RollCall Build Sheet — FA³–FA■ Essentials

Minimal, print■friendly spec to generate the MVP app via AI builder (no frills).

Scope

MVP vertical video app focused on authenticity. Build via AI agents using Next.js + TypeScript, Supabase (Postgres), Prisma, simple upload pipeline (Cloudinary or tus), Vercel deploy.

Core Principles

• No beauty/face■altering filters. • 60■second max video. • Trim■only at launch. • Transparent ranking. • Evidence artifacts after each step.

1) MVP Feature Checklist

- Auth: email + magic link (Supabase).
- Feed: vertical autoplay, infinite scroll, like, comment, share link.
- Record/Upload: ≤60s, trim-only, auto-volume normalize; reject >60s.
- Post Page: video, caption (≤80 chars), time, likes, comments.
- Profile: user posts list; settings (EN/ES toggle, caption size).
- Moderation: report → review queue; transparency panel (“why I’m seeing this”).
- Ranking: recency x engagement (24h capped). Chronological fallback.
- Evidence: plan.md, schema.png, api.md, test-report.html, deploy-log.md + preview URL.

Editing Layer (start simple, swappable later)

- Provider: "cloudinary" (server transforms) OR "ffmpeg" (client ffmpeg.wasm) OR "sdk" (IMG.
- Start flags: trim=true, merge=false, speed=false, text=false, music=false, filters.beauty=fals

Lives Placeholder (don’t build yet)

Tables: Stream(id, creatorId, status, ingestUrl, playbackUrl, startedAt, endedAt)
LiveChatMessage(id, streamId, userId, body, createdAt)
Routes: POST /api/live/create | /start | /end | GET /api/live/:id
Provider later: ivs | mux | livepeer | agora

2) Data Model (Prisma schema)

```
model User {
  id           String   @id @default(cuid())
  email        String   @unique
  handle       String   @unique
  displayName  String
  bio          String?
  createdAt    DateTime @default(now())
  posts        Post[]
  likes        Like[]
  comments     Comment[]
  language     String   @default("en")
}

model Post {
  id          String   @id @default(cuid())
  authorId    String
  author      User     @relation(fields: [authorId], references: [id])
  videoUrl    String
  thumbUrl    String
  durationSec Int
  caption     String   @db.VarChar(80)
  hash        String   @unique
  createdAt   DateTime @default(now())
  likes       Like[]
  comments    Comment[]
  visibility  String   @default("public") // public, private, review
  flags       Int      @default(0)
  score       Float    @default(0)
}

model Like {
  userId String
  postId String
  user   User @relation(fields: [userId], references: [id])
  post   Post @relation(fields: [postId], references: [id])
  createdAt DateTime @default(now())
  @@id([userId, postId])
}

model Comment {
  id       String   @id @default(cuid())
  postId   String
  userId   String
  body     String   @db.VarChar(200)
  createdAt DateTime @default(now())
  post     Post     @relation(fields: [postId], references: [id])
  user     User     @relation(fields: [userId], references: [id])
}
```

3) API Routes (contracts)

```
POST /api/upload            -> {videoUrl, thumbUrl, hash, durationSec}  // reject if >60s
POST /api/posts             body: {caption, videoUrl, thumbUrl, durationSec, hash, edits?}
GET  /api/feed?cursor=...   -> list of posts (ranked), pagination
POST /api/posts/:id/like    -> 200
POST /api/posts/:id/unlike  -> 200
POST /api/posts/:id/comment body: {body} (<=200 chars) -> 200
POST /api/moderation/report body: {postId, reason} -> {queueId}

// FA³ transparency & moderation
GET  /api/transparency/events?userId=... -> events
POST /api/moderation/act body: {postId, action, notes?} -> 200

// FA■ wallet (placeholder: can be OFF initially)
GET  /api/wallet/balance -> {balance, currency}
POST /api/wallet/tip     -> {toUserId, postId?, amountCents} -> {eventId}
POST /api/payouts/request -> {amountCents} -> {payoutId}

// FA■ metrics & feedback
POST /api/metrics/log     -> {key, postId?, value?} -> 200
POST /api/feedback/create -> {kind, message, postId?} -> {id}
```

4) Feature Flags (features.json)

```
{
  "editing": {
    "provider": "cloudinary",
    "trim": true, "merge": false, "speed": false, "text": false, "music": false,
    "filters": { "beauty": false, "faceMorph": false }
  },
  "fa3": { "enabled": true,  "transparencyPanel": true },
  "fa4": { "enabled": false, "wallet": true, "tips": true, "revShare": {"creator": 0.93, "platfo
  "fa5": { "enabled": false, "metrics": true, "a_b": true },
  "live":{ "enabled": false, "provider": "stub" }
}
```

Policy Gates to Enforce

```
- Reject uploads > 60s; show friendly toast.
- Never expose beauty/face-morph filters when filters.beauty=false.
- Add small 'UNFILTERED' badge on posts recorded in-app.
- Rate-limit likes, comments, and reports to prevent spam.
```

5) Milestones & AI■Builder Commands

Milestones:
M1: Auth + Feed scaffold → preview URL + plan.md/schema.png/api.md
M2: Upload + Post create (≤60s, trim-only) → tests + preview
M3: Likes + Comments + Ranking → tests + preview
M4: Report + Review queue + Transparency Panel → preview
M5: Polish + i18n (EN/ES) + Production deploy

AI■Builder Commands (say these literally):
- Read this spec and confirm modules and data models.
- Create Next.js app with Supabase auth, Prisma, Tailwind. Show plan.md.
- Implement Feed page with autoplay + infinite scroll. Deploy preview.
- Add Upload (≤60s, trim-only) + reject >60s with tests. Re-deploy preview.
- Create Post page (+ likes & comments endpoints). Show test report.
- Implement ranking (recency + engagement 24h cap). Show function + docs.
- Add /report → ReviewQueue + Transparency events. Re-deploy preview.
- Generate artifacts and provide preview + production URLs.