Bleed-AI: Flagship Design System & Architecture Specification

An ultra-premium, dark-themed native Android design system and component architecture engineered to rival Linear, Raycast, and Apple Human Interface guidelines. This system strictly eliminates muddy grays and emoji glyphs in favor of chromatic obsidian elevation, directional specular edge lighting, precision vector iconography, terminal-grade typography, and spring physics.

1. Multi-Tier Chromatic Token Architecture

```
Layer 0: True Zero Canvas
 └─── Canvas Pure Black:      #000000 (OLED emissive off)
 └─── Midnight Obsidian:      #08080C (Deep cold obsidian with 2% cobalt underto

Layer 1: Structural Surfaces & Elevation Cards
 └─── Surface Base:           #0F0F16 (Frosted base elevation)
 └─── Surface Raised:         #151520 (Interactive cards & grouped sections)
 └─── Surface Active:         #1D1D2C (Hover, pressed, and focus target)
 └─── Border Ambient:         #232334 (Ambient contour, 1px)
 └─── Border Specular:        LinearGradient(White 12% -> White 2%) (Directional

Layer 2: Overlays & Floating Islands
 └─── Glass Floating Island:  #12121C at 88% Alpha with 24dp blur
 └─── Modal Sheet:            #0E0E17 at 96% Alpha

Layer 3: Quantum Accents (Zero-Muted Pigments)
 └─── Radiant Amber:          #F59E0B -> #D97706 (Deep cognitive reasoning aura)
 └─── Quantum Cyan:           #06B6D4 -> #0284C7 (Web intelligence & external da
 └─── Emerald Pulse:          #10B981 (Active cryptographic handshake / 200 OK)
 └─── Crimson Flare:          #EF4444 (Destructive purge & fail-safe interrupts)

Layer 4: Editorial Typography (Alabaster Contrast)
 └─── Text Primary:           #F8FAFC (98% contrast, crisp alabaster)
 └─── Text Secondary:         #94A3B8 (Subdued metadata, 65% contrast)
 └─── Text Tertiary:          #64748B (Structural labels & timestamps, 45% contr
```

2. Micro-Interaction Physics & Haptic Choreography

Every interaction is driven by continuous spring curves rather than fixed cubic-bezier tweens:

Trigger / Gesture	Target Property	Spring Physics Parameters	Haptic Detent

Pill / Button Press	Scale `1.0f -> 0.965f`	`dampingRatio = 0.75f, stiffness = 600f`	`HapticFeedbackConstants.KEYBOARD_TAP`
Drawer Navigation Swipe	Horizontal Translation	`dampingRatio = 0.82f, stiffness = 300f`	`HapticFeedbackConstants.CLOCK_TICK`
Reasoning Accordion	Expand / Collapse Height	`dampingRatio = 0.88f, stiffness = 220f`	`HapticFeedbackConstants.VIRTUAL_KEY`
Streaming Chunk Arrival	Monospace Token Fade	Alpha interpolator `0f -> 1f` (100ms)	Silent (zero thermal overhead)
Copy / Action Commit	Icon Scale `1.25f -> 1.0f`	`dampingRatio = 0.60f, stiffness = 500f`	`HapticFeedbackConstants.CONFIRM`

3. Core Theme & Motion Foundation

`ui/theme/Color.kt`

```
package com.zai.chat.ui.theme

import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color

// Layer 0: Canvas
val CanvasPureBlack = Color(0xFF000000)
val MidnightObsidian = Color(0xFF08080C)

// Layer 1: Structural Surfaces
val SurfaceBase = Color(0xFF0F0F16)
val SurfaceRaised = Color(0xFF151520)
val SurfaceActive = Color(0xFF1D1D2C)
val BorderAmbient = Color(0xFF232334)

// Layer 2: Glassmorphism & Specular Highlights
val GlassIslandBackground = Color(0xE012121C)
val GlassIslandBorder = Color(0x33FFFFFF)

val SpecularEdgeTop = Color(0x1FFFFFFF)
val SpecularEdgeBottom = Color(0x05FFFFFF)
val SpecularGradientBrush = Brush.verticalGradient(
    colors = listOf(SpecularEdgeTop, SpecularEdgeBottom)
```

```
)

// Layer 3: Quantum Accents
val RadiantAmber = Color(0xFFF59E0B)
val RadiantAmberGlow = Color(0x33F59E0B)
val QuantumCyan = Color(0xFF06B6D4)
val QuantumCyanGlow = Color(0x3306B6D4)
val EmeraldPulse = Color(0xFF10B981)
val CrimsonFlare = Color(0xFFEF4444)

// Layer 4: Typography
val TextPrimary = Color(0xFFF8FAFC)
val TextSecondary = Color(0xFF94A3B8)
val TextTertiary = Color(0xFF64748B)

// Syntax Token Highlighting Palette (Tokyo Night Storm Variant)
val SyntaxKeyword = Color(0xFFBB9AF7)
val SyntaxFunction = Color(0xFF7AA2F7)
val SyntaxString = Color(0xFF9ECE6A)
val SyntaxType = Color(0xFF2AC3DE)
val SyntaxNumber = Color(0xFFFF9E64)
val SyntaxComment = Color(0xFF565F89)
val SyntaxBackground = Color(0xFF0A0A10)
```

ui/theme/Type.kt

```
package com.zai.chat.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

val BleedAiTypography = Typography(
    headlineLarge = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.Bold,
        fontSize = 28.sp,
        lineHeight = 34.sp,
        letterSpacing = (-0.5).sp,
        color = TextPrimary
    ),
    headlineMedium = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.SemiBold,
        fontSize = 20.sp,
        lineHeight = 26.sp,
        letterSpacing = (-0.3).sp,
        color = TextPrimary
    ),
    titleMedium = TextStyle(
        fontFamily = FontFamily.SansSerif,
```

```
        fontWeight = FontWeight.Medium,
        fontSize = 15.sp,
        lineHeight = 22.sp,
        letterSpacing = (-0.1).sp,
        color = TextPrimary
    ),
    bodyLarge = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 25.6.sp, // 1.6x editorial line-height
        letterSpacing = 0.1.sp,
        color = TextPrimary
    ),
    bodyMedium = TextStyle(
        fontFamily = FontFamily.SansSerif,
        fontWeight = FontWeight.Normal,
        fontSize = 14.sp,
        lineHeight = 21.sp,
        letterSpacing = 0.05.sp,
        color = TextSecondary
    ),
    labelSmall = TextStyle(
        fontFamily = FontFamily.Monospace,
        fontWeight = FontWeight.Medium,
        fontSize = 10.5.sp,
        lineHeight = 14.sp,
        letterSpacing = 0.8.sp,
        color = TextTertiary
    )
)
```

ui/theme/Motion.kt

```
package com.zai.chat.ui.theme

import androidx.compose.animation.core.Spring
import androidx.compose.animation.core.SpringSpec
import androidx.compose.animation.core.spring

object BleedMotion {
    val PressScaleSpring: SpringSpec<Float> = spring(
        dampingRatio = 0.75f,
        stiffness = 600f
    )

    val FluidNavigationSpring: SpringSpec<Float> = spring(
        dampingRatio = 0.82f,
        stiffness = 300f
    )

    val PanelExpandSpring: SpringSpec<Float> = spring(
        dampingRatio = 0.88f,
```

```
        stiffness = 220f
    )

    val CheckmarkPopSpring: SpringSpec<Float> = spring(
        dampingRatio = Spring.DampingRatioMediumBouncy,
        stiffness = 500f
    )
}
```

ui/components/SpecularModifier.kt

Zero-dependency Compose modifier providing tactile scale-down press physics and directional specular top borders.

```
package com.zai.chat.ui.components

import android.view.HapticFeedbackConstants
import androidx.compose.animation.core.animateFloatAsState
import androidx.compose.foundation.border
import androidx.compose.foundation.gestures.awaitFirstDown
import androidx.compose.foundation.gestures.waitForUpOrCancellation
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.composed
import androidx.compose.ui.graphics.Shape
import androidx.compose.ui.graphics.graphicsLayer
import androidx.compose.ui.input.pointer.pointerInput
import androidx.compose.ui.platform.LocalView
import androidx.compose.ui.unit.Dp
import androidx.compose.ui.unit.dp
import com.zai.chat.ui.theme.BleedMotion
import com.zai.chat.ui.theme.SpecularGradientBrush

fun Modifier.tactilePress(
    targetScale: Float = 0.965f,
    onClick: (() -> Unit)? = null
): Modifier = composed {
    val view = LocalView.current
    var isPressed by remember { mutableStateOf(false) }
    val scale by animateFloatAsState(
        targetValue = if (isPressed) targetScale else 1.0f,
        animationSpec = BleedMotion.PressScaleSpring,
        label = "tactile_scale"
    )

    this
        .graphicsLayer {
            scaleX = scale
            scaleY = scale
```

```
        }
        .pointerInput(Unit) {
            while (true) {
                awaitPointerEventScope {
                    awaitFirstDown(requireUnconsumed = false)
                    isPressed = true
                    view.performHapticFeedback(HapticFeedbackConstants.KEYBOARD_

                    val upOrCancel = waitForUpOrCancellation()
                    isPressed = false
                    if (upOrCancel != null) {
                        onClick?.invoke()
                    }
                }
            }
        }
}

fun Modifier.specularBorder(
    shape: Shape = RoundedCornerShape(16.dp),
    strokeWidth: Dp = 1.dp
): Modifier = this.border(
    width = strokeWidth,
    brush = SpecularGradientBrush,
    shape = shape
)
```

4. Studio-Grade Markdown & Syntax Highlighting Engine

`ui/components/SyntaxHighlighter.kt`

High-performance deterministic tokenizer supporting Kotlin, Python, JS/TS, SQL, JSON, Rust, and Shell without runtime webview bridges.

```
package com.zai.chat.ui.components

import androidx.compose.ui.text.AnnotatedString
import androidx.compose.ui.text.SpanStyle
import androidx.compose.ui.text.buildAnnotatedString
import com.zai.chat.ui.theme.SyntaxComment
import com.zai.chat.ui.theme.SyntaxFunction
import com.zai.chat.ui.theme.SyntaxKeyword
import com.zai.chat.ui.theme.SyntaxNumber
import com.zai.chat.ui.theme.SyntaxString
import com.zai.chat.ui.theme.SyntaxType
import com.zai.chat.ui.theme.TextPrimary
import java.util.regex.Pattern

object SyntaxHighlighter {

    private val KEYWORDS = Pattern.compile(
        "\\b(val|var|fun|class|interface|object|return|if|else|when|for|while|im
```

```
        "def|async|await|const|let|struct|impl|fn|select|from|where|match|pub|ty
    )
    private val STRINGS = Pattern.compile("\"(\\\\.|[^\"\\\\])*\"|'(\\\\.|[^'\\\
    private val NUMBERS = Pattern.compile("\\b(0x[0-9a-fA-F]+|\\d+(\\.\\d+)?([eE
    private val COMMENTS = Pattern.compile("//.*|/\\*[\\s\\S]*?\\*/|#.*")
    private val FUNCTIONS = Pattern.compile("\\b([a-zA-Z_][a-zA-Z0-9_]*)(?=\\()"
    private val TYPES = Pattern.compile("\\b([A-Z][a-zA-Z0-9_]*)\\b")

    fun highlight(code: String, language: String): AnnotatedString {
        return buildAnnotatedString {
            append(code)
            addStyle(SpanStyle(color = TextPrimary), 0, code.length)

            // 1. Strings
            applyPattern(code, STRINGS, SpanStyle(color = SyntaxString))

            // 2. Numbers
            applyPattern(code, NUMBERS, SpanStyle(color = SyntaxNumber))

            // 3. Keywords
            applyPattern(code, KEYWORDS, SpanStyle(color = SyntaxKeyword))

            // 4. Function invocations
            applyPattern(code, FUNCTIONS, SpanStyle(color = SyntaxFunction))

            // 5. Types and Classes
            applyPattern(code, TYPES, SpanStyle(color = SyntaxType))

            // 6. Comments (Overrides prior styling)
            applyPattern(code, COMMENTS, SpanStyle(color = SyntaxComment))
        }
    }

    private fun AnnotatedString.Builder.applyPattern(
        text: String,
        pattern: Pattern,
        style: SpanStyle
    ) {
        val matcher = pattern.matcher(text)
        while (matcher.find()) {
            addStyle(style, matcher.start(), matcher.end())
        }
    }
}
```

ui/components/CodeBlockView.kt

Terminal-grade specular code frame with line wrapping, active line numbering, and vector animated copy validation.

```
package com.zai.chat.ui.components

import android.content.ClipData
```

```
import android.content.ClipboardManager
import android.content.Context
import android.view.HapticFeedbackConstants
import androidx.compose.animation.AnimatedContent
import androidx.compose.animation.core.animateFloatAsState
import androidx.compose.foundation.background
import androidx.compose.foundation.horizontalScroll
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.rounded.Check
import androidx.compose.material.icons.rounded.ContentCopy
import androidx.compose.material.icons.rounded.FormatLineSpacing
import androidx.compose.material.icons.rounded.Terminal
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.rememberCoroutineScope
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalView
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zai.chat.ui.theme.BleedMotion
import com.zai.chat.ui.theme.BorderAmbient
import com.zai.chat.ui.theme.EmeraldPulse
import com.zai.chat.ui.theme.SurfaceActive
import com.zai.chat.ui.theme.SurfaceRaised
import com.zai.chat.ui.theme.SyntaxBackground
import com.zai.chat.ui.theme.TextSecondary
import com.zai.chat.ui.theme.TextTertiary
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch

@Composable
fun CodeBlockView(
    code: String,
```

```
    language: String = "kotlin",
    modifier: Modifier = Modifier
) {
    val context = LocalContext.current
    val view = LocalView.current
    val scope = rememberCoroutineScope()

    var isCopied by remember { mutableStateOf(false) }
    var wrapLines by remember { mutableStateOf(false) }
    var showLineNumbers by remember { mutableStateOf(true) }

    val highlightedText = remember(code, language) {
        SyntaxHighlighter.highlight(code, language)
    }

    val lines = remember(code) { code.lines() }
    val shape = RoundedCornerShape(12.dp)

    Column(
        modifier = modifier
            .fillMaxWidth()
            .clip(shape)
            .background(SyntaxBackground)
            .specularBorder(shape)
    ) {
        // Specular Top Console Bar
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .background(SurfaceRaised)
                .padding(horizontal = 12.dp, vertical = 6.dp),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically
        ) {
            Row(verticalAlignment = Alignment.CenterVertically) {
                Icon(
                    imageVector = Icons.Rounded.Terminal,
                    contentDescription = null,
                    tint = TextSecondary,
                    modifier = Modifier.size(15.dp)
                )
                Spacer(modifier = Modifier.width(6.dp))
                Text(
                    text = language.uppercase(),
                    style = MaterialTheme.typography.labelSmall.copy(color = Tex
                )
            }

            Row(verticalAlignment = Alignment.CenterVertically) {
                // Toggle Line Numbers
                IconButton(
                    onClick = {
                        view.performHapticFeedback(HapticFeedbackConstants.KEYBO
                        showLineNumbers = !showLineNumbers
                    },
                    modifier = Modifier.size(28.dp)
```

```
        ) {
            Icon(
                imageVector = Icons.Rounded.FormatLineSpacing,
                contentDescription = "Line numbers",
                tint = if (showLineNumbers) EmeraldPulse else TextTertia
                modifier = Modifier.size(14.dp)
            )
        }

        // Copy Action Button
        IconButton(
            onClick = {
                val cm = context.getSystemService(Context.CLIPBOARD_SERV
                cm.setPrimaryClip(ClipData.newPlainText("Source Code", c
                view.performHapticFeedback(HapticFeedbackConstants.CONFI
                isCopied = true
                scope.launch {
                    delay(2200)
                    isCopied = false
                }
            },
            modifier = Modifier.size(28.dp)
        ) {
            AnimatedContent(
                targetState = isCopied,
                label = "copy_icon_transition"
            ) { copied ->
                if (copied) {
                    Icon(
                        imageVector = Icons.Rounded.Check,
                        contentDescription = "Copied",
                        tint = EmeraldPulse,
                        modifier = Modifier.size(14.dp)
                    )
                } else {
                    Icon(
                        imageVector = Icons.Rounded.ContentCopy,
                        contentDescription = "Copy code",
                        tint = TextSecondary,
                        modifier = Modifier.size(14.dp)
                    )
                }
            }
        }
    }
}

// Code Editor Body
val horizontalScrollState = rememberScrollState()
Row(
    modifier = Modifier
        .fillMaxWidth()
        .padding(vertical = 12.dp)
) {
    if (showLineNumbers) {
        Column(
```

```
                    modifier = Modifier
                        .padding(start = 12.dp, end = 8.dp),
                    horizontalAlignment = Alignment.End
                ) {
                    lines.indices.forEach { index ->
                        Text(
                            text = "${index + 1}",
                            style = MaterialTheme.typography.labelSmall.copy(
                                color = TextTertiary.copy(alpha = 0.5f),
                                fontSize = 12.sp,
                                lineHeight = 18.sp
                            )
                        )
                    }
                }
            }

            Box(
                modifier = Modifier
                    .weight(1f)
                    .then(if (wrapLines) Modifier.padding(end = 12.dp) else Modi
            ) {
                Text(
                    text = highlightedText,
                    style = MaterialTheme.typography.bodyMedium.copy(
                        fontFamily = FontFamily.Monospace,
                        fontSize = 12.5.sp,
                        lineHeight = 18.sp
                    )
                )
            }
        }
    }
}
```

5. Quantum Reasoning Accordion Component

`ui/components/ThinkingBlock.kt`

Dynamic neural thought inspector with continuous beacon pulse, stopwatch tracker, and zero emoji glyphs.

```
package com.zai.chat.ui.components

import android.view.HapticFeedbackConstants
import androidx.compose.animation.AnimatedVisibility
import androidx.compose.animation.animateContentSize
import androidx.compose.animation.core.FastOutSlowInEasing
import androidx.compose.animation.core.LinearEasing
import androidx.compose.animation.core.RepeatMode
import androidx.compose.animation.core.animateFloat
import androidx.compose.animation.core.infiniteRepeatable
```

```
import androidx.compose.animation.core.rememberInfiniteTransition
import androidx.compose.animation.core.tween
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.heightIn
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.rounded.KeyboardArrowDown
import androidx.compose.material.icons.rounded.KeyboardArrowUp
import androidx.compose.material.icons.rounded.Memory
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableFloatStateOf
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.geometry.Offset
import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.drawscope.Stroke
import androidx.compose.ui.platform.LocalView
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zai.chat.ui.theme.BleedMotion
import com.zai.chat.ui.theme.BorderAmbient
import com.zai.chat.ui.theme.RadiantAmber
import com.zai.chat.ui.theme.RadiantAmberGlow
import com.zai.chat.ui.theme.SurfaceBase
import com.zai.chat.ui.theme.SurfaceRaised
import com.zai.chat.ui.theme.TextPrimary
import com.zai.chat.ui.theme.TextSecondary
import com.zai.chat.ui.theme.TextTertiary
import kotlinx.coroutines.delay
```

```
@Composable
fun ThinkingBlock(
    reasoningText: String,
    isStreaming: Boolean,
    modifier: Modifier = Modifier
) {
    var isExpanded by remember { mutableStateOf(isStreaming) }
    val view = LocalView.current
    var elapsedTimeSeconds by remember { mutableFloatStateOf(0.0f) }

    LaunchedEffect(isStreaming) {
        if (isStreaming) {
            isExpanded = true
            val startTime = System.currentTimeMillis()
            while (isStreaming) {
                elapsedTimeSeconds = (System.currentTimeMillis() - startTime) / 1
                delay(100)
            }
        }
    }

    val shape = RoundedCornerShape(12.dp)

    Column(
        modifier = modifier
            .fillMaxWidth()
            .clip(shape)
            .background(SurfaceBase)
            .specularBorder(shape)
            .animateContentSize(animationSpec = BleedMotion.PanelExpandSpring)
    ) {
        // Quantum Header Bar
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .clickable {
                    view.performHapticFeedback(HapticFeedbackConstants.VIRTUAL_KE
                    isExpanded = !isExpanded
                }
                .padding(horizontal = 14.dp, vertical = 10.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {
            // Live Status Beacon
            if (isStreaming) {
                PulsingCognitiveBeacon()
            } else {
                Box(
                    modifier = Modifier
                        .size(6.dp)
                        .clip(CircleShape)
                        .background(RadiantAmber)
                )
            }

            Spacer(modifier = Modifier.width(10.dp))
```

```
        Icon(
            imageVector = Icons.Rounded.Memory,
            contentDescription = null,
            tint = RadiantAmber,
            modifier = Modifier.size(16.dp)
        )

        Spacer(modifier = Modifier.width(8.dp))

        Text(
            text = if (isStreaming) "COGNITIVE SYNTHESIS IN PROGRESS" else "I
            style = MaterialTheme.typography.labelSmall.copy(
                fontWeight = FontWeight.Bold,
                color = if (isStreaming) RadiantAmber else TextSecondary
            )
        )

        Spacer(modifier = Modifier.weight(1f))

        // Duration Indicator
        Text(
            text = "${"%.1f".format(elapsedTimeSeconds)}s",
            style = MaterialTheme.typography.labelSmall.copy(color = TextTer
        )

        Spacer(modifier = Modifier.width(6.dp))

        Icon(
            imageVector = if (isExpanded) Icons.Rounded.KeyboardArrowUp else
            contentDescription = null,
            tint = TextTertiary,
            modifier = Modifier.size(18.dp)
        )
    }

    // Expandable Monospace Thought Stream
    AnimatedVisibility(visible = isExpanded) {
        Column(
            modifier = Modifier
                .fillMaxWidth()
                .heightIn(max = 260.dp)
                .background(SurfaceRaised.copy(alpha = 0.5f))
                .padding(horizontal = 14.dp, vertical = 12.dp)
        ) {
            val scrollState = rememberScrollState()
            Text(
                text = reasoningText.ifBlank { "Initializing neural weights
                style = MaterialTheme.typography.bodyMedium.copy(
                    fontFamily = FontFamily.Monospace,
                    fontSize = 12.5.sp,
                    lineHeight = 19.sp,
                    color = TextSecondary
                ),
                modifier = Modifier.verticalScroll(scrollState)
            )
        }
```

```
        }
    }
}

@Composable
fun PulsingCognitiveBeacon() {
    val transition = rememberInfiniteTransition(label = "beacon_pulse")
    val scale by transition.animateFloat(
        initialValue = 0.8f,
        targetValue = 1.4f,
        animationSpec = infiniteRepeatable(
            animation = tween(900, easing = FastOutSlowInEasing),
            repeatMode = RepeatMode.Reverse
        ),
        label = "scale"
    )
    val alpha by transition.animateFloat(
        initialValue = 0.3f,
        targetValue = 1.0f,
        animationSpec = infiniteRepeatable(
            animation = tween(900, easing = FastOutSlowInEasing),
            repeatMode = RepeatMode.Reverse
        ),
        label = "alpha"
    )

    Box(contentAlignment = Alignment.Center, modifier = Modifier.size(10.dp)) {
        Box(
            modifier = Modifier
                .size(10.dp * scale)
                .clip(CircleShape)
                .background(RadiantAmber.copy(alpha = alpha * 0.3f))
        )
        Box(
            modifier = Modifier
                .size(6.dp)
                .clip(CircleShape)
                .background(RadiantAmber)
        )
    }
}
```

6. Asymmetric Message Architecture & Floating Action Island

`ui/components/MessageBubble.kt`

Editorial document layout for model responses paired with organic asymmetric user bubbles and a glassmorphic floating action island.

```
package com.zai.chat.ui.components

import android.content.ClipData
```

```
import android.content.ClipboardManager
import android.content.Context
import android.content.Intent
import android.view.HapticFeedbackConstants
import androidx.compose.animation.AnimatedVisibility
import androidx.compose.animation.fadeIn
import androidx.compose.animation.fadeOut
import androidx.compose.animation.scaleIn
import androidx.compose.animation.scaleOut
import androidx.compose.foundation.ExperimentalFoundationApi
import androidx.compose.foundation.background
import androidx.compose.foundation.combinedClickable
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.layout.widthIn
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.rounded.ContentCopy
import androidx.compose.material.icons.rounded.Edit
import androidx.compose.material.icons.rounded.ForkRight
import androidx.compose.material.icons.rounded.Refresh
import androidx.compose.material.icons.rounded.Share
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalView
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import com.zai.chat.data.model.Message
import com.zai.chat.data.model.MessageRole
import com.zai.chat.ui.theme.BorderAmbient
import com.zai.chat.ui.theme.GlassIslandBackground
import com.zai.chat.ui.theme.QuantumCyan
import com.zai.chat.ui.theme.SurfaceActive
```

```
import com.zai.chat.ui.theme.SurfaceRaised
import com.zai.chat.ui.theme.TextPrimary
import com.zai.chat.ui.theme.TextSecondary
import com.zai.chat.ui.theme.TextTertiary

@OptIn(ExperimentalFoundationApi::class)
@Composable
fun MessageBubble(
    message: Message,
    isStreaming: Boolean,
    onEditAndResend: ((Message) -> Unit)? = null,
    onRegenerate: (() -> Unit)? = null,
    modifier: Modifier = Modifier
) {
    val context = LocalContext.current
    val view = LocalView.current
    var showActionIsland by remember { mutableStateOf(false) }

    val isUser = message.role == MessageRole.USER
    val userShape = RoundedCornerShape(topStart = 18.dp, topEnd = 4.dp, bottomSt

    Column(
        modifier = modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 6.dp),
        horizontalAlignment = if (isUser) Alignment.End else Alignment.Start
    ) {
        if (isUser) {
            // USER: Asymmetric Organic Squircle
            Surface(
                shape = userShape,
                color = SurfaceRaised,
                modifier = Modifier
                    .widthIn(max = 310.dp)
                    .specularBorder(userShape)
                    .combinedClickable(
                        onClick = {
                            view.performHapticFeedback(HapticFeedbackConstants.K
                            showActionIsland = !showActionIsland
                        },
                        onLongClick = {
                            view.performHapticFeedback(HapticFeedbackConstants.L
                            showActionIsland = true
                        }
                    )
            ) {
                Text(
                    text = message.content,
                    style = MaterialTheme.typography.bodyLarge.copy(color = Text
                    modifier = Modifier.padding(horizontal = 16.dp, vertical = 1
                )
            }
        } else {
            // ASSISTANT: Edge-to-Edge Editorial Typography Layout
            Column(
                modifier = Modifier
```

```
            .fillMaxWidth()
            .combinedClickable(
                onClick = { showActionIsland = !showActionIsland },
                onLongClick = {
                    view.performHapticFeedback(HapticFeedbackConstants.L
                    showActionIsland = true
                }
            )
    ) {
        // Identity Bar
        Row(
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier.padding(bottom = 8.dp)
        ) {
            Box(
                modifier = Modifier
                    .size(6.dp)
                    .clip(CircleShape)
                    .background(QuantumCyan)
            )
            Spacer(modifier = Modifier.width(8.dp))
            Text(
                text = "INTELLIGENCE CORE",
                style = MaterialTheme.typography.labelSmall.copy(fontWei
            )
        }

        // Reasoning Accordion
        if (!message.reasoning.isNullOrBlank()) {
            ThinkingBlock(
                reasoningText = message.reasoning,
                isStreaming = isStreaming && message.content.isEmpty()
            )
            Spacer(modifier = Modifier.height(12.dp))
        }

        // Document Body Content
        Box(modifier = Modifier.fillMaxWidth()) {
            Text(
                text = message.content,
                style = MaterialTheme.typography.bodyLarge.copy(color =
            )
        }

        // Web Citations Strip
        if (message.citations.isNotEmpty()) {
            Spacer(modifier = Modifier.height(10.dp))
            WebCitationStrip(citations = message.citations)
        }
    }
}

// Floating Specular Action Island
AnimatedVisibility(
    visible = showActionIsland,
    enter = fadeIn() + scaleIn(initialScale = 0.94f),
```

```
        exit = fadeOut() + scaleOut(targetScale = 0.94f)
    ) {
        Surface(
            shape = RoundedCornerShape(20.dp),
            color = GlassIslandBackground,
            modifier = Modifier
                .padding(top = 8.dp)
                .specularBorder(RoundedCornerShape(20.dp))
        ) {
            Row(
                modifier = Modifier.padding(horizontal = 6.dp, vertical = 2.
                horizontalArrangement = Arrangement.spacedBy(2.dp),
                verticalAlignment = Alignment.CenterVertically
            ) {
                IconButton(
                    onClick = {
                        val cm = context.getSystemService(Context.CLIPBOARD_
                        cm.setPrimaryClip(ClipData.newPlainText("Message Con
                        view.performHapticFeedback(HapticFeedbackConstants.C
                        showActionIsland = false
                    },
                    modifier = Modifier.size(32.dp)
                ) {
                    Icon(Icons.Rounded.ContentCopy, contentDescription = "Co
                }

                IconButton(
                    onClick = {
                        val shareIntent = Intent(Intent.ACTION_SEND).apply {
                            type = "text/plain"
                            putExtra(Intent.EXTRA_TEXT, message.content)
                        }
                        context.startActivity(Intent.createChooser(shareInte
                        showActionIsland = false
                    },
                    modifier = Modifier.size(32.dp)
                ) {
                    Icon(Icons.Rounded.Share, contentDescription = "Share",
                }

                if (isUser) {
                    IconButton(
                        onClick = {
                            onEditAndResend?.invoke(message)
                            showActionIsland = false
                        },
                        modifier = Modifier.size(32.dp)
                    ) {
                        Icon(Icons.Rounded.Edit, contentDescription = "Edit
                    }
                } else if (!isStreaming) {
                    IconButton(
                        onClick = {
                            onRegenerate?.invoke()
                            showActionIsland = false
                        },
```

```
                            modifier = Modifier.size(32.dp)
                        ) {
                            Icon(Icons.Rounded.Refresh, contentDescription = "Re
                        }
                    }
                }
            }
        }
    }
}
```

7. Floating Island Composer with Quantum Auras

`ui/screens/chat/ChatComposer.kt`

Detached squircle composer floating 12dp above the gesture bar with glowing auras and spring-animated morphing controls.

```
package com.zai.chat.ui.screens.chat

import android.view.HapticFeedbackConstants
import androidx.compose.animation.animateColorAsState
import androidx.compose.animation.core.FastOutSlowInEasing
import androidx.compose.animation.core.RepeatMode
import androidx.compose.animation.core.animateFloat
import androidx.compose.animation.core.infiniteRepeatable
import androidx.compose.animation.core.rememberInfiniteTransition
import androidx.compose.animation.core.tween
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.clickable
import androidx.compose.foundation.horizontalScroll
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.heightIn
import androidx.compose.foundation.layout.imePadding
import androidx.compose.foundation.layout.navigationBarsPadding
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.BasicTextField
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.rounded.ArrowUpward
import androidx.compose.material.icons.rounded.AttachFile
```

```
import androidx.compose.material.icons.rounded.Memory
import androidx.compose.material.icons.rounded.Public
import androidx.compose.material.icons.rounded.Stop
import androidx.compose.material3.DropdownMenu
import androidx.compose.material3.DropdownMenuItem
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.SolidColor
import androidx.compose.ui.platform.LocalView
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zai.chat.ui.components.specularBorder
import com.zai.chat.ui.theme.BleedMotion
import com.zai.chat.ui.theme.BorderAmbient
import com.zai.chat.ui.theme.CanvasPureBlack
import com.zai.chat.ui.theme.CrimsonFlare
import com.zai.chat.ui.theme.GlassIslandBackground
import com.zai.chat.ui.theme.QuantumCyan
import com.zai.chat.ui.theme.QuantumCyanGlow
import com.zai.chat.ui.theme.RadiantAmber
import com.zai.chat.ui.theme.RadiantAmberGlow
import com.zai.chat.ui.theme.SurfaceActive
import com.zai.chat.ui.theme.SurfaceRaised
import com.zai.chat.ui.theme.TextPrimary
import com.zai.chat.ui.theme.TextSecondary
import com.zai.chat.ui.theme.TextTertiary

@Composable
fun ChatComposer(
    isStreaming: Boolean,
    selectedModel: String,
    availableModels: List<String>,
    webSearchActive: Boolean,
    deepThinkingActive: Boolean,
    onSendMessage: (String) -> Unit,
    onStopStreaming: () -> Unit,
    onToggleWebSearch: (Boolean) -> Unit,
    onToggleDeepThinking: (Boolean) -> Unit,
    onSelectModel: (String) -> Unit,
    modifier: Modifier = Modifier
) {
```

```
var textInput by remember { mutableStateOf("") }
val view = LocalView.current
var modelMenuOpen by remember { mutableStateOf(false) }

val islandShape = RoundedCornerShape(24.dp)

Box(
    modifier = modifier
        .fillMaxWidth()
        .navigationBarsPadding()
        .imePadding()
        .padding(horizontal = 14.dp, vertical = 10.dp)
) {
    Surface(
        shape = islandShape,
        color = GlassIslandBackground,
        modifier = Modifier
            .fillMaxWidth()
            .specularBorder(islandShape)
    ) {
        Column(
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 14.dp, vertical = 10.dp)
        ) {
            // Micro-Feature Dynamic Chips
            Row(
                modifier = Modifier
                    .fillMaxWidth()
                    .horizontalScroll(rememberScrollState()),
                horizontalArrangement = Arrangement.spacedBy(8.dp),
                verticalAlignment = Alignment.CenterVertically
            ) {
                // Model Selector Pill
                Box {
                    Surface(
                        shape = RoundedCornerShape(14.dp),
                        color = SurfaceActive,
                        modifier = Modifier.clickable {
                            view.performHapticFeedback(HapticFeedbackConstan
                            modelMenuOpen = true
                        }
                    ) {
                        Text(
                            text = selectedModel.uppercase(),
                            style = MaterialTheme.typography.labelSmall.copy
                            modifier = Modifier.padding(horizontal = 10.dp,
                        )
                    }

                    DropdownMenu(
                        expanded = modelMenuOpen,
                        onDismissRequest = { modelMenuOpen = false },
                        modifier = Modifier.background(GlassIslandBackground
                    ) {
                        availableModels.forEach { model ->
```

```
            DropdownMenuItem(
                text = { Text(model, color = TextPrimary) },
                onClick = {
                    onSelectModel(model)
                    modelMenuOpen = false
                }
            )
        }
    }
}

// Quantum Web Search Toggle
val searchAura = if (webSearchActive) QuantumCyanGlow else C
Surface(
    shape = RoundedCornerShape(14.dp),
    color = if (webSearchActive) SurfaceActive else SurfaceR
    border = androidx.compose.foundation.BorderStroke(1.dp, 
    modifier = Modifier
        .clickable {
            view.performHapticFeedback(HapticFeedbackConstan
            onToggleWebSearch(!webSearchActive)
        }
) {
    Row(
        modifier = Modifier.padding(horizontal = 10.dp, vert
        verticalAlignment = Alignment.CenterVertically
    ) {
        Icon(
            imageVector = Icons.Rounded.Public,
            contentDescription = null,
            tint = if (webSearchActive) QuantumCyan else Tex
            modifier = Modifier.size(13.dp)
        )
        Spacer(modifier = Modifier.width(6.dp))
        Text(
            text = "WEB SEARCH",
            style = MaterialTheme.typography.labelSmall.copy
                fontWeight = FontWeight.Bold,
                color = if (webSearchActive) QuantumCyan els
            )
        )
    }
}

// Radiant Thinking Toggle
Surface(
    shape = RoundedCornerShape(14.dp),
    color = if (deepThinkingActive) SurfaceActive else Surfa
    border = androidx.compose.foundation.BorderStroke(1.dp, 
    modifier = Modifier
        .clickable {
            view.performHapticFeedback(HapticFeedbackConstan
            onToggleDeepThinking(!deepThinkingActive)
        }
) {
    Row(
```

```
                modifier = Modifier.padding(horizontal = 10.dp, vert
                verticalAlignment = Alignment.CenterVertically
            ) {
                Icon(
                    imageVector = Icons.Rounded.Memory,
                    contentDescription = null,
                    tint = if (deepThinkingActive) RadiantAmber else
                    modifier = Modifier.size(13.dp)
                )
                Spacer(modifier = Modifier.width(6.dp))
                Text(
                    text = "REASONING",
                    style = MaterialTheme.typography.labelSmall.copy
                        fontWeight = FontWeight.Bold,
                        color = if (deepThinkingActive) RadiantAmber
                    )
                )
            }
        }
    }

    Spacer(modifier = Modifier.height(8.dp))

    // Input Box with Morphing Action Button
    Row(
        modifier = Modifier.fillMaxWidth(),
        verticalAlignment = Alignment.Bottom
    ) {
        IconButton(
            onClick = { view.performHapticFeedback(HapticFeedbackCon
            modifier = Modifier.size(36.dp)
        ) {
            Icon(Icons.Rounded.AttachFile, contentDescription = "Att
        }

        Box(
            modifier = Modifier
                .weight(1f)
                .padding(horizontal = 6.dp, vertical = 8.dp)
        ) {
            if (textInput.isEmpty()) {
                Text(
                    text = "Query model or execute command...",
                    style = MaterialTheme.typography.bodyLarge.copy(
                )
            }
            BasicTextField(
                value = textInput,
                onValueChange = { textInput = it },
                textStyle = TextStyle(
                    color = TextPrimary,
                    fontSize = 16.sp,
                    lineHeight = 24.sp
                ),
                cursorBrush = SolidColor(QuantumCyan),
                modifier = Modifier
```

```
                .fillMaxWidth()
                .heightIn(max = 140.dp)
            )
        }

        // Morphing Send / Cancel Action Pill
        val actionButtonBg by animateColorAsState(
            targetValue = when {
                isStreaming -> CrimsonFlare
                textInput.isNotBlank() -> TextPrimary
                else -> SurfaceActive
            },
            animationSpec = BleedMotion.PressScaleSpring,
            label = "action_btn_bg"
        )

        Box(
            modifier = Modifier
                .size(36.dp)
                .clip(CircleShape)
                .background(actionButtonBg)
                .clickable {
                    view.performHapticFeedback(HapticFeedbackConstan
                    if (isStreaming) {
                        onStopStreaming()
                    } else if (textInput.isNotBlank()) {
                        onSendMessage(textInput)
                        textInput = ""
                    }
                },
            contentAlignment = Alignment.Center
        ) {
            Icon(
                imageVector = if (isStreaming) Icons.Rounded.Stop el
                contentDescription = if (isStreaming) "Stop" else "D
                tint = if (isStreaming || textInput.isNotBlank()) Ca
                modifier = Modifier.size(18.dp)
            )
        }
    }
  }
 }
}
}
```

8. Editorial Timeline Drawer & Live Fuzzy Search

`ui/screens/drawer/DrawerChatList.kt`

Chronological grouping (Pinned, Today, Yesterday, Previous 7 Days) with tactile swipe actions and a live session health card.

```
package com.zai.chat.ui.screens.drawer

import android.view.HapticFeedbackConstants
import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.rounded.Add
import androidx.compose.material.icons.rounded.Bolt
import androidx.compose.material.icons.rounded.DeleteOutline
import androidx.compose.material.icons.rounded.PushPin
import androidx.compose.material.icons.rounded.Search
import androidx.compose.material.icons.rounded.Settings
import androidx.compose.material3.DismissDirection
import androidx.compose.material3.DismissValue
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.OutlinedTextFieldDefaults
import androidx.compose.material3.SwipeToDismiss
import androidx.compose.material3.Text
import androidx.compose.material3.rememberDismissState
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalView
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.zai.chat.data.model.Chat
import com.zai.chat.ui.components.specularBorder
import com.zai.chat.ui.theme.BorderAmbient
import com.zai.chat.ui.theme.CrimsonFlare
```

```
import com.zai.chat.ui.theme.EmeraldPulse
import com.zai.chat.ui.theme.MidnightObsidian
import com.zai.chat.ui.theme.RadiantAmber
import com.zai.chat.ui.theme.SurfaceActive
import com.zai.chat.ui.theme.SurfaceBase
import com.zai.chat.ui.theme.SurfaceRaised
import com.zai.chat.ui.theme.TextPrimary
import com.zai.chat.ui.theme.TextSecondary
import com.zai.chat.ui.theme.TextTertiary

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DrawerChatList(
    viewModel: DrawerViewModel,
    onSelectChat: (String) -> Unit,
    onNewChat: () -> Unit,
    onOpenSettings: () -> Unit,
    modifier: Modifier = Modifier
) {
    val state by viewModel.uiState.collectAsState()
    val view = LocalView.current

    Column(
        modifier = modifier
            .fillMaxHeight()
            .width(320.dp)
            .background(MidnightObsidian)
            .padding(16.dp)
    ) {
        // Brand Header
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically
        ) {
            Row(verticalAlignment = Alignment.CenterVertically) {
                Box(
                    modifier = Modifier
                        .size(10.dp)
                        .clip(CircleShape)
                        .background(EmeraldPulse)
                )
                Spacer(modifier = Modifier.width(10.dp))
                Text(
                    text = "BLEED-AI",
                    style = MaterialTheme.typography.titleMedium.copy(fontWeight
                )
            }

            Row {
                IconButton(onClick = {
                    view.performHapticFeedback(HapticFeedbackConstants.KEYBOARD_
                    onNewChat()
                }) {
                    Icon(Icons.Rounded.Add, contentDescription = "New Session",
                }
```

```
        IconButton(onClick = {
            view.performHapticFeedback(HapticFeedbackConstants.KEYBOARD_T
            onOpenSettings()
        }) {
            Icon(Icons.Rounded.Settings, contentDescription = "Settings"
        }
    }
}

Spacer(modifier = Modifier.height(12.dp))

// Specular Search Field
OutlinedTextField(
    value = state.searchQuery,
    onValueChange = { viewModel.onSearchQueryChanged(it) },
    placeholder = { Text("Search conversations...", style = MaterialThem
    leadingIcon = { Icon(Icons.Rounded.Search, contentDescription = null
    shape = RoundedCornerShape(12.dp),
    colors = OutlinedTextFieldDefaults.colors(
        focusedContainerColor = SurfaceBase,
        unfocusedContainerColor = SurfaceBase,
        focusedBorderColor = BorderAmbient,
        unfocusedBorderColor = Color.Transparent
    ),
    modifier = Modifier
        .fillMaxWidth()
        .specularBorder(RoundedCornerShape(12.dp))
)

Spacer(modifier = Modifier.height(16.dp))

// Chronological List Stream
LazyColumn(
    modifier = Modifier
        .fillMaxWidth()
        .weight(1f),
    verticalArrangement = Arrangement.spacedBy(6.dp)
) {
    // Group Pinned Chats
    val pinned = state.chats.filter { it.pinned }
    if (pinned.isNotEmpty()) {
        item {
            Text(
                text = "PINNED THREADS",
                style = MaterialTheme.typography.labelSmall.copy(fontWei
                modifier = Modifier.padding(vertical = 4.dp)
            )
        }
        items(pinned, key = { "pinned_${it.id}" }) { chat ->
            TimelineChatCard(
                chat = chat,
                onClick = { onSelectChat(chat.id) },
                onTogglePin = { viewModel.togglePin(chat.id, chat.pinned
            )
        }
    }
```

```
        // Group Recent Chats
        val recent = state.chats.filter { !it.pinned }
        if (recent.isNotEmpty()) {
            item {
                Text(
                    text = "ACTIVE RECENT",
                    style = MaterialTheme.typography.labelSmall.copy(fontWei
                    modifier = Modifier.padding(top = 10.dp, bottom = 4.dp)
                )
            }
            items(recent, key = { it.id }) { chat ->
                val dismissState = rememberDismissState(
                    confirmValueChange = { value ->
                        if (value == DismissValue.DismissedToStart) {
                            view.performHapticFeedback(HapticFeedbackConstan
                            viewModel.deleteChat(chat.id)
                            true
                        } else false
                    }
                )

                SwipeToDismiss(
                    state = dismissState,
                    directions = setOf(DismissDirection.EndToStart),
                    background = {
                        Box(
                            modifier = Modifier
                                .fillMaxWidth()
                                .fillMaxHeight()
                                .clip(RoundedCornerShape(10.dp))
                                .background(CrimsonFlare.copy(alpha = 0.8f))
                                .padding(horizontal = 16.dp),
                            contentAlignment = Alignment.CenterEnd
                        ) {
                            Icon(Icons.Rounded.DeleteOutline, contentDescrip
                        }
                    },
                    dismissContent = {
                        TimelineChatCard(
                            chat = chat,
                            onClick = { onSelectChat(chat.id) },
                            onTogglePin = { viewModel.togglePin(chat.id, cha
                        )
                    }
                )
            }
        }
    }

    // Live Session Telemetry Footer
    Surface(
        shape = RoundedCornerShape(12.dp),
        color = SurfaceBase,
        modifier = Modifier
            .fillMaxWidth()
```

```
                .specularBorder(RoundedCornerShape(12.dp))
        ) {
            Row(
                modifier = Modifier.padding(12.dp),
                verticalAlignment = Alignment.CenterVertically
            ) {
                Box(
                    modifier = Modifier
                        .size(8.dp)
                        .clip(CircleShape)
                        .background(EmeraldPulse)
                )
                Spacer(modifier = Modifier.width(10.dp))
                Column(modifier = Modifier.weight(1f)) {
                    Text("Z.AI HANDSHAKE ACTIVE", style = MaterialTheme.typograp
                    Text("Latency: 42ms • TLS 1.3", style = MaterialTheme.typogr
                }
                Icon(Icons.Rounded.Bolt, contentDescription = null, tint = Emera
            }
        }
    }
}

@Composable
fun TimelineChatCard(
    chat: Chat,
    onClick: () -> Unit,
    onTogglePin: () -> Unit
) {
    val view = LocalView.current
    val shape = RoundedCornerShape(10.dp)

    Surface(
        shape = shape,
        color = SurfaceBase,
        modifier = Modifier
            .fillMaxWidth()
            .specularBorder(shape)
            .clickable {
                view.performHapticFeedback(HapticFeedbackConstants.KEYBOARD_TAP)
                onClick()
            }
    ) {
        Row(
            modifier = Modifier.padding(horizontal = 12.dp, vertical = 10.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {
            Column(modifier = Modifier.weight(1f)) {
                Text(
                    text = chat.title
```