# DualCam: Product Requirements Document

**Document Version:** 1.0
**Date:** October 24, 2025
**Product Name:** DualCam
**Platform:** iOS 18+
**Technology Stack:** Swift 6, AVFoundation, SwiftUI

---

## Executive Summary

DualCam is a next-generation dual camera iOS application designed to capture simultaneous front and back camera footage with a premium liquid glass aesthetic. The app leverages iOS 18's latest AVFoundation multi-camera APIs to deliver professional-grade dual perspective recording while maintaining an intuitive, consumer-friendly interface.

### Vision Statement

To provide content creators, vloggers, and everyday users with the most powerful, reliable, and beautifully designed dual camera recording experience on iOS, enabling authentic storytelling through multi-perspective capture.

### Business Objectives

1. **Market Differentiation**: Stand out from competitors (Mixcam, DoubleTake) through superior reliability, design, and feature completeness
2. **User Delight**: Deliver a premium, Apple-native experience with liquid glass design language
3. **Technical Excellence**: Leverage cutting-edge iOS 18+ and Swift 6 capabilities
4. **Flexible Output**: Provide three recording outputs (front, back, combined) for maximum editing flexibility

### Success Metrics

- **Technical Performance**:
- App crash rate < 0.1%
- 99% successful save rate to Photos library
- < 5% hardware cost threshold for multi-cam recording
- Thermal management prevents overheating during 10+ minute recordings
- **User Engagement**:
- Daily active users retention > 40% after 30 days
- Average session length > 5 minutes
- Feature adoption rate > 60% for advanced controls
- **App Store Performance**:
- Target 4.5+ star rating
- Featured in App Store "New Apps We Love"

- Top 10 in Photo & Video category within 3 months

---

## Product Overview

### Product Description

DualCam is a professional-quality dual camera recording application that enables users to simultaneously capture video from both front and back cameras on their iPhone. The app produces three distinct outputs: a front-only video, a back-only video, and a combined picture-in-picture or split-screen video, all saved directly to the Photos library.

### Target Platform

- **iOS Version**: 18.0 or later
- **Device Requirements**:
- iPhone XS, XS Max, XR and later models
- A12 Bionic chip or newer
- Multi-camera support capability (verified via `AVCaptureMultiCamSession.isMultiCamSupported`)
- Minimum 4GB RAM
- 100MB available storage (excluding recorded videos)

### Design Philosophy

DualCam embraces Apple's liquid glass/glassmorphism design language, featuring:
- Semi-transparent UI elements with background blur
- Soft, glowing borders on interactive components
- Layered depth hierarchy
- Smooth animations and transitions
- Dark mode optimized
- Accessibility-first approach

---

## Target Audience

### Primary Personas

#### 1. Content Creator Clara (Age 22-35)

- **Background**: Social media influencer, YouTuber, TikToker
- **Goals**: Create engaging reaction videos, vlogs, and dual-perspective content
- **Pain Points**: Existing apps crash, have recording limits, poor quality
- **Needs**: Reliable recording, high quality output, easy sharing to social media
- **Tech Savvy**: High - understands video settings and formats

#### 2. Interview Ian (Age 28-45)

- **Background**: Journalist, podcaster, event organizer
- **Goals**: Capture interviews with simultaneous host and guest perspectives
- **Pain Points**: Needs professional reliability, doesn't want to learn complex tools
- **Needs**: Easy operation, multiple output files for editing, good audio
- **Tech Savvy**: Medium - familiar with iOS but not video production

#### 3. Vlogger Vanessa (Age 18-28)

- **Background**: College student, travel enthusiast, lifestyle vlogger
- **Goals**: Document experiences with authentic front+back perspective
- **Pain Points**: Limited budget, wants professional results without expensive gear
- **Needs**: Fun UI, good stabilization, filters and effects
- **Tech Savvy**: Medium-high - comfortable with iPhone features

#### 4. Parent Paul (Age 35-50)

- **Background**: Family documenter, captures kids' moments
- **Goals**: Capture special moments while also capturing family reactions
- **Pain Points**: Doesn't want complexity, just wants it to work
- **Needs**: Simple interface, automatic settings, easy sharing
- **Tech Savvy**: Low-medium - uses iPhone casually

### Secondary Personas

- Educators creating instructional content
- Fitness instructors recording workout tutorials
- Musicians capturing performance and audience simultaneously
- Event photographers offering dual perspective coverage

---

## Feature Specifications

### 1. Core Camera Functionality

#### 1.1 Multi-Camera Capture System

**Priority:** P0 (Critical)

**Description:** Simultaneous capture from front and back cameras using AVCaptureMultiCamSession.

**Technical Requirements:**
- Implement AVCaptureMultiCamSession with manual connection management
- Configure separate AVCaptureDeviceInput for front and back cameras
- Use `addInputWithNoConnections()` and `addOutputWithNoConnections()` methods
- Create explicit AVCaptureConnection instances for each camera-output pair
- Monitor hardware cost to stay below 1.0 threshold
- Implement dynamic quality adjustment if approaching hardware limits

**User Stories:**
- As a user, I want to see both front and back camera previews simultaneously
- As a user, I want smooth, synchronized recording from both cameras
- As a user, I want the app to handle hardware limitations gracefully

**Acceptance Criteria:**
- ✅ Both camera feeds display in real-time with < 100ms latency
- ✅ Recording starts and stops simultaneously on both cameras
- ✅ Hardware cost monitoring prevents session failure
- ✅ Graceful fallback to single camera if multi-cam unavailable
- ✅ Support for all camera combinations (wide, ultra-wide, telephoto)

---

## 1.2 Three-Output Recording System

**Priority:** P0 (Critical)

**Description:** Simultaneously record and save three distinct video files: front-only, back-only, and combined view.

**Technical Requirements:**
- Create three separate AVCaptureVideoDataOutput instances:
1. Front camera output → front_only.mov
2. Back camera output → back_only.mov
3. Combined output → combined.mov (composited using Metal)
- Use AVAssetWriter for each output to write to separate file URLs
- Implement Metal shader for real-time compositing of combined view
- Ensure frame synchronization across all three outputs
- Handle different frame rates/resolutions between cameras gracefully

**Recording Outputs:**

**Output 1: Front Camera Only**
- Full resolution front camera video
- File naming: `DualCam_Front_YYYYMMDD_HHMMSS.mov`
- Independent of combined view composition

**Output 2: Back Camera Only**
- Full resolution back camera video
- File naming: `DualCam_Back_YYYYMMDD_HHMMSS.mov`
- Independent of combined view composition

**Output 3: Combined View**
- Picture-in-picture or split-screen composite
- User-selectable layout (see Layout Options below)
- File naming: `DualCam_Combined_YYYYMMDD_HHMMSS.mov`
- Includes both camera feeds in single frame

**User Stories:**
- As a video editor, I want separate files for each camera to edit independently
- As a content creator, I want a ready-to-share combined video immediately
- As a user, I want all three files saved automatically without extra steps

**Acceptance Criteria:**
- ✅ Three distinct video files created for each recording session
- ✅ All files saved to Photos library automatically
- ✅ Files maintain frame synchronization (< 33ms drift)
- ✅ No data loss or corruption in any output
- ✅ Combined view matches real-time preview exactly
- ✅ All files include proper metadata (creation date, location, etc.)

---

## 1.3 Combined View Layout Options

**Priority:** P0 (Critical)

**Description:** Multiple composition layouts for the combined output.

**Available Layouts:**

1. **Picture-in-Picture (PiP)**
   - Large background: Back camera (full screen)
   - Small overlay: Front camera (moveable, resizable)
   - PiP positions: 4 corners + center
   - PiP sizes: Small (15%), Medium (25%), Large (40%)
   - Rounded corners with liquid glass border

2. **Split-Screen Vertical**
   - Top: Front camera (50% height)
   - Bottom: Back camera (50% height)
   - Thin divider line with glass effect

3. **Split-Screen Horizontal**
   - Left: Front camera (50% width)
   - Right: Back camera (50% width)
   - Thin divider line with glass effect

4. **Background Dominance**
   - Back camera: 70% of screen
   - Front camera: 30% of screen (bottom overlay)
   - Preserves more of back camera view

5. **Equal Focus**
   - Both cameras: 50% each
   - Side-by-side or top-bottom
   - Equal priority for both perspectives

**Interactive Controls:**
- Tap layout icon to cycle through options
- Drag PiP window to reposition during preview
- Pinch PiP window to resize
- Layout settings persist between sessions

**Acceptance Criteria:**
- ✅ All 5 layouts available and functional
- ✅ Layout changes reflected in real-time preview
- ✅ PiP window draggable to any screen position
- ✅ Layout changes during recording applied to subsequent frames
- ✅ Settings saved and restored on app launch

---

## 2. Camera Preview Interface

### 2.1 Stacked Dual Preview

**Priority:** P0 (Critical)

**Description:** Real-time preview of both camera feeds before and during recording.

**Layout:**
- **Primary View**: Larger camera feed (user-selectable: front or back)
- **Secondary View**: Smaller overlay or split-screen
- **Match Combined Output**: Preview reflects selected layout
- **Real-time Updates**: < 60ms latency from sensor to screen

**Technical Implementation:**
- Use AVCaptureVideoPreviewLayer for each camera
- Composite using Metal for combined preview
- Maintain 60fps preview when possible
- Adaptive quality based on device capabilities

**User Stories:**
- As a user, I want to see exactly what will be recorded
- As a user, I want smooth, lag-free preview
- As a creator, I want to frame both shots before recording

**Acceptance Criteria:**
- ✅ Both camera feeds visible simultaneously
- ✅ Preview matches final recording output
- ✅ Smooth 60fps preview on supported devices
- ✅ Preview maintains aspect ratio correctly
- ✅ No black bars or distortion

---

## 3. Recording Controls

### 3.1 Record Button

**Priority:** P0 (Critical)

**Design:**
- Large circular button (84pt diameter)
- Center bottom of screen
- Liquid glass background with white/red fill
- State indicators:
- **Ready**: White fill, glass border
- **Recording**: Red fill, pulsing animation
- **Processing**: Loading spinner

**Behavior:**
- Single tap: Start/stop recording
- Long press: Reserved for future quick settings
- Haptic feedback on tap
- Visual state transitions (smooth animations)

**Recording States:**
- **Idle**: Ready to record
- **Recording**: Active recording with timer display
- **Processing**: Saving files to library
- **Success**: Visual confirmation of save
- **Error**: Clear error message with retry option

**Acceptance Criteria:**
- ✅ Immediate response to tap (< 100ms)
- ✅ Clear visual feedback for all states
- ✅ Haptic feedback for state changes
- ✅ Recording timer visible during capture
- ✅ Cancel/stop always available

---

### 3.2 Recording Timer

**Priority:** P0 (Critical)

**Display:**
- HH:MM:SS format for recordings > 1 hour
- MM:SS format for recordings < 1 hour
- SS format for first minute
- Position: Top center of screen
- Liquid glass background
- Pulsing red dot indicator

**Features:**
- Real-time update every second
- No recording time limit (only storage dependent)
- Battery level warning when < 20%
- Storage warning when < 500MB available
- Temperature warning if device overheating

**Acceptance Criteria:**
- ✅ Timer accurate to ±1 second
- ✅ Timer visible in all screen orientations
- ✅ Red recording indicator pulsing
- ✅ Warning indicators don’t obstruct view
- ✅ Timer resets properly between recordings

---

## 4. Advanced Camera Features

### 4.1 Independent Zoom Controls

**Priority:** P0 (Critical)

**Description:** Separate zoom control for front and back cameras.

**Implementation:**
- **Pinch Gesture**: Zoom active camera (indicated by border highlight)
- **Zoom Slider**: Dedicated control with camera toggle
- **Preset Buttons**: 0.5x, 1x, 2x, 5x (device dependent)
- **Zoom Range**: 0.5x to 10x digital zoom
- **Smooth Transitions**: Animated zoom changes

**UI Elements:**
- Zoom level indicator (e.g., “2.0x”)
- Camera toggle button to switch zoom target

- Slider for fine-grained control
- Quick preset tap buttons

**Technical Requirements:**
- Use `AVCaptureDevice.videoZoomFactor` for zoom
- Respect `activeFormat.videoMaxZoomFactor` limits
- Smooth zoom ramp with configurable duration
- Independent zoom state for each camera

**User Stories:**
- As a user, I want to zoom in on my back camera without affecting front camera
- As a creator, I want quick zoom presets for common focal lengths
- As a user, I want smooth zoom transitions during recording

**Acceptance Criteria:**
- ✅ Independent zoom for front and back cameras
- ✅ Pinch-to-zoom gesture works intuitively
- ✅ Zoom level indicator always visible
- ✅ Preset buttons work instantly
- ✅ Zoom state persists between recordings

---

### 4.2 Video Quality Settings

**Priority:** P1 (High)

**Resolution Options:**
- **720p HD** (1280x720) - Basic quality, smaller file size
- **1080p Full HD** (1920x1080) - Standard quality (default)
- **4K UHD** (3840x2160) - Maximum quality (if hardware cost allows)

**Frame Rate Options:**
- **24fps** - Cinematic look
- **30fps** - Standard video (default)
- **60fps** - Smooth motion (if supported)

**Codec Options:**
- **H.264** - Universal compatibility (default)
- **H.265/HEVC** - Better compression, smaller files
- **ProRes** - Professional editing (future enhancement)

**Quality Presets:**
- **High Efficiency**: 1080p, 30fps, H.265
- **Balanced**: 1080p, 30fps, H.264 (default)
- **Maximum Quality**: 4K, 60fps, H.265
- **Social Media**: 1080p, 30fps, H.264, optimized bitrate

**UI:**
- Settings panel with liquid glass design
- Visual previews of quality differences
- File size estimates for each option
- Real-time hardware cost indicator

**Acceptance Criteria:**
- ✅ All resolution options functional
- ✅ Frame rate changes applied correctly
- ✅ Codec selection works as expected
- ✅ Presets provide sensible defaults
- ✅ Hardware limitations communicated clearly

---

### 4.3 Exposure & Focus Controls

**Priority:** P1 (High)

**Exposure Controls:**
- **Auto Exposure (AE)**: Default automatic mode
- **AE Lock**: Tap-and-hold to lock exposure
- **Exposure Compensation**: ±3 EV slider adjustment
- **Manual Exposure**:
- ISO range: 29-2000 (device dependent)
- Shutter speed: 1/8000s to 1/3s
- **Exposure Indicator**: Visual meter showing current level

**Focus Controls:**
- **Auto Focus (AF)**: Continuous autofocus (default)
- **Tap-to-Focus**: Tap on preview to focus
- **AF Lock**: Lock focus at current distance
- **Manual Focus**: Slider for lens position (0.0-1.0)
- **Focus Peaking**: Highlight in-focus areas (optional)

**UI Design:**
- Tap preview: Show AF/AE reticle
- Swipe up/down on reticle: Adjust exposure compensation
- Settings panel: Manual controls
- Yellow square: AF/AE target indicator
- Lock icon: Indicates locked state

**Technical Implementation:**

```
// Exposure control
device.exposureMode = .continuousAutoExposure
device.setExposureTargetBias(compensation) { _ in }

// Focus control
device.focusMode = .continuousAutoFocus
device.focusPointOfInterest = CGPoint(x: 0.5, y: 0.5)
```

**Acceptance Criteria:**
- ✅ Tap-to-focus works instantly
- ✅ Exposure compensation visible in real-time
- ✅ Manual controls provide precise adjustment
- ✅ AF/AE locks function properly
- ✅ Reticle disappears after 3 seconds

---

### 4.4 Flash & Torch Controls

**Priority:** P1 (High)

**Flash Modes:**
- **Off**: No flash
- **On**: Always fire flash
- **Auto**: Flash based on ambient light

**Torch Mode (Video Light):**
- **Off**: No torch
- **On**: Continuous light during recording
- **Auto**: Torch based on scene brightness
- **Brightness Levels**: 0-100% adjustable

**UI:**
- Flash/torch icon button (top-left of preview)
- States cycle: Off → Auto → On
- Icon changes color to indicate state
- Torch brightness slider (when torch enabled)

**Technical Requirements:**
- Check `device.hasTorch` and `device.hasFlash`
- Use `device.torchMode` for video recording
- Adjust `device.setTorchModeOn(level:)` for brightness
- Monitor battery impact (show warning if < 20%)

**Acceptance Criteria:**
- ✅ Flash modes work for compatible cameras
- ✅ Torch enables smoothly during recording
- ✅ Brightness adjustment visible in real-time
- ✅ Low battery warning shown if using torch
- ✅ Graceful handling of non-torch devices

---

### 4.5 Video Stabilization

**Priority:** P1 (High)

**Stabilization Modes:**
- **Off**: No stabilization
- **Standard**: Basic software stabilization (default)
- **Enhanced**: Advanced stabilization
- **Cinematic**: Smooth, film-like motion
- **Action Mode**: Maximum stability for high motion

**Implementation:**

```
if connection.isVideoStabilizationSupported {
    connection.preferredVideoStabilizationMode = .cinematicExtended
}
```

**UI:**
- Settings panel option
- Visual icon indicating active mode
- Real-time preview of stabilization effect
- Warning if mode not supported

**Acceptance Criteria:**
- ✅ All modes functional on supported devices
- ✅ Stabilization visible in preview
- ✅ Clear indication of active mode
- ✅ Performance impact minimal
- ✅ Fallback for unsupported modes

---

### 4.6 Audio Settings

**Priority:** P1 (High)

**Audio Features:**
- **Microphone Selection**:
- Built-in microphone (front, back, bottom)
- External microphone (Lightning/USB-C)
- Bluetooth microphone
- **Audio Monitoring**: Real-time level meters
- **Manual Gain Control**: Adjust input level (-12dB to +12dB)
- **Wind Noise Reduction**: Toggle on/off
- **Audio Format**:
- Sample rate: 48kHz (default), 44.1kHz
- Bit depth: 16-bit, 24-bit
- Channels: Mono, Stereo

**UI:**
- Audio settings in settings panel
- Level meters during recording
- Microphone icon indicates active input
- Mute toggle (video only recording)

**Technical Implementation:**

```
let audioDevice = AVCaptureDevice.default(for: .audio)
let audioInput = try AVCaptureDeviceInput(device: audioDevice!)
session.addInput(audioInput)
```

**Acceptance Criteria:**
- ✅ Audio levels display in real-time
- ✅ External microphone detected automatically
- ✅ Wind noise reduction audible effect
- ✅ Gain control functional
- ✅ Mute option works correctly

---

### 4.7 Grid Overlays

**Priority:** P2 (Medium)

**Grid Types:**
- **None**: No grid (default for beginners)
- **Rule of Thirds**: 3x3 grid (default)
- **Golden Ratio**: Fibonacci spiral
- **Center Cross**: Crosshair alignment
- **Square**: 1:1 aspect ratio guide

**UI:**
- Toggle in quick settings
- Semi-transparent overlay (20% opacity)
- Liquid glass aesthetic
- Doesn't appear in final video

**Acceptance Criteria:**
- ✅ Grid visible in preview only
- ✅ All grid types render correctly
- ✅ Grid doesn't impact performance
- ✅ Preference saved between sessions
- ✅ Grid adapts to screen orientation

---

### 4.8 Timer & Self-Timer

**Priority:** P2 (Medium)

**Timer Options:**
- **Off**: Immediate recording
- **3 seconds**: Short delay
- **10 seconds**: Standard delay
- **Custom**: User-defined (5-60 seconds)

**Countdown Display:**
- Large numbers in center of screen
- Countdown audio beeps
- Visual countdown animation
- Cancel option during countdown

**UI:**
- Timer icon button in controls
- Shows selected delay time
- Countdown overlay during countdown

**Acceptance Criteria:**
- ✅ Timer countdown accurate
- ✅ Audio beeps at 3, 2, 1
- ✅ Cancel button functional
- ✅ Recording starts automatically after countdown
- ✅ Timer setting persists

---

### 4.9 Filters & Effects

**Priority:** P2 (Medium)

**Available Filters:**
- **None**: Natural color
- **Vivid**: Increased saturation
- **Dramatic**: High contrast
- **Cool**: Blue tint
- **Warm**: Orange/yellow tint
- **B&W**: Black and white
- **Sepia**: Vintage tone
- **Cinematic**: Film-like grade

**Real-time Application:**
- Filters applied during recording
- Preview shows filter effect
- Metal shaders for performance
- Adjustable filter intensity (0-100%)

**UI:**
- Horizontal filter carousel
- Thumbnail previews of each filter
- Live preview when swiping
- Filter name displayed

**Acceptance Criteria:**
- ✅ All filters render in real-time
- ✅ No performance degradation
- ✅ Filters apply to combined output
- ✅ Separate files unaffected by filters
- ✅ Filter choice saved between sessions

---

## 5. User Interface Design

### 5.1 Liquid Glass Design System

**Priority:** P0 (Critical)

**Design Principles:**
- **Translucency**: 20-40% opacity for glass elements
- **Background Blur**: 20-30px blur radius
- **Borders**: 1px soft borders with white 40% opacity
- **Shadows**: Soft shadows (radius 10-20px, 20-30% opacity)
- **Colors**:
- Light mode: White with subtle blue tint
- Dark mode: Dark gray with subtle purple tint
- **Typography**: San Francisco (system font)
- Regular weight for body
- Semibold for emphasis
- Bold for headers

**SwiftUI Materials:**

```
.background(.ultraThinMaterial)  // Lightest glass
.background(.thinMaterial)       // Light glass
.background(.regularMaterial)    // Standard glass
.background(.thickMaterial)      // Heavy glass
```

**Component Library:**
- **GlassButton**: Circular buttons with glass background
- **GlassPanel**: Rectangular panels for settings/info
- **GlassSlider**: Slider controls with glass track
- **GlassCard**: Content cards with glass effect
- **GlassBadge**: Small indicators and labels

**Acceptance Criteria:**
- ✅ Consistent glass aesthetic throughout app
- ✅ Respects accessibility settings (reduce transparency)
- ✅ Performs smoothly on all supported devices
- ✅ Adapts to light and dark mode
- ✅ Maintains readability (WCAG AA contrast)

---

### 5.2 Main Recording Screen

**Priority:** P0 (Critical)

**Layout:**

**Top Bar (Floating):**
- Status indicators (top-left to top-right):
- Settings gear icon
- Flash/torch toggle
- Timer display (when recording)
- Battery indicator
- Close/back button

**Center:**
- Full-screen camera preview
- AF/AE reticle (when active)
- Grid overlay (if enabled)
- Filter preview

**Bottom Bar (Floating):**
- Gallery thumbnail (bottom-left)
- Record button (bottom-center)
- Camera flip button (bottom-right)
- Zoom controls (when pinching or tapping zoom)

**Settings Panel (Slide-up):**
- Swipe up from bottom to reveal
- Glass panel with controls:
- Resolution & frame rate

- Layout options
- Exposure & focus
- Audio settings
- Filters
- Grid toggle

**Visual Design:**
- Full bleed camera preview
- Floating glass controls don't obstruct view
- Smooth animations (0.3s ease-in-out)
- Auto-hide controls after 3s of inactivity
- Tap screen to show controls again

**Acceptance Criteria:**
- ✅ All controls accessible within 2 taps
- ✅ Preview uses 100% of screen
- ✅ Controls auto-hide during recording
- ✅ Settings panel smooth slide animation
- ✅ Touch targets minimum 44x44pt

---

### 5.3 Settings/Configuration Screen

**Priority:** P1 (High)

**Categories:**

**Video Settings:**
- Resolution (720p, 1080p, 4K)
- Frame rate (24, 30, 60fps)
- Codec (H.264, H.265)
- Bitrate (Low, Medium, High, Maximum)

**Audio Settings:**
- Microphone selection
- Sample rate
- Wind noise reduction
- Audio monitoring

**Recording Settings:**
- Layout default
- Auto-save to library
- File naming convention
- Maximum recording duration

**Interface Settings:**
- Grid type
- Control auto-hide duration
- Haptic feedback toggle
- Sound effects toggle

**About:**
- App version

- Device compatibility info
- Privacy policy
- Support/feedback

**UI Design:**
- Standard iOS Settings-style interface
- Grouped table view sections
- Glass-styled cells
- Inline pickers and toggles
- Immediate preview of changes

**Acceptance Criteria:**
- ✅ All settings accessible
- ✅ Changes save immediately
- ✅ Settings persist between launches
- ✅ Clear descriptions for each option
- ✅ Restore defaults option available

---

### 5.4 Gallery/Library View

**Priority:** P1 (High)

**Features:**
- Thumbnail grid of recorded videos
- Group by recording session (3 files per session)
- Preview thumbnails for each file type
- Metadata display (date, duration, file size, resolution)
- Multi-select for batch operations

**Actions:**
- **Tap thumbnail**: Preview video in full screen
- **Long press**: Show context menu
- Share
- Delete
- Rename
- Export
- View details
- **Swipe**: Quick delete

**Batch Operations:**
- Select multiple sessions
- Delete selected
- Share selected
- Export selected

**UI Design:**
- Grid layout (2-3 columns)
- Glass overlay showing file info
- Combined view thumbnail as primary
- Small badges for front/back files
- Filter by date, duration, resolution

**Acceptance Criteria:**
- ✅ All recorded videos appear in gallery
- ✅ Thumbnails generate quickly
- ✅ Actions perform correctly
- ✅ Batch operations work smoothly
- ✅ Gallery syncs with Photos library

---

## 6. Photos Library Integration

### 6.1 Automatic Save to Photos

**Priority:** P0 (Critical)

**Behavior:**
- All recordings automatically saved to Photos library
- Three separate video assets created per recording
- Assets grouped in "DualCam" album
- Metadata included (date, location, camera model)

**Implementation:**

```
import Photos

PHPhotoLibrary.requestAuthorization { status in
    if status == .authorized {
        PHPhotoLibrary.shared().performChanges({
            PHAssetChangeRequest.creationRequestForAssetFromVideo(atFileURL: videoURL)
        })
    }
}
```

**Album Organization:**
- Custom "DualCam" album created automatically
- Recordings also appear in "Recents"
- Smart album creation (by date, camera used)
- Proper asset collections API usage

**User Stories:**
- As a user, I want my recordings saved automatically
- As a user, I want to find my recordings in the Photos app
- As a user, I want my videos organized in a dedicated album

**Acceptance Criteria:**
- ✅ All three videos saved for each recording
- ✅ Save success rate > 99%
- ✅ Custom album created on first launch
- ✅ Metadata preserved (EXIF, GPS if enabled)
- ✅ No duplicate assets created

---

### 6.2 Permissions Handling

**Priority:** P0 (Critical)

**Required Permissions:**
- **Camera**: Required for video capture
- **Microphone**: Required for audio recording
- **Photos Library**: Required for saving videos

**Permission Flow:**
1. **First Launch**: Request all permissions upfront
2. **Permission Denied**: Show educational screen explaining why needed
3. **Settings Link**: Direct link to iOS Settings if user denied
4. **Graceful Degradation**: Limited functionality if permissions denied

**UI/UX:**
- Clear permission prompts with custom messaging
- Before iOS system prompt, show app screen explaining benefits
- Visual indicators when permissions missing
- In-app settings to review/change permissions
- Never repeatedly prompt (respect user choice)

**Info.plist Entries:**

```xml
<key>NSCameraUsageDescription</key>
<string>DualCam needs camera access to record dual-perspective videos</string>

<key>NSMicrophoneUsageDescription</key>
<string>DualCam needs microphone access to record audio with your videos</string>

<key>NSPhotoLibraryAddUsageDescription</key>
<string>DualCam needs Photos access to save your recorded videos</string>

<key>NSPhotoLibraryUsageDescription</key>
<string>DualCam needs Photos access to show your recorded videos in the gallery</
string>

<key>NSLocationWhenInUseUsageDescription</key>
<string>DualCam can add location data to your videos if you choose</string>
```

**Acceptance Criteria:**
- ✅ All permission prompts clear and friendly
- ✅ Custom messaging before system prompts
- ✅ Settings link functional
- ✅ App doesn’t crash if permissions denied
- ✅ User can change permissions from within app

---

## 7. Technical Architecture

### 7.1 Technology Stack

**Priority:** P0 (Critical)

**Languages & Frameworks:**
- **Swift 6**: Latest Swift with strict concurrency checking

- **SwiftUI**: Modern UI framework
- **UIKit**: Where SwiftUI limitations exist (AVFoundation integration)
- **AVFoundation**: Camera capture and recording
- **Metal**: Real-time video compositing
- **Photos/PhotoKit**: Library integration
- **Combine**: Reactive state management
- **CoreMotion**: Device orientation detection

**Minimum Requirements:**
- iOS 18.0+
- Xcode 16+
- Swift 6 language mode
- Strict concurrency checking enabled

**Project Structure:**

```
DualCam/
├── App/
│   ├── DualCamApp.swift
│   └── AppDelegate.swift (if needed)
├── Core/
│   ├── Camera/
│   │   ├── MultiCameraManager.swift
│   │   ├── CameraConfiguration.swift
│   │   ├── RecordingSession.swift
│   │   └── CameraPermissions.swift
│   ├── Recording/
│   │   ├── RecordingCoordinator.swift
│   │   ├── VideoWriter.swift
│   │   ├── AudioManager.swift
│   │   └── MetalCompositor.swift
│   ├── Storage/
│   │   ├── PhotoLibraryManager.swift
│   │   ├── FileManager+Extensions.swift
│   │   └── RecordingMetadata.swift
│   └── Models/
│       ├── CameraSettings.swift
│       ├── RecordingSettings.swift
│       └── LayoutConfiguration.swift
├── UI/
│   ├── Screens/
│   │   ├── CameraView.swift
│   │   ├── SettingsView.swift
│   │   └── GalleryView.swift
│   ├── Components/
│   │   ├── GlassButton.swift
│   │   ├── GlassPanel.swift
│   │   ├── RecordButton.swift
│   │   ├── ZoomControl.swift
│   │   └── LayoutSelector.swift
│   └── Styles/
│       ├── LiquidGlassModifiers.swift
│       └── ColorPalette.swift
├── ViewModels/
│   ├── CameraViewModel.swift
│   ├── SettingsViewModel.swift
│   └── GalleryViewModel.swift
└── Resources/
    ├── Assets.xcassets
    ├── Info.plist
    └── Localizable.strings
```

**Acceptance Criteria:**
- ✅ Clean architecture with clear separation of concerns
- ✅ Swift 6 concurrency safety compliance
- ✅ MVVM pattern for state management
- ✅ Reusable components throughout app
- ✅ Comprehensive documentation

---

### 7.2 AVCaptureMultiCamSession Implementation

**Priority:** P0 (Critical)

**Architecture Overview:**

```
class MultiCameraManager {
    private let multiCamSession = AVCaptureMultiCamSession()

    // Camera inputs
    private var backCameraInput: AVCaptureDeviceInput?
    private var frontCameraInput: AVCaptureDeviceInput?

    // Video outputs for separate files
    private let backVideoOutput = AVCaptureVideoDataOutput()
    private let frontVideoOutput = AVCaptureVideoDataOutput()

    // For combined view
    private let combinedVideoOutput = AVCaptureVideoDataOutput()

    // Preview layers
    private let backPreviewLayer: AVCaptureVideoPreviewLayer
    private let frontPreviewLayer: AVCaptureVideoPreviewLayer

    func setupMultiCamSession() async throws {
        guard AVCaptureMultiCamSession.isMultiCamSupported else {
            throw CameraError.multiCamNotSupported
        }

        multiCamSession.beginConfiguration()

        // Add cameras without automatic connections
        try addBackCamera()
        try addFrontCamera()

        // Add outputs without automatic connections
        addVideoOutputs()

        // Manually create connections
        createConnections()

        // Monitor hardware cost
        guard multiCamSession.hardwareCost <= 1.0 else {
            throw CameraError.hardwareCostExceeded
        }

        multiCamSession.commitConfiguration()
    }
}
```

**Key Components:**

1. **Session Setup:**
   - Check `isMultiCamSupported` before initialization
   - Use `beginConfiguration()` / `commitConfiguration()` for atomic changes
   - Add inputs with `addInputWithNoConnections()`
   - Add outputs with `addOutputWithNoConnections()`

2. **Manual Connections:**
   - Create explicit connections between input ports and outputs
   - Separate connection for each camera-output pair
   - No connection reuse or fanout

3. **Hardware Cost Management:**
   - Monitor `hardwareCost` property (must stay < 1.0)

   - Dynamically reduce resolution/frame rate if approaching limit
   - Use sensor binning when appropriate
   - Display warning to user if cost high

4. **Format Selection:**
   - Choose compatible formats for both cameras
   - Balance quality vs. hardware cost
   - Respect user's resolution/frame rate preferences
   - Fallback to lower quality if needed

**Acceptance Criteria:**
- ✅ Multi-cam session initializes successfully on compatible devices
- ✅ Hardware cost stays below 1.0
- ✅ Both cameras stream simultaneously
- ✅ Manual connections established correctly
- ✅ Error handling for unsupported devices

---

### 7.3 Metal Compositor for Combined View

**Priority:** P0 (Critical)

**Purpose:**
Real-time compositing of front and back camera feeds into single frame for combined output.

**Metal Pipeline:**

```swift
class MetalCompositor {
    private let device: MTLDevice
    private let commandQueue: MTLCommandQueue
    private let pipelineState: MTLRenderPipelineState

    func composite(
        backFrame: CVPixelBuffer,
        frontFrame: CVPixelBuffer,
        layout: LayoutConfiguration
    ) -> CVPixelBuffer {
        // Create render pass
        // Apply layout transformations
        // Composite frames
        // Return composited buffer
    }
}
```

**Shader Implementation:**
- Vertex shader for layout positioning
- Fragment shader for blending and effects
- Support for PiP, split-screen, and custom layouts
- Glass borders and shadows rendered in shader
- Efficient GPU-based processing

**Layout Rendering:**
- **PiP**: Scale and position small frame over large frame
- **Split-Screen**: Crop and position both frames side-by-side

- **Borders**: Render liquid glass borders using alpha blending
- **Shadows**: Apply soft shadows to small frame

**Performance:**
- Target 30fps minimum for composition
- Optimize shader for real-time performance
- Use texture caching for efficiency
- Asynchronous command buffer execution

**Acceptance Criteria:**
- ✅ Composites 30+ fps consistently
- ✅ All layouts render correctly
- ✅ No visible artifacts or tearing
- ✅ Efficient GPU utilization (< 50%)
- ✅ Proper synchronization with audio

---

## 7.4 Three-Output Recording Strategy

**Priority:** P0 (Critical)

**Architecture:**

```swift
class RecordingCoordinator {
    private var backVideoWriter: VideoWriter
    private var frontVideoWriter: VideoWriter
    private var combinedVideoWriter: VideoWriter

    func startRecording() async throws {
        // Create three separate file URLs
        let backURL = generateFileURL(suffix: "Back")
        let frontURL = generateFileURL(suffix: "Front")
        let combinedURL = generateFileURL(suffix: "Combined")

        // Initialize video writers
        backVideoWriter = VideoWriter(url: backURL, settings: settings)
        frontVideoWriter = VideoWriter(url: frontURL, settings: settings)
        combinedVideoWriter = VideoWriter(url: combinedURL, settings: settings)

        // Start writing
        try await backVideoWriter.start()
        try await frontVideoWriter.start()
        try await combinedVideoWriter.start()
    }

    func processFrames(
        backBuffer: CMSampleBuffer,
        frontBuffer: CMSampleBuffer
    ) async {
        // Write to separate writers
        await backVideoWriter.append(backBuffer)
        await frontVideoWriter.append(frontBuffer)

        // Composite and write combined
        let composited = compositor.composite(back: backBuffer, front: frontBuffer)
        await combinedVideoWriter.append(composited)
    }
}
```

**VideoWriter Class:**
- Uses `AVAssetWriter` for each output file
- Separate `AVAssetWriterInput` for video and audio
- Handles frame timing and synchronization
- Manages file I/O on background queue
- Implements error recovery

**Frame Synchronization:**
- Use presentation timestamps (PTS) from CMSampleBuffer
- Ensure all three outputs have synchronized frames
- Handle frame drops gracefully
- Maintain A/V sync for all outputs

**File Management:**
- Temporary files during recording
- Atomic move to Photos library on completion
- Cleanup on error or cancel
- Progress tracking for each writer

**Acceptance Criteria:**
- ✅ Three files created for every recording

- ✅ Frame synchronization < 33ms drift
- ✅ All files playable and valid
- ✅ Proper cleanup on errors
- ✅ No data loss or corruption

---

### 7.5 State Management with Combine

**Priority:** P1 (High)

**ViewModel Pattern:**

```swift
@MainActor
class CameraViewModel: ObservableObject {
    @Published var isRecording = false
    @Published var recordingDuration: TimeInterval = 0
    @Published var zoomLevelBack: CGFloat = 1.0
    @Published var zoomLevelFront: CGFloat = 1.0
    @Published var selectedLayout: LayoutConfiguration = .pipBottomRight
    @Published var hardwareCost: Float = 0.0
    @Published var systemPressure: AVCaptureSystemPressureLevel?

    private let cameraManager: MultiCameraManager
    private let recordingCoordinator: RecordingCoordinator
    private var cancellables = Set<AnyCancellable>()

    init(cameraManager: MultiCameraManager,
         recordingCoordinator: RecordingCoordinator) {
        self.cameraManager = cameraManager
        self.recordingCoordinator = recordingCoordinator

        setupBindings()
    }

    func toggleRecording() async {
        if isRecording {
            await stopRecording()
        } else {
            await startRecording()
        }
    }
}
```

**State Flow:**
- User interaction → ViewModel → Manager → Update state
- State changes published via `@Published` properties
- SwiftUI views react to state changes
- Actor-isolated async operations for camera work

**Acceptance Criteria:**
- ✅ State changes propagate immediately to UI
- ✅ No race conditions or data races
- ✅ Thread-safe state updates
- ✅ Memory leaks prevented (weak references)
- ✅ Clear separation between UI and business logic

---

### 7.6 Error Handling Strategy

**Priority:** P0 (Critical)

**Error Categories:**

```
enum CameraError: LocalizedError {
    case multiCamNotSupported
    case hardwareCostExceeded
    case cameraUnavailable(position: AVCaptureDevice.Position)
    case permissionDenied(type: PermissionType)
    case recordingFailed(reason: String)
    case insufficientStorage
    case deviceOverheating
    case formatNotSupported

    var errorDescription: String? {
        // User-friendly error messages
    }

    var recoverySuggestion: String? {
        // Actionable recovery steps
    }
}
```

**Error Handling Flow:**
1. **Catch errors** at appropriate level
2. **Log errors** for debugging
3. **Show user-friendly message** in UI
4. **Provide recovery action** when possible
5. **Graceful degradation** if feature unavailable

**User-Facing Errors:**
- **Alert dialogs** for critical errors
- **Toast notifications** for warnings
- **Inline messages** for validation errors
- **Recovery buttons** (e.g., “Open Settings”, “Try Again”)

**Example Implementation:**

```swift
func startRecording() async {
    do {
        try await recordingCoordinator.startRecording()
        isRecording = true
    } catch CameraError.insufficientStorage {
        showError(
            title: "Not Enough Storage",
            message: "Free up space and try again",
            action: .openSettings
        )
    } catch {
        showError(
            title: "Recording Failed",
            message: error.localizedDescription,
            action: .retry
        )
    }
}
```

**Acceptance Criteria:**
- ✅ All errors caught and handled
- ✅ User-friendly error messages
- ✅ Recovery actions provided
- ✅ Errors logged for debugging
- ✅ App never crashes from recoverable errors

---

### 7.7 Performance Optimization

**Priority:** P1 (High)

**Optimization Strategies:**

1. **Background Queue Usage:**
   - Camera operations on dedicated serial queue
   - Video writing on separate background queue
   - UI updates only on main thread
   - Metal rendering on GPU

2. **Memory Management:**
   - Release sample buffers immediately after use
   - Reuse pixel buffer pools
   - Monitor memory pressure
   - Clear caches on memory warning

3. **Thermal Management:**
   - Monitor `AVCaptureDevice.systemPressureState`
   - Reduce quality when pressure is elevated
   - Show warning to user before shutdown
   - Implement cool-down period if needed

4. **Battery Optimization:**
   - Use efficient codecs (H.265 vs H.264)
   - Disable torch when not needed

   - Reduce preview frame rate when not recording
   - Batch file I/O operations

5. **Frame Rate Maintenance:**
   - Target 30fps for recording minimum
   - 60fps preview when possible
   - Drop preview frames before recording frames
   - Use appropriate video stabilization mode

**Monitoring:**
- Real-time hardware cost display
- FPS counter (debug mode)
- Memory usage indicator
- Thermal state indicator
- Battery level display

**Acceptance Criteria:**
- ✅ Maintains 30fps recording consistently
- ✅ Memory usage stays reasonable (< 500MB)
- ✅ Thermal management prevents overheating
- ✅ Battery drain acceptable (< 20%/hour recording)
- ✅ App responds smoothly to user interaction

---

## 8. Development Timeline

### Phase 1: Foundation (Weeks 1-2)

**Deliverables:**
- Project setup with Swift 6 and iOS 18 SDK
- Basic app structure and navigation
- Camera permissions implementation
- Single camera preview working

**Key Milestones:**
- ✅ Project builds successfully
- ✅ Camera permission flow complete
- ✅ Single camera preview displays
- ✅ Basic UI framework in place

---

### Phase 2: Multi-Camera Core (Weeks 3-4)

**Deliverables:**
- AVCaptureMultiCamSession implementation
- Dual camera preview working
- Hardware cost monitoring
- Basic recording to single file

**Key Milestones:**
- ✅ Both cameras preview simultaneously
- ✅ Multi-cam session stable

- ✅ Recording starts and stops reliably
- ✅ Basic file saved to Photos library

---

### Phase 3: Three-Output System (Weeks 5-6)

**Deliverables:**
- Metal compositor for combined view
- Three separate video writers
- Frame synchronization
- All three files saving correctly

**Key Milestones:**
- ✅ Metal compositor functional
- ✅ Three output files created
- ✅ Frames synchronized across outputs
- ✅ No data loss or corruption

---

### Phase 4: Liquid Glass UI (Weeks 7-8)

**Deliverables:**
- Complete liquid glass design system
- All main screens with glass aesthetic
- Animations and transitions
- Light and dark mode support

**Key Milestones:**
- ✅ Glass components library complete
- ✅ Main recording screen polished
- ✅ Settings screen functional
- ✅ Gallery view implemented

---

### Phase 5: Advanced Features (Weeks 9-11)

**Deliverables:**
- Independent zoom controls
- Exposure and focus controls
- Video quality settings
- Audio configuration
- Filters and effects
- Grid overlays

**Key Milestones:**
- ✅ All P0 and P1 features implemented
- ✅ Feature settings persist correctly
- ✅ Advanced controls intuitive
- ✅ Performance remains stable

---

### Phase 6: Polish & Testing (Weeks 12-14)

**Deliverables:**
- Comprehensive testing on multiple devices
- Bug fixes and optimizations
- Accessibility improvements
- Documentation and help screens
- App Store assets preparation

**Key Milestones:**
- ✅ All critical bugs fixed
- ✅ Performance targets met
- ✅ Accessibility audit passed
- ✅ App Store submission ready

---

### Phase 7: Launch & Iteration (Week 15+)

**Deliverables:**
- App Store submission
- Marketing materials
- User feedback collection
- Iterative improvements

**Key Milestones:**
- ✅ App approved by Apple
- ✅ Public launch
- ✅ Initial user reviews positive
- ✅ Roadmap for v1.1 defined

---

## 9. Success Metrics & KPIs

### Technical Performance Metrics

**Stability:**
- Crash-free rate: > 99.9%
- Recording success rate: > 99%
- File save success rate: > 99.5%

**Performance:**
- Recording frame rate: 30fps minimum, 60fps target
- Preview latency: < 100ms
- UI response time: < 100ms
- App launch time: < 3 seconds
- Memory footprint: < 500MB during recording
- Battery consumption: < 20% per hour of recording

**Quality:**
- Frame synchronization drift: < 33ms (1 frame at 30fps)
- Audio/video sync: < 50ms offset
- No visible artifacts in recordings
- Color accuracy maintained

### User Engagement Metrics

**Acquisition:**
- App Store impressions → downloads conversion: > 20%
- First-time user activation rate: > 80%

**Engagement:**
- Daily active users (DAU): Track growth
- Weekly active users (WAU): Track growth
- Sessions per user per week: > 3
- Average session duration: > 5 minutes
- Recording completion rate: > 90%

**Retention:**
- Day 1 retention: > 60%
- Day 7 retention: > 40%
- Day 30 retention: > 25%
- 90-day retention: > 15%

**Feature Adoption:**
- Advanced controls usage: > 40% of users
- Layout switching: > 60% of users
- Gallery access: > 70% of users
- Filters usage: > 50% of users

---

### Business Metrics

**App Store Performance:**
- Average rating: 4.5+ stars
- Total ratings: 1,000+ within 3 months
- Review response time: < 48 hours
- Feature requests collected: Continuous

**Growth:**
- Month-over-month user growth: > 20%
- Organic vs. paid acquisition ratio: Track
- Social sharing rate: > 10% of recordings

**Satisfaction:**
- Net Promoter Score (NPS): > 50
- User satisfaction score: > 4.0/5.0
- Support ticket resolution rate: > 95%

---

## 10. Risk Assessment & Mitigation

### Technical Risks

**Risk 1: Hardware Limitations**
- **Description**: Some devices may not support multi-cam or have insufficient hardware
- **Probability**: Medium

- **Impact**: High
- **Mitigation**:
- Comprehensive device compatibility checking
- Graceful fallback to single-camera mode
- Clear communication of device requirements
- Dynamic quality adjustment based on hardware cost

**Risk 2: Thermal Throttling**
- **Description**: Extended recording may cause device overheating
- **Probability**: High
- **Impact**: Medium
- **Mitigation**:
- Real-time thermal monitoring
- Proactive quality reduction before shutdown
- User warnings with cool-down suggestions
- Optimization of video encoding settings

**Risk 3: Battery Drain**
- **Description**: Multi-camera recording is power-intensive
- **Probability**: High
- **Impact**: Medium
- **Mitigation**:
- Battery level display and warnings
- Efficient codec usage (H.265)
- Optional power-saving mode
- Background task management

**Risk 4: Frame Synchronization**
- **Description**: Maintaining sync across three outputs challenging
- **Probability**: Medium
- **Impact**: High
- **Mitigation**:
- Use presentation timestamps consistently
- Buffer management to handle timing variations
- Comprehensive testing with different scenarios
- Fallback mechanism if sync drifts

---

## User Experience Risks

**Risk 5: Complexity Overload**
- **Description**: Too many features may overwhelm users
- **Probability**: Medium
- **Impact**: Medium
- **Mitigation**:
- Progressive disclosure of advanced features
- Smart defaults that work for most users
- Onboarding tutorial for first-time users
- Settings organized in clear categories

**Risk 6: Storage Consumption**
- **Description**: Three files per recording uses significant storage
- **Probability**: High
- **Impact**: Medium
- **Mitigation**:
- Storage warning when space low
- Optional setting to save only combined view
- Automatic cleanup of temporary files
- Clear file size indicators in settings

---

### Business Risks

**Risk 7: Competition**
- **Description**: Established apps (Mixcam, DoubleTake) have market presence
- **Probability**: High
- **Impact**: Medium
- **Mitigation**:
- Differentiate with superior design and reliability
- Leverage iOS 18 features competitors lack
- Build community through social media
- Responsive to user feedback

**Risk 8: App Store Approval**
- **Description**: Rejection or delays in review process
- **Probability**: Low
- **Impact**: High
- **Mitigation**:
- Follow Apple guidelines strictly
- Comprehensive testing before submission
- Clear privacy policy and permissions explanations
- Responsive to reviewer feedback

---

## 11. Future Enhancements (Post-V1)

### V1.1 - Advanced Recording Features

- Live streaming support
- Multi-device sync (use multiple iPhones)
- Remote control via Apple Watch
- Cloud backup integration
- Social media direct upload

### V1.2 - Professional Tools

- Manual color grading controls
- LUT (Look-Up Table) support
- Professional audio meters
- Timecode synchronization
- ProRes recording codec

### V1.3 - AI-Powered Features

- Auto-framing and tracking
- Scene detection and auto-settings
- AI-powered filters
- Automatic highlight generation
- Intelligent noise reduction

### V2.0 - Platform Expansion

- iPad optimization
- Mac Catalyst version
- Apple Vision Pro support
- Cross-device continuity
- Multi-user collaboration

---

# Appendix

## A. Competitive Analysis

| Feature | DualCam | Mixcam | DoubleTake | Dualgram |
|---|---|---|---|---|
| **Price** | TBD | $1.99/week | Free | Subscription |
| **Max Resolution** | 4K | 4K | 1080p | Unknown |
| **iOS Version** | 18+ | 13+ | 13+ | 13+ |
| **Swift Version** | 6 | Unknown | Objective-C/ Swift | Unknown |
| **Separate Outputs** | ✅ 3 files | ❌ 1 file | ✅ 2 files | ❌ 1 file |
| **Layout Options** | 5 options | 4 options | 3 options | 2 options |
| **Independent Zoom** | ✅ | ❌ | ❌ | ❌ |
| **Liquid Glass UI** | ✅ | ❌ | ❌ | ❌ |
| **Advanced Controls** | ✅ Full | ⚠️ Limited | ⚠️ Limited | ❌ Basic |
| **Reliability** | Target 99%+ | ⚠️ Issues | ⚠️ Issues | ✅ Good |
| **Photo Capture** | Future | ✅ | ❌ | ✅ |

## B. Device Compatibility Matrix

| Device | Multi-Cam | 4K Recording | Metal | Recommendation |
|---|---|---|---|---|
| iPhone 15 Pro Max | ✅ | ✅ | ✅ | Excellent |
| iPhone 15 Pro | ✅ | ✅ | ✅ | Excellent |
| iPhone 15 | ✅ | ✅ | ✅ | Excellent |
| iPhone 14 Pro Max | ✅ | ✅ | ✅ | Excellent |
| iPhone 14 Pro | ✅ | ✅ | ✅ | Excellent |
| iPhone 14 | ✅ | ✅ | ✅ | Excellent |
| iPhone 13 Pro Max | ✅ | ✅ | ✅ | Great |
| iPhone 13 Pro | ✅ | ✅ | ✅ | Great |
| iPhone 13 | ✅ | ✅ | ✅ | Great |
| iPhone 12 Pro Max | ✅ | ✅ | ✅ | Great |
| iPhone 12 Pro | ✅ | ✅ | ✅ | Great |
| iPhone 12 | ✅ | ✅ | ✅ | Good |
| iPhone 11 Pro Max | ✅ | ⚠️ Limited | ✅ | Good |
| iPhone 11 Pro | ✅ | ⚠️ Limited | ✅ | Good |
| iPhone 11 | ✅ | ⚠️ Limited | ✅ | Good |
| iPhone XS Max | ✅ | ⚠️ Limited | ✅ | Acceptable |
| iPhone XS | ✅ | ⚠️ Limited | ✅ | Acceptable |
| iPhone XR | ✅ | ❌ | ✅ | Acceptable |

## C. Glossary of Terms

- **AVCaptureMultiCamSession**: Apple's API for simultaneous multi-camera capture
- **Hardware Cost**: Resource utilization metric (0.0-1.0) for multi-camera capture
- **Liquid Glass**: UI design style with translucent, blurred backgrounds

- **PiP**: Picture-in-Picture layout with one camera overlaid on another
- **H.265/HEVC**: High Efficiency Video Codec for better compression
- **Metal**: Apple's GPU framework for high-performance graphics
- **CMSampleBuffer**: Core Media sample buffer containing video/audio frames
- **Presentation Timestamp (PTS)**: Timing information for media frames
- **Sensor Binning**: Combining adjacent pixels to reduce resolution and power

## D. References

1. Apple Developer Documentation - AVFoundation
2. Apple Human Interface Guidelines - iOS 18
3. WWDC 2019 Session 249 - Camera Capture Beyond the Basics
4. WWDC 2023 - What's New in Camera Capture
5. Swift Concurrency Documentation
6. Metal Programming Guide
7. PhotoKit Framework Documentation

---

**Document Status:** ✅ Approved for Development
**Next Review Date:** November 15, 2025
**Owner:** DualCam Development Team
**Stakeholders:** Product, Engineering, Design, QA