



Lets formulate specs to build this ios app quickly;

Actually I want to discuss building an app called dealflow where entrepreneurs can record a 30 second elevator pitch video or a 5 minute deck presentation video and request investors to q&a them. In this set up the founders have free access to leave a 30 second pitch or pay \$5 to create a 5 min presentation while investors must pay a monthly or annual subscription to view investment opportunities

Dealflow can be specified as a lightweight two-sided video pitch marketplace where founders upload short pitch videos and investors pay subscriptions to browse opportunities and ask Q&A.
[\[1\]](#) [\[2\]](#)

Core user stories

- **Founder – free pitch**
 - Record or upload a 30s elevator pitch video from iOS camera or library.[\[3\]](#)
 - Create a startup profile (name, sector, stage, geography, ticket size, links).
 - Publish pitch into a browsable feed visible to subscribed investors.
- Founder – paid 5-minute presentation
 - Pay a one-off in-app purchase of £4.99 (US: 4.99) to unlock a 5-minute pitch slot for a given startup.[\[4\]](#)
 - Upload/record a 5-minute pitch (deck walkthrough, demo etc.).[\[5\]](#) [\[3\]](#)
 - Tag video with round details (pre-seed/seed, amount, timing).
- Investor – discovery & Q&A
 - Create investor profile (firm, ticket size, sector focus, geography).[\[2\]](#) [\[1\]](#)
 - Browse a Tinder-style or card feed of pitches with filters (sector, stage, location, round size).
 - Tap into a deal to:
 - Watch 30s or 5-min video.
 - View key deal metadata (round amount, cap, traction bullets).
 - Ask questions via:
 - Asynchronous text Q&A.
 - Optional async video questions (30s clips).
 - Bookmark or “follow” startups and receive notifications on replies and updates.
- Matching & notifications

- Investors can “shortlist” or “pass”; founders see anonymised interest or named interest depending on settings.^[1]
- Push notifications for:
 - New investor questions.
 - New answers from founders.
 - Subscription renewal reminders.

Monetisation & pricing model

- **Founders**
 - Free tier:
 - 1 × 30s elevator pitch per startup.
 - Basic startup profile and inbound Q&A.
 - Paid add-ons:
 - One-off 5-minute pitch slot sold via Apple In-App Purchase (compliant with Apple rules for digital content).^[4]
 - Future: bundles (e.g. 3 × 5-minute slots, featured placement).
- **Investors**
 - Subscription (consumable via Apple IAP or, in EU, optional external purchase with DMA trade-offs).^{[6] [4]}
 - Monthly plan (e.g. £29.99).
 - Annual plan with discount (e.g. £249.99).
 - Entitlements:
 - Unlimited viewing of all pitches.
 - Unlimited Q&A messages.
 - Saved lists and export of notes.
- Compliance constraints
 - All payments for viewing digital content in the iOS app must go through Apple IAP unless using EU-only external purchase entitlement with added reporting and extra fees.^{[6] [4]}
 - Founder 5-minute pitch unlock is clearly “digital content” and must be IAP if bought in-app.^[4]

MVP feature spec (v1)

- **Authentication & onboarding**
 - Apple Sign-In plus email/password.
 - Role selection: Founder or Investor (can later support dual roles).
 - KYC-lite for investors (LinkedIn URL + firm website) to reduce low-quality accounts.^[1]

- Video capture & upload
 - In-app recorder with:
 - 30s timer (hard stop).
 - 5-minute timer (hard stop) once 5-minute entitlement is unlocked.
 - Support for uploading pre-recorded video files.
 - Background upload with progress and retry.
- Content model
 - Startup: name, tagline, location, website, sector tags, stage, round size, traction highlights.
 - PitchVideo: type (30s free / 5-min paid), language, subtitle status, createdAt, viewCount.
 - InvestorProfile: fund size band, cheque size band, sectors, geo, stage, notes.^[1]
- Q&A system
 - Thread per investor-startup pair.
 - Messages can be:
 - Text.
 - Optional video question/answer clips.
 - Basic moderation (flag message; simple block).
- Discovery UI
 - Home feed for investors:
 - Card layout with:
 - Founder name + startup logo.
 - 30s preview (tap to play).
 - Sector, stage, location chips.
 - Filters and search.
 - Founder dashboard:
 - Stats: views, unique investors, questions received.
 - List of pitches and status (draft/published).
- Payments & subscriptions
 - Integrate StoreKit:
 - Non-consumable product: “5-minute pitch unlock” per startup or global, depending on your economics.
 - Auto-renewable subscriptions: “Investor Monthly”, “Investor Annual”.
 - Backend verification of receipts for entitlements.^[4]

Architecture & build-fast choices

- **Frontend (iOS)**
 - Option A (fastest native): SwiftUI + AVFoundation for capture and playback.
 - Option B (cross-platform ready): React Native/Expo with proper native modules for video; you already use Expo, so this aligns with existing stack.
- Backend
 - Simple JSON API for v1 (REST):
 - Auth, profiles.
 - Pitches, Q&A, subscriptions, analytics.
 - Cloud storage for video files with signed URLs (e.g. GCS/S3); front door via CDN.
 - Optional AI add-ons later:
 - Auto-transcription and subtitles.
 - AI summary of each pitch for quicker investor triage.^[5] ^[1]
- Moderation & trust
 - Basic terms acceptance on sign-up.
 - Manual admin dashboard (web) to:
 - Remove abusive content.
 - Verify investors on request.

Next step

If helpful, the next step can be a 1-page “Dealfow MVP spec” with concrete screens (Founder Pitch Flow, Investor Browse Flow, Payments Flow) and a task breakdown for a 4–6 week sprint so you can brief a contractor or move straight into implementation.

**

Dealfow MVP Developer Specification

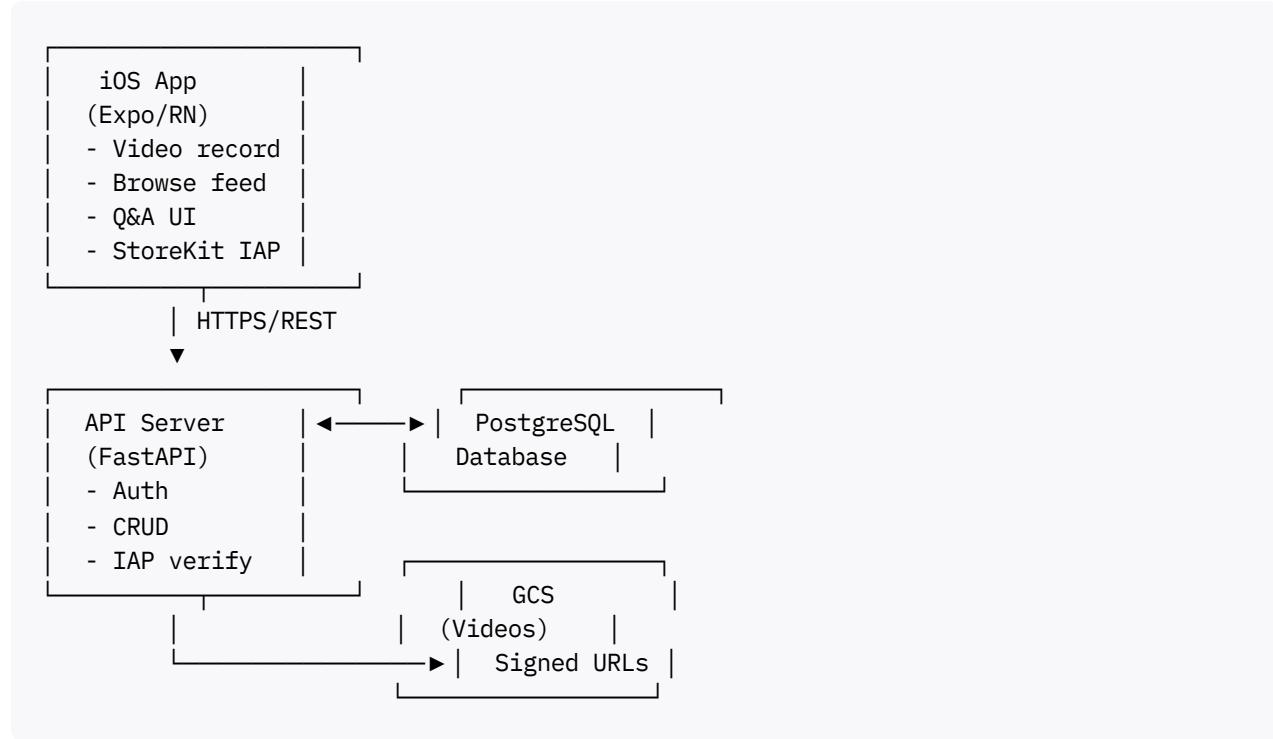
Project: Dealfow - Video pitch marketplace for founders & investors

Tech Stack: React Native (Expo), Python (FastAPI), PostgreSQL, Google Cloud Storage, Firebase Auth

Target: iOS first, Android-ready architecture

Timeline: 4–6 week MVP sprint

1. System Architecture



2. Project File Structure

```
dealflow/
  └── mobile/                                # Expo React Native app
      ├── app.json
      ├── package.json
      ├── App.js
      └── src/
          ├── navigation/
          │   └── AppNavigator.js
          ├── screens/
          │   ├── auth/
          │   │   ├── LoginScreen.js
          │   │   ├── SignupScreen.js
          │   │   └── RoleSelectScreen.js
          │   ├── founder/
          │   │   ├── FounderDashboardScreen.js
          │   │   ├── CreatePitchScreen.js
          │   │   ├── RecordVideoScreen.js
          │   │   ├── PitchStatsScreen.js
          │   │   └── FounderQAScreen.js
          │   ├── investor/
          │   │   ├── InvestorFeedScreen.js
          │   │   ├── PitchDetailScreen.js
          │   │   ├── InvestorQAScreen.js
          │   │   └── SubscriptionScreen.js
          │   └── shared/
          │       ├── ProfileScreen.js
          │       └── SettingsScreen.js
          └── components/
```

```
    └── VideoPlayer.js
    └── VideoRecorder.js
    └── PitchCard.js
    └── FilterBar.js
    └── QAThread.js
    └── services/
        ├── api.js
        ├── auth.js
        ├── iap.js          # StoreKit wrapper
        └── video.js
    └── hooks/
        ├── useAuth.js
        ├── useSubscription.js
        └── useVideoUpload.js
    └── context/
        └── AuthContext.js
    └── utils/
        ├── constants.js
        └── validators.js
    └── ios/
        └── dealflow.storekit      # StoreKit config file

└── backend/
    ├── requirements.txt
    ├── main.py
    └── app/
        ├── __init__.py
        ├── config.py
        ├── database.py
        ├── models/
            ├── __init__.py
            ├── user.py
            ├── startup.py
            ├── pitch.py
            ├── investor.py
            ├── qa.py
            └── subscription.py
        ├── schemas/
            ├── __init__.py
            ├── user.py
            ├── pitch.py
            ├── investor.py
            └── qa.py
        ├── api/
            ├── __init__.py
            ├── deps.py          # Dependencies (auth, db)
            ├── auth.py
            ├── pitches.py
            ├── investors.py
            ├── qa.py
            ├── subscriptions.py
            └── uploads.py
        └── services/
            ├── __init__.py
            ├── firebase.py       # Firebase Auth verify
            └── gcs.py            # GCS signed URLs
```

```

    └── iap_verify.py      # Apple IAP receipt verification
    └── utils/
        ├── __init__.py
        └── security.py
    └── alembic/           # DB migrations
        └── versions/
            └── env.py

```

3. Database Schema (PostgreSQL)

```

-- users table
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    firebase_uid VARCHAR(128) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    role VARCHAR(20) NOT NULL CHECK (role IN ('founder', 'investor')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- founder_profiles table
CREATE TABLE founder_profiles (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    full_name VARCHAR(255),
    linkedin_url VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(user_id)
);

-- startups table
CREATE TABLE startups (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    founder_id UUID REFERENCES users(id) ON DELETE CASCADE,
    name VARCHAR(255) NOT NULL,
    tagline VARCHAR(500),
    website VARCHAR(500),
    sector VARCHAR(100),
    stage VARCHAR(50) CHECK (stage IN ('idea', 'pre_seed', 'seed', 'series_a')),
    location VARCHAR(255),
    round_size_min INTEGER,
    round_size_max INTEGER,
    traction_bullets TEXT[],
    logo_url VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- pitches table
CREATE TABLE pitches (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    startup_id UUID REFERENCES startups(id) ON DELETE CASCADE,
    type VARCHAR(20) NOT NULL CHECK (type IN ('30s_free', '5min_paid')),
    video_url VARCHAR(1000) NOT NULL,

```

```

    thumbnail_url VARCHAR(1000),
    duration_seconds INTEGER,
    status VARCHAR(20) DEFAULT 'draft' CHECK (status IN ('draft', 'published', 'archived'),
    view_count INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- investor_profiles table
CREATE TABLE investor_profiles (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    full_name VARCHAR(255),
    firm_name VARCHAR(255),
    linkedin_url VARCHAR(500),
    website VARCHAR(500),
    ticket_size_min INTEGER,
    ticket_size_max INTEGER,
    sectors TEXT[],
    stages TEXT[],
    geographies TEXT[],
    is_verified BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(user_id)
);

-- subscriptions table
CREATE TABLE subscriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    investor_id UUID REFERENCES users(id) ON DELETE CASCADE,
    plan_type VARCHAR(20) NOT NULL CHECK (plan_type IN ('monthly', 'annual')),
    apple_transaction_id VARCHAR(255) UNIQUE,
    apple_original_transaction_id VARCHAR(255),
    status VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'expired', 'cancelled')),
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- pitch_unlocks table (for 5-min paid pitches)
CREATE TABLE pitch_unlocks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    startup_id UUID REFERENCES startups(id) ON DELETE CASCADE,
    founder_id UUID REFERENCES users(id) ON DELETE CASCADE,
    apple_transaction_id VARCHAR(255) UNIQUE NOT NULL,
    product_id VARCHAR(100) NOT NULL,
    purchased_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(startup_id, founder_id)
);

-- qa_threads table
CREATE TABLE qa_threads (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pitch_id UUID REFERENCES pitches(id) ON DELETE CASCADE,
    investor_id UUID REFERENCES users(id) ON DELETE CASCADE,

```

```

        startup_id UUID REFERENCES startups(id) ON DELETE CASCADE,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        UNIQUE(pitch_id, investor_id)
    );

-- qa_messages table
CREATE TABLE qa_messages (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    thread_id UUID REFERENCES qa_threads(id) ON DELETE CASCADE,
    sender_id UUID REFERENCES users(id) ON DELETE CASCADE,
    message_type VARCHAR(20) CHECK (message_type IN ('text', 'video')),
    content TEXT,
    video_url VARCHAR(1000),
    is_read BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- pitch_views table (analytics)
CREATE TABLE pitch_views (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pitch_id UUID REFERENCES pitches(id) ON DELETE CASCADE,
    investor_id UUID REFERENCES users(id) ON DELETE SET NULL,
    viewed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_pitches_startup ON pitches(startup_id);
CREATE INDEX idx_pitches_status ON pitches(status);
CREATE INDEX idx_qa_threads_pitch ON qa_threads(pitch_id);
CREATE INDEX idx_qa_threads_investor ON qa_threads(investor_id);
CREATE INDEX idx_qa_messages_thread ON qa_messages(thread_id);
CREATE INDEX idx_subscriptions_investor ON subscriptions(investor_id);
CREATE INDEX idx_subscriptions_status ON subscriptions(status, expires_at);

```

4. Backend API Endpoints (FastAPI)

Authentication Endpoints

```

# POST /api/v1/auth/signup
Request:
{
    "firebase_token": "eyJhbGc...",
    "role": "founder", # or "investor"
    "email": "user@example.com"
}
Response: {
    "user": {"id": "uuid", "email": "...", "role": "founder"},
    "token": "jwt_token"
}

# POST /api/v1/auth/login
Request: {"firebase_token": "eyJhbGc..."}
Response: {"user": {...}, "token": "jwt_token"}

```

```
# GET /api/v1/auth/me
Headers: Authorization: Bearer <token>
Response: {"user": {...}, "profile": {...}}
```

Founder Endpoints

```
# POST /api/v1/startups
Headers: Authorization: Bearer <token>
Request: {
    "name": "Acme Inc",
    "tagline": "We make widgets",
    "sector": "SaaS",
    "stage": "seed",
    "location": "London, UK",
    "round_size_min": 500000,
    "round_size_max": 1000000,
    "traction_bullets": ["10K MRR", "50% MoM growth"]
}
Response: {"id": "uuid", "name": "Acme Inc", ...}

# GET /api/v1/startups/{startup_id}
Response: {"id": "uuid", "name": "...", "pitches": [...]}

# POST /api/v1/pitches/upload-url
Headers: Authorization: Bearer <token>
Request: {
    "startup_id": "uuid",
    "type": "30s_free", # or "5min_paid"
    "filename": "pitch.mp4",
    "content_type": "video/mp4"
}
Response: {
    "upload_url": "https://storage.googleapis.com/...",
    "video_id": "uuid"
}

# POST /api/v1/pitches/{pitch_id}/publish
Headers: Authorization: Bearer <token>
Request: {"duration_seconds": 28}
Response: {"id": "uuid", "status": "published"}

# GET /api/v1/founder/dashboard
Headers: Authorization: Bearer <token>
Response: {
    "startups": [...],
    "stats": {
        "total_views": 245,
        "unique_investors": 18,
        "questions_received": 7
    }
}

# POST /api/v1/iap/verify-unlock
Headers: Authorization: Bearer <token>
```

```

Request: {
    "startup_id": "uuid",
    "receipt_data": "base64_apple_receipt",
    "transaction_id": "1000000123456789"
}
Response: {"unlocked": true, "pitch_type": "5min_paid"}

```

Investor Endpoints

```

# GET /api/v1/pitches/feed
Headers: Authorization: Bearer <token>
Query: ?sector=SaaS&stage=seed&location=UK&limit=20&offset=0
Response: {
    "pitches": [
        {
            "id": "uuid",
            "startup": {"name": "...", "tagline": "...", "sector": "..."},
            "type": "30s_free",
            "thumbnail_url": "...",
            "view_count": 45,
            "created_at": "2025-12-20T10:00:00Z"
        }
    ],
    "total": 150
}

# GET /api/v1/pitches/{pitch_id}
Headers: Authorization: Bearer <token>
Response: {
    "id": "uuid",
    "startup": {...},
    "video_url": "https://storage.googleapis.com/... (signed URL)",
    "type": "30s_free",
    "duration_seconds": 28
}

# POST /api/v1/pitches/{pitch_id}/view
Headers: Authorization: Bearer <token>
Response: {"viewed": true}

# POST /api/v1/iap/verify-subscription
Headers: Authorization: Bearer <token>
Request: {
    "receipt_data": "base64_apple_receipt",
    "transaction_id": "1000000123456789",
    "product_id": "com.dealflow.investor.monthly"
}
Response: {
    "subscription": {
        "status": "active",
        "expires_at": "2026-01-22T00:00:00Z",
        "plan_type": "monthly"
    }
}

```

```
# GET /api/v1/investor/subscription
Headers: Authorization: Bearer <token>
Response: {
  "status": "active",
  "plan_type": "monthly",
  "expires_at": "2026-01-22T00:00:00Z"
}
```

Q&A Endpoints

```
# POST /api/v1/qa/threads
Headers: Authorization: Bearer <token>
Request: {"pitch_id": "uuid"}
Response: {"thread_id": "uuid", "created_at": "..."}  
  

# GET /api/v1/qa/threads
Headers: Authorization: Bearer <token>
Response: {
  "threads": [
    {
      "id": "uuid",
      "pitch": {...},
      "startup": {...},
      "last_message": {...},
      "unread_count": 2
    }
  ]
}  
  

# POST /api/v1/qa/threads/{thread_id}/messages
Headers: Authorization: Bearer <token>
Request: {
  "message_type": "text",
  "content": "What's your CAC?"
}
Response: {"id": "uuid", "created_at": "..."}  
  

# GET /api/v1/qa/threads/{thread_id}/messages
Headers: Authorization: Bearer <token>
Response: {
  "messages": [
    {
      "id": "uuid",
      "sender_id": "uuid",
      "sender_role": "investor",
      "message_type": "text",
      "content": "What's your CAC?",
      "created_at": ...
    }
  ]
}
```

5. Mobile App Implementation (React Native/Expo)

Key Dependencies (package.json)

```
{  
  "dependencies": {  
    "expo": "~52.0.0",  
    "expo-camera": "~16.0.0",  
    "expo-av": "~15.0.0",  
    "expo-media-library": "~17.0.0",  
    "expo-file-system": "~18.0.0",  
    "@react-navigation/native": "^6.1.0",  
    "@react-navigation/stack": "^6.4.0",  
    "react-native-iap": "^12.15.0",  
    "axios": "^1.7.0",  
    "firebase": "^10.7.0",  
    "@react-native-firebase/auth": "^20.0.0",  
    "react-native-video": "^6.0.0"  
  }  
}
```

Video Recording Component [11] [12] [13]

```
// src/components/VideoRecorder.js  
import { Camera, CameraView } from 'expo-camera';  
import { useState, useRef } from 'react';  
import { View, TouchableOpacity, Text } from 'react-native';  
  
export default function VideoRecorder({ maxDuration = 30, onVideoRecorded }) {  
  const [isRecording, setIsRecording] = useState(false);  
  const [hasPermission, setHasPermission] = useState(null);  
  const cameraRef = useRef(null);  
  const timerRef = useRef(null);  
  
  // Request camera & microphone permissions  
  useEffect(() => {  
    (async () => {  
      const { status } = await Camera.requestCameraPermissionsAsync();  
      const audioStatus = await Camera.requestMicrophonePermissionsAsync();  
      setHasPermission(status === 'granted' && audioStatus.status === 'granted');  
    })();  
  }, []);  
  
  const startRecording = async () => {  
    if (!cameraRef.current) return;  
  
    setIsRecording(true);  
    const video = await cameraRef.current.recordAsync({  
      maxDuration: maxDuration, // 30 or 300 seconds  
      quality: '720p',  
      mute: false  
    });  
    onVideoRecorded(video);  
  };  
}
```

```

        setIsRecording(false);
        onVideoRecorded(video); // {uri: 'file://...', duration: 28.5}
    };

    const stopRecording = () => {
        if (cameraRef.current && isRecording) {
            cameraRef.current.stopRecording();
        }
    };

    if (hasPermission === null) return <Text>Requesting permissions...</Text>;
    if (hasPermission === false) return <Text>No camera access</Text>;

    return (
        <View style={{ flex: 1 }}>
            <CameraView
                ref={cameraRef}
                style={{ flex: 1 }}
                facing="front"
                mode="video"
            />
            <TouchableOpacity
                onPress={isRecording ? stopRecording : startRecording}
                style={styles.recordButton}
            >
                <Text>{isRecording ? 'Stop' : 'Record'}</Text>
            </TouchableOpacity>
        </View>
    );
}

```

Video Upload Service [\[14\]](#) [\[15\]](#)

```

// src/services/video.js
import * as FileSystem from 'expo-file-system';
import api from './api';

export const uploadVideo = async (videoUri, startupId, type) => {
    // Step 1: Request signed upload URL from backend
    const { data } = await api.post('/pitches/upload-url', {
        startup_id: startupId,
        type: type, // '30s_free' or '5min_paid'
        filename: videoUri.split('/').pop(),
        content_type: 'video/mp4'
    });

    const { upload_url, video_id } = data;

    // Step 2: Upload video directly to GCS using signed URL
    const uploadResult = await FileSystem.uploadAsync(upload_url, videoUri, {
        httpMethod: 'PUT',
        headers: {
            'Content-Type': 'video/mp4',
        },
        uploadType: FileSystem.FileSystemUploadType.BINARY_CONTENT,
    });
}

```

```

    });

    if (uploadResult.status !== 200) {
      throw new Error('Video upload failed');
    }

    return video_id;
};


```

StoreKit Integration (iOS IAP) [\[16\]](#) [\[17\]](#)

```

// src/services/iap.js
import RNIap, {
  purchaseErrorListener,
  purchaseUpdatedListener,
  finishTransaction,
} from 'react-native-iap';

const PRODUCT_IDS = {
  FIVE_MIN_PITCH: 'com.dealflow.founder.5min',
  INVESTOR_MONTHLY: 'com.dealflow.investor.monthly',
  INVESTOR_ANNUAL: 'com.dealflow.investor.annual',
};

export const initIAP = async () => {
  try {
    await RNIap.initConnection();
    const products = await RNIap.getProducts({
      skus: Object.values(PRODUCT_IDS)
    });
    return products;
  } catch (err) {
    console.error('IAP init error:', err);
  }
};

export const purchaseFiveMinPitch = async (startupId) => {
  try {
    const purchase = await RNIap.requestPurchase({
      sku: PRODUCT_IDS.FIVE_MIN_PITCH,
    });

    // Verify with backend
    const response = await api.post('/iap/verify-unlock', {
      startup_id: startupId,
      receipt_data: purchase.transactionReceipt,
      transaction_id: purchase.transactionId,
    });

    await finishTransaction({ purchase });
    return response.data;
  } catch (err) {
    console.error('Purchase error:', err);
    throw err;
  }
};


```

```

};

export const subscribeInvestor = async (planType) => {
  try {
    const sku = planType === 'monthly'
      ? PRODUCT_IDS.INVESTOR_MONTHLY
      : PRODUCT_IDS.INVESTOR_ANNUAL;

    const purchase = await RNIap.requestSubscription({ sku });

    // Verify with backend
    const response = await api.post('/iap/verify-subscription', {
      receipt_data: purchase.transactionReceipt,
      transaction_id: purchase.transactionId,
      product_id: sku,
    });

    await finishTransaction({ purchase });
    return response.data;
  } catch (err) {
    console.error('Subscription error:', err);
    throw err;
  }
};

```

6. Backend Service Implementation (Python/FastAPI)

GCS Signed URL Service [\[15\]](#) [\[14\]](#)

```

# app/services/gcs.py
from google.cloud import storage
from datetime import timedelta
import uuid

class GCSService:
    def __init__(self, bucket_name: str):
        self.client = storage.Client()
        self.bucket = self.client.bucket(bucket_name)

    def generate_upload_url(
        self,
        filename: str,
        content_type: str = "video/mp4",
        expiration: int = 15
    ) -> tuple[str, str]:
        """Generate signed URL for uploading video"""
        blob_name = f"videos/{uuid.uuid4()}/{filename}"
        blob = self.bucket.blob(blob_name)

        url = blob.generate_signed_url(
            version="v4",
            expiration=timedelta(minutes=expiration),
            method="PUT",

```

```

        content_type=content_type,
    )

    return url, blob_name

def generate_download_url(
    self,
    blob_name: str,
    expiration: int = 60
) -> str:
    """Generate signed URL for viewing video"""
    blob = self.bucket.blob(blob_name)

    url = blob.generate_signed_url(
        version="v4",
        expiration=timedelta(minutes=expiration),
        method="GET",
    )

    return url

```

Apple IAP Verification Service^[17]

```

# app/services/iap_verify.py
import requests
from typing import Optional, Dict
import base64

SANDBOX_URL = "https://sandbox.itunes.apple.com/verifyReceipt"
PRODUCTION_URL = "https://buy.itunes.apple.com/verifyReceipt"

class IAPVerifier:
    def __init__(self, shared_secret: str):
        self.shared_secret = shared_secret

    def verify_receipt(self, receipt_data: str) -> Optional[Dict]:
        """Verify Apple IAP receipt"""
        payload = {
            "receipt-data": receipt_data,
            "password": self.shared_secret,
            "exclude-old-transactions": True
        }

        # Try production first
        response = requests.post(PRODUCTION_URL, json=payload)
        data = response.json()

        # If sandbox receipt, retry with sandbox URL
        if data.get("status") == 21007:
            response = requests.post(SANDBOX_URL, json=payload)
            data = response.json()

        if data.get("status") == 0:
            return data

```

```

        return None

def extract_subscription_info(self, receipt_data: Dict) -> Dict:
    """Extract subscription details from receipt"""
    latest_info = receipt_data.get("latest_receipt_info", [])

    if not latest_info:
        return None

    latest = latest_info[0]

    return {
        "transaction_id": latest["transaction_id"],
        "original_transaction_id": latest["original_transaction_id"],
        "product_id": latest["product_id"],
        "expires_date_ms": int(latest["expires_date_ms"]),
    }

```

API Route Example

```

# app/api/pitches.py
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from app.api.deps import get_current_user, get_db
from app.services.gcs import GCSService
from app import models, schemas

router = APIRouter()
gcs_service = GCSService(bucket_name="dealflow-videos")

@router.post("/upload-url", response_model=schemas.UploadURLResponse)
async def get_upload_url(
    request: schemas.UploadURLRequest,
    db: Session = Depends(get_db),
    current_user: models.User = Depends(get_current_user)
):
    # Verify user owns the startup
    startup = db.query(models.Startup).filter(
        models.Startup.id == request.startup_id,
        models.Startup.founder_id == current_user.id
    ).first()

    if not startup:
        raise HTTPException(status_code=404, detail="Startup not found")

    # Check if 5min pitch requires unlock
    if request.type == "5min_paid":
        unlock = db.query(models.PitchUnlock).filter(
            models.PitchUnlock.startup_id == request.startup_id,
            models.PitchUnlock.founder_id == current_user.id
        ).first()

        if not unlock:
            raise HTTPException(
                status_code=402,

```

```

        detail="5-minute pitch requires payment"
    )

# Generate signed upload URL
upload_url, blob_name = gcs_service.generate_upload_url(
    filename=request.filename,
    content_type=request.content_type
)

# Create pitch record
pitch = models.Pitch(
    startup_id=request.startup_id,
    type=request.type,
    video_url=blob_name,
    status="draft"
)
db.add(pitch)
db.commit()
db.refresh(pitch)

return {
    "upload_url": upload_url,
    "video_id": str(pitch.id)
}

```

7. Implementation Phases

Phase 1: Foundation (Week 1)

- Set up Expo project with navigation
- Set up FastAPI backend with PostgreSQL
- Implement Firebase Auth on mobile + backend verification
- Create database schema and migrations
- Build login/signup flows

Phase 2: Core Video Features (Week 2)

- Implement VideoRecorder component [\[12\]](#) [\[11\]](#)
- Build GCS signed URL service [\[14\]](#) [\[15\]](#)
- Create founder pitch creation flow
- Build video upload with progress tracking
- Implement basic pitch dashboard

Phase 3: Investor Features (Week 3)

- Build investor feed with filters
- Implement video playback with signed URLs
- Create pitch detail view
- Add view tracking
- Build Q&A thread system

Phase 4: Monetization (Week 4)

- Integrate StoreKit/react-native-iap [\[16\]](#) [\[17\]](#)
- Build subscription verification flow
- Implement 5-minute pitch unlock
- Add payment status checks throughout app
- Create subscription screen

Phase 5: Polish & Testing (Weeks 5-6)

- Add push notifications (Expo Notifications)
- Implement error handling and loading states
- Create onboarding flows
- TestFlight beta testing
- Bug fixes and optimization

8. Environment Variables

Backend (.env)

```
DATABASE_URL=postgresql://user:pass@localhost:5432/dealflo  
FIREBASE_PROJECT_ID=dealflo-prod  
GOOGLE_APPLICATION_CREDENTIALS=/path/to/service-account.json  
GCS_BUCKET_NAME=dealflo-videos  
APPLE_SHARED_SECRET=your_apple_shared_secret  
JWT_SECRET=your_jwt_secret  
ENVIRONMENT=development # or production
```

Mobile (app.json extras)

```
{  
  "expo": {  
    "extra": {  
      "apiUrl": "https://api.dealflo.com",  
      "firebaseConfig": {
```

```
        "apiKey": "...",
        "authDomain": "...",
        "projectId": "dealflow-prod"
    }
}
}
```

9. Key Implementation Notes

- 1. Video Quality:** Use 720p for balance between quality and file size [\[18\]](#) [\[12\]](#)
- 2. Upload Progress:** Implement retry logic and background uploads for reliability
- 3. IAP Testing:** Use StoreKit Configuration File for local testing before App Store Connect [\[17\]](#)
- 4. Subscription Checks:** Verify subscription status on every protected API call
- 5. Signed URLs:** Generate download URLs with 60min expiry; regenerate on expiry
- 6. Security:** Never store Apple receipts; verify and extract info only [\[17\]](#)
- 7. Video Storage:** Structure GCS as /videos/{uuid}/{filename} for organization [\[14\]](#)

This spec is ready for AI code generation tools. Each section provides exact file paths, schemas, and implementation details needed to generate working code.

*

1. <https://qubit.capital/blog/matchmaking-platforms-tools>
2. <https://investormatch.ai/founders/>
3. <https://www.heygen.com/video/startup-pitch-video-maker>
4. <https://www.revenuecat.com/blog/growth/apple-eu-dma-update-june-2025/>
5. <https://www.motion.com/use-cases/en/startup-pitch-video-generator>
6. <https://www.paddle.com/blog/apple-revises-eu-app-store-rules-what-developers-need-to-know-2025>
7. <https://www.synthesia.io>
8. <https://pitch.com>
9. https://www.reddit.com/r/startups/comments/ivttg0/best_animation_software_for_pitching_app_idea/
10. <https://blueoceancompetition.org/powerful-apps-for-creating-your-successful-blue-ocean-pitch/>
11. https://www.youtube.com/watch?v=C_O4IXww_A0
12. <https://docs.expo.dev/versions/latest/sdk/camera/>
13. <https://www.youtube.com/watch?v=n7xFdFx88rg>
14. <https://stackoverflow.com/questions/71423715/signed-url-for-video-upload-causing-problem-in-gcp>
15. <https://www.youtube.com/watch?v=fd6lqZ32axY>
16. <https://www.youtube.com/watch?v=ubMSuFj-loY>
17. <https://www.createwithswift.com/implementing-subscriptions-in-app-purchases-with-storekit-2/>
18. <https://github.com/expo/expo/issues/33042>

19. <https://www.smashingmagazine.com/2018/04/audio-video-recording-react-native-expo/>
20. <https://stackoverflow.com/questions/79628624/can-i-record-system-audio-and-video-simultaneously-using-expo-react-native>