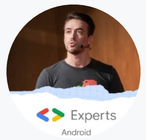
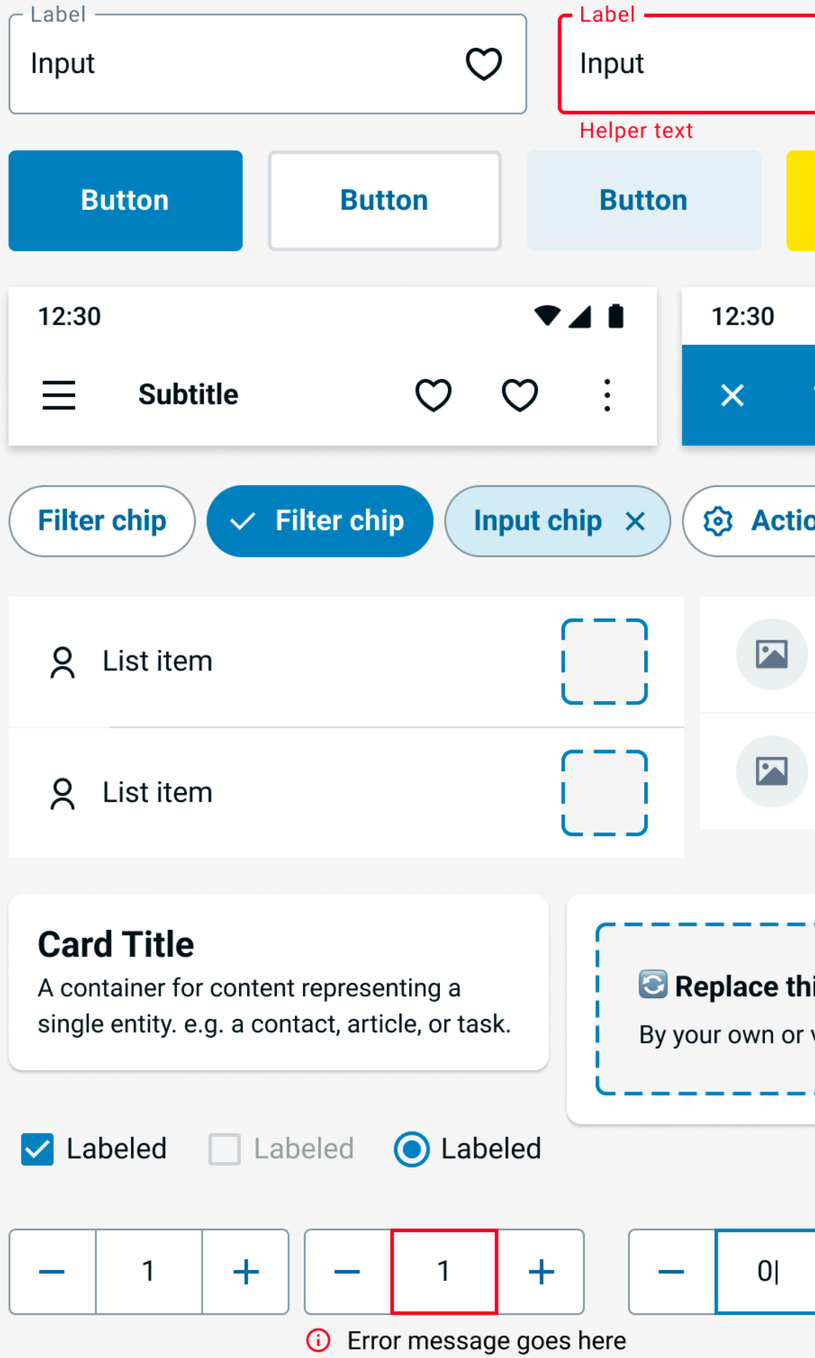
TAKE YOUR SHOT OF VITAMIN!

Cross platform Design System



@GerardPaligot



VITAMIN? CROSS DESIGN SYSTEM!

IMPLEMENTATIONS

2,026,487 Figma instances

519 Editors

4411 Viewers

1,501,741 Npm downloads

4 variants (CSS, Svelte, Vue, React)

208 stars

356,041 Maven downloads

2 variants (XML, Compose)

237 stars

2 variants (UIKit, SwiftUI)

28 stars

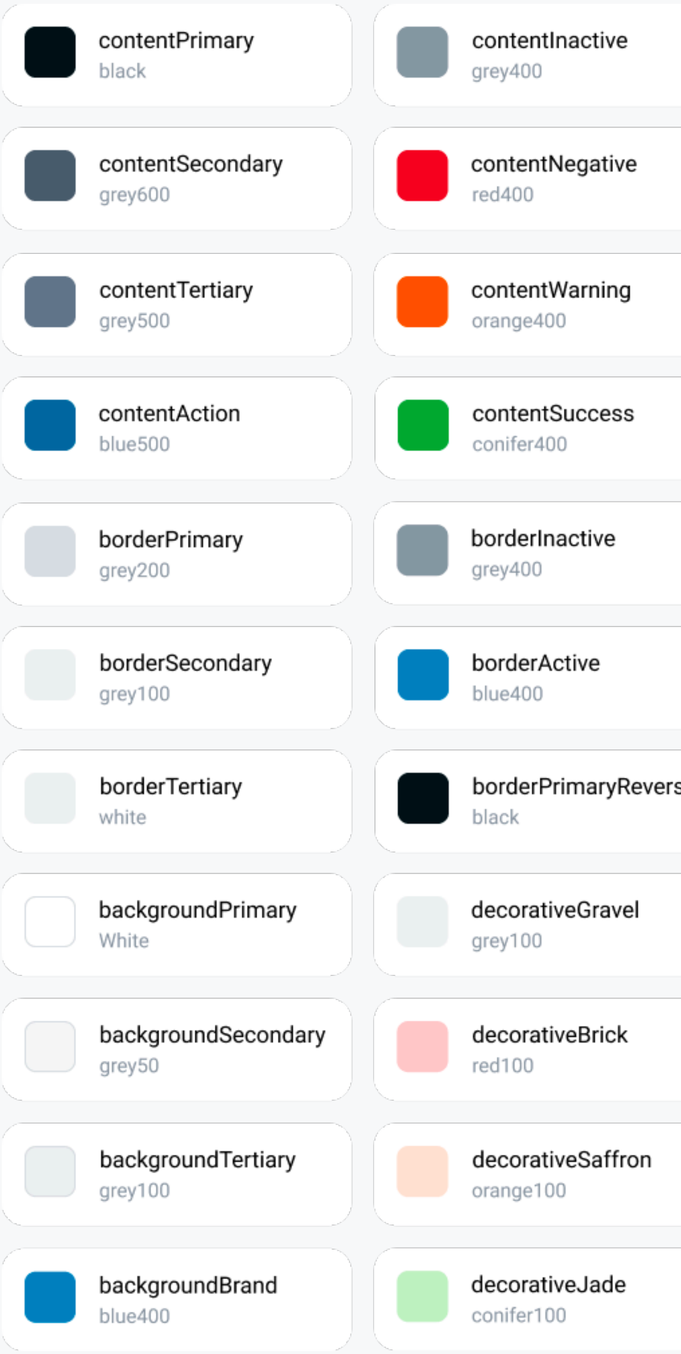
COMPOSE IMPLEMENTATION

COLOR PALETTE

```
object VitaminPalette {
    val vtmnPurple50 = Color(242, 237, 242)
    val vtmnPurple100 = Color(220, 207, 221)
    val vtmnPurple200 = Color(172, 141, 175)
    val vtmnPurple300 = Color(150, 111, 154)
    val vtmnPurple400 = Color(108, 78, 111)
    val vtmnPurple500 = Color(91, 65, 93)
    val vtmnPurple600 = Color(73, 53, 75)
    val vtmnPurple700 = Color(44, 32, 45)

    // same for blue, green, conifer, yellow,
    // orange, red and grey
}
```

SEMANTIC COLORS

```
val vtmnLightColors = VitaminColors(
    vtmnBackgroundPrimary = VitaminPalette.vtmnWhite,
    vtmnBackgroundSecondary = VitaminPalette.vtmnGrey50,
    vtmnBackgroundTertiary = VitaminPalette.vtmnGrey100,
    vtmnBackgroundBrandPrimary = VitaminPalette.vtmnBlue400,
    vtmnBackgroundBrandSecondary = VitaminPalette.vtmnBlue50,
    vtmnBackgroundAccent = VitaminPalette.vtmnYellow400,
    vtmnBackgroundDiscount = VitaminPalette.vtmnRed400,
    vtmnBackgroundPrimaryReversed = VitaminPalette.vtmnBlack,
    vtmnBackgroundBrandPrimaryReversed = VitaminPalette.vtmnWhite,

    // same for contents, borders and decoratives
)

// same for vtmnDarkColors
```



FONT FAMILY

```
private val robotoCondensed = FontFamily(
    Font(R.font.roboto_condensed_regular, FontWeight.Normal, FontStyle.Normal),
    Font(R.font.roboto_condensed_regularitalic, FontWeight.Normal, FontStyle.Italic),
    Font(R.font.roboto_condensed_bold, FontWeight.Bold, FontStyle.Normal),
    Font(R.font.roboto_condensed_bolditalic, FontWeight.Bold, FontStyle.Italic),
    Font(R.font.roboto_condensed_light, FontWeight.Light, FontStyle.Normal),
    Font(R.font.roboto_condensed_lightitalic, FontWeight.Light, FontStyle.Italic),
)

private val roboto = FontFamily(
    Font(R.font.roboto_regular, FontWeight.Normal, FontStyle.Normal),
    Font(R.font.roboto_italic, FontWeight.Normal, FontStyle.Italic),
    Font(R.font.roboto_bold, FontWeight.Bold, FontStyle.Normal),
    Font(R.font.roboto_bolditalic, FontWeight.Bold, FontStyle.Italic),
    Font(R.font.roboto_light, FontWeight.Light, FontStyle.Normal),
    Font(R.font.roboto_lightitalic, FontWeight.Light, FontStyle.Italic),
)
```

SEMANTIC TYPOGRAPHY

```
val vtmnTypography = VitaminTypography(
    h1 = TextStyle(
        fontFamily = robotoCondensed,
        fontSize = 42.sp,
        fontWeight = FontWeight.W700,
        lineHeight = 44.sp
    ),
    // h2, h3, h4, h5, h6, subtitle1, subtitle2
    text1 = TextStyle(
        fontFamily = roboto,
        fontSize = 17.sp,
        fontWeight = FontWeight.W400,
        lineHeight = 28.sp
    ),
    // text2, text3, text1 bold, text2 bold and text3 bold
    button = TextStyle(
        fontFamily = roboto,
        fontSize = 16.sp,
        fontWeight = FontWeight.W700,
        lineHeight = 16.sp
    ),
    // caption, overline
)
```

HEADLINE1

Headline2

Headline3

Headline4

Headline5

Headline6

Subtitle1

Subtitle2

Body/Regular/Text1

Body/Regular/Text2

Body/Regular/Text3

Body/Bold/Text1

Body/Bold/Text2

Body/Bold/Text3

Caption

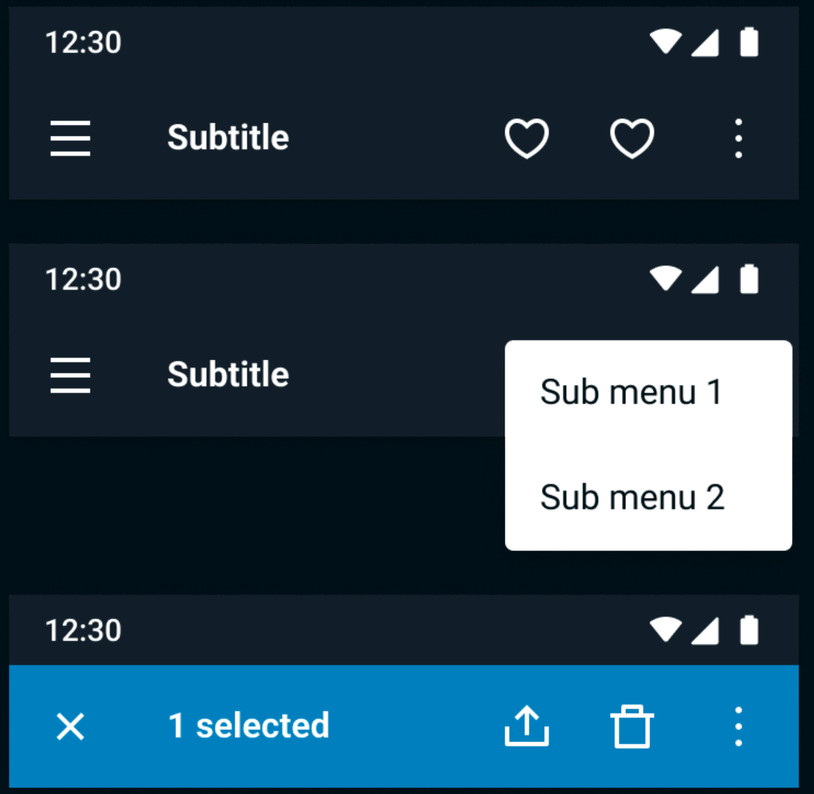
Overline

SEMANTIC SHAPES

```
private val radius100 = 4.dp
private val radius200 = 8.dp
private val radius300 = 12.dp
private val radius400 = 16.dp
private val radius500 = 20.dp
private val radius600 = 24.dp
private val radius700 = 32.dp
private val radius800 = 48.dp

val vtmnShapes = VitaminShapes(
    radius100 = RoundedCornerShape(radius100),
    radius200 = RoundedCornerShape(radius200),
    radius300 = RoundedCornerShape(radius300),
    radius400 = RoundedCornerShape(radius400),
    radius500 = RoundedCornerShape(radius500),
    radius600 = RoundedCornerShape(radius600),
    radius700 = RoundedCornerShape(radius700),
    radius800 = RoundedCornerShape(radius800)
)
```

TopAppBar

Vitamin Specification



MATERIAL 2

```
@Composable
fun TopAppBar(
    title: @Composable () → Unit,
    modifier: Modifier = Modifier,
    navigationIcon: @Composable (() → Unit)? = null,
    actions: @Composable RowScope.() → Unit = {},
    backgroundColor: Color = MaterialTheme.colors.primarySurface,
    contentColor: Color = contentColorFor(backgroundColor),
    elevation: Dp = AppBarDefaults.TopAppBarElevation
)
```

MATERIAL 2

```
@Composable
fun TopAppBar(
    title: @Composable () → Unit,
    modifier: Modifier = Modifier,
    navigationIcon: @Composable (() → Unit)? = null,
    actions: @Composable RowScope.() → Unit = {},
    backgroundColor: Color = MaterialTheme.colors.primarySurface,
    contentColor: Color = contentColorFor(backgroundColor),
    elevation: Dp = AppBarDefaults.TopAppBarElevation
)
```

MATERIAL 2

```
@Composable
fun TopAppBar(
    title: @Composable () → Unit,
    modifier: Modifier = Modifier,
    navigationIcon: @Composable (() → Unit)? = null,
    actions: @Composable RowScope.() → Unit = {},
    backgroundColor: Color = MaterialTheme.colors.primarySurface,
    contentColor: Color = contentColorFor(backgroundColor),
    elevation: Dp = AppBarDefaults.TopAppBarElevation
)
```

VITAMIN

```
object VitaminTopBars {
    private const val MAX_ACTIONS = 3

    @Composable
    fun Primary(
        title: String,
        modifier: Modifier = Modifier,
        maxActions: Int = MAX_ACTIONS,
        actions: List<ActionItem> = emptyList(),
        expandedMenu: MutableState<Boolean> = remember { mutableStateOf(false) },
        colors: TopBarColors = VitaminTopBarColors.primary(),
        onDismissOverflowMenu: (() → Unit)? = null,
        overflowIcon: (@Composable VitaminMenuIconButtons.() → Unit)? = null,
        navigationIcon: (@Composable VitaminNavigationIconButtons.() → Unit)? = null
    )
}
```

VITAMIN

```
object VitaminTopBars {
    private const val MAX_ACTIONS = 3

    @Composable
    fun Primary(
        title: String,
        modifier: Modifier = Modifier,
        maxActions: Int = MAX_ACTIONS,
        actions: List<ActionItem> = emptyList(),
        expandedMenu: MutableState<Boolean> = remember { mutableStateOf(false) },
        colors: TopBarColors = VitaminTopBarColors.primary(),
        onDismissOverflowMenu: (() → Unit)? = null,
        overflowIcon: (@Composable VitaminMenuIconButtons.() → Unit)? = null,
        navigationIcon: (@Composable VitaminNavigationIconButtons.() → Unit)? = null
    )
}
```

VITAMIN

```
object VitaminTopBars {
    private const val MAX_ACTIONS = 3

    @Composable
    fun Primary(
        title: String,
        modifier: Modifier = Modifier,
        maxActions: Int = MAX_ACTIONS,
        actions: List<ActionItem> = emptyList(),
        expandedMenu: MutableState<Boolean> = remember { mutableStateOf(false) },
        colors: TopBarColors = VitaminTopBarColors.primary(),
        onDismissOverflowMenu: (() → Unit)? = null,
        overflowIcon: (@Composable VitaminMenuIconButtons.() → Unit)? = null,
        navigationIcon: (@Composable VitaminNavigationIconButtons.() → Unit)? = null
    )
}
```

VITAMIN

```
open class ActionItem(
    val icon: Painter? = null,
    val contentDescription: String?,
    val content: @Composable () → Unit = {},
    val onClick: () → Unit,
)
```

VITAMIN

```
VitaminTopBars.Primary(
    title = "Title",
    actions = arrayListOf(
        ActionItem(
            icon = painterResource(R.drawable.ic_vtmn_android_line),
            contentDescription = "Android",
            onClick = { }
        ),
        ActionItem(
            contentDescription = null,
            content = { Text("Windows") },
            onClick = { }
        ),
        ActionItem(
            icon = painterResource(R.drawable.ic_vtmn_apple_line),
            contentDescription = "Apple",
            onClick = { }
        )
    )
)
```

MATERIAL 2

```
TopAppBar(
    title = { Text(text = "Title") },
    actions = {
        IconButton(onClick = { }) {
            Icon(
                painter = painterResource(R.drawable.ic_vtmn_android_line),
                contentDescription = "Android"
            )
        }
        TextButton(onClick = { }) {
            Text(text = "Windows")
        }
        IconButton(onClick = { }) {
            Icon(
                painter = painterResource(R.drawable.ic_vtmn_apple_line),
                contentDescription = "Apple"
            )
        }
    }
)
```

VITAMIN

```
object VitaminTopBars {
    private const val MAX_ACTIONS = 3

    @Composable
    fun Primary(
        title: String,
        modifier: Modifier = Modifier,
        maxActions: Int = MAX_ACTIONS,
        actions: List<ActionItem> = emptyList(),
        expandedMenu: MutableState<Boolean> = remember { mutableStateOf(false) },
        colors: TopBarColors = VitaminTopBarColors.primary(),
        onDismissOverflowMenu: (() → Unit)? = null,
        overflowIcon: (@Composable VitaminMenuIconButtons.() → Unit)? = null,
        navigationIcon: (@Composable VitaminNavigationIconButtons.() → Unit)? = null
    )
}
```

VITAMIN

```
object VitaminTopBarColors {
    @Composable
    fun primary(
        background: Color = VitaminTheme.colors.vtmnBackgroundPrimary,
        contentColor: Color = VitaminTheme.colors.vtmnContentPrimary,
        iconColor: Color = VitaminTheme.colors.vtmnContentPrimary
    ): TopBarColors = // ...

    @Composable
    fun contextual(
        background: Color = VitaminTheme.colors.vtmnContentActive,
        contentColor: Color = VitaminTheme.colors.vtmnContentPrimaryReversed,
        iconColor: Color = VitaminTheme.colors.vtmnContentPrimaryReversed
    ): TopBarColors = // ...
}
```

MATERIAL 2

```
// primary
TopAppBar(
    title = { Text(text = "Title") },
    backgroundColor = MaterialTheme.colors.surface,
    contentColor = MaterialTheme.colors.onSurface
)

// contextual
TopAppBar(
    title = { Text(text = "Title") },
    backgroundColor = MaterialTheme.colors.primarySurface,
    contentColor = MaterialTheme.colors.onPrimary
)
```

VITAMIN

```
object VitaminTopBars {
    private const val MAX_ACTIONS = 3

    @Composable
    fun Primary(
        title: String,
        modifier: Modifier = Modifier,
        maxActions: Int = MAX_ACTIONS,
        actions: List<ActionItem> = emptyList(),
        expandedMenu: MutableState<Boolean> = remember { mutableStateOf(false) },
        colors: TopBarColors = VitaminTopBarColors.primary(),
        onDismissOverflowMenu: (() → Unit)? = null,
        overflowIcon: (@Composable VitaminMenuIconButtons.() → Unit)? = null,
        navigationIcon: (@Composable VitaminNavigationIconButtons.() → Unit)? = null
    )
}
```

VITAMIN

```
object VitaminNavigationIconButtons {
    @Composable
    fun PreviousPage(
        // ...
    )

    @Composable
    fun Drawer(
        // ...
    )

    @Composable
    fun Context(
        // ...
    )

    @Composable
    fun Close(
        // ...
    )
}
```

VITAMIN

```
object VitaminNavigationIconButtons {
    @Composable
    fun PreviousPage(
        onClick: () → Unit,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        enabled: Boolean = true,
        interactionSource: MutableInteractionSource = remember { MutableInteractionSource() },
    ) {
        IconButtons(
            onClick = onClick,
            modifier = modifier,
            enabled = enabled,
            interactionSource = interactionSource,
            icon = {
                Icon(
                    painter = painterResource(R.drawable.ic_vtmn_chevron_left_line),
                    contentDescription = contentDescription
                )
            }
        )
    }
}
```

VITAMIN

```
VitaminTopBars.Primary(
    title = "Title",
    navigationIcon = {
        PreviousPage(
            onClick = { },
            contentDescription = "Come back to previous page"
        )
    }
)
```

MATERIAL 2

```
TopAppBar(
    title = { Text(text = "Title") },
    navigationIcon = {
        IconButton(
            onClick = { },
            content = {
                Icon(
                    painter = painterResource(R.drawable.ic_vtmn_chevron_left_line),
                    contentDescription = "Come back to previous page"
                )
            }
        )
    }
)
```

VITAMIN

```
object VitaminTopBars {
    private const val MAX_ACTIONS = 3

    @Composable
    fun Primary(
        title: String,
        modifier: Modifier = Modifier,
        maxActions: Int = MAX_ACTIONS,
        actions: List<ActionItem> = emptyList(),
        expandedMenu: MutableState<Boolean> = remember { mutableStateOf(false) },
        colors: TopBarColors = VitaminTopBarColors.primary(),
        onDismissOverflowMenu: (() → Unit)? = null,
        overflowIcon: (@Composable VitaminMenuIconButtons.() → Unit)? = null,
        navigationIcon: (@Composable VitaminNavigationIconButtons.() → Unit)? = null
    )
}
```

VITAMIN

```
object VitaminMenuIconButtons {
    @Composable
    fun More(
        onClick: () → Unit,
        contentDescription: String?,
        modifier: Modifier = Modifier,
        enabled: Boolean = true,
        interactionSource: MutableInteractionSource = remember { MutableInteractionSource() },
    ) {
        IconButtons(
            onClick = onClick,
            modifier = modifier,
            enabled = enabled,
            interactionSource = interactionSource,
            icon = {
                Icon(
                    painter = painterResource(R.drawable.ic_vtmn_more_2_line),
                    contentDescription = contentDescription
                )
            }
        )
    }
}
```

VITAMIN

```
@Composable
internal fun OverflowMenu(
    actions: List<ActionItem>,
    modifier: Modifier = Modifier,
    expanded: MutableState<Boolean> = remember { mutableStateOf(false) },
    interactionSource: MutableInteractionSource = remember { MutableInteractionSource() },
    onDismissRequest: (() → Unit)? = null,
    overflowIcon: @Composable VitaminMenuIconButtons.() → Unit
)
```

VITAMIN

```
@Composable
internal fun OverflowMenu(
    actions: List<ActionItem>,
    modifier: Modifier = Modifier,
    expanded: MutableState<Boolean> = remember { mutableStateOf(false) },
    interactionSource: MutableInteractionSource = remember { MutableInteractionSource() },
    onDismissRequest: (() -> Unit)? = null,
    overflowIcon: @Composable VitaminMenuIconButtons.() -> Unit
)
```

VITAMIN

```
object VitaminMenus {
    @Composable
    fun Dropdown(
        anchor: @Composable () → Unit,
        modifier: Modifier = Modifier,
        expanded: MutableState<Boolean> = remember { mutableStateOf(false) },
        interactionSource: MutableInteractionSource = remember { MutableInteractionSource() },
        onDismissRequest: () → Unit = {},
        children: @Composable VitaminMenuItems.() → Unit
    )
}
```

VITAMIN

```
object VitaminMenus {
    @Composable
    fun Dropdown(
        anchor: @Composable () → Unit,
        modifier: Modifier = Modifier,
        expanded: MutableState<Boolean> = remember { mutableStateOf(false) },
        interactionSource: MutableInteractionSource = remember { MutableInteractionSource() },
        onDismissRequest: () → Unit = {},
        children: @Composable VitaminMenuItems.() → Unit
    )
}
```

VITAMIN

```
val expanded = remember { mutableStateOf(false) }
VitaminTopBars.Primary(
    title = "Title",
    overflowIcon = {
        More(
            onClick = { expanded.value = true },
            contentDescription = "More",
        )
    },
    expandedMenu = expanded
)
```

MATERIAL 2

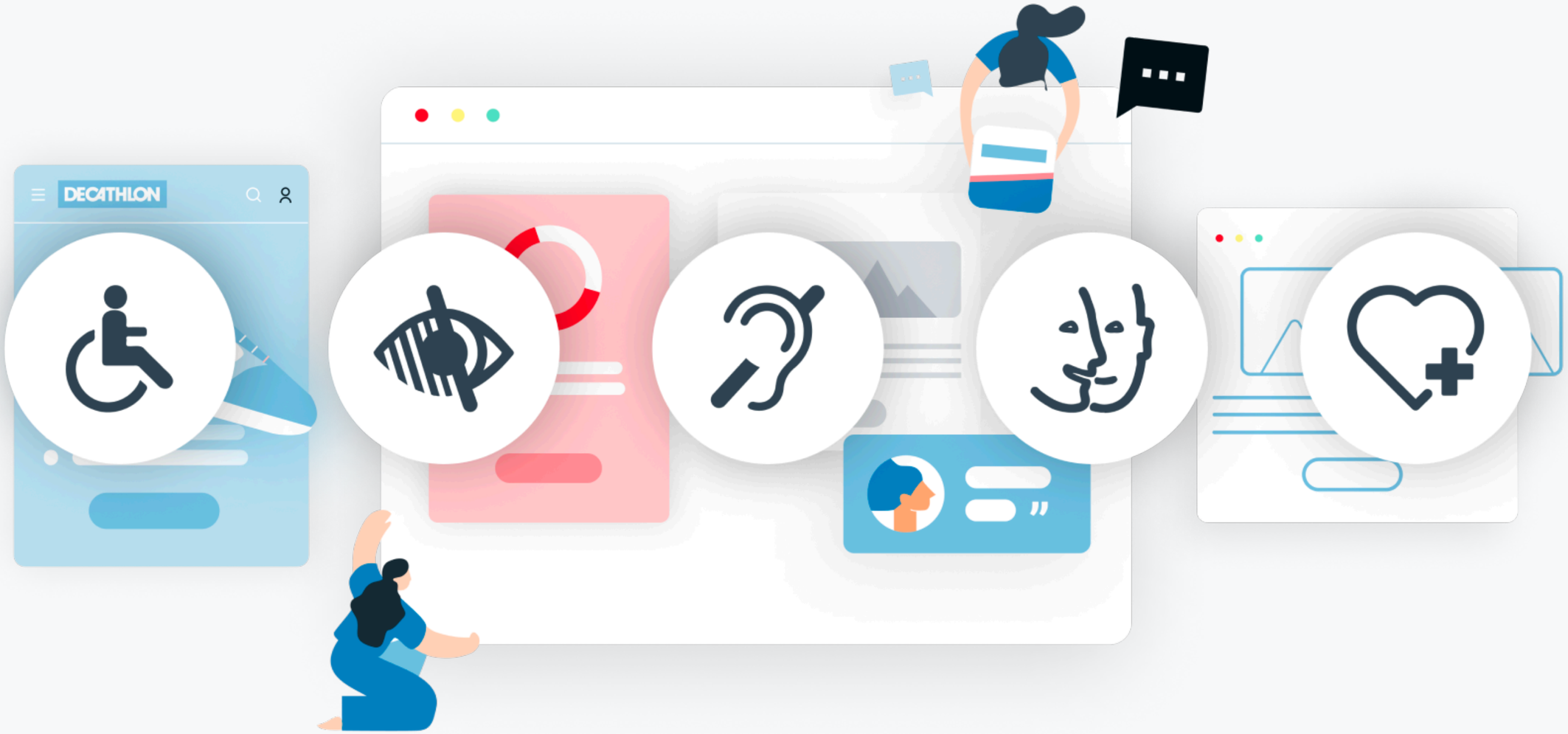
```
@Composable
fun OverflowMenu() {
    val expanded = remember { mutableStateOf(false) }
    Box {
        IconButton(onClick = { expanded.value = true }) {
            Icon(
                painter = painterResource(id = R.drawable.ic_vtmn_more_2_line),
                contentDescription = "More"
            )
        }
        DropdownMenu(
            expanded = expanded.value,
            onDismissRequest = { expanded.value = false },
            content = {
                DropdownMenuItem(
                    onClick = { },
                    content = {
                        Text(text = "Sub menu 1")
                    }
                )
            }
        )
    }
}
```

MATERIAL 2

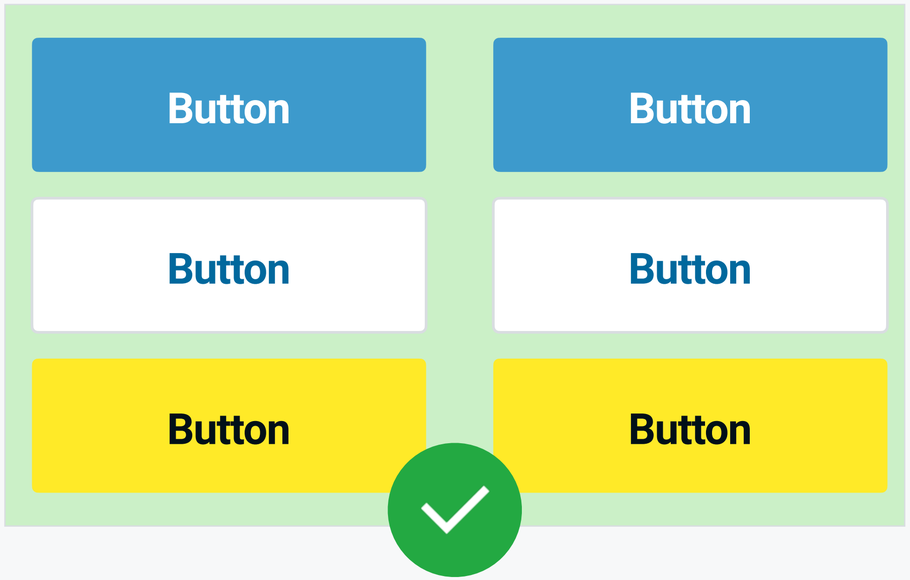
```
TopAppBar(
    title = { Text(text = "Title") },
    actions = {
        OverflowMenu()
    }
)
```

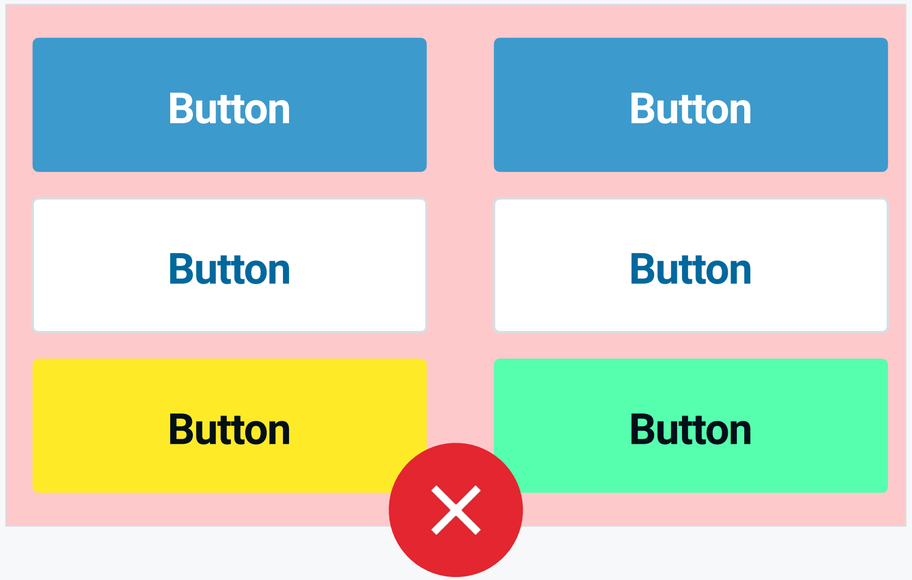
WHAT'S NEXT?!

VITAMIN 💙 ACCESSIBILITY

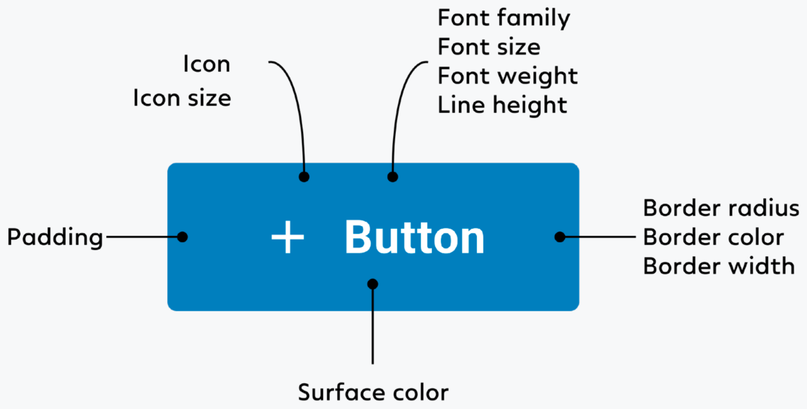


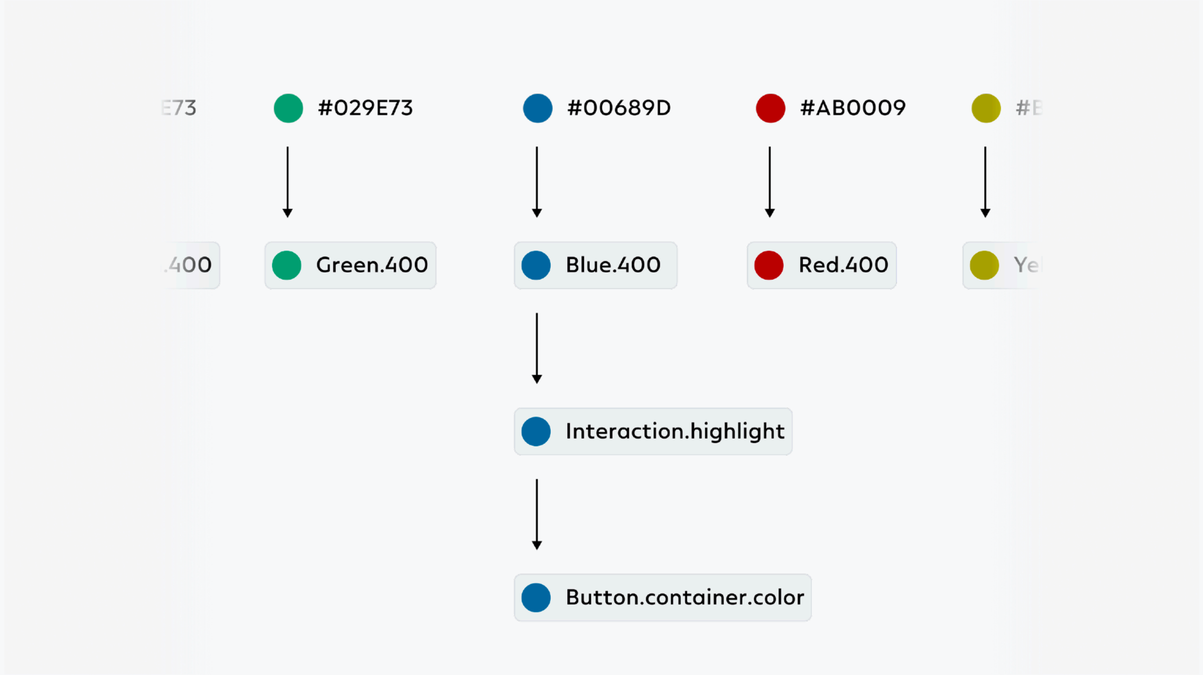
VITAMIN 💙 TESTING





VITAMIN 💙 TOKENISATION





VITAMIN 💙 MATERIAL

```
@Composable
fun VitaminTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () → Unit
) {
    val vtmnColors = if (darkTheme) vtmnDarkColors else vtmnLightColors
    ProvideVitaminResources(vtmnColors, vtmnTypography, vtmnShapes) {
        MaterialTheme(
            colors = mdColors(darkTheme, palette),
            typography = mdTypography,
            shapes = mdShapes,
            content = content
        )
    }
}
```

VITAMIN 💙 MATERIAL

```
fun mdColors(
    darkTheme: Boolean,
    vtmnColors: VitaminColors
) = Colors(
    primary = vtmnColors.vtmnBackgroundBrandPrimary,
    primaryVariant = vtmnColors.vtmnBackgroundBrandPrimary,
    onPrimary = vtmnColors.vtmnContentPrimaryReversed,
    secondary = vtmnColors.vtmnBackgroundAccent,
    secondaryVariant = vtmnColors.vtmnBackgroundAccent,
    onSecondary = vtmnColors.vtmnContentPrimaryReversed,
    background = vtmnColors.vtmnBackgroundSecondary,
    onBackground = vtmnColors.vtmnContentPrimary,
    surface = vtmnColors.vtmnBackgroundPrimary,
    onSurface = vtmnColors.vtmnContentPrimary,
    error = vtmnColors.vtmnContentNegative,
    onError = vtmnColors.vtmnContentPrimaryReversed,
    isLight = !darkTheme
)
```

REFERENCES

Vitamin Figma Community https://www.figma.com/@decathlon

Vitamin Compose https://github.com/Decathlon/vitamin-compose

Vitamin Android https://github.com/Decathlon/vitamin-android

Vitamin iOS https://github.com/Decathlon/vitamin-ios

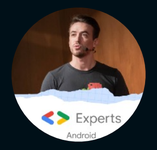
Vitamin Web https://github.com/Decathlon/vitamin-web

Vitamin Slack https://decathlon-design.slack.com/

Conferences4Hall https://github.com/GerardPaligot/conferences4hall

TAKE YOUR SHOT OF VITAMIN!

Thank you! Any questions?



@GerardPaligot