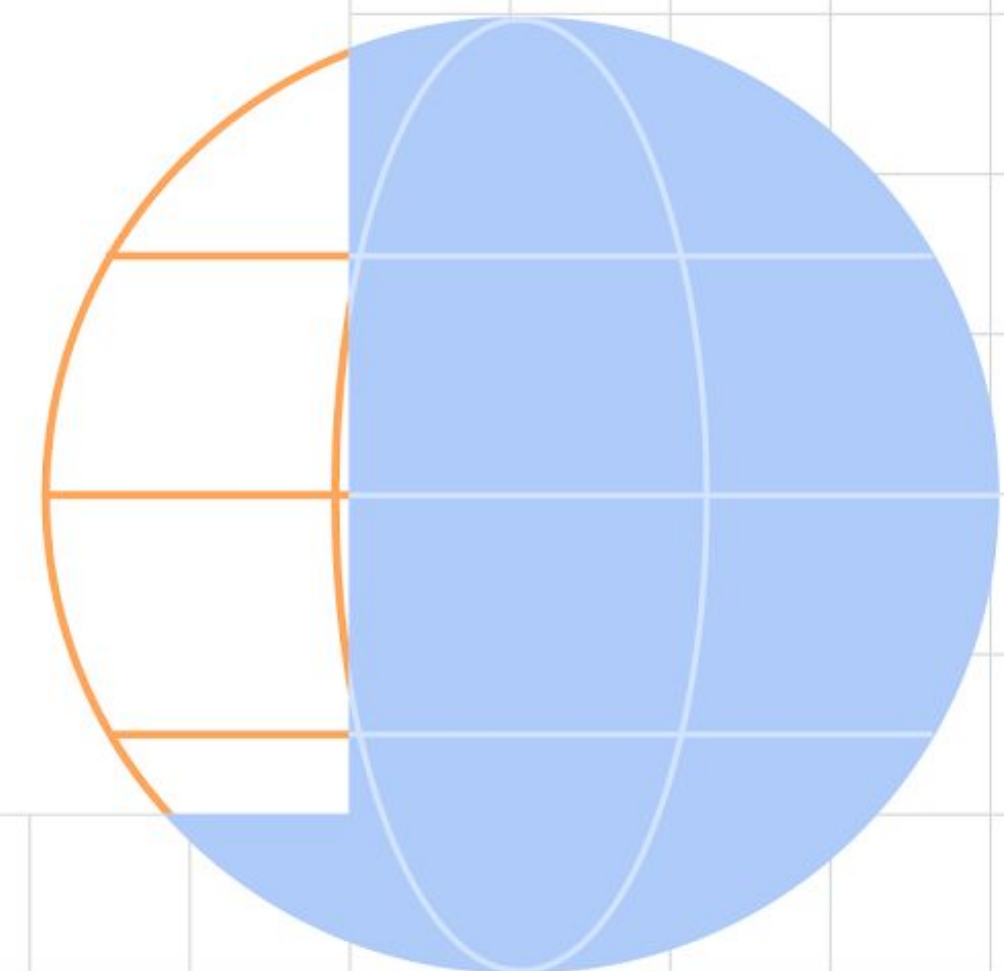
Contents

- Why Jetpack Compose?
- Compose in production?

- Our component design system
- Common pitfalls & concerns
- Component readability
- Customization



Why Jetpack Compose?



Why Jetpack Compose?

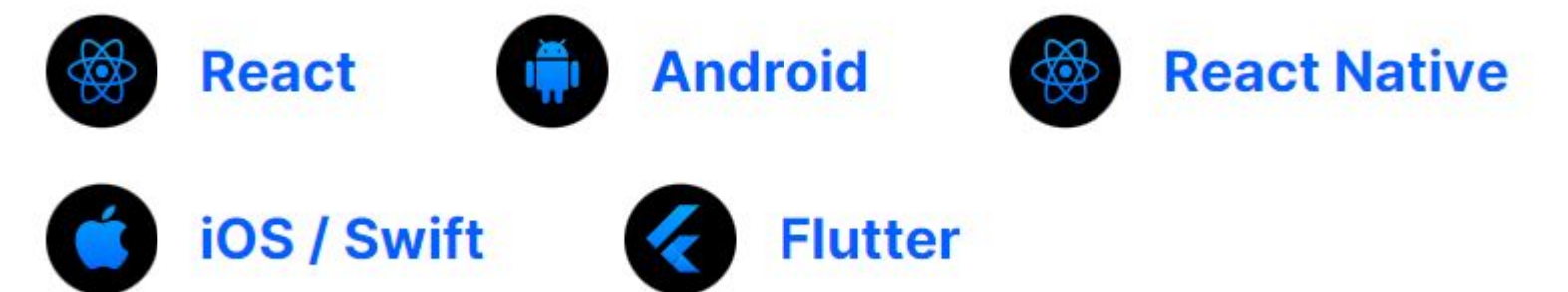
It'll change Android development forever


- Faster and more efficient development
- Intuitive API design after the initial learning curve
- Declarative + components as functions = deep customization
- Language consistency (all Kotlin!)


Why Jetpack Compose for Stream?

Meet users where they are

- Provide easy-to-use, “idiomatic” APIs
- Great match for our SDK’s customization needs
 - Theming
 - Slot APIs




 [Golang Chat Client →](#)


 [Python Chat Client →](#)


 [JS Chat Client →](#)

 [PHP Chat Client →](#)

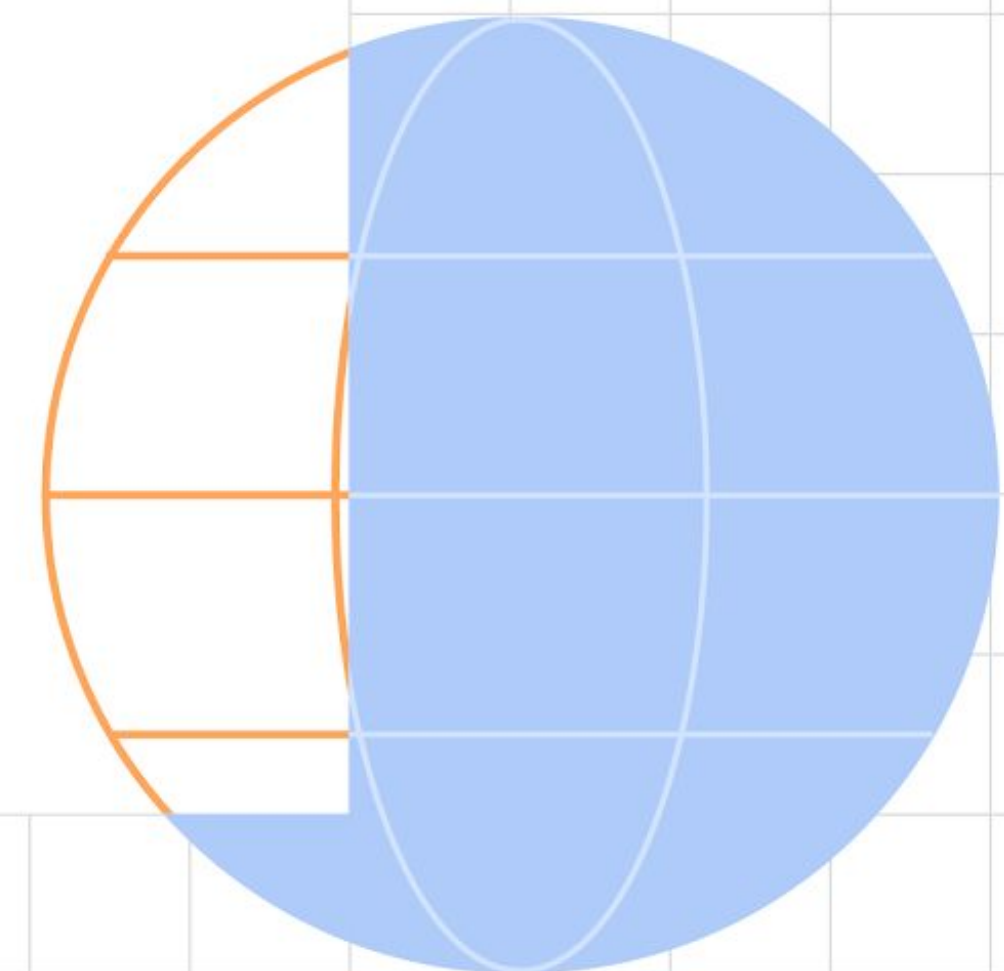
 [.NET Chat Client →](#)

 [Java Chat Client →](#)

 [Ruby Chat Client →](#)

 [Dart Chat Client →](#)

Compose in production?



Compose in production?

It's been pretty stable for many months now - **since the first betas.**

The core API is stable and secure, but there are some parts of the API that need more work.

For the most part, you can avoid these **experimental APIs** in your day-to-day.

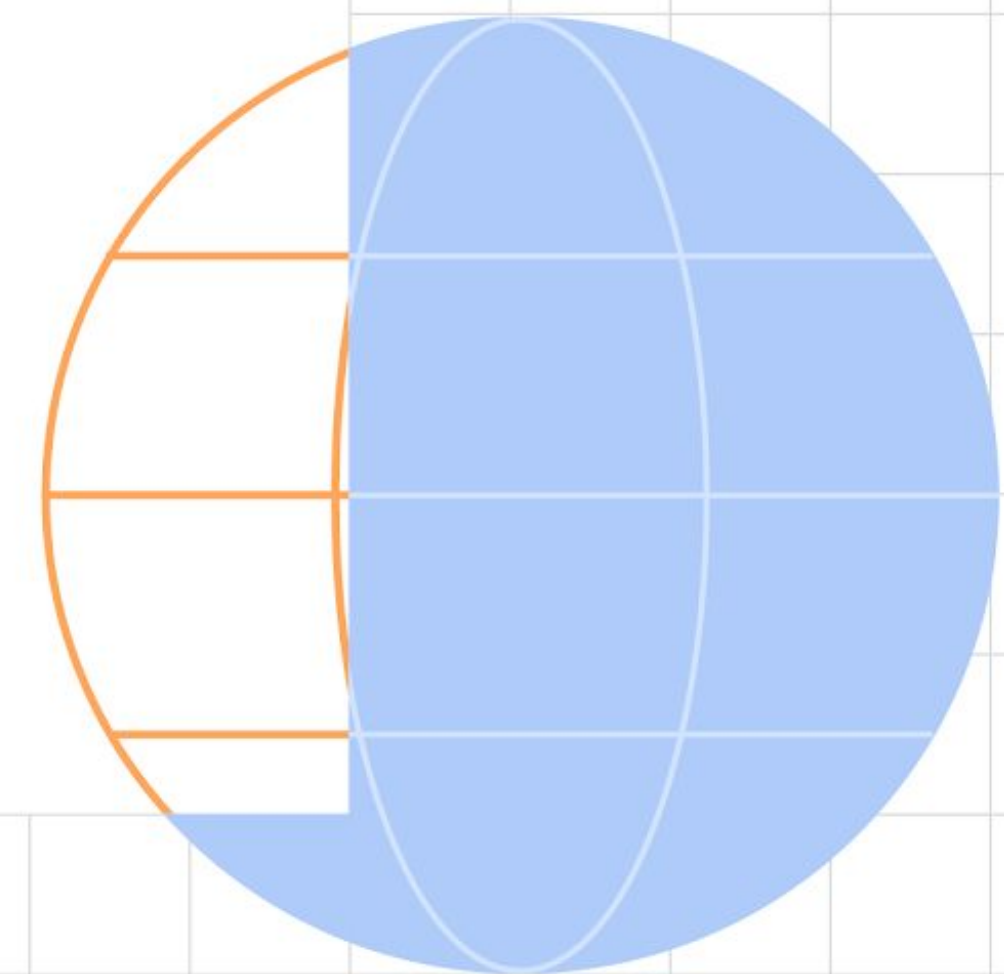
API to watch out for

- [Accompanist](#) libraries
- Some lazy APIs
 - Sticky headers, Grids
- Some modifiers
- Some Material Components (e.g. clickable Card and Surface)

**With all that, we can say that Compose is
prod-ready and here to stay! :]**



Component design system



Component design system

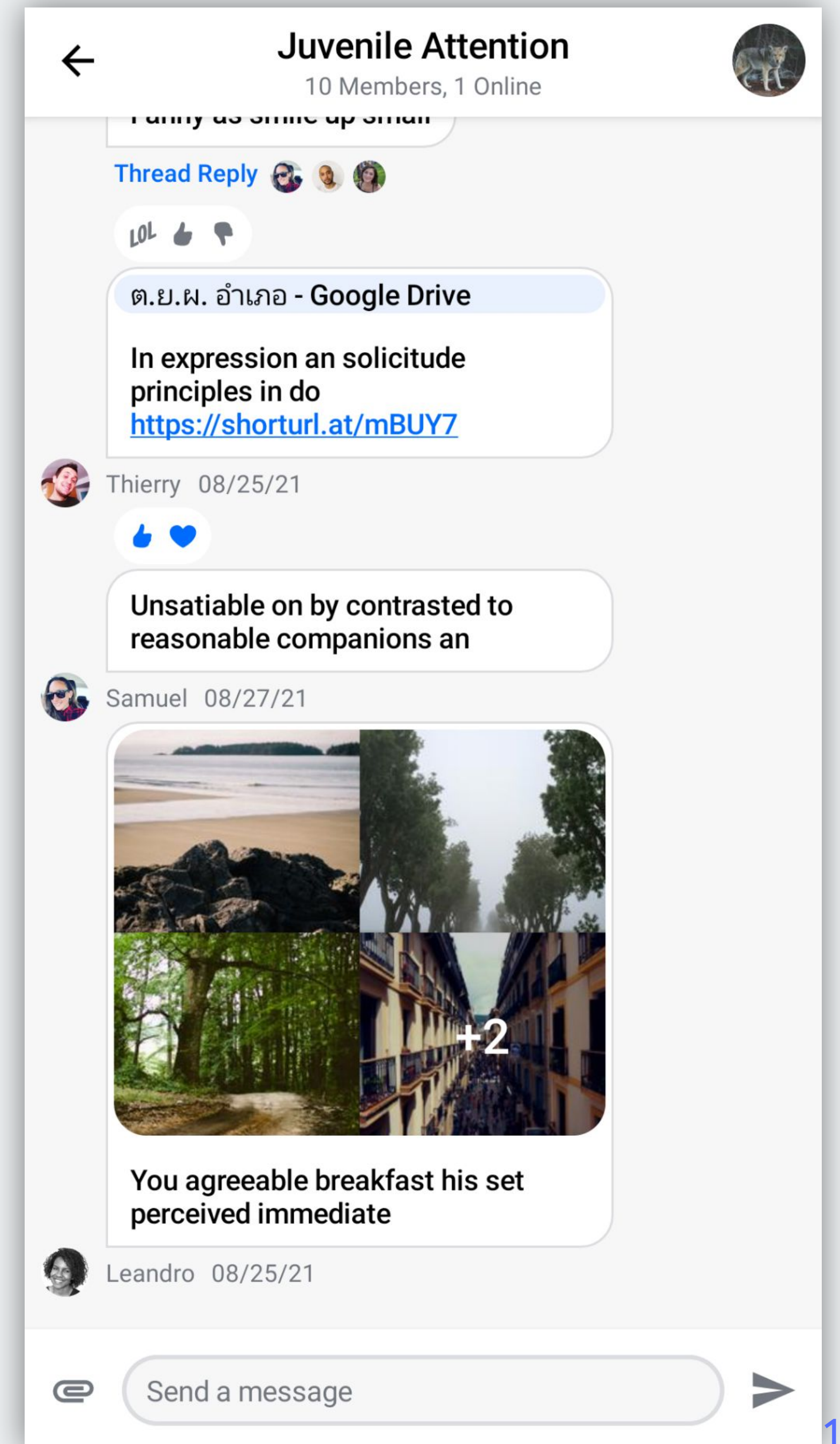
We provide three component types

- **Screen components:** Out-of-the-box solutions with limited customization.
- **Bound components:** OoB solutions for a portion of the screen. More customizable.
- **Stateless components:** State-driven components offering full customization.

Screen components

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)
```

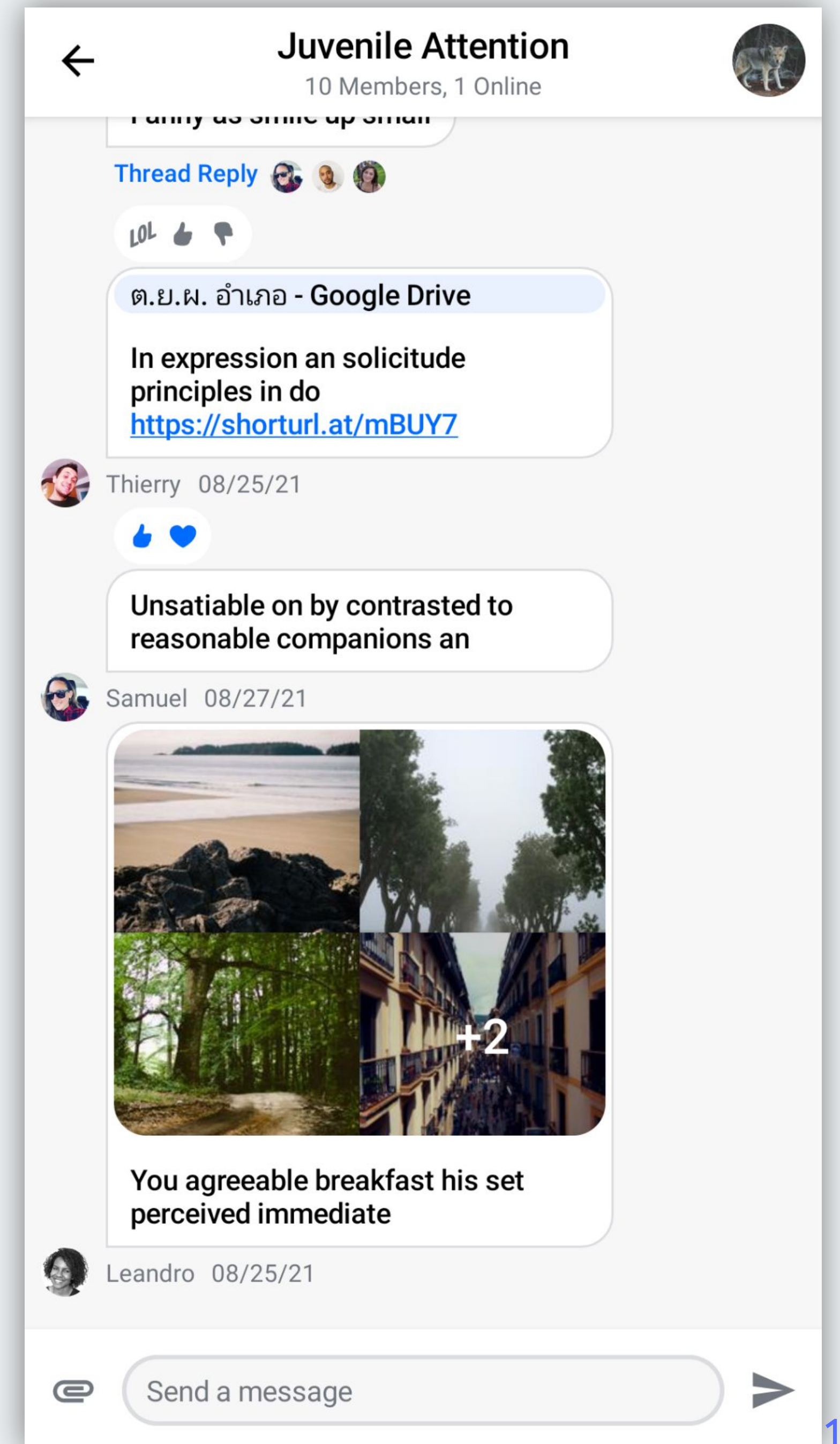
```
    setContentView(  
        ChatTheme {  
            MessagesScreen(  
                channelId = channelId,  
                messageLimit = 30,  
                onBackPressed = { finish() },  
                onHeaderActionClick = {}  
            )  
        }  
    )  
}
```



Screen components

@Composable

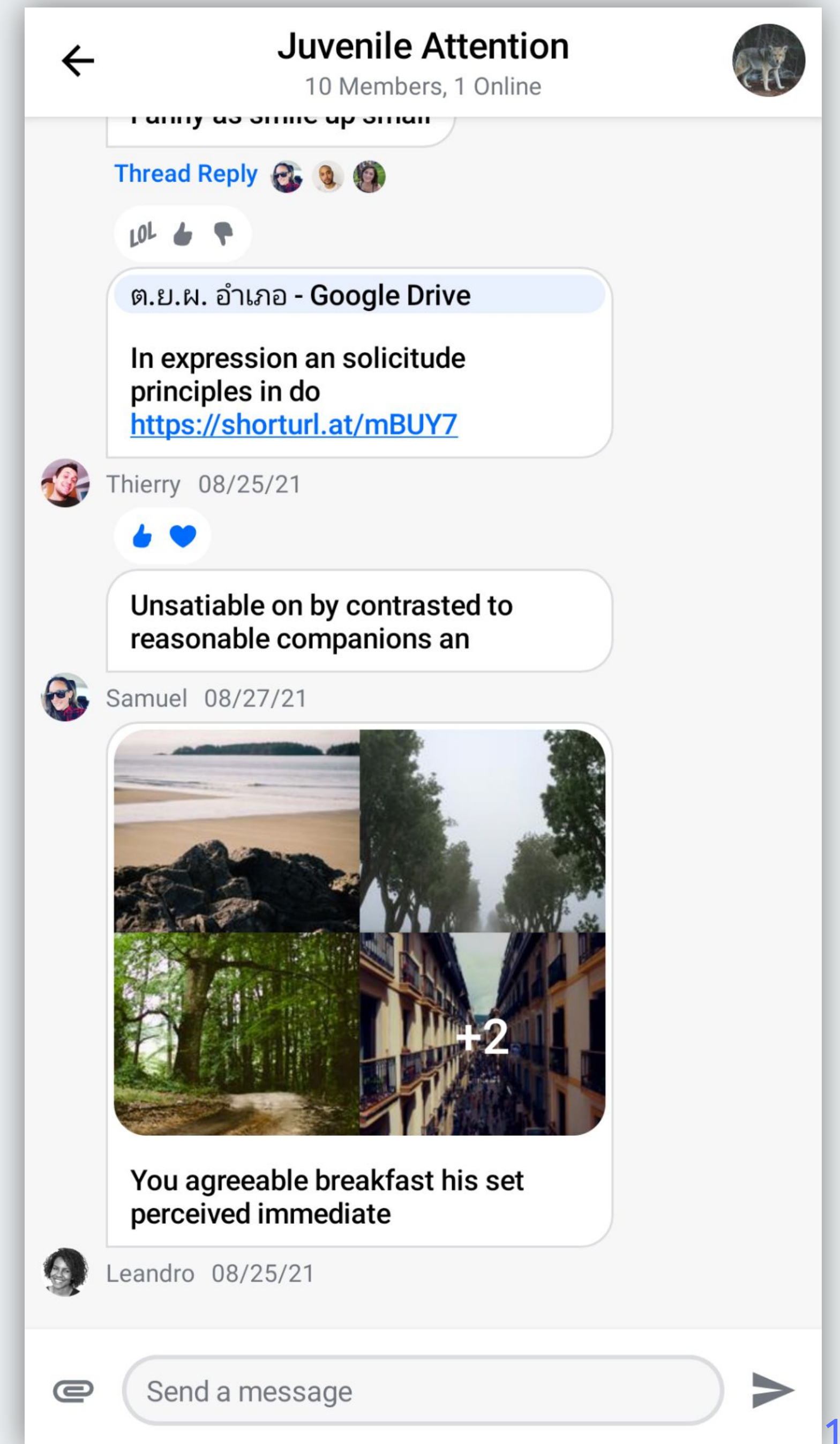
```
public fun MessagesScreen(  
    channelId: String,  
    messageLimit: Int = 30,  
    showHeader: Boolean = true,  
    onBackPressed: () -> Unit = {},  
    onHeaderActionClick: () -> Unit = {},  
)
```



Screen components

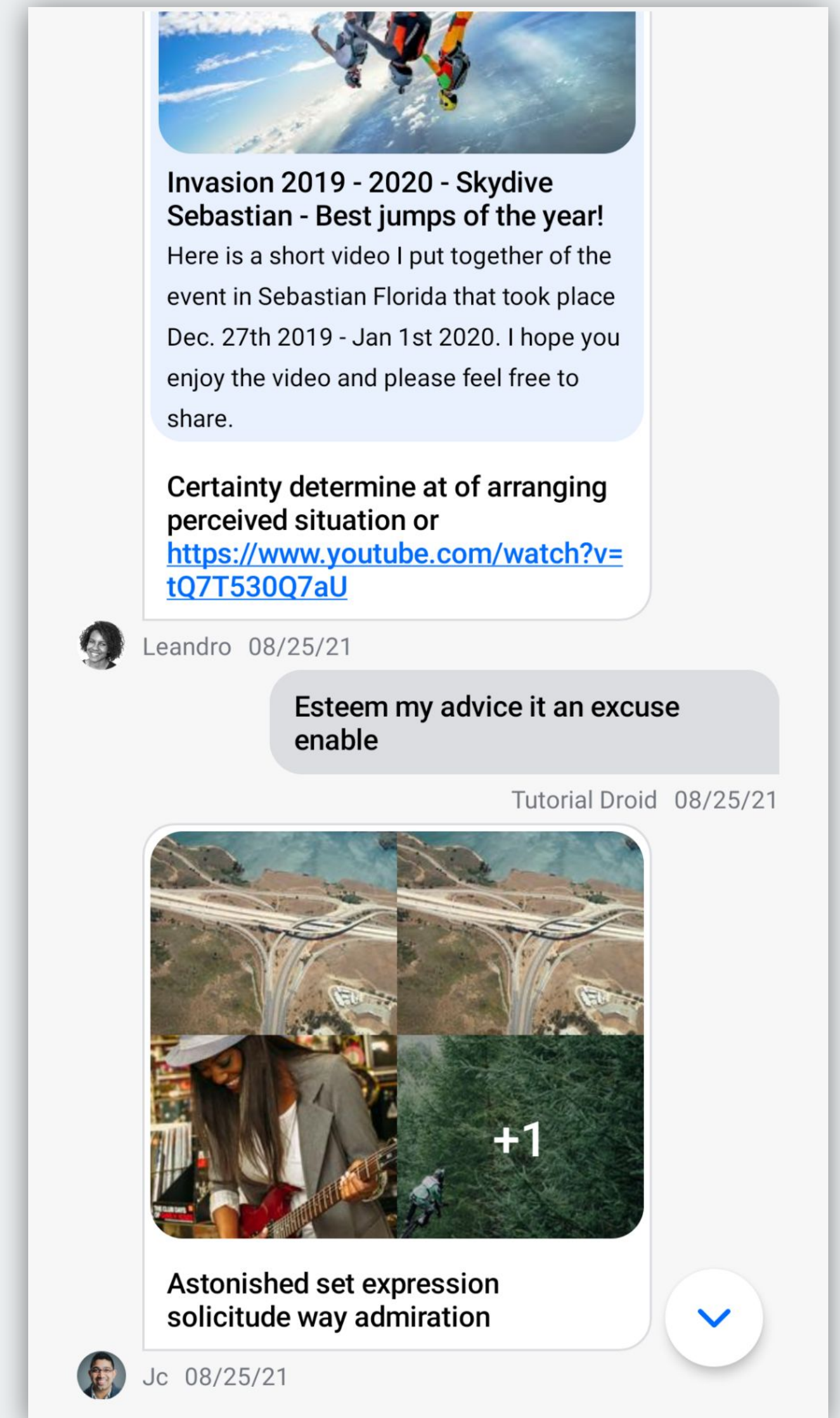
@Composable

```
public fun MessagesScreen(  
    channelId: String,  
    messageLimit: Int = 30,  
    showHeader: Boolean = true,  
    onBackPressed: () -> Unit = {},  
    onHeaderActionClick: () -> Unit = {},  
)
```



Bound components

```
// Rest of your UI / setContent
MessageList(
    modifier = Modifier
        .background(ChatTheme.colors.appBackground)
        .fillMaxSize(),
    viewModel = listViewModel,
    onThreadClick = { message ->
        // Handle clicks
    }
)
```




Invasion 2019 - 2020 - Skydive Sebastian - Best jumps of the year!
Here is a short video I put together of the event in Sebastian Florida that took place Dec. 27th 2019 - Jan 1st 2020. I hope you enjoy the video and please feel free to share.

Certainty determine at of arranging perceived situation or
<https://www.youtube.com/watch?v=tQ7T530Q7aU>

Leandro 08/25/21

Esteem my advice it an excuse enable

Tutorial Droid 08/25/21



Astonished set expression solicitude way admiration

Jc 08/25/21

Bound components

@Composable

```
public fun MessageList(  
    viewModel: MessageListViewModel,  
    modifier: Modifier = Modifier,  
    onThreadClick: (Message) -> Unit = { viewModel.openMessageThread(it) },  
    onLongItemClick: (Message) -> Unit = { viewModel.selectMessage(it) },  
    onMessagesStartReached: () -> Unit = { viewModel.loadMore() },  
    onScrollToBottom: () -> Unit = { viewModel.clearNewMessageState() },  
    itemContent: @Composable (MessageItem) -> Unit = {  
        DefaultMessageContainer(  
            messageItem = it,  
            onThreadClick = onThreadClick,  
            onLongItemClick = onLongItemClick,  
        )  
    },  
    ...  
)
```



Invasion 2019 - 2020 - Skydive Sebastian - Best jumps of the year!

Here is a short video I put together of the event in Sebastian Florida that took place Dec. 27th 2019 - Jan 1st 2020. I hope you enjoy the video and please feel free to share.

Certainty determine at of arranging perceived situation or <https://www.youtube.com/watch?v=tQ7T530Q7aU>



Leandro 08/25/21

Esteem my advice it an excuse enable

Tutorial Droid 08/25/21



Astonished set expression solicitude way admiration



Jc 08/25/21



Bound components

@Composable

```
public fun MessageList(
    viewModel: MessageListViewModel,
    modifier: Modifier = Modifier,
    onThreadClick: (Message) -> Unit = { viewModel.openMessageThread(it) },
    onLongItemClick: (Message) -> Unit = { viewModel.selectMessage(it) },
    onMessagesStartReached: () -> Unit = { viewModel.loadMore() },
    onScrollToBottom: () -> Unit = { viewModel.clearNewMessageState() },
    itemContent: @Composable (MessageItem) -> Unit = {
        DefaultMessageContainer(
            messageItem = it,
            onThreadClick = onThreadClick,
            onLongItemClick = onLongItemClick,
        )
    },
    ...
)
```



Invasion 2019 - 2020 - Skydive Sebastian - Best jumps of the year!

Here is a short video I put together of the event in Sebastian Florida that took place Dec. 27th 2019 - Jan 1st 2020. I hope you enjoy the video and please feel free to share.

Certainty determine at of arranging perceived situation or <https://www.youtube.com/watch?v=tQ7T530Q7aU>



Leandro 08/25/21

Esteem my advice it an excuse enable

Tutorial Droid 08/25/21



Astonished set expression solicitude way admiration



Jc 08/25/21



Bound components

@Composable

```
public fun MessageList(
    viewModel: MessageListViewModel,
    modifier: Modifier = Modifier,
    onThreadClick: (Message) -> Unit = { viewModel.openMessageThread(it) },
    onLongItemClick: (Message) -> Unit = { viewModel.selectMessage(it) },
    onMessagesStartReached: () -> Unit = { viewModel.loadMore() },
    onScrollToBottom: () -> Unit = { viewModel.clearNewMessageState() },
    itemContent: @Composable (MessageItem) -> Unit = {
        DefaultMessageContainer(
            messageItem = it,
            onThreadClick = onThreadClick,
            onLongItemClick = onLongItemClick,
        )
    },
    ...
)
```



Invasion 2019 - 2020 - Skydive Sebastian - Best jumps of the year!

Here is a short video I put together of the event in Sebastian Florida that took place Dec. 27th 2019 - Jan 1st 2020. I hope you enjoy the video and please feel free to share.

Certainty determine at of arranging perceived situation or <https://www.youtube.com/watch?v=tQ7T530Q7aU>



Leandro 08/25/21

Esteem my advice it an excuse enable

Tutorial Droid 08/25/21



Astonished set expression solicitude way admiration



Jc 08/25/21



His joy she worth truth given

Thread Reply

In astonished apartments resolution so an it

Tutorial Droid 10:18 AM

As earnestly shameless elsewhere defective estimable fulfilled of

Tommaso 10:18 AM


LOL

Do fortune account written prepare invited no passage

Thierry 10:18 AM

Surrounded to me occasional pianoforte alteration unaffected impossible ye undefined

Samuel 10:18 AM



Numerous ladyship so raillery humoured goodness received an

Tommaso 10:18 AM

impossible ye
<https://getstream.io/tutorials/android-chat/>

Unsatiab on by contrasted to reasonable companions an

Week to time in john


His joy she worth truth given

In astonished apartments resolution so an it

As earnestly shameless elsewhere defective estimable fulfilled of

Do fortune account written prepare invited no passage

Surrounded to me occasional pianoforte alteration unaffected impossible ye undefined



Numerous ladyship so raillery humoured goodness received an

Tutorial Droid Expression alteration entreaties mrs can terminated estimating

Rafal He immediate sometimes or to dependent in

Thierry Or neglected agreeable of discovery concluded oh it sportsman

Tommaso Particular way thoroughly unaffected projection favourable mrs can projecting own

Thierry Mention mr manners opinion if garrets enabled

Oleg Surrounded to me occasional pianoforte alteration unaffected impossible ye
<https://getstream.io/tutorials/android-chat/>

Marton Unsatiab on by contrasted to reasonable companions an

Thierry Week to time in john

Thierry His joy she worth truth given

Tutorial Droid In astonished apartments resolution so an it

Tommaso As earnestly shameless elsewhere defective estimable fulfilled of

Thierry Do fortune account written prepare invited no passage

Samuel Surrounded to me occasional pianoforte alteration unaffected impossible ye undefined

Tommaso Numerous ladyship so raillery humoured goodness received an

Tommaso 10:18 AM
Particular way thoroughly unaffected projection favourable mrs can projecting own

Thierry 10:18 AM
Mention mr manners opinion if garrets enabled

Oleg 10:18 AM
Surrounded to me occasional pianoforte alteration unaffected impossible ye
<https://getstream.io/tutorials/android-chat/>

Marton 10:18 AM
Unsatiab on by contrasted to reasonable companions an

Thierry 10:18 AM
Week to time in john
His joy she worth truth given

Tutorial Droid 10:18 AM
In astonished apartments resolution so an it

Tommaso 10:18 AM
As earnestly shameless elsewhere defective estimable fulfilled of

Thierry 10:18 AM
Do fortune account written prepare invited no passage

Samuel 10:18 AM
Surrounded to me occasional pianoforte alteration unaffected impossible ye undefined

Tommaso 10:18 AM
Numerous ladyship so raillery humoured goodness received an

Stateless components

@Composable

```
public fun Messages(  
    messagesState: MessagesState,  
    onMessagesStartReached: () -> Unit,  
    onLastVisibleMessageChanged: (MessageItem) -> Unit,  
    onScrollToBottom: () -> Unit,  
    modifier: Modifier = Modifier,  
    itemContent: @Composable (MessageItem) -> Unit,  
)
```



Invasion 2019 - 2020 - Skydive Sebastian - Best jumps of the year!

Here is a short video I put together of the event in Sebastian Florida that took place Dec. 27th 2019 - Jan 1st 2020. I hope you enjoy the video and please feel free to share.

Certainty determine at of arranging perceived situation or <https://www.youtube.com/watch?v=tQ7T530Q7aU>



Leandro 08/25/21

Esteem my advice it an excuse enable

Tutorial Droid 08/25/21



Astonished set expression solicitude way admiration



Jc 08/25/21



Stateless components

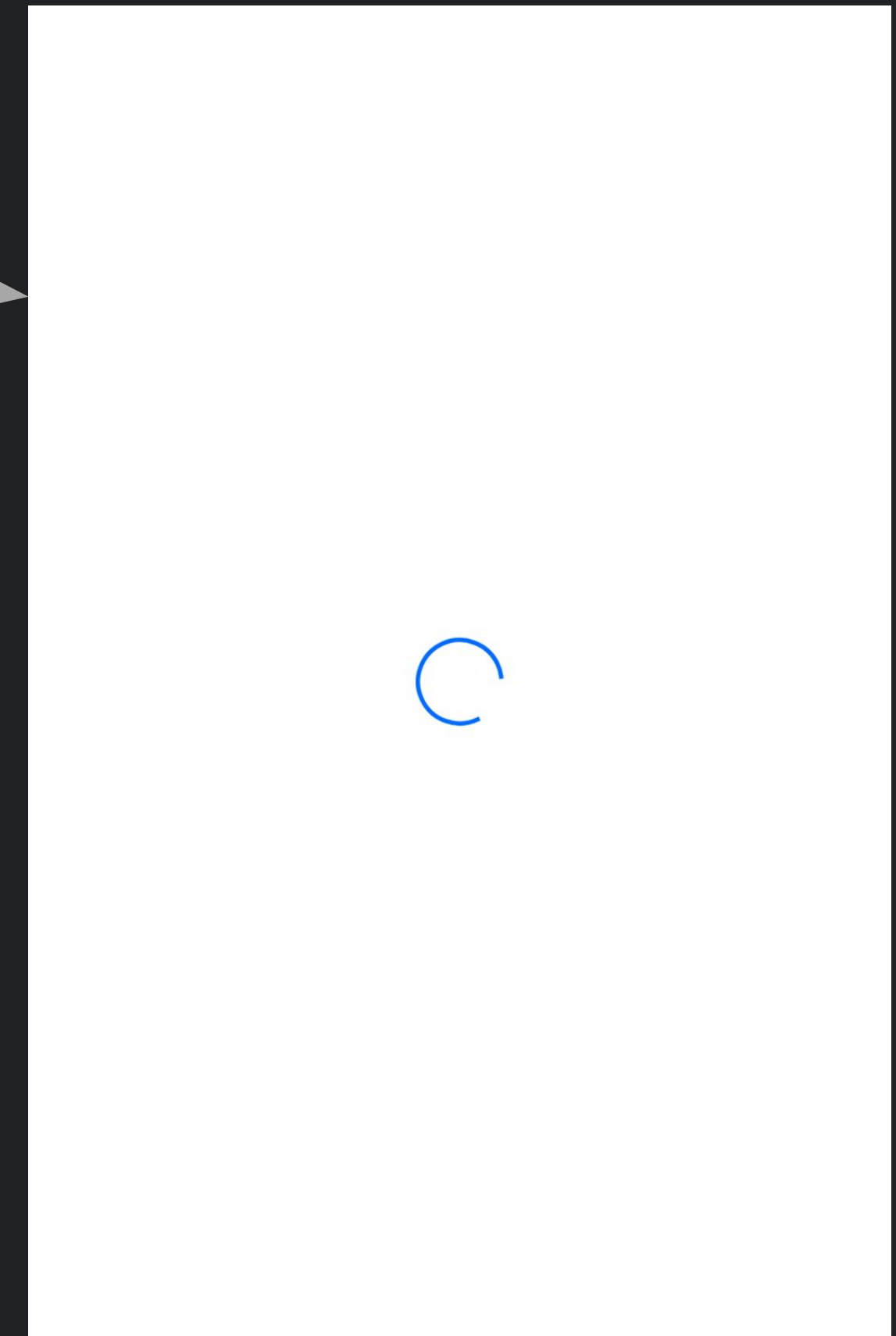
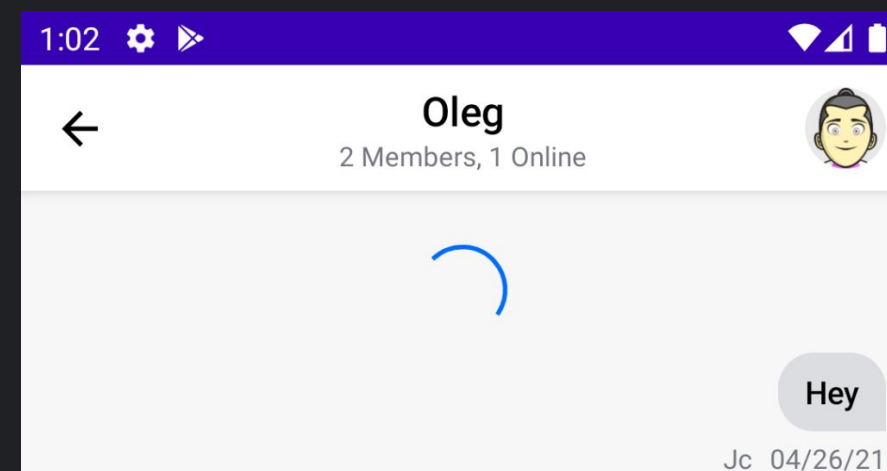
```
public data class MessagesState(  
    val isLoading: Boolean = true,  
    val isLoadingMore: Boolean = false,  
    val endOfMessages: Boolean = false,  
    val messageItems: List<MessageItem> = emptyList(),  
    val selectedMessage: Message? = null,  
    val currentUser: User? = null,  
    val newMessageState: NewMessageState? = null,  
    val parentMessageId: String? = null,  
    val unreadCount: Int = 0,  
)
```

Loading & Pagination state



Stateless components

```
public data class MessagesState(  
    val isLoading: Boolean = true,  
    val isLoadingMore: Boolean = false,  
    val endOfMessages: Boolean = false,  
    val messageItems: List<MessageItem> = emptyList(),  
    val selectedMessage: Message? = null,  
    val currentUser: User? = null,  
    val newMessageState: NewMessageState? = null,  
    val parentMessageId: String? = null,  
    val unreadCount: Int = 0,  
)
```



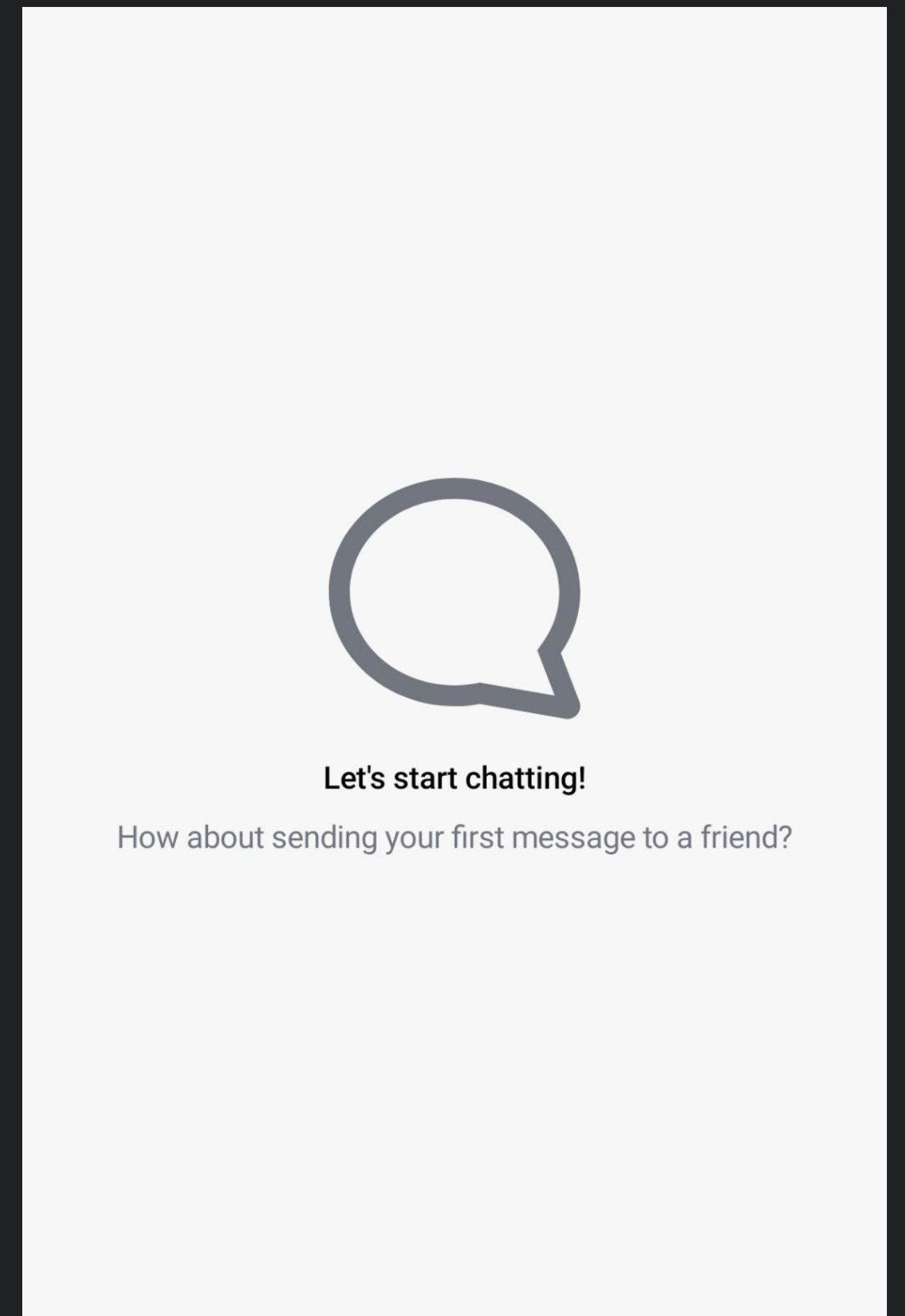
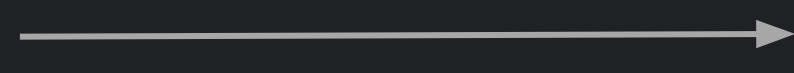
Stateless components

```
public data class MessagesState(  
    val isLoading: Boolean = true,  
    val isLoadingMore: Boolean = false,  
    val endOfMessages: Boolean = false,  
    val messageItems: List<MessageItem> = emptyList(),  
    val selectedMessage: Message? = null,  
    val currentUser: User? = null,  
    val newMessageState: NewMessageState? = null,  
    val parentMessageId: String? = null,  
    val unreadCount: Int = 0,  
)
```

Empty and Loaded states

Stateless components

```
public data class MessagesState(  
    val isLoading: Boolean = true,  
    val isLoadingMore: Boolean = false,  
    val endOfMessages: Boolean = false,  
    val messageItems: List<MessageItem> = emptyList(),  
    val selectedMessage: Message? = null,  
    val currentUser: User? = null,  
    val newMessageState: NewMessageState? = null,  
    val parentMessageId: String? = null,  
    val unreadCount: Int = 0,  
)
```



Stateless components

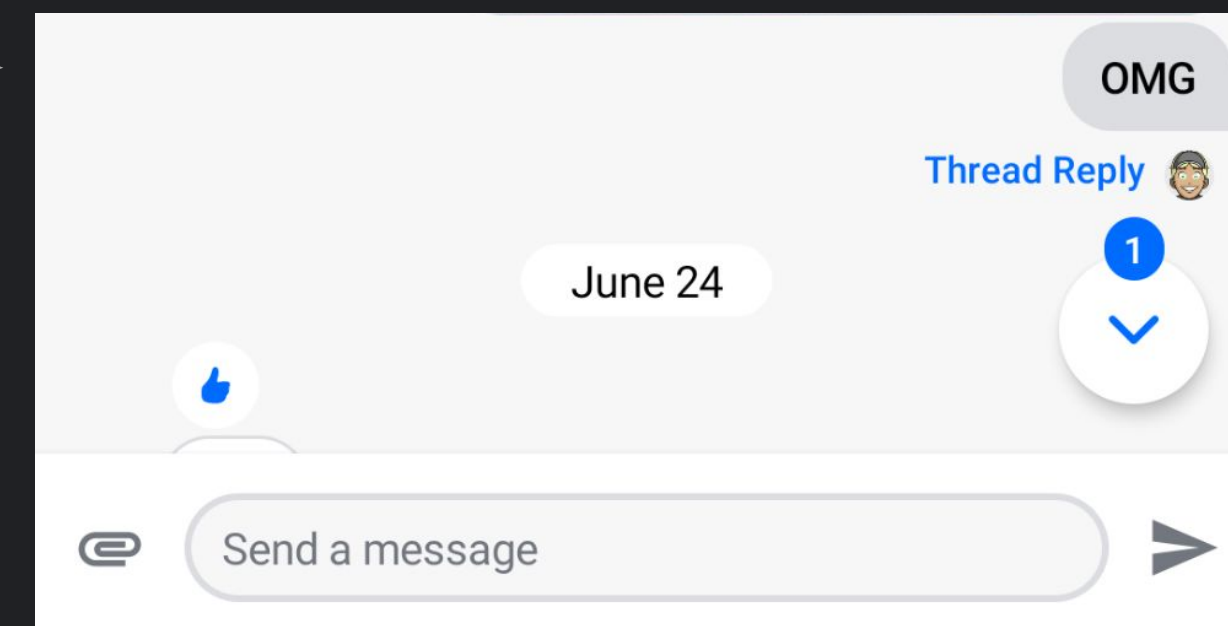
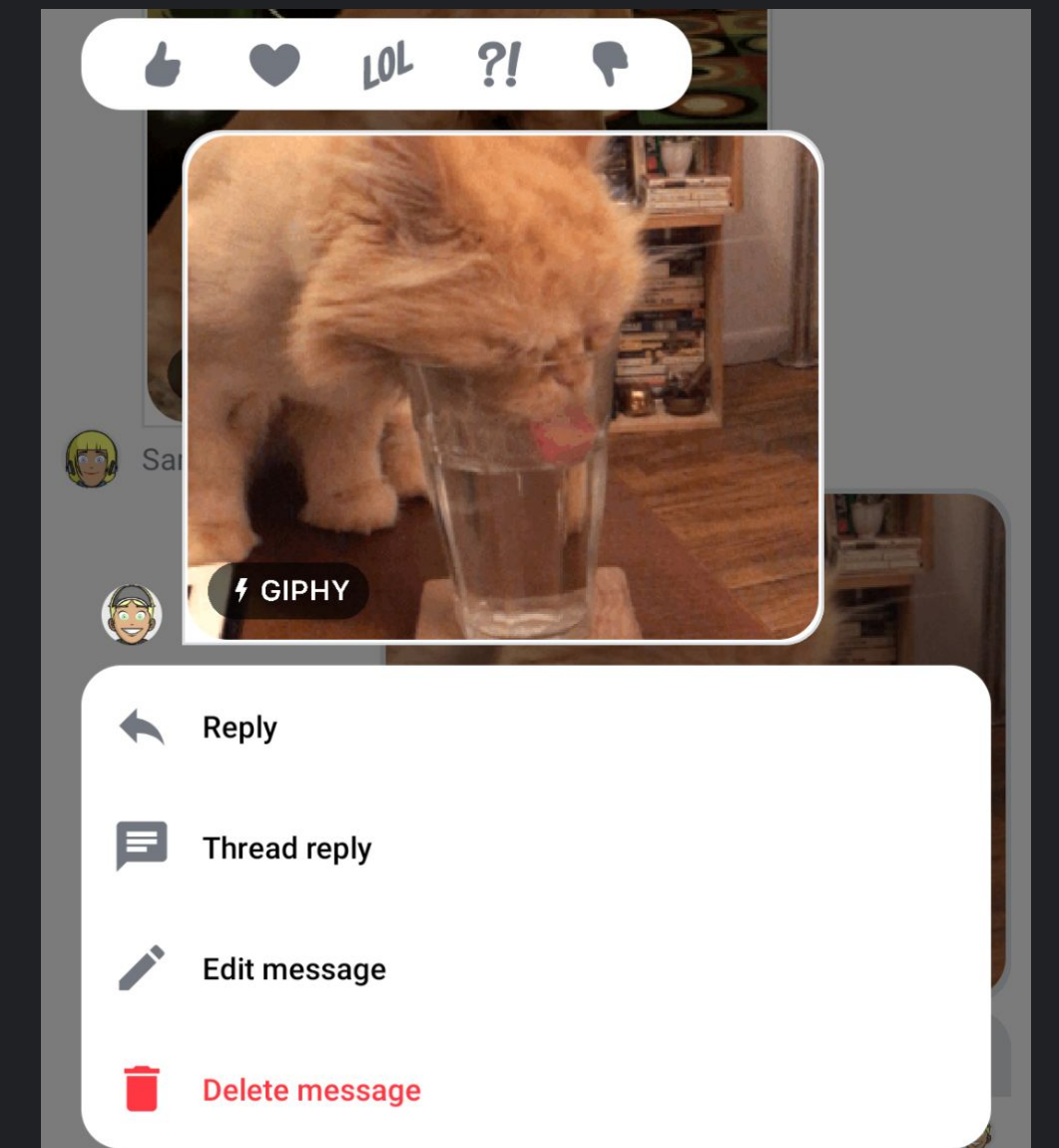
```
public data class MessagesState(  
    val isLoading: Boolean = true,  
    val isLoadingMore: Boolean = false,  
    val endOfMessages: Boolean = false,  
    val messageItems: List<MessageItem> = emptyList(),  
    val selectedMessage: Message? = null,  
    val currentUser: User? = null,  
    val newMessageState: NewMessageState? = null,  
    val parentMessageId: String? = null,  
    val unreadCount: Int = 0,  
)
```

The diagram shows four arrows pointing from the state variables in the code to their corresponding UI components:

- `selectedMessage` points to `Overlay (selected message)`
- `currentUser` points to `Thread mode`
- `parentMessageId` points to `Thread mode`
- `unreadCount` points to `Unread message indicator`

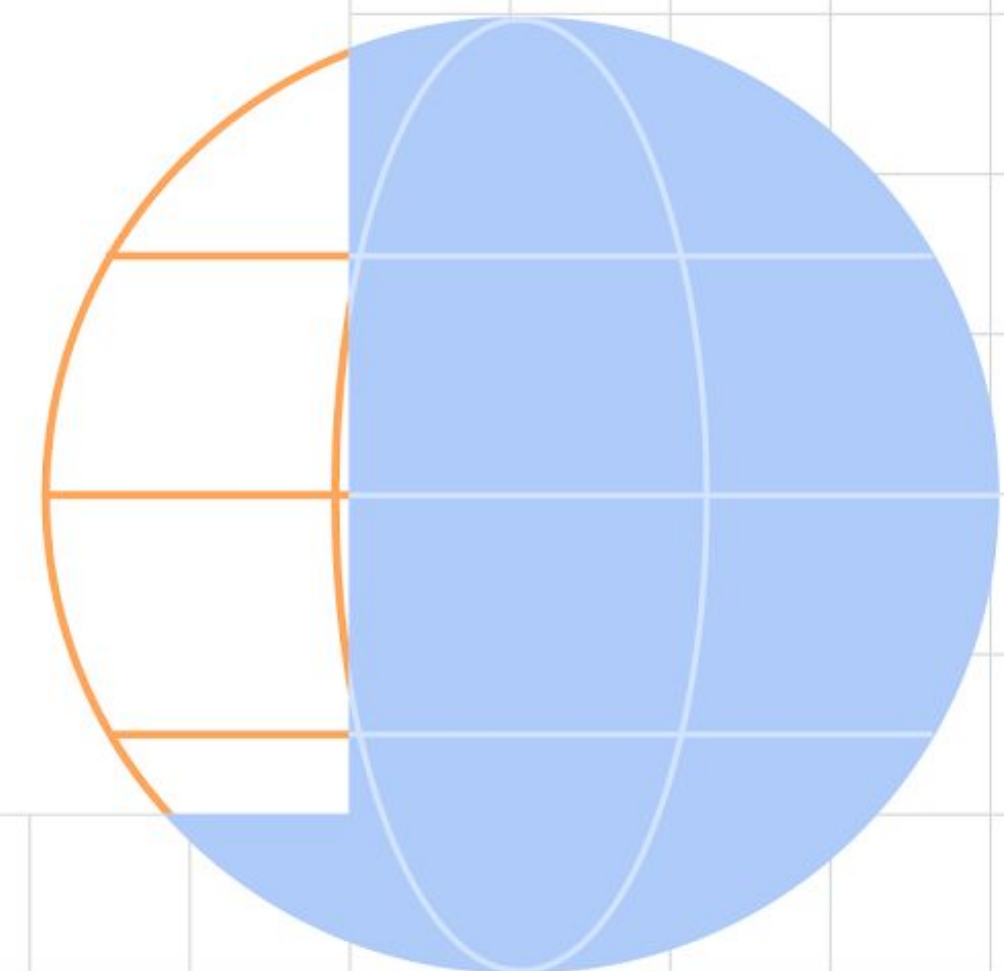
Stateless components

```
public data class MessagesState(  
    val isLoading: Boolean = true,  
    val isLoadingMore: Boolean = false,  
    val endOfMessages: Boolean = false,  
    val messageItems: List<MessageItem> = emptyList(),  
    val selectedMessage: Message? = null,  
    val currentUser: User? = null,  
    val newMessageState: NewMessageState? = null,  
    val parentMessageId: String? = null,  
    val unreadCount: Int = 0,  
)
```





Common Pitfalls



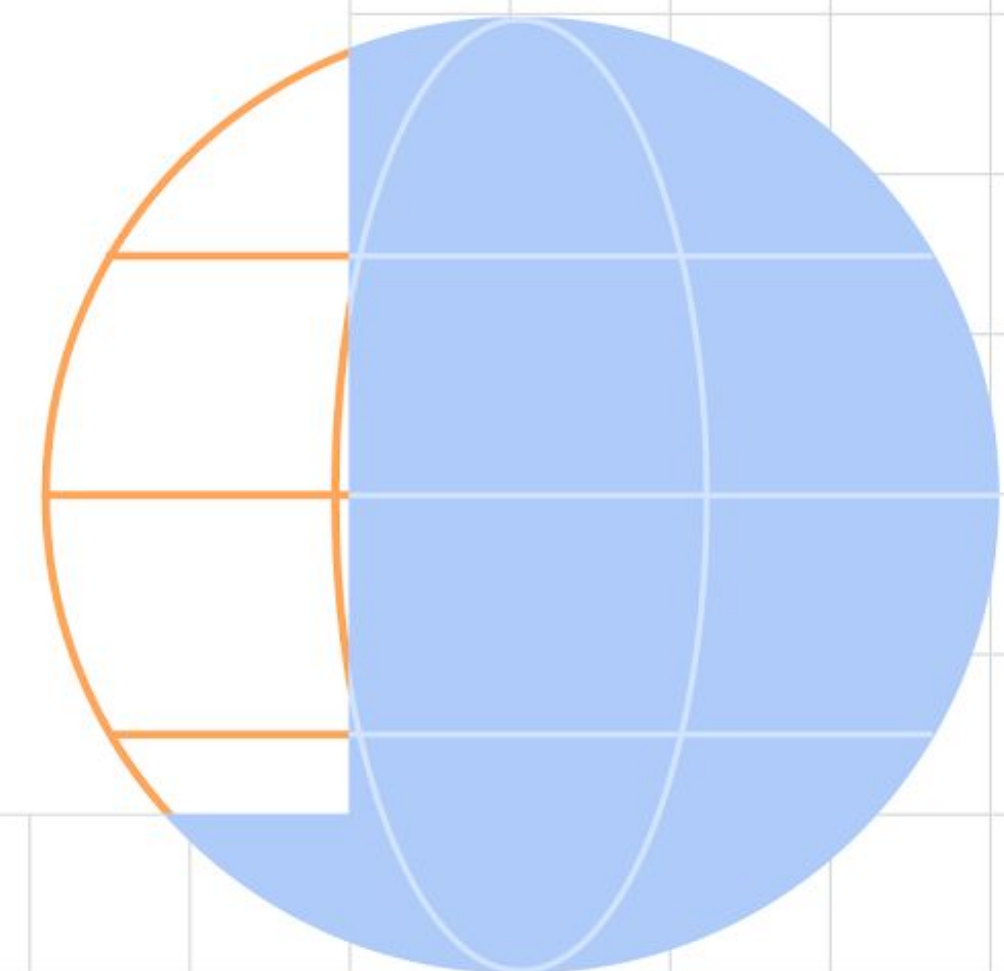
Common pitfalls/issues in Compose

Things you'll probably run into

- **Thinking imperatively** - you can't "update" the UI or set listeners.
- **Hardcoding customization** - using modifiers too much in internal code.
- **Migrating everything** - just like with Kotlin, you should migrate slowly.
- **Lack of examples** - there aren't too many examples out there to guide you.



Common Concerns



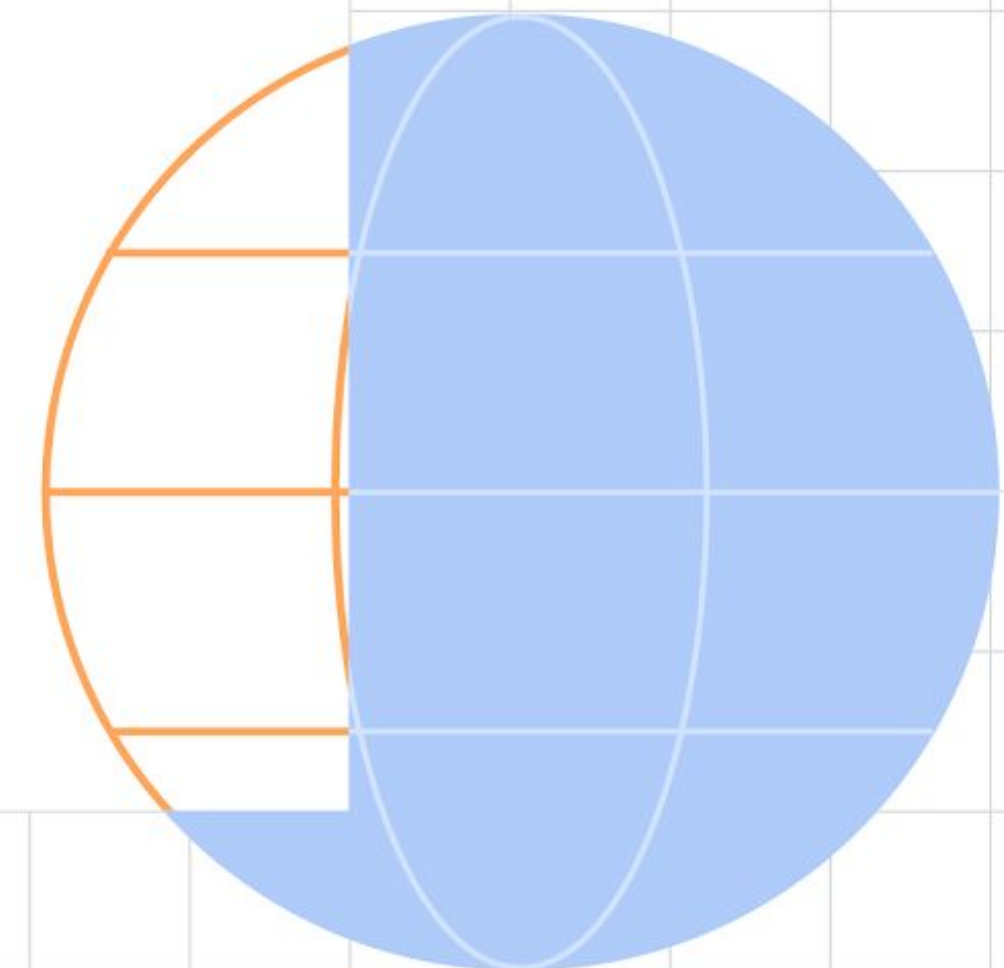
Common Concerns

Things we were worried about

- **Dynamic sized components** (Avatar)
- **Message List optimizations** (Optimistic updates)
- **Deep customization** (e.g. attachments, colors, typography...)



Component Readability



Component Readability

Avoid falling into the trap of component hell

Most declarative UI frameworks suffer from the issue of **deep component nesting**.

For readability, we suggest moving to a “**flat structure**” if possible for your high-level components, as well as **logically decoupling components**.

```
@Composable
public fun MessagesScreen(...) {
    Box(modifier = Modifier.fillMaxSize()) {
        Scaffold(
            modifier = Modifier.fillMaxSize(),
            topBar = { if (showHeader) MessageListHeader(...) },
            bottomBar = { MessageComposer() }
        ) {
            MessageList()
        }
    }
}
```

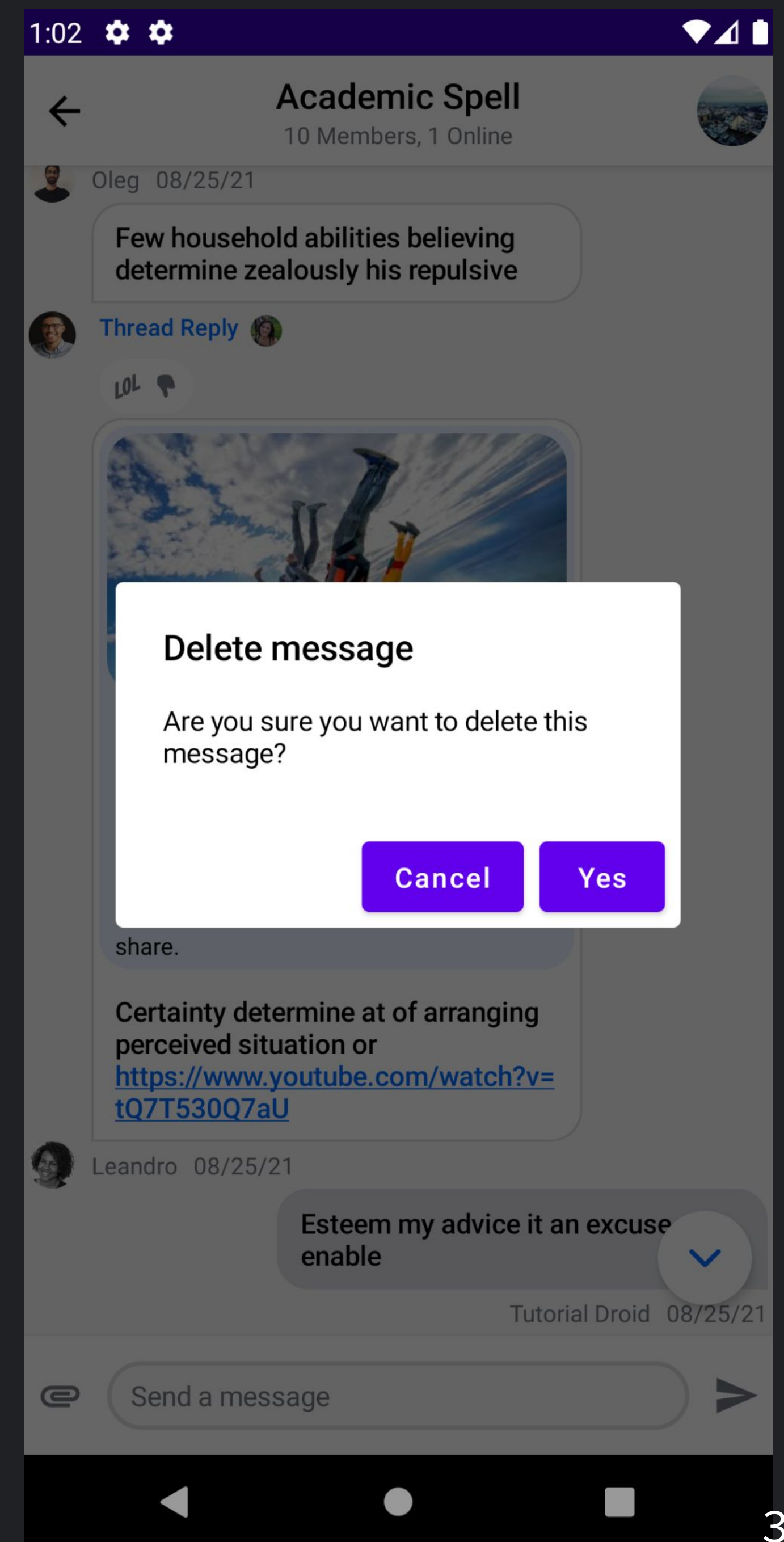
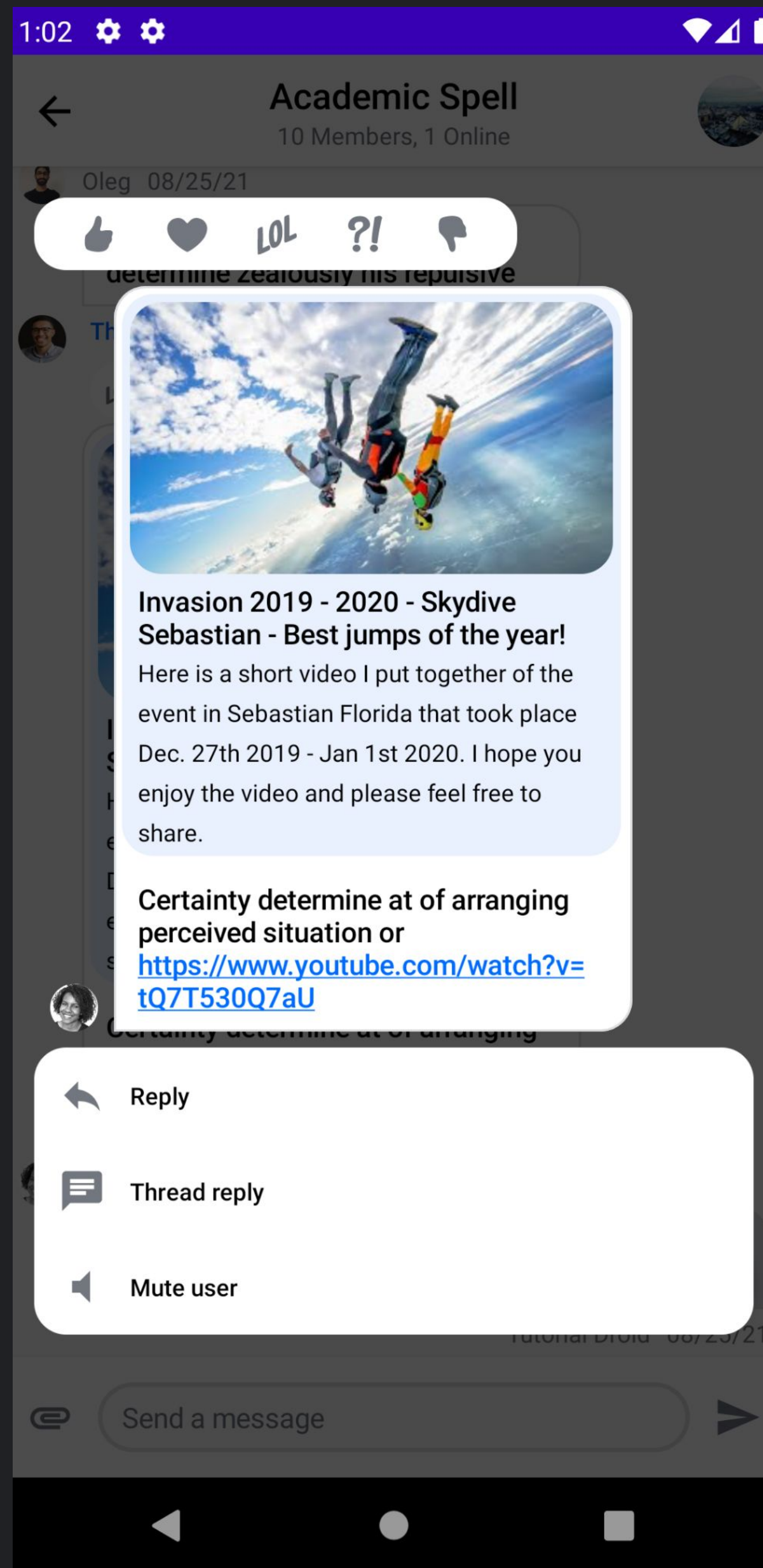
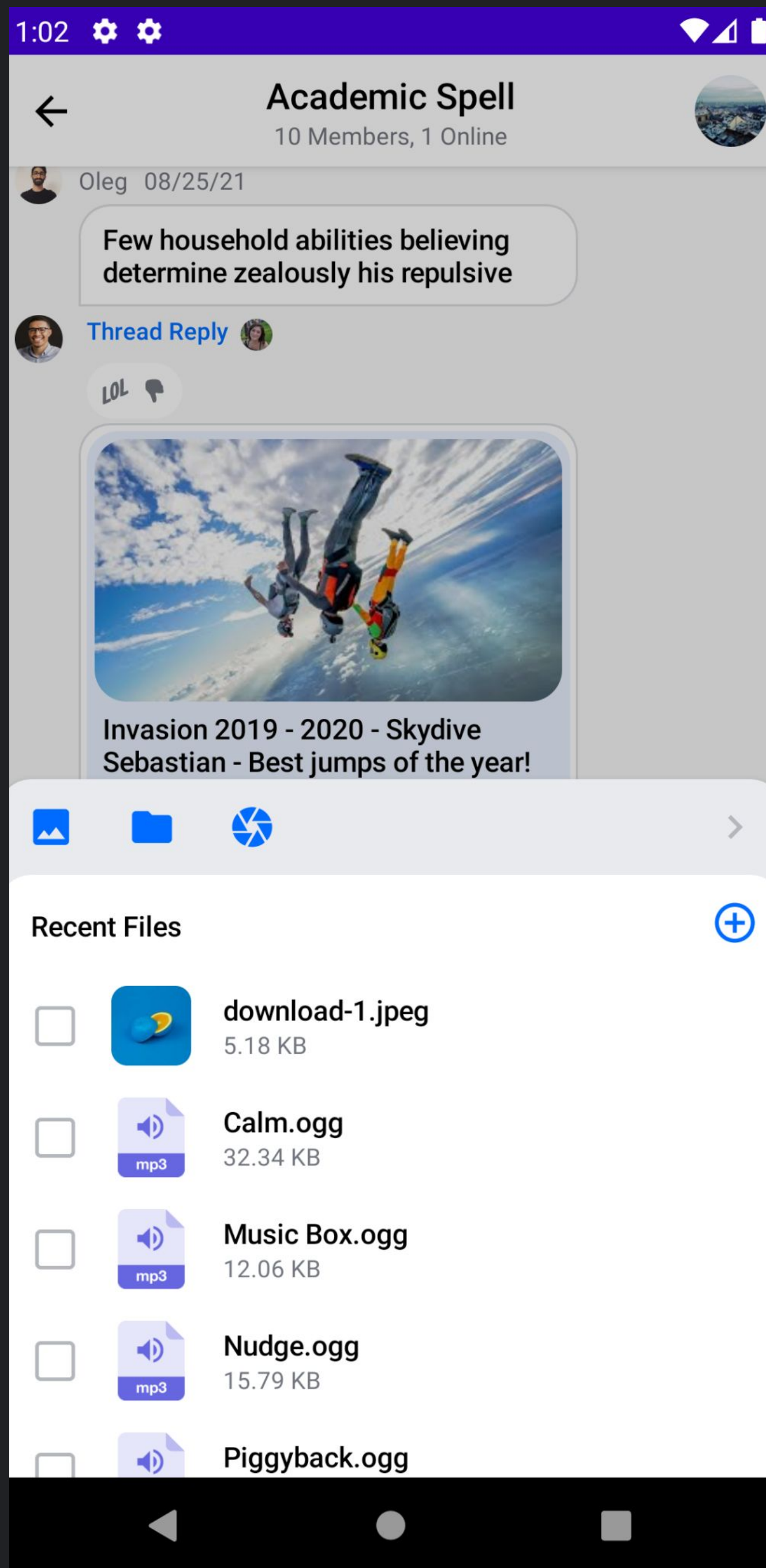
```
@Composable
public fun MessagesScreen(...) {
    Box(modifier = Modifier.fillMaxSize()) {
        Scaffold(...) {
            MessageList()
        }

        if (selectedMessage != null) SelectedMessageOverlay()

        if (isShowingAttachments) AttachmentsPicker()

        val deleteAction = messageActions.firstOrNull { it is Delete }

        if (deleteAction != null) SimpleDialog()
    }
}
```



```
Box(modifier = Modifier.fillMaxSize()) {
    BottomDrawer(drawerContent = {
        AnimatedVisibility(visible = isShowingAttachments) {
            AttachmentsPicker(...)
        }
    }) {
        Scaffold(topBar = { ... },
            bottomBar = { ... }) {
            MessageList(itemContent = {

                })
        }
    }
}

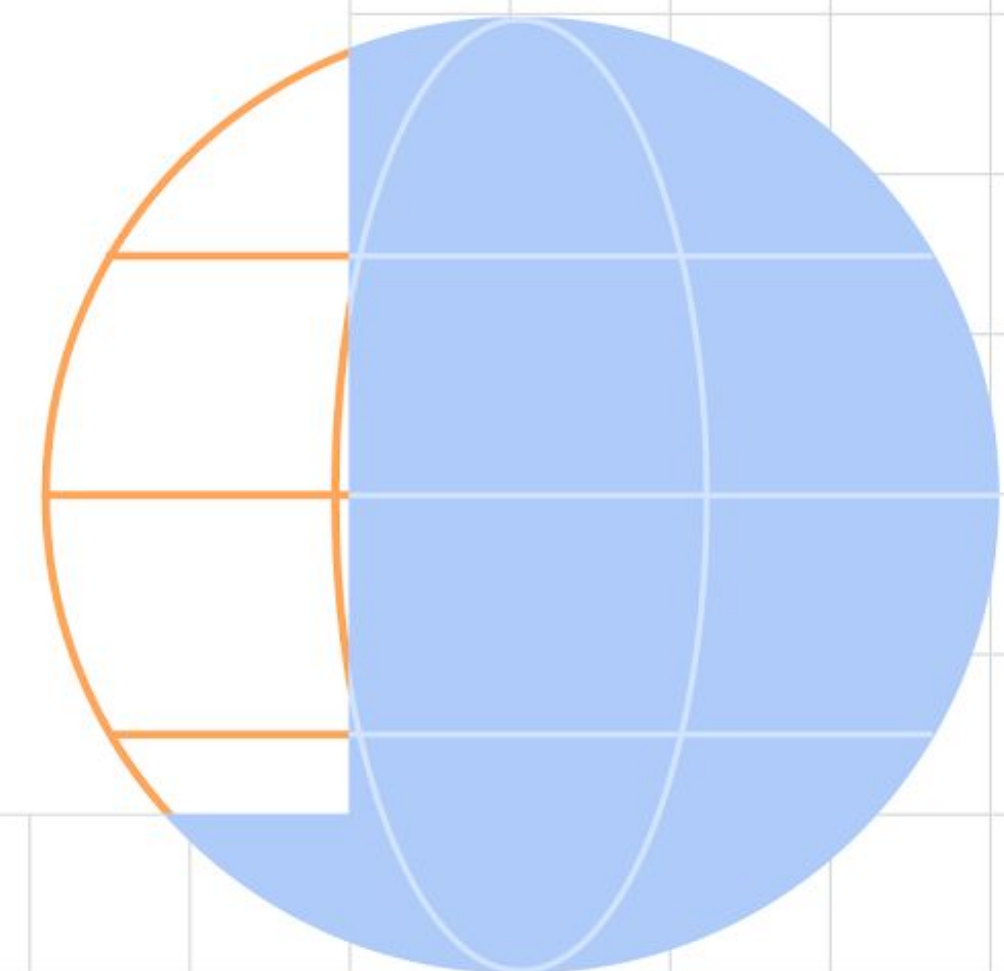
if (messageActions.any { it is Delete }) {

}

}
```



Customization mindset



Customization Mindset

Give out-of-the-box solutions, allow customization

- Provide a **modifier** in components people will use - [Jetpack Compose API Guidelines](#).
- Think about exposing **content** parameters where appropriate - [Slot APIs](#).
- Think about **theming** and building a design system.
- Provide customization for **state** and **action handlers**.

Component parameters

```
@Composable
public fun MessageComposer(
    viewModel: MessageComposerViewModel,

    onSendMessage: (Message) -> Unit = { viewModel.sendMessage(it) },
    onAttachmentsClick: () -> Unit = {},
    onValueChange: (String) -> Unit = { viewModel.setMessageInput(it) },
    onAttachmentRemoved: (Attachment) -> Unit = { viewModel.removeSelectedAttachment(it) },
    onCancelAction: () -> Unit = { viewModel.dismissMessageActions() },

    modifier: Modifier = Modifier,
    integrations: @Composable RowScope.() -> Unit = {
        DefaultComposerIntegrations(onAttachmentsClick)
    },
    label: @Composable () -> Unit = { DefaultComposerLabel() },
    input: @Composable RowScope.() -> Unit = {
        MessageInput(
            modifier = Modifier
                .fillMaxWidth()
                .weight(1f),
            label = label,
            value = viewModel.input,
            attachments = viewModel.selectedAttachments,
            activeAction = viewModel.activeAction,
            onValueChange = onValueChange,
            onAttachmentRemoved = onAttachmentRemoved
        )
    },
)
```

State

Action handlers

UI & Slot APIs

State

```
@Composable
```

```
public fun MessageComposer(  
    viewModel: MessageComposerViewModel,  
)
```

State

```
@Composable
```

```
public fun MessageComposer(  
    viewModel: MessageComposerViewModel,  
)
```

```
@Composable
```

```
public fun MessageComposer(  
    value: String,  
    attachments: List<Attachment>,  
    activeAction: MessageAction?,  
)
```

Action handlers

@Composable

```
public fun MessageComposer(  
    onSendMessage: (Message) -> Unit = { viewModel.sendMessage(it) },  
    onAttachmentsClick: () -> Unit = {},  
    onValueChange: (String) -> Unit = { viewModel.setMessageInput(it) },  
    onAttachmentRemoved: (Attachment) -> Unit = { viewModel.removeSelectedAttachment(it) },  
    onCancelAction: () -> Unit = { viewModel.dismissMessageActions() },  
)
```

Action handlers

@Composable

```
public fun MessageComposer(  
    onSendMessage: (Message) -> Unit = { viewModel.sendMessage(it) },  
    onAttachmentsClick: () -> Unit = {},  
    onValueChange: (String) -> Unit = { viewModel.setMessageInput(it) },  
    onAttachmentRemoved: (Attachment) -> Unit = { viewModel.removeSelectedAttachment(it) },  
    onCancelAction: () -> Unit = { viewModel.dismissMessageActions() },  
)
```

```
MessageComposer(  
    viewModel = composerViewModel,  
    onSendMessage = { message ->  
        message.text = message.text.replace("XML", "Jetpack Compose")  
        composerViewModel.sendMessage(message)  
    }  
)
```

Action handlers

```
@Composable
```

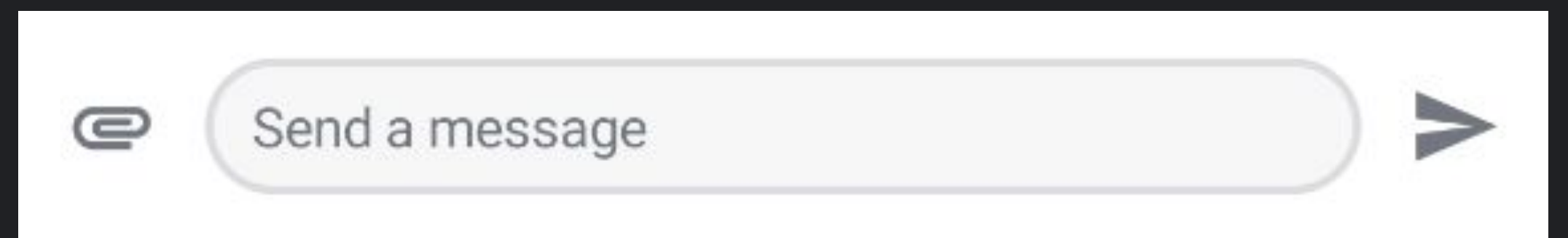
```
public fun MessageComposer(  
    onSendMessage: (Message) -> Unit = { viewModel.sendMessage(it) },  
    onAttachmentsClick: () -> Unit = {},  
    onValueChange: (String) -> Unit = { viewModel.setMessageInput(it) },  
    onAttachmentRemoved: (Attachment) -> Unit = { viewModel.removeSelectedAttachment(it) },  
    onCancelAction: () -> Unit = { viewModel.dismissMessageActions() },  
)
```

```
MessageComposer(  
    viewModel = composerViewModel,  
    onSendMessage = { message ->  
        message.text = message.text.replace("XML", "Jetpack Compose")  
        composerViewModel.sendMessage(message)  
    }  
)
```

Slot APIs

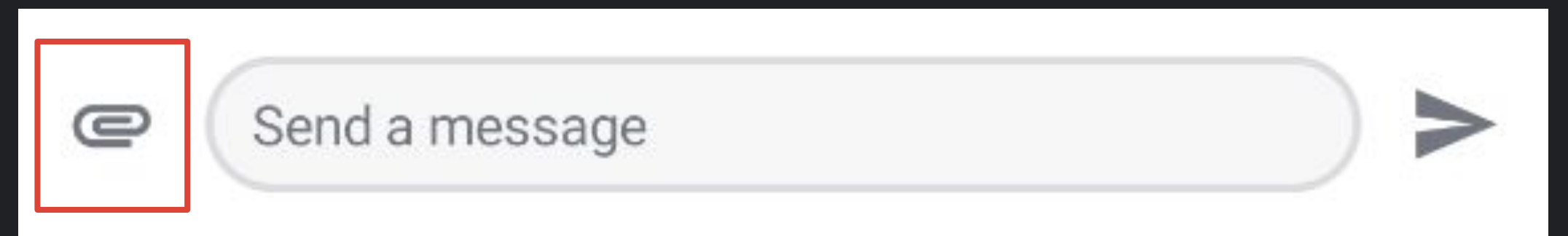
@Composable

```
public fun MessageComposer(  
    integrations: @Composable RowScope.() -> Unit = {  
        DefaultComposerIntegrations(onAttachmentsClick)  
    },  
    label: @Composable () -> Unit = { DefaultComposerLabel() },  
    input: @Composable RowScope.() -> Unit = {  
        MessageInput(  
            modifier = Modifier.fillMaxWidth().weight(1f),  
            label = label,  
            value = viewModel.input,  
            attachments = viewModel.selectedAttachments,  
            activeAction = viewModel.activeAction,  
            onValueChange = onValueChange,  
            onAttachmentRemoved = onAttachmentRemoved  
        )  
    },  
)
```



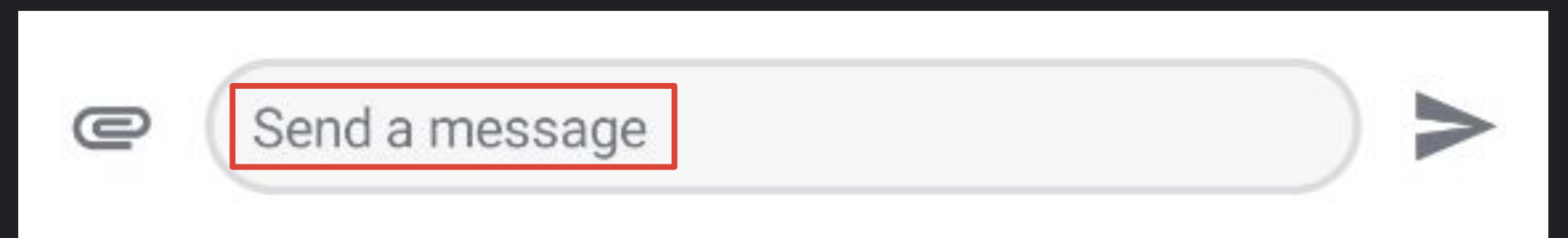
Slot APIs

```
@Composable
public fun MessageComposer(
    integrations: @Composable RowScope.() -> Unit = {
        DefaultComposerIntegrations(onAttachmentsClick)
    },
    label: @Composable () -> Unit = { DefaultComposerLabel() },
    input: @Composable RowScope.() -> Unit = {
        MessageInput(
            modifier = Modifier.fillMaxWidth().weight(1f),
            label = label,
            value = viewModel.input,
            attachments = viewModel.selectedAttachments,
            activeAction = viewModel.activeAction,
            onValueChange = onValueChange,
            onAttachmentRemoved = onAttachmentRemoved
        )
    },
)
```



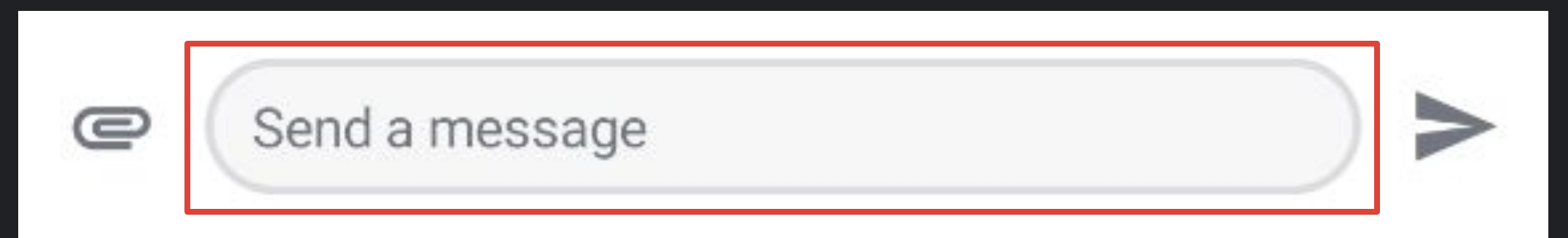
Slot APIs

```
@Composable
public fun MessageComposer(
    integrations: @Composable RowScope.() -> Unit = {
        DefaultComposerIntegrations(onAttachmentsClick)
    },
    label: @Composable () -> Unit = { DefaultComposerLabel() },
    input: @Composable RowScope.() -> Unit = {
        MessageInput(
            modifier = Modifier.fillMaxWidth().weight(1f),
            label = label,
            value = viewModel.input,
            attachments = viewModel.selectedAttachments,
            activeAction = viewModel.activeAction,
            onValueChange = onValueChange,
            onAttachmentRemoved = onAttachmentRemoved
        )
    },
)
)
```



Slot APIs

```
@Composable
public fun MessageComposer(
    integrations: @Composable RowScope.() -> Unit = {
        DefaultComposerIntegrations(onAttachmentsClick)
    },
    label: @Composable () -> Unit = { DefaultComposerLabel() },
    input: @Composable RowScope.() -> Unit = {
        MessageInput(
            modifier = Modifier.fillMaxWidth().weight(1f),
            label = label,
            value = viewModel.input,
            attachments = viewModel.selectedAttachments,
            activeAction = viewModel.activeAction,
            onValueChange = onValueChange,
            onAttachmentRemoved = onAttachmentRemoved
        )
    },
)
)
```



```

@Composable
public fun ChatTheme(
    isInDarkMode: Boolean = isSystemInDarkTheme(),
    colors: StreamColors = if (isInDarkMode) StreamColors.defaultDarkColors() else StreamColors.defaultColors(),
    typography: StreamTypography = StreamTypography.default,
    shapes: StreamShapes = StreamShapes.default,
    attachmentFactories: List<AttachmentFactory> = StreamAttachmentFactories.defaultFactories,
    reactionTypes: Map<String, Int> = defaultReactionTypes,
    content: @Composable () -> Unit,
) {
    CompositionLocalProvider(
        LocalColors provides colors,
        LocalTypography provides typography,
        LocalShapes provides shapes,
        LocalAttachmentFactories provides attachmentFactories,
        LocalReactionTypes provides reactionTypes,
    ) {
        content()
    }
}

```

@Composable

public fun ChatTheme(

 isInDarkMode: Boolean = isSystemInDarkTheme(),

 colors: StreamColors = if (isInDarkMode) StreamColors.defaultDarkColors() else StreamColors.defaultColors(),

 typography: StreamTypography = StreamTypography.default,

 shapes: StreamShapes = StreamShapes.default,

 attachmentFactories: List<AttachmentFactory> = StreamAttachmentFactories.defaultFactories,

 reactionTypes: Map<String, Int> = defaultReactionTypes,

 content: @Composable () -> Unit,

) {

 CompositionLocalProvider(

 LocalColors provides colors,

 LocalTypography provides typography,

 LocalShapes provides shapes,

 LocalAttachmentFactories provides attachmentFactories,

 LocalReactionTypes provides reactionTypes,

) {

 content()

 }



























}

```

@Composable
public fun ChatTheme(
    isInDarkMode: Boolean = isSystemInDarkTheme(),
    colors: StreamColors = if (isInDarkMode) StreamColors.defaultDarkColors() else StreamColors.defaultColors(),
    typography: StreamTypography = StreamTypography.default,
    shapes: StreamShapes = StreamShapes.default,
    attachmentFactories: List<AttachmentFactory> = StreamAttachmentFactories.defaultFactories,
    reactionTypes: Map<String, Int> = defaultReactionTypes,
    content: @Composable () -> Unit,
) {
    CompositionLocalProvider(
        LocalColors provides colors,
        LocalTypography provides typography,
        LocalShapes provides shapes,
        LocalAttachmentFactories provides attachmentFactories,
        LocalReactionTypes provides reactionTypes,
    ) {
        content()
    }
}

```

```
public data class StreamColors(  
    public val textHighEmphasis: Color,  
    public val textLowEmphasis: Color,  
    public val disabled: Color,  
    public val borders: Color,  
    public val inputBackground: Color,  
    public val appBackground: Color,  
    public val barsBackground: Color,  
    public val linkBackground: Color,  
    public val primaryAccent: Color,  
    public val errorAccent: Color,  
    public val infoAccent: Color,  
    public val highlight: Color,  
)
```

CS-light	CS-dark
 text-high-emphasis	 text-high-emphasis
 text-low-emphasis	 text-low-emphasis
 disabled	 disabled
 borders	 borders
 input-bg	 input-bg
 app-bg	 app-bg
 bars-bg	 bars-bg
 link-bg	 link-bg
 accent-primary	 accent-primary
 accent-error	 accent-error
 accent-info	 accent-info
 section-bg-gradient	 section-bg-gradient
 highlight	 highlight

Chat SDK Resources

- Compose Chat SDK on GitHub ✨
 - <https://github.com/GetStream/stream-chat-android/>
- Compose Tutorial
 - <https://getstream.io/chat/compose/tutorial/>
- Compose SDK docs
 - <https://getstream.io/chat/docs/sdk/android/compose/overview/>
- Twitter
 - [@getstream_io](https://twitter.com/getstream_io)

Jetpack Compose Resources

- Official Jetpack Compose API Guidelines
 - <https://github.com/androidx/androidx/blob/androidx-main/compose/docs/compose-api-guidelines.md>
- Google Compose pathway
 - <https://developer.android.com/courses/pathways/compose>
- GDG Osijek YT
 - www.youtube.com/channel/UCAXQuBMZ5B7f4thG5gO_3Rw



Questions? stream



Filip Babić
Jetpack Compose Lead

@filbabic



Márton Braun
Android Dev Advocate

@zsmb13