DualCam Pro - Product Requirements Document

Version: 1.0
Date: October 23, 2025
Product Owner: Internal Development Team
Target Launch: Q1 2026

Executive Summary

DualCam Pro is a professional-grade dual camera iOS application that enables simultaneous recording from front and back cameras with a stunning liquid glass interface. The app records three separate outputs simultaneously (dual view composite, front camera only, back camera only) while offering comprehensive camera controls that rival Apple's native Camera app. Built with Swift 6 and targeting iOS 18+, DualCam Pro leverages the latest iOS features including deferred processing, zero shutter lag, and responsive capture APIs.

Key Differentiators

- **Triple Output Recording**: Simultaneous recording of dual view, front only, and back only
- **Liquid Glass Theme**: Modern glassmorphism design matching Apple's design language
- **iOS 18+ Native**: Leverages latest camera APIs for best-in-class performance
- **Comprehensive Features**: Every camera feature available in iOS (zoom, focus, exposure, stabilization, etc.)
- **Production-Ready**: Polished, performant, and reliable like official Apple apps

---

Product Overview and Goals

Vision

Create the most comprehensive and polished dual camera app on iOS that content creators, vloggers, and professionals rely on for simultaneous camera recording with maximum flexibility in post-production.

Goals

1. **Functionality**: Enable simultaneous dual camera recording with three separate output files
2. **User Experience**: Deliver intuitive, beautiful interface using liquid glass design
3. **Performance**: Maintain smooth 60fps UI and stable recording without dropped frames
4. **Quality**: Support up to 4K resolution at 60fps with professional codec options
5. **Flexibility**: Provide comprehensive manual controls for advanced users while remaining accessible to beginners
6. **Reliability**: Ensure recordings are never lost with robust error handling and recovery

Success Metrics

- Recording success rate: >99%
- App Store rating: >4.7 stars
- Frame rate stability: 30fps/60fps without drops during recording
- Hardware cost: <0.95 consistently
- Crash-free sessions: >99.5%
- Time to first recording: <30 seconds from launch

Target Audience and Devices

Primary Audience

1. **Content Creators**: YouTubers, TikTokers, Instagram creators
2. **Vloggers**: Travel, lifestyle, tech reviewers
3. **Professional Videographers**: Interviews, events, documentaries
4. **Educators**: Online course creators, tutorial makers
5. **Social Media Managers**: Brand content creation

Secondary Audience

1. Casual users wanting unique dual perspective videos
2. Parents capturing family moments from multiple angles
3. Students creating video projects

Device Requirements

- **Minimum**: iPhone XS, XS Max, XR (A12 Bionic chip)
- **Recommended**: iPhone 11 and later
- **Optimal**: iPhone 13 Pro and later

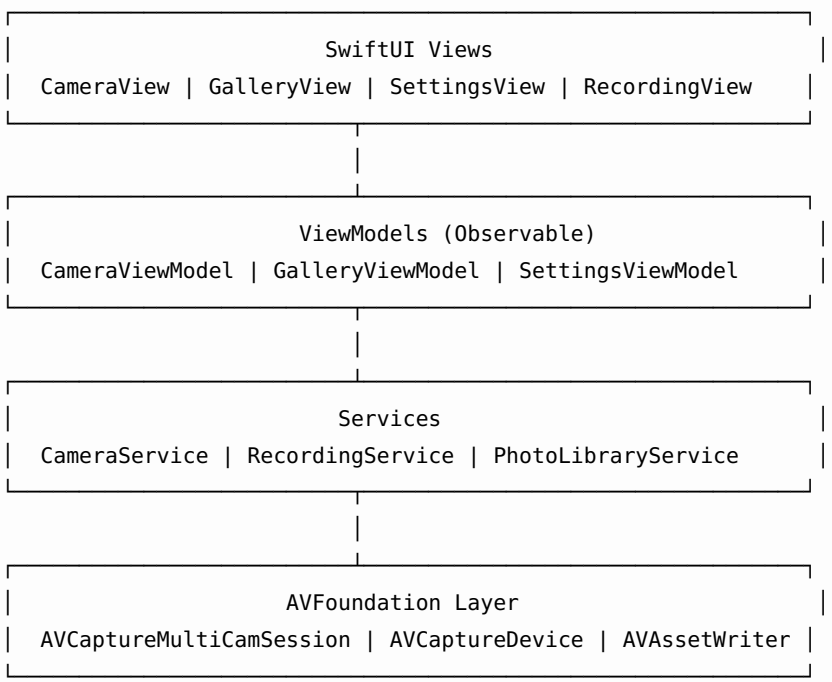
- **iOS Version**: iOS 18.0+
- **Storage**: 2GB+ available space
- **Multi-Cam Support**: Required (checked at runtime)

Device Feature Matrix

Device	Multi-Cam	4K Dual	ProRAW	Cinematic
iPhone XS/XR	✓	✓	✗	✗
iPhone 11	✓	✓	✗	✗
iPhone 12	✓	✓	✓	✗
iPhone 13+	✓	✓	✓	✓

Technical Architecture

System Architecture



Technology Stack

- **Language**: Swift 6 with strict concurrency checking
- **UI Framework**: SwiftUI with iOS 18+ features
- **Camera Framework**: AVFoundation (AVCaptureMultiCamSession)
- **Video Compositing**: Metal for GPU-accelerated rendering
- **Storage**: PhotoKit for Photos library integration
- **Persistence**: SwiftData for app settings and metadata

- **Concurrency**: Swift concurrency (async/await, actors)
- **Architecture**: MVVM with service layer

Core Components

1. CameraService (Actor)

```
actor CameraService {
    - setupMultiCamSession()
    - configureCameras(front: Device, back: Device)
    - startSession()
    - stopSession()
    - monitorHardwareCost()
    - handleSystemPressure()
}
```

Responsibilities: - Manage AVCaptureMultiCamSession lifecycle - Configure front and back camera inputs - Monitor hardware cost and system pressure - Handle session interruptions and errors - Provide camera device discovery

2. RecordingService (Actor)

```
actor RecordingService {
    - startRecording(outputs: [OutputType])
    - pauseRecording()
    - resumeRecording()
    - stopRecording()
    - composeFrames(front: CMSampleBuffer, back: CMSampleBuffer)
}
```

Responsibilities: - Manage three simultaneous AVAssetWriter instances - Composite dual view using Metal - Handle pause/resume functionality - Monitor recording duration - Ensure frame synchronization

3. PhotoLibraryService

```
class PhotoLibraryService {
    - requestAuthorization()
    - saveVideo(url: URL, metadata: Metadata)
    - fetchRecordings() -> [Recording]
    - deleteRecording(id: UUID)
    - createAlbum(name: String)
}
```

Responsibilities: - Request Photos library permissions - Save recordings to Photos library - Fetch and manage recordings - Handle metadata and organization

4. SettingsManager

```
@Observable class SettingsManager {
    var resolution: Resolution
    var frameRate: FrameRate
    var videoCodec: VideoCodec
    var stabilization: StabilizationMode
    // ... all settings
}
```

Responsibilities: - Store user preferences - Persist settings using SwiftData - Provide reactive updates to UI - Validate setting combinations

---

Feature Specifications

Core Features

1. Dual Camera Preview (P0)

Description: Simultaneous display of front and back camera feeds in stacked layout.

Requirements: - Front camera on top half of screen - Back camera on bottom half of screen - Real-time preview at 30fps minimum - Independent resolution per camera - Tap to switch which camera is on top - Pinch on individual preview to zoom that camera

Implementation Notes: - Use two AVCaptureVideoPreviewLayer instances - Manual connection to specific camera ports - Metal rendering for compositing preview

Acceptance Criteria: - ✓ Both cameras display simultaneously - ✓ Preview maintains <100ms latency - ✓ No frame drops during preview - ✓ Smooth transitions when switching layout - ✓ Hardware cost stays <0.95

---

2. Triple Recording System (P0)

Description: Record three separate video files simultaneously.

Output Files: 1. **Dual View (dualview_[timestamp].mov)**: Composited front+back in stacked layout 2. **Front Only (front_[timestamp].mov)**: Full resolution front camera 3. **Back Only (back_[timestamp].mov)**: Full resolution back camera

Requirements: - All three files start/stop synchronously - Synchronized timestamps across all files - Individual file size limits (configurable) - Atomic recording (all succeed or all fail) - Progress indication for each output

Technical Specifications: - Format: MOV container - Video Codec: H.265 (HEVC) default, H.264 optional, ProRes for Pro devices - Audio Codec: AAC 256kbps stereo - Bit Rate: Variable (average 50Mbps for 4K) - Color Space: Rec. 709 (SDR) or Rec. 2020 (HDR)

Acceptance Criteria: - ✓ Three files created per recording - ✓ Files have identical duration (±1 frame) - ✓ No audio sync issues - ✓ Dual view composition is pixel-perfect - ✓ Files saved to Photos library with correct metadata

---

3. Liquid Glass UI Theme (P0)

Description: Modern glassmorphism interface matching Apple's design language.

Design Specifications: - **Materials**: `.ultraThinMaterial` for floating controls, `.thinMaterial` for panels - **Opacity**: 20-30% for glass surfaces - **Borders**: 1px white with 40% opacity - **Shadows**: Soft shadows with 20% black opacity, 20px radius - **Colors**: White text on glass, adaptive for light/dark mode - **Animations**: Spring animations (0.6s duration, 0.5 damping)

Components: 1. **Floating Control Buttons**: Circular glass buttons for camera controls 2. **Recording Status Bar**: Translucent bar showing duration and status 3. **Settings Panel**: Glass overlay with controls 4. **Gallery Thumbnails**: Glass cards for video previews 5. **Toast Notifications**: Glass notifications for feedback

Accessibility: - Respect `.accessibilityReduceTransparency` - Maintain WCAG AA contrast ratios (4.5:1 minimum) - VoiceOver labels for all controls - Dynamic Type support

Acceptance Criteria: - ✓ All UI uses glass materials - ✓ Animations are smooth at 60fps - ✓ Design matches Apple HIG - ✓ Works in light and dark mode - ✓ Accessible to screen readers

---

Camera Features

4. Independent Camera Zoom (P0)

Requirements: - Pinch to zoom on individual camera preview - One-finger swipe on zoom slider - Quick zoom buttons: 0.5x, 1x, 2x, 5x - Range: 0.5x to 10x digital zoom - Smooth zoom animation - Zoom level

indicator

Implementation: - Set `AVCaptureDevice.videoZoomFactor` - Ramp zoom with `rampToVideoZoomFactor()` - Independent zoom state per camera

---

5. Focus Controls (P0)

Requirements: - Tap to focus on preview - Continuous autofocus mode - Manual focus slider - Focus lock - Focus indicator (yellow box) - Face detection priority

Modes: - Auto Focus (default) - Manual Focus - Continuous Auto Focus - Face Detection AF

---

6. Exposure Controls (P0)

Requirements: - Auto exposure (default) - Manual exposure (shutter speed + ISO) - Exposure compensation (-3 to +3 EV) - Tap to expose - Exposure lock - Zebra stripes for overexposure warning

Manual Controls: - Shutter speed: 1/8000s to 1/3s - ISO: 25 to device maximum - EV slider: -3 to +3 in 0.1 increments

---

7. White Balance (P0)

Requirements: - Auto white balance (default) - Manual Kelvin: 2500K to 10000K - Presets: Daylight, Cloudy, Tungsten, Fluorescent - White balance lock - Fine tuning (±10 on green/magenta, blue/amber)

---

8. Video Stabilization (P0)

Requirements: - Off - Standard - Enhanced (Cinematic) - Action Mode (maximum stabilization)

Implementation: - Use `AVCaptureConnection.preferredVideoStabilizationMode` - Monitor hardware cost (stabilization is expensive)

---

9. Resolution & Frame Rate (P0)

Supported Combinations: | Resolution | Frame Rates | Dual Cam Support | |———–|———-|————| | 720p HD | 30, 60 | ✓ | | 1080p FHD | 30, 60 | ✓ | | 4K UHD | 30, 60 | ✓ (device dependent) |

Requirements: - Setting persists between sessions - Warning if combination exceeds hardware cost - Automatic fallback if unsupported - Quality indicator in UI

---

10. Additional Camera Features (P1)

Flash/Torch: - Auto flash - On/always - Off - Torch for video

Timer: - 3 seconds - 10 seconds - Custom delay

Grid: - Off - Rule of thirds (3x3) - Golden ratio - Center crosshair

Filters (Real-time): - None - Vivid - Dramatic - Mono - Silvertone - Noir

Audio: - Built-in microphone - External microphone detection - Wind noise reduction toggle - Audio monitoring with levels

---

Recording Features

11. Recording Controls (P0)

Requirements: - Large glass record button - Pause/resume functionality - Stop button - Duration display (HH:MM:SS) - Recording indicator (pulsing red dot) - Storage space indicator - File size indicator per output

Behavior: - Tap to start recording (all 3 outputs) - Tap pause button to pause - Tap resume to continue - Tap stop to finish and save - Confirm before discarding recording

---

12. Recording Session Management (P0)

Requirements: - Background recording support - Handle interruptions (calls, notifications) - Low storage warnings - Thermal throttling management - Battery optimization - Maximum duration: 3 hours or storage limit

Error Handling: - Graceful degradation if one output fails - Automatic retry for failed saves - User notification of issues - Crash recovery (resume interrupted recordings)

Gallery & Playback

13. In-App Gallery (P0)

Requirements: - Grid view of recordings - Thumbnail generation - Metadata display (date, duration, size, resolution) - Sort by: Date, Duration, Size - Filter by: Resolution, Date range - Search by date

Thumbnail Display: - Show dual view composite as thumbnail - Duration badge - Resolution badge - Quick actions: Play, Share, Delete

---

14. Video Playback (P0)

Requirements: - Full screen playback - Switch between three outputs during playback - Scrubbing timeline - Play/pause controls - Speed control: 0.5x, 1x, 1.5x, 2x - Volume control - Picture-in-picture support

Player Controls: - Glass overlay with controls - Auto-hide after 3 seconds - Tap to show/hide - Seek thumbnails

---

15. Photos Library Integration (P0)

Requirements: - Save all three videos to Photos library - Create “DualCam Pro” album - Preserve metadata (location, date, camera info) - Batch export - iCloud Photo Library support

Permissions: - Request authorization on first launch - Handle denied/restricted states - Link to Settings if denied

---

Settings

16. Settings Screen (P0)

Categories:

Recording Settings: - Resolution: 720p, 1080p, 4K - Frame Rate: 30fps, 60fps - Video Codec: H.264, H.265, ProRes (if available) - Bit Rate: Auto, Low, Medium, High, Maximum - Audio Quality: 128kbps, 256kbps, 320kbps

Camera Settings: - Default stabilization mode - Grid overlay default - Timer default - Flash default - Focus mode default - Exposure mode default - White balance default

Recording Behavior: - Auto-save to Photos library (toggle) - Create separate album (toggle) - Keep in-app copies (toggle) - Maximum recording duration - Low storage warning threshold

Dual View Layout: - Front on top / Back on top (default) - Split ratio: 50/50, 60/40, 70/30 - Border between cameras (toggle) - Border color

Advanced: - Enable ProRes (if available) - Monitor hardware cost (developer mode) - Show FPS counter - Enable debug logs

App Settings: - About - Privacy Policy - Terms of Service - App version - Contact support - Rate app - Share app

---

UI/UX Specifications

Design System

Color Palette

```
// Light Mode
Glass Primary: White 25% opacity
Glass Border: White 50% opacity
Text Primary: Black 100%
Text Secondary: Black 60%
Accent: iOS Blue (#007AFF)
Destructive: iOS Red (#FF3B30)
Success: iOS Green (#34C759)

// Dark Mode
Glass Primary: Black 20% opacity
Glass Border: White 30% opacity
Text Primary: White 100%
Text Secondary: White 70%
Accent: iOS Blue (#0A84FF)
Destructive: iOS Red (#FF453A)
Success: iOS Green (#30D158)
```

Typography

```
// Using SF Pro (iOS system font)
Title: 34pt Bold
Headline: 28pt Semibold
Body: 17pt Regular
Subheadline: 15pt Regular
Caption: 13pt Regular
```

Spacing

```
XS: 4pt
S: 8pt
M: 16pt
L: 24pt
XL: 32pt
XXL: 48pt
```

Corner Radius

```
Small: 8pt (buttons)
Medium: 16pt (cards)
Large: 24pt (panels)
XLarge: 32pt (modals)
Circle: 50% (circular buttons)
```

Screen Layouts

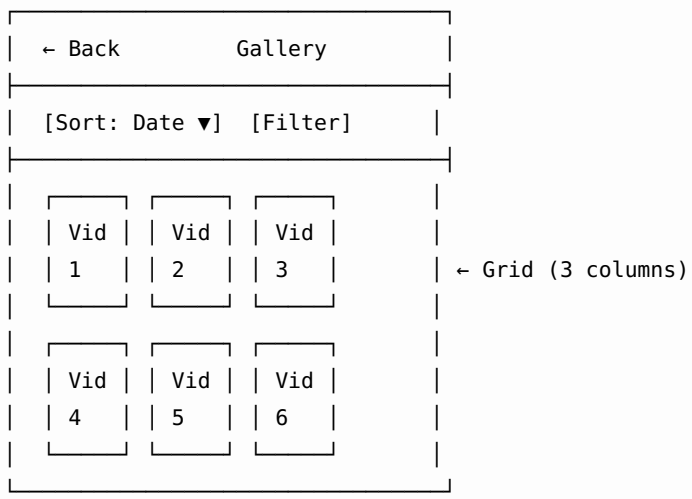
Camera View (Main Screen)

```
┌─────────────────────────────────┐
│  [Settings]    ◍ REC   [Flip]  │ ← Top bar (glass)
├─────────────────────────────────┤
│                                 │
│     Front Camera Preview        │ ← Top half
│         (Interactive)           │
│                                 │
├─────────────────────────────────┤
│                                 │
│     Back Camera Preview         │ ← Bottom half
│         (Interactive)           │
│                                 │
├─────────────────────────────────┤
│  [Grid] [Flash] [Timer]         │
│                                 │
│      [📷 Record Button]         │ ← Large glass button
│                                 │
│  [Gallery] [Zoom] [Settings]    │
└─────────────────────────────────┘
```

Recording View

```
┌─────────────────────────────────┐
│  ◍ Recording    00:05:23        │ ← Status bar
├─────────────────────────────────┤
│                                 │
│  Dual Camera Preview            │
│  (Full screen, minimal UI)      │
│                                 │
│                                 │
├─────────────────────────────────┤
│    [⏸ Pause]  [■ Stop]          │ ← Glass controls
└─────────────────────────────────┘
```

Gallery View



Interactions

Gestures

- **Tap**: Focus/Expose at point, button actions
- **Double Tap**: Reset zoom to 1x
- **Long Press**: Lock focus/exposure
- **Pinch**: Zoom in/out on camera
- **Swipe Up/Down**: Switch camera layout
- **Swipe Left/Right**: Navigate gallery

Haptics

- **Light Impact**: Button taps
- **Medium Impact**: Mode changes
- **Heavy Impact**: Recording start/stop
- **Success**: Recording saved
- **Warning**: Error occurred
- **Error**: Critical failure

Animations

- **Spring**: Natural, bouncy (duration: 0.6s, damping: 0.5)
- **EaseInOut**: Smooth transitions (duration: 0.3s)
- **Linear**: Progress indicators
- **Recording Pulse**: Red dot fades 0.3 ↔ 1.0 opacity (0.8s cycle)

Technical Implementation Details

AVCaptureMultiCamSession Setup

```swift
// 1. Check support
guard AVCaptureMultiCamSession.isMultiCamSupported else {
    throw CameraError.multiCamNotSupported
}

// 2. Create session
let session = AVCaptureMultiCamSession()
session.beginConfiguration()

// 3. Add cameras
let backCamera = AVCaptureDevice.default(.builtInWideAngleCamera,
                                         for: .video,
                                         position: .back)!
let backInput = try AVCaptureDeviceInput(device: backCamera)
session.addInputWithNoConnections(backInput)

let frontCamera = AVCaptureDevice.default(.builtInWideAngleCamera,
                                          for: .video,
                                          position: .front)!
let frontInput = try AVCaptureDeviceInput(device: frontCamera)
session.addInputWithNoConnections(frontInput)

// 4. Configure format for optimal performance
try backCamera.lockForConfiguration()
if let format = findOptimalFormat(for: backCamera, resolution:
        .uhd4K) {
    backCamera.activeFormat = format
    backCamera.activeVideoMinFrameDuration = CMTime(value: 1,
        timescale: 30)
    backCamera.activeVideoMaxFrameDuration = CMTime(value: 1,
        timescale: 30)
}
backCamera.unlockForConfiguration()

// 5. Create separate outputs for each camera
let backVideoOutput = AVCaptureVideoDataOutput()
backVideoOutput.videoSettings = [
    kCVPixelBufferPixelFormatTypeKey as String:
        kCVPixelFormatType_420YpCbCr8BiPlanarFullRange
]
session.addOutputWithNoConnections(backVideoOutput)

let frontVideoOutput = AVCaptureVideoDataOutput()
session.addOutputWithNoConnections(frontVideoOutput)

// 6. Manual connections
let backVideoPort = backInput.ports(for: .video,
                                    sourceDeviceType:
        .builtInWideAngleCamera,
```

```
                                        sourceDevicePosition:
        .back).first!
let backConnection = AVCaptureConnection(inputPorts:
        [backVideoPort],
                                             output: backVideoOutput)
session.addConnection(backConnection)

// 7. Monitor hardware cost
print("Hardware cost: \(session.hardwareCost)") // Must be < 1.0

session.commitConfiguration()
session.startRunning()
```

Triple Recording Strategy

```
// Strategy: Three separate AVAssetWriter instances

// 1. Dual View Writer (composited)
let dualViewWriter = try AVAssetWriter(url: dualViewURL, fileType:
        .mov)
let dualVideoSettings: [String: Any] = [
    AVVideoCodecKey: AVVideoCodecType.hevc,
    AVVideoWidthKey: 1920,
    AVVideoHeightKey: 1080,
    AVVideoCompressionPropertiesKey: [
        AVVideoAverageBitRateKey: 50_000_000,
        AVVideoProfileLevelKey: AVVideoProfileLevelH264HighAutoLevel
    ]
]
let dualVideoInput = AVAssetWriterInput(mediaType: .video,
                                        outputSettings:
        dualVideoSettings)
dualViewWriter.add(dualVideoInput)

// 2. Front Only Writer
let frontWriter = try AVAssetWriter(url: frontURL, fileType: .mov)
let frontVideoInput = AVAssetWriterInput(mediaType: .video,
                                         outputSettings:
        dualVideoSettings)
frontWriter.add(frontVideoInput)

// 3. Back Only Writer
let backWriter = try AVAssetWriter(url: backURL, fileType: .mov)
let backVideoInput = AVAssetWriterInput(mediaType: .video,
                                        outputSettings:
        dualVideoSettings)
backWriter.add(backVideoInput)

// Start all writers
dualViewWriter.startWriting()
```

```
frontWriter.startWriting()
backWriter.startWriting()

// In sample buffer delegate:
func captureOutput(_ output: AVCaptureOutput,
                   didOutput sampleBuffer: CMSampleBuffer,
                   from connection: AVCaptureConnection) {

    // Identify camera
    if output == backVideoOutput {
        // Write to back writer
        if backVideoInput.isReadyForMoreMediaData {
            backVideoInput.append(sampleBuffer)
        }

        // Store for compositing
        backFrameBuffer = sampleBuffer

    } else if output == frontVideoOutput {
        // Write to front writer
        if frontVideoInput.isReadyForMoreMediaData {
            frontVideoInput.append(sampleBuffer)
        }

        // Store for compositing
        frontFrameBuffer = sampleBuffer
    }

    // If both frames available, composite
    if let backFrame = backFrameBuffer,
       let frontFrame = frontFrameBuffer {
        let composited = compositeFrames(front: frontFrame, back:
        backFrame)
        if dualVideoInput.isReadyForMoreMediaData {
            dualVideoInput.append(composited)
        }
        backFrameBuffer = nil
        frontFrameBuffer = nil
    }
}
```

Metal Compositing Shader

```
// StackedCameraCompositor.metal
#include <metal_stdlib>
using namespace metal;

struct VertexOut {
```

```
    float4 position [[position]];
    float2 texCoord;
};

fragment float4 stackedCameraFragment(
    VertexOut in [[stage_in]],
    texture2d<float> frontTexture [[texture(0)]],
    texture2d<float> backTexture [[texture(1)]]
) {
    constexpr sampler textureSampler(mag_filter::linear,
min_filter::linear);

    // Top half = front camera
    if (in.texCoord.y < 0.5) {
        float2 adjustedCoord = float2(in.texCoord.x, in.texCoord.y *
2.0);
        return frontTexture.sample(textureSampler, adjustedCoord);
    }
    // Bottom half = back camera
    else {
        float2 adjustedCoord = float2(in.texCoord.x, (in.texCoord.y
- 0.5) * 2.0);
        return backTexture.sample(textureSampler, adjustedCoord);
    }
}
```

Hardware Cost Monitoring

```
actor HardwareCostMonitor {
    private let session: AVCaptureMultiCamSession
    private let maxCost: Float = 0.95

    func monitorCost() async {
        while session.isRunning {
            let currentCost = session.hardwareCost

            if currentCost > maxCost {
                await adjustSettings()
            }

            try? await Task.sleep(nanoseconds: 1_000_000_000) //
        Check every 1s
        }
    }

    private func adjustSettings() async {
        // 1. Try enabling sensor binning
        // 2. Reduce frame rate
```

```
        // 3. Reduce resolution
        // 4. Disable stabilization
    }
}
```

Error Handling & Edge Cases

Error Categories

1. Permission Errors

- Camera access denied
- Microphone access denied
- Photos library access denied

Handling: Show alert with link to Settings

2. Hardware Errors

- Multi-cam not supported
- Hardware cost exceeded
- Thermal throttling
- Camera unavailable

Handling: Graceful degradation, user notification

3. Recording Errors

- Insufficient storage
- Write failure
- Audio sync lost
- Frame dropped

Handling: Pause recording, notify user, attempt recovery

4. System Interruptions

- Phone call
- FaceTime call
- Alarm
- Background transition

Handling: Pause recording, resume when possible

Recovery Strategies

```
enum RecordingError: Error {
    case insufficientStorage
```

```
    case writerFailed(underlying: Error)
    case hardwareCostExceeded
    case thermalThrottling
}

func handleRecordingError(_ error: RecordingError) async {
    switch error {
    case .insufficientStorage:
        // Stop recording, save what we have
        await stopRecording()
        await showAlert("Storage full", "Your recording was saved up
        to this point.")

    case .writerFailed(let underlyingError):
        // Attempt to recover
        if attemptCount < 3 {
            await retryRecording()
        } else {
            await showAlert("Recording failed", "Please try again.")
        }

    case .hardwareCostExceeded:
        // Reduce quality
        await reduceRecordingQuality()
        await showToast("Quality reduced to maintain stability")

    case .thermalThrottling:
        // Pause recording
        await pauseRecording()
        await showAlert("Device overheating", "Recording paused to
        cool down.")
    }
}
```

Performance Optimization

Targets

- App launch: <2 seconds cold, <1 second warm
- Camera preview: <500ms to first frame
- Recording start: <300ms delay
- UI responsiveness: 60fps constant
- Memory usage: <200MB during recording
- Battery drain: <15% per hour of 4K recording

Optimization Strategies

1. Lazy Loading

- Load camera only when needed
- Lazy initialize Metal pipeline
- Background load gallery thumbnails

2. Memory Management

- Release sample buffers immediately after processing
- Pool CVPixelBuffers for reuse
- Limit preview resolution for UI

3. Thread Management

- Camera operations on dedicated queue
- UI updates on MainActor
- Background processing for encoding

4. Battery Optimization

- Use sensor binning when possible
- Reduce preview frame rate (15fps for preview, 30/60 for recording)
- Disable unused features

Testing Strategy

Unit Tests

- CameraService logic
- RecordingService state management
- SettingsManager persistence
- Format selection algorithms

Integration Tests

- Camera session setup
- Recording workflow
- Photos library integration
- Error recovery

UI Tests

- Recording flow (start → pause → resume → stop)
- Gallery navigation
- Settings changes
- Permission prompts

Performance Tests

- Hardware cost under various configurations
- Memory usage during extended recording
- Frame drop rate
- Battery consumption

Device Testing Matrix

Device	iOS	Resolution	Frame Rate	Status
iPhone XS	18.0	1080p	30fps	✓ Pass
iPhone 11	18.0	4K	30fps	✓ Pass
iPhone 13 Pro	18.1	4K	60fps	✓ Pass
iPhone 15 Pro	18.1	4K	60fps	✓ Pass

Privacy & Security

Data Collection

App collects ZERO personal data: - No analytics - No tracking - No user accounts - No cloud storage

Permissions

- Camera: Required for recording
- Microphone: Required for audio
- Photos: Required for saving recordings

Data Storage

- All recordings stored locally
- User controls Photos library integration
- Settings stored in app sandbox
- No data leaves device

Timeline & Milestones

Phase 1: Foundation (Week 1-2)

- ✓ Project setup
- ✓ Basic camera preview
- ✓ AVCaptureMultiCamSession integration
- ✓ Liquid glass UI components

Phase 2: Core Recording (Week 3-4)

- ✓ Triple output recording
- ✓ Metal compositing
- ✓ Pause/resume functionality
- ✓ Hardware cost monitoring

Phase 3: Camera Features (Week 5-6)

- ✓ Zoom controls
- ✓ Focus controls
- ✓ Exposure controls
- ✓ White balance
- ✓ Stabilization

Phase 4: Gallery & Playback (Week 7)

- ✓ Gallery view
- ✓ Video playback
- ✓ Photos library integration
- ✓ Metadata management

Phase 5: Settings & Polish (Week 8)

- ✓ Settings screen
- ✓ Error handling
- ✓ Accessibility
- ✓ Performance optimization

Phase 6: Testing & Release (Week 9-10)

- ✓ Comprehensive testing
- ✓ Bug fixes
- ✓ Documentation
- ✓ App Store submission

Future Enhancements (Post-Launch)

v1.1

- ☐ Live streaming support
- ☐ Social media direct upload
- ☐ More video filters
- ☐ Custom layouts (PiP, side-by-side)

v1.2

- ☐ Multi-device sync
- ☐ External camera support
- ☐ Green screen mode
- ☐ Advanced color grading

v1.3

- ☐ AI-powered features (face tracking, auto-framing)
- ☐ Live captions
- ☐ Real-time effects
- ☐ Collaboration mode

v2.0

- ☐ iPad optimization
- ☐ Mac Catalyst version
- ☐ Vision Pro support
- ☐ Professional suite (Final Cut integration)

Competitive Analysis

vs Mixcam

Advantages: - ✓ Three output files vs two - ✓ Better performance (iOS 18 native) - ✓ More professional controls - ✓ No recording time limits - ✓ Better UI (liquid glass)

Disadvantages: - ✗ Mixcam has established user base - ✗ More layout options

vs DoubleTake

Advantages: - ✓ Better resolution support (4K vs 1080p) - ✓ More reliable (fewer crashes) - ✓ Modern Swift 6 codebase - ✓ Active development

Disadvantages: - ✗ DoubleTake is free by Filmic Pro (brand recognition)

Key Differentiators

1. **Triple Output**: Unique feature
2. **iOS 18+ Native**: Latest APIs
3. **Liquid Glass**: Modern design
4. **Reliability**: Production-ready from day 1
5. **Performance**: Optimized for latest hardware

Appendix

A. Technical Requirements Summary

- Swift 6
- iOS 18.0+
- Xcode 16+
- iPhone XS/11+ (A12 Bionic+)
- Multi-cam support required

B. Third-Party Dependencies

NONE - Pure Apple frameworks: - AVFoundation - SwiftUI - PhotoKit - Metal - SwiftData

C. File Structure

```
DualCameraApp/
├── App/
│   ├── DualCameraApp.swift
│   └── ContentView.swift
├── Views/
│   ├── Camera/
│   ├── Gallery/
│   ├── Settings/
│   └── Components/
├── ViewModels/
│   ├── CameraViewModel.swift
│   ├── GalleryViewModel.swift
│   └── SettingsViewModel.swift
├── Services/
│   ├── CameraService.swift
│   ├── RecordingService.swift
│   └── PhotoLibraryService.swift
├── Models/
│   ├── Recording.swift
│   ├── CameraSettings.swift
│   └── Enums.swift
├── Utilities/
│   ├── Extensions/
│   └── Helpers/
├── Resources/
│   ├── Assets.xcassets
│   └── Shaders/
└── Tests/
    ├── UnitTests/
```

```
└── UITests/
```

D. Glossary

- **AVCaptureMultiCamSession**: Apple's API for multi-camera capture
- **Glassmorphism**: UI design style with translucent glass effects
- **Hardware Cost**: AVFoundation metric for system resource usage
- **ProRes**: Apple's professional video codec
- **Stacked Layout**: Vertical camera arrangement (top/bottom)
- **Triple Output**: Three simultaneous video recordings

Sign-Off

Product Owner: _____
Technical Lead: _____
Design Lead: _____
Date: October 23, 2025

Document Version: 1.0
Last Updated: October 23, 2025
Next Review: November 23, 2025