

BACKEND_STRUCTURE.md — VERDICT

AI-Powered Deposition Coaching & Trial Preparation Platform
Version: 1.0.0 — Hackathon Edition | February 21, 2026
Runtime: Node.js 22.13.0 LTS | Framework: Fastify 5.2.1 | ORM: Prisma 6.3.1

TABLE OF CONTENTS

- 1. [Architecture Overview](#)
- 2. [Database Schema](#)
- 3. [API Endpoints](#)
- 4. [Authentication & Authorization](#)
- 5. [Data Validation Rules](#)
- 6. [Error Handling](#)
- 7. [Caching Strategy](#)
- 8. [Rate Limiting](#)
- 9. [Database Migrations](#)
- 10. [Backup & Recovery](#)
- 11. [API Versioning](#)

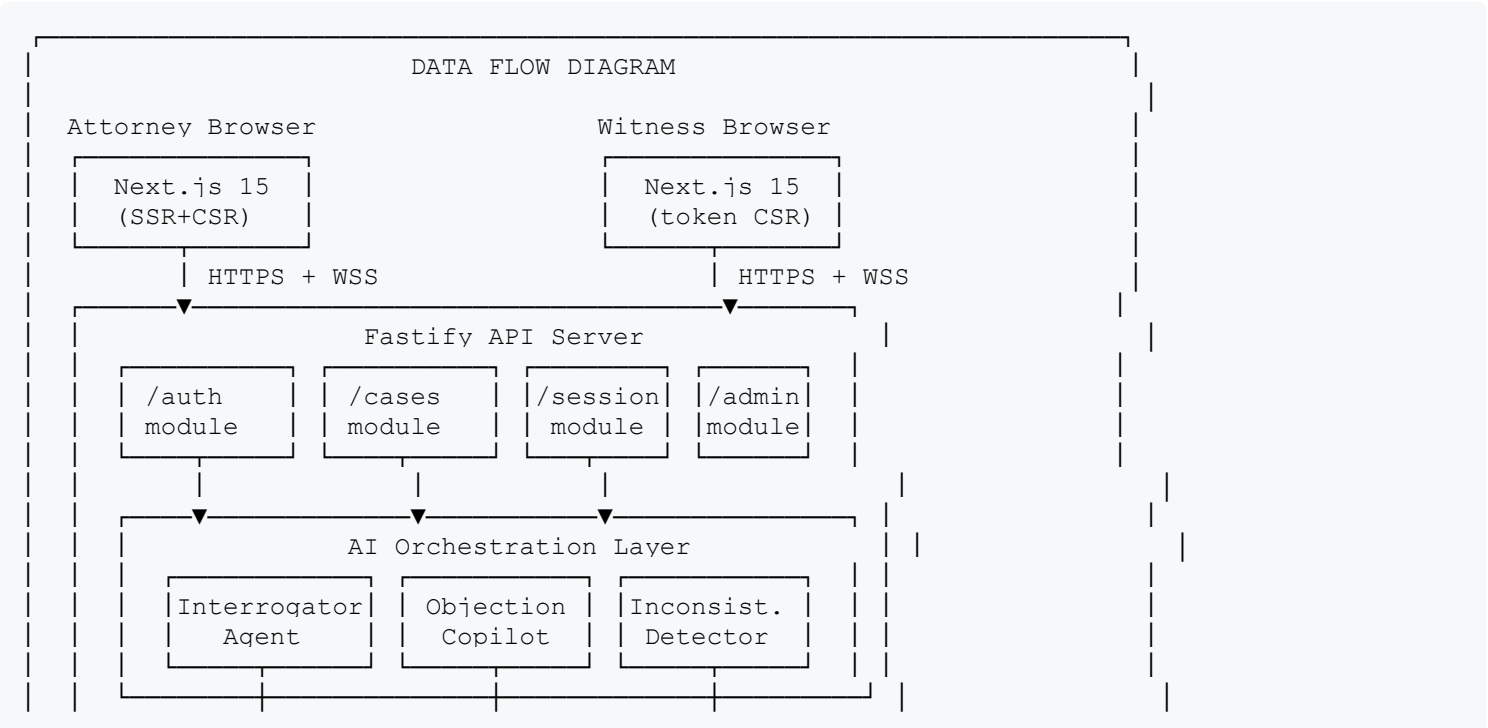
1. ARCHITECTURE OVERVIEW

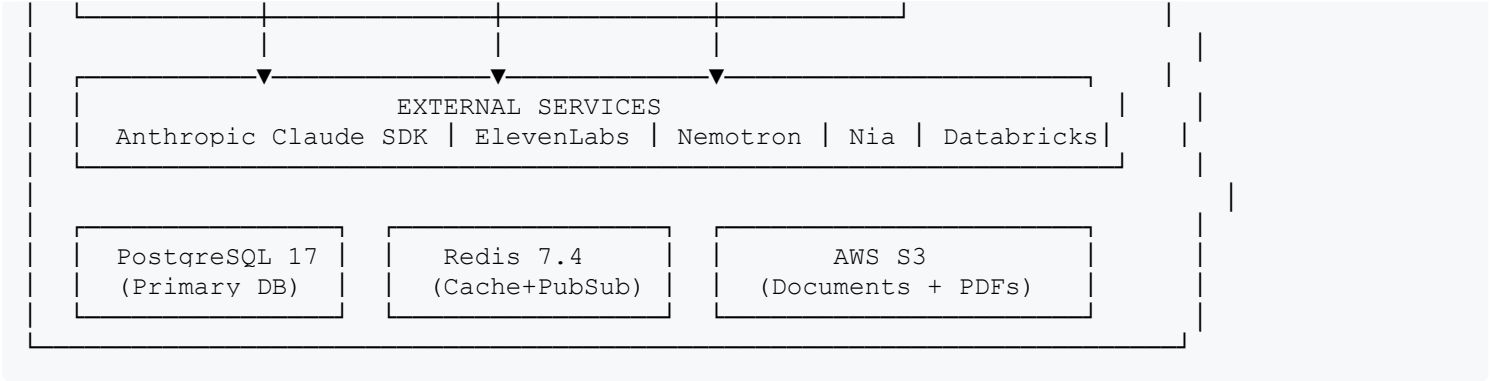
System Architecture Pattern

RESTful API + WebSocket (Socket.io)

VERDICT's backend is a **Fastify monolith with clear module boundaries**, exposing:

- **REST API** at `/api/v1/*` — CRUD operations, document ingestion, brief generation
- **WebSocket namespaces** at `/ws/session/:sessionId` — live session real-time events (alerts, transcript, agent status)
- **AI Agent pipeline** — server-side orchestration of Claude SDK → Nia → Nemotron per session event





Authentication Strategy

Method	Flow	Storage
JWT (Access Token)	httpOnly cookie, 8h TTL	Client cookie only
JWT (Refresh Token)	httpOnly cookie, 30d TTL	Cookie + Redis allowlist
SAML 2.0	Okta/AzureAD → /auth/saml/callback	Issues JWT on success
Witness Token	nanoid(24) URL param	Redis with 72h TTL
Brief Share Token	nanoid(24) query param	PostgreSQL with expiry timestamp

Caching Strategy (Summary)

Layer	Store	What	TTL
Session tokens	Redis	Refresh token allowlist	30 days
Witness tokens	Redis	witness:{token} → sessionId	72 hours
Rate limit counters	Redis	Per-IP / per-user endpoint counters	1 minute
Session event buffer	Redis	Live session events (60s auto-flush)	300 seconds
Dashboard stats	Redis	Firm-level aggregated stats	5 minutes
Ingestion status	Redis	ingestion:{documentId} → status + progress	10 minutes

2. DATABASE SCHEMA

Convention: All tables use snake_case. All IDs are CUID (not UUID) generated by Prisma. All timestamps are UTC. All monetary values and scores are stored as integers or floats, never VARCHAR.

Table: firms

Purpose: Stores law firm accounts — the top-level tenant entity. All other data is scoped to a firm.

Column	Type	Constraints	Description
--------	------	-------------	-------------

id	VARCHAR (30)	PK, NOT NULL	CUID generated by Prisma
name	VARCHAR (255)	NOT NULL	Law firm display name
slug	VARCHAR (100)	UNIQUE, NOT NULL	URL-safe firm identifier (e.g., "kirkland-ellis")
sso_metadata_url	VARCHAR (2048)	NULL	SAML 2.0 IdP metadata URL for enterprise SSO
sso_metadata_xml	TEXT	NULL	Raw SAML metadata XML (fallback if URL unavailable)
sso_enabled	BOOLEAN	NOT NULL, DEFAULT FALSE	Whether SSO is configured and active
password_login_enabled	BOOLEAN	NOT NULL, DEFAULT TRUE	Whether email/password login is permitted
retention_days	INTEGER	NOT NULL, DEFAULT 90	Days to retain session/case data before deletion
sentinel_enabled	BOOLEAN	NOT NULL, DEFAULT FALSE	Firm-level Behavioral Sentinel (P1.4) toggle
seats	INTEGER	NOT NULL	Maximum number of active users
plan	VARCHAR (50)	NOT NULL, DEFAULT 'trial'	'trial', 'starter', 'professional', 'enterprise'
plan_expires_at	TIMESTAMP	NULL	NULL = no expiry
setup_complete	BOOLEAN	NOT NULL, DEFAULT FALSE	True after onboarding Step 4 completes
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Record creation time
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification time

Indexes:

- PRIMARY KEY (id)
- UNIQUE INDEX firms_slug_unique (slug)
- INDEX firms_plan_idx (plan)

Relationships:

- One firm → Many users (CASCADE DELETE)
- One firm → Many cases (CASCADE DELETE)

Table: users

Purpose: Attorneys, associates, paralegals, and admins within a firm.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Owning firm
email	VARCHAR(255)	UNIQUE, NOT NULL	Login email; RFC 5321 format
name	VARCHAR(255)	NOT NULL	Full display name
role	VARCHAR(30)	NOT NULL	'PARTNER', 'ASSOCIATE', 'PARALEGAL', 'ADMIN'
password_hash	VARCHAR(60)	NULL	bcrypt hash (cost 12); NULL if SSO-only user
password_reset_token	VARCHAR(64)	NULL	Hashed reset token; cleared on use
password_reset_expires_at	TIMESTAMP	NULL	Expiry for password reset token
email_verified	BOOLEAN	NOT NULL, DEFAULT FALSE	True after email verification click
email_verify_token	VARCHAR(64)	NULL	Hashed email verification token; cleared on use
notify_brief_ready	BOOLEAN	NOT NULL, DEFAULT TRUE	Email when coaching brief generated
notify_plateau_alert	BOOLEAN	NOT NULL, DEFAULT TRUE	Email when witness score plateaus
notify_session_reminder	BOOLEAN	NOT NULL, DEFAULT TRUE	Email 24h before session
notify_ingestion_complete	BOOLEAN	NOT NULL, DEFAULT TRUE	Email when document indexing finishes
last_login_at	TIMESTAMP	NULL	Timestamp of most recent successful login
login_attempts	INTEGER	NOT NULL, DEFAULT 0	Failed login counter; reset on success
locked_until	TIMESTAMP	NULL	Account locked until this time (brute-force protection)
is_active	BOOLEAN	NOT NULL, DEFAULT TRUE	FALSE = deactivated (soft delete)
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Record creation

updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification
------------	-----------	-------------------------	-------------------

Indexes:

- PRIMARY KEY (id)
- UNIQUE INDEX users_email_unique (email)
- INDEX users_firm_id_idx (firm_id)
- INDEX users_firm_role_idx (firm_id, role) — admin panel user listing
- INDEX users_active_firm_idx (firm_id, is_active) — seat count queries

Relationships:

- Many users → One firms (firm_id)
- One user → Many cases (as owner)
- One user → Many refresh_tokens

Table: refresh_tokens

Purpose: Allowlist of active refresh tokens. Enables selective revocation (logout one device) or full revocation (logout all devices).

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
user_id	VARCHAR(30)	FK → users.id ON DELETE CASCADE, NOT NULL	Owning user
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Firm context (for multi-tenant queries)
token_hash	VARCHAR(64)	UNIQUE, NOT NULL	SHA-256 hash of the raw token
device_hint	VARCHAR(255)	NULL	User-Agent snippet for "active sessions" UI display
ip_address	VARCHAR(45)	NULL	IPv4 or IPv6; stored for security audit
expires_at	TIMESTAMP	NOT NULL	30 days from issuance
revoked_at	TIMESTAMP	NULL	NULL = active; set on logout
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Issuance time
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification

Indexes:

- PRIMARY KEY (id)
- UNIQUE INDEX refresh_tokens_hash_unique (token_hash)
- INDEX refresh_tokens_user_id_idx (user_id)
- INDEX refresh_tokens_expires_at_idx (expires_at) — for nightly expiry cleanup job

Relationships:

- Many refresh_tokens → One users

Table: cases

Purpose: The primary entity — a legal matter for which deposition prep is being conducted.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Owning firm
owner_id	VARCHAR(30)	FK → users.id ON DELETE RESTRICT, NOT NULL	Attorney who created the case
name	VARCHAR(255)	NOT NULL	e.g., "Chen v. Metropolitan Hospital"
case_type	VARCHAR(50)	NOT NULL	'MEDICAL_MALPRACTICE', 'EMPLOYMENT_DISCRIMINATION', 'COMMERCIAL_DISPUTE', 'CONTRACT_BREACH', 'OTHER'
case_type_custom	VARCHAR(255)	NULL	Free-text for 'OTHER' case type
opposing_firm	VARCHAR(255)	NULL	Opposing counsel firm name
deposition_date	DATE	NULL	Target deposition date for countdown badge
is_archived	BOOLEAN	NOT NULL, DEFAULT FALSE	Soft delete / archive
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Record creation
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification

Indexes:

- PRIMARY KEY (id)
- INDEX cases_firm_id_idx (firm_id)
- INDEX cases_owner_id_idx (owner_id)
- INDEX cases_firm_archived_idx (firm_id, is_archived) — dashboard listing
- INDEX cases_deposition_date_idx (firm_id, deposition_date) — sorted by soonest deposition

Relationships:

- Many cases → One firms
- Many cases → One users (owner)

- One case → Many documents
- One case → Many witnesses
- One case → Many sessions

Table: documents

Purpose: Case files uploaded by the attorney — PDFs, DOCX, TXT. Tracked through the Nia ingestion pipeline.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
case_id	VARCHAR(30)	FK → cases.id ON DELETE CASCADE, NOT NULL	Owning case
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Denormalized for row-level security
uploader_id	VARCHAR(30)	FK → users.id ON DELETE SET NULL, NULL	User who uploaded; NULL if user deactivated
filename	VARCHAR(500)	NOT NULL	Original filename as uploaded
s3_key	VARCHAR(1024)	NOT NULL, UNIQUE	S3 object key: {firmId}/{caseId}/{docId}/{filename}
file_size_bytes	BIGINT	NOT NULL	File size in bytes (max 209,715,200 = 200MB)
mime_type	VARCHAR(100)	NOT NULL	'application/pdf', 'application/vnd.openxmlformats-officedocument.wordprocessingml.document', 'text/plain'
doc_type	VARCHAR(50)	NOT NULL	'PRIOR_DEPOSITION', 'MEDICAL_RECORDS', 'FINANCIAL_RECORDS', 'CORRESPONDENCE', 'EXHIBIT', 'OTHER'
page_count	INTEGER	NULL	Set after ingestion succeeds
ingestion_status	VARCHAR(30)	NOT NULL, DEFAULT 'PENDING'	'PENDING', 'UPLOADING', 'INDEXING', 'READY', 'FAILED'
ingestion_started_at	TIMESTAMP	NULL	When Nia pipeline kicked off

ingestion_completed_at	TIMESTAMP	NULL	When status became READY or FAILED
ingestion_error	TEXT	NULL	Error message if status = FAILED
nia_index_id	VARCHAR(255)	NULL	Nia's internal document identifier post-indexing
extracted_facts	JSONB	NULL	{ parties: [], keyDates: [], disputedFacts: [], priorStatements: [] }
facts_confirmed_at	TIMESTAMP	NULL	When attorney confirmed extracted facts
file_hash	VARCHAR(64)	NULL	SHA-256 of file content for duplicate detection
version	INTEGER	NOT NULL, DEFAULT 1	Increments when document is re-uploaded
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Upload initiated
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last status change

Indexes:

- PRIMARY KEY (id)
- UNIQUE INDEX documents_s3_key_unique (s3_key)
- INDEX documents_case_id_idx (case_id)
- INDEX documents_firm_id_idx (firm_id)
- INDEX documents_ingestion_status_idx (case_id, ingestion_status) — readiness gate queries
- INDEX documents_file_hash_case_idx (case_id, file_hash) — duplicate detection
- GIN INDEX documents_extracted_facts_gin (extracted_facts) — JSONB search on prior statements

Relationships:

- Many documents → One cases
- Many documents → One firms

Table: witnesses

Purpose: Individual deponent records within a case.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
case_id	VARCHAR(30)	FK → cases.id ON DELETE	Owning case

		CASCADE, NOT NULL	
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Denormalized for isolation
name	VARCHAR(255)	NOT NULL	Witness full name
email	VARCHAR(255)	NOT NULL	Email for practice link delivery
role	VARCHAR(50)	NOT NULL	'DEFENDANT', 'PLAINTIFF', 'EXPERT', 'CORPORATE_REPRESENTATIVE', 'OTHER'
notes	TEXT	NULL	Attorney prep notes (e.g., known weak points)
linked_document_ids	VARCHAR(30) []	NOT NULL, DEFAULT "	PostgreSQL array of document IDs associated with this witness
session_count	INTEGER	NOT NULL, DEFAULT 0	Denormalized count — incremented on each session creation
latest_score	INTEGER	NULL	Denormalized — updated after each session brief
baseline_score	INTEGER	NULL	Score from Session 1 — never updated
plateau_detected	BOOLEAN	NOT NULL, DEFAULT FALSE	Set to TRUE when no improvement over 3 sessions
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Record creation
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification

Indexes:

- PRIMARY KEY (id)
- INDEX witnesses_case_id_idx (case_id)
- INDEX witnesses_firm_id_idx (firm_id)
- INDEX witnesses_plateau_idx (firm_id, plateau_detected) — dashboard plateau alerts
- GIN INDEX witnesses_linked_doc_ids_gin (linked_document_ids) — document-witness association queries

Relationships:

- Many witnesses → One cases
- One witness → Many sessions

Table: sessions

Purpose: A single deposition simulation session between an attorney and a witness.

Column	Type	Constraints	Description
id	VARCHAR (30)	PK, NOT NULL	CUID
case_id	VARCHAR (30)	FK → cases.id ON DELETE CASCADE, NOT NULL	Owning case
witness_id	VARCHAR (30)	FK → witnesses.id ON DELETE CASCADE, NOT NULL	Witness being prepared
firm_id	VARCHAR (30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Denormalized
attorney_id	VARCHAR (30)	FK → users.id ON DELETE SET NULL, NULL	Attorney who configured the session
session_number	INTEGER	NOT NULL	1-based sequence per witness (1 = first session)
status	VARCHAR (30)	NOT NULL, DEFAULT 'CONFIGURED'	'CONFIGURED', 'LOBBY', 'ACTIVE', 'PAUSED', 'COMPLETE', 'FAILED'
duration_minutes	INTEGER	NOT NULL	15, 30, 45, or 60
focus_areas	VARCHAR (50) []	NOT NULL, DEFAULT "	'TIMELINE_CHRONOLOGY', 'FINANCIAL_DETAILS', 'COMMUNICATIONS', 'RELATIONSHIPS', 'ACTIONS_TAKEN', 'PRIOR_STATEMENTS'
aggression_level	VARCHAR (20)	NOT NULL, DEFAULT 'STANDARD'	'STANDARD', 'ELEVATED', 'HIGH_STAKES'
objection_copilot	BOOLEAN	NOT NULL, DEFAULT TRUE	Objection Copilot (P0.2) active
behavioral_sentinel	BOOLEAN	NOT NULL, DEFAULT FALSE	Behavioral Sentinel (P1.4) active
witness_consented	BOOLEAN	NOT NULL, DEFAULT FALSE	Witness granted camera permission for Sentinel
witness_token	VARCHAR (24)	UNIQUE, NULL	nanoid token for witness access link
witness_token_expires_at	TIMESTAMP	NULL	72h from generation

witness_joined_at	TIMESTAMP	NULL	When witness entered the session lobby
started_at	TIMESTAMP	NULL	When attorney clicked "Begin Session"
paused_at	TIMESTAMP	NULL	Most recent pause timestamp
ended_at	TIMESTAMP	NULL	When session completed or was ended early
question_count	INTEGER	NOT NULL, DEFAULT 0	Incrementing counter per question delivered
session_score	INTEGER	NULL	0–100; computed post-session by Review Orchestrator
consistency_rate	FLOAT	NULL	0.0–1.0; proportion of answers with no flags
improvement_delta	INTEGER	NULL	session_score minus witness.baseline_score
transcript_raw	TEXT	NULL	Full speaker-tagged session transcript
nia_session_context_id	VARCHAR(255)	NULL	Nia session context handle for this session's RAG queries
prior_weak_areas	JSONB	NULL	Weak areas from prior sessions injected into Interrogator
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Session record creation
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification

Indexes:

- PRIMARY KEY (id)
- UNIQUE INDEX sessions_witness_token_unique (witness_token)
- INDEX sessions_case_id_idx (case_id)
- INDEX sessions_witness_id_idx (witness_id)
- INDEX sessions_firm_id_idx (firm_id)
- INDEX sessions_attorney_id_idx (attorney_id)
- INDEX sessions_status_idx (firm_id, status) — active session queries
- UNIQUE INDEX sessions_witness_number_unique (witness_id, session_number) — enforces sequential numbering

Relationships:

- Many sessions → One cases
 - Many sessions → One witnesses
 - Many sessions → One users (attorney)
 - One session → Many alerts
 - One session → One brief (after completion)
 - One session → Many session_events
-

Table: session_events

Purpose: Granular event log for every meaningful action within a live session. Auto-flushed from Redis buffer every 60 seconds. Backs the Databricks Delta Live Tables streaming pipeline.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
session_id	VARCHAR(30)	FK → sessions.id ON DELETE CASCADE, NOT NULL	Owning session
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Denormalized
event_type	VARCHAR(50)	NOT NULL	See event types below
question_number	INTEGER	NULL	Q-number at time of event
speaker	VARCHAR(20)	NULL	'INTERROGATOR', 'WITNESS', 'SYSTEM'
text_content	TEXT	NULL	Question or answer text
audio_latency_ms	INTEGER	NULL	ElevenLabs TTS latency in milliseconds
stt_latency_ms	INTEGER	NULL	ElevenLabs STT latency in milliseconds
emotion_vectors	JSONB	NULL	Behavioral Sentinel: { fear: 0.82, contempt: 0.12, ... }
facs_au_vector	JSONB	NULL	Raw FACS Action Unit values { au4: 0.8, au6: 0.2, ... }
facs_duration_ms	INTEGER	NULL	Duration of detected expression in milliseconds
metadata	JSONB	NULL	Event-type-specific extra data
occurred_at	TIMESTAMP	NOT NULL	Precise event timestamp
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	DB insert time (may be later than occurred_at due to buffer)

Event Types:

SESSION START	— session began
SESSION PAUSE	— attorney paused
SESSION RESUME	— attorney resumed
SESSION END	— session ended (timer or early)
QUESTION DELIVERED	— Interrogator delivered question (with audio_latency_ms)
ANSWER RECEIVED	— Witness answer transcribed (with stt_latency_ms)
HESITATION DETECTED	— Witness silence > 4 seconds
TOPIC CHANGE	— Interrogator moved to next focus area
OBJECTION FIRED	— Objection Copilot fired alert
INCONSISTENCY_FIRED	— Inconsistency Detector fired alert

COMPOSURE ALERT	— Behavioral Sentinel fired (with emotion_vectors, facts_au_vector)
ATTORNEY NOTE	— Attorney timestamped annotation
WITNESS_DISCONNECT	— Witness WebSocket dropped
WITNESS_RECONNECT	— Witness WebSocket restored
AGENT_DEGRADED	— API fallback activated (e.g., Nemotron → Claude-only)

Indexes:

- PRIMARY KEY (id)
- INDEX session_events_session_id_idx (session_id)
- INDEX session_events_firm_id_idx (firm_id)
- INDEX session_events_type_idx (session_id, event_type) — event type filtering
- INDEX session_events_occurred_at_idx (session_id, occurred_at) — time-ordered streaming
- GIN INDEX session_events_emotion_vectors_gin (emotion_vectors) — Behavioral Sentinel queries

Relationships:

- Many session_events → One sessions

Table: alerts

Purpose: Persistent record of every objection, inconsistency, and composure alert — both live-fired and secondary-review-queued.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
session_id	VARCHAR(30)	FK → sessions.id ON DELETE CASCADE, NOT NULL	Owning session
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Denormalized
alert_type	VARCHAR(50)	NOT NULL	'OBJECTION', 'INCONSISTENCY', 'INCONSISTENCY_SECONDARY', 'COMPOSURE'
question_number	INTEGER	NOT NULL	Q-number when alert fired
fired_at	TIMESTAMP	NOT NULL	When the alert was generated
question_text	TEXT	NOT NULL	The question that triggered the analysis
answer_text	TEXT	NULL	Witness answer (for INCONSISTENCY, COMPOSURE)
fre_rule	VARCHAR(50)	NULL	e.g., 'FRE 611(c)' — for OBJECTION alerts
objection_category	VARCHAR(50)	NULL	'LEADING', 'HEARSAY', 'COMPOUND', 'ASSUMES_FACTS', 'SPECULATION'

prior_quote	TEXT	NULL	Exact prior sworn statement for INCONSISTENCY alerts
prior_document_id	VARCHAR(30)	NULL	FK → documents.id (nullable — document may be deleted)
prior_document_page	INTEGER	NULL	Page number in prior document
prior_document_line	INTEGER	NULL	Line number in prior document
contradiction_confidence	FLOAT	NULL	Nemotron score 0.0–1.0
emotion_category	VARCHAR(30)	NULL	'FEAR', 'CONTEMPT', 'DISGUST', 'ANGER', 'SURPRISE' — for COMPOSURE
emotion_duration_ms	INTEGER	NULL	Duration of expression in ms
behavioral_a_u_vectors	JSONB	NULL	{ au4: 0.8, au20: 0.7, ... }
impeachment_risk	VARCHAR(20)	NULL, DEFAULT 'STANDARD'	'STANDARD', 'HIGH' — HIGH when Sentinel corroborates
is_live_fired	BOOLEAN	NOT NULL, DEFAULT TRUE	FALSE = secondary queue (confidence 0.50–0.74)
attorney_decision	VARCHAR(20)	NULL	'CONFIRMED', 'REJECTED', 'ANNOTATED'
attorney_note	TEXT	NULL	Free-text annotation from attorney
decision_at	TIMESTAMP	NULL	When attorney acted on the alert
in_brief	BOOLEAN	NOT NULL, DEFAULT FALSE	Whether this alert appears in the coaching brief
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Alert record created
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification

Indexes:

- PRIMARY KEY (id)
- INDEX alerts_session_id_idx (session_id)
- INDEX alerts_firm_id_idx (firm_id)
- INDEX alerts_type_session_idx (session_id, alert_type) — brief generation queries
- INDEX alerts_decision_idx (session_id, attorney_decision) — confirmed flags for brief
- INDEX alerts_persisting_idx (session_id, attorney_decision, alert_type) — multi-session persisting flags
- INDEX alerts_fired_at_idx (session_id, fired_at) — time-ordered alert replay

Relationships:

- Many alerts → One sessions

Table: briefs

Purpose: Post-session coaching brief generated by the Review Orchestrator. One brief per completed session.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
session_id	VARCHAR(30)	FK → sessions.id ON DELETE CASCADE, NOT NULL, UNIQUE	One-to-one with session
firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Denormalized
witness_id	VARCHAR(30)	FK → witnesses.id ON DELETE CASCADE, NOT NULL	Denormalized for witness profile queries
generation_status	VARCHAR(30)	NOT NULL, DEFAULT 'PENDING'	'PENDING', 'GENERATING', 'COMPLETE', 'FAILED'
generation_started_at	TIMESTAMP	NULL	When Review Orchestrator started
generation_completed_at	TIMESTAMP	NULL	When brief became available
generation_error	TEXT	NULL	Error message if FAILED
session_score	INTEGER	NULL	0–100 composite score
consistency_rate	FLOAT	NULL	0.0–1.0
improvement_delta	INTEGER	NULL	vs witness.baseline_score
confirmed_flags	INTEGER	NOT NULL, DEFAULT 0	Count of confirmed inconsistencies
objection_count	INTEGER	NOT NULL, DEFAULT 0	Total objections fired
composure_alert_count	INTEGER	NOT NULL, DEFAULT 0	Total composure alerts
top_recommendations	TEXT[]	NOT NULL, DEFAULT "	Array of 3 coaching recommendation strings

narrative_text	TEXT	NULL	Full Claude-generated narrative (attorney-quality prose)
weakness_map_scores	JSONB	NULL	{ timeline: 78, financial: 34, communications: 65, relationships: 80, actions: 71, prior statements: 55, composure: 60 }
elevenlabs_audio_manifest	JSONB	NULL	[{ alertId, s3Key, durationMs }] — coach voice clips
pdf_s3_key	VARCHAR(1024)	NULL	S3 key for rendered PDF
pdf_generated_at	TIMESTAMP	NULL	When PDF render completed
share_token	VARCHAR(24)	UNIQUE, NULL	nanoid — 7-day expiring share link token
share_token_expires_at	TIMESTAMP	NULL	Expiry for share_token
share_token_issued_at	TIMESTAMP	NULL	When the share token was created
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Brief record created
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification

Indexes:

- PRIMARY KEY (id)
- UNIQUE INDEX briefs_session_id_unique (session_id) — one-to-one enforcement
- UNIQUE INDEX briefs_share_token_unique (share_token)
- INDEX briefs_firm_id_idx (firm_id)
- INDEX briefs_witness_id_idx (witness_id) — witness profile brief history
- INDEX briefs_generation_status_idx (generation_status) — polling for pending briefs
- INDEX briefs_share_token_expiry_idx (share_token_expires_at) — expired link cleanup

Relationships:

- One brief ↔ One sessions (1:1)
- Many briefs → One witnesses

Table: attorney_annotations

Purpose: Timestamped free-text notes added by attorney during a live session. Separate from alert decisions.

Column	Type	Constraints	Description
id	VARCHAR(30)	PK, NOT NULL	CUID
session_id	VARCHAR(30)	FK → sessions.id ON DELETE CASCADE, NOT NULL	Owning session

firm_id	VARCHAR(30)	FK → firms.id ON DELETE CASCADE, NOT NULL	Denormalized
attorney_id	VARCHAR(30)	FK → users.id ON DELETE SET NULL, NULL	Annotating attorney
question_number	INTEGER	NULL	Q-number at time of annotation
note_text	TEXT	NOT NULL	The annotation content
session_timestamp_ms	INTEGER	NOT NULL	Milliseconds into the session when note was added
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Annotation created
updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()	Last modification

Indexes:

- PRIMARY KEY (id)
- INDEX annotations_session_id_idx (session_id)

3. API ENDPOINTS

Base URL: https://prod.verdict.law/api/v1
Content-Type: application/json (all requests and responses)
Auth header when JWT required: Cookie: access_token=Bearer.{jwt} (httpOnly)
All timestamps: ISO 8601 UTC (e.g., "2026-02-21T14:32:00.000Z")

AUTH MODULE

POST /auth/login

Purpose: Authenticates an attorney or admin with email/password. Issues access + refresh tokens.

Authentication: Public

Request Body:

```
{
  "email": "sarah.chen@kirklandellis.com",
  "password": "Secure!Pass#2026"
}
```

Validation Rules:

- email : required, valid RFC 5321 format, max 255 chars
- password : required, min 8 chars, max 128 chars

Response (200):

```
{
  "success": true,
  "data": {
    "user": {
      "id": "clxyz1234",
      "firmId": "clxyz_firm",
      "email": "sarah.chen@kirklandellis.com",
      "name": "Sarah Chen",
      "role": "PARTNER"
    }
  }
}
```

Sets two *httpOnly* cookies:

- `access_token` (8h, Secure, SameSite=Strict)
- `refresh_token` (30d, Secure, SameSite=Strict, HttpOnly)

Errors:

- 400 `VALIDATION_ERROR` — Missing or malformed fields
- 401 `INVALID_CREDENTIALS` — Email not found OR password mismatch (same message — no enumeration)
- 401 `EMAIL_NOT_VERIFIED` — Account not yet email-verified
- 403 `ACCOUNT_LOCKED` — Locked due to too many failed attempts (`locked_until` in future)
- 403 `SSO_REQUIRED` — Firm has `sso_enabled=true` and `password_login_enabled=false`
- 403 `ACCOUNT_INACTIVE` — `is_active=false`

Side Effects:

- On success: `last_login_at` updated; `login_attempts` reset to 0; refresh token inserted into `refresh_tokens` table
- On failure: `login_attempts` incremented; if `login_attempts >= 10`: `locked_until` set to `NOW() + 15` minutes

POST `/auth/logout`

Purpose: Revokes the current session's refresh token and clears auth cookies.

Authentication: Required

Request Body: (*none*)

Response (200):

```
{
  "success": true,
  "message": "Logged out successfully"
}
```

Side Effects:

- Refresh token row: `revoked_at` set to `NOW()`
- Clears `access_token` and `refresh_token` cookies
- Redis: `DEL session:{userId}:{tokenId}` if cached

POST `/auth/logout-all`

Purpose: Revokes ALL refresh tokens for the current user (log out every device).

Authentication: Required

Request Body: *(none)*

Response (200):

```
{
  "success": true,
  "message": "All sessions revoked",
  "sessionsRevoked": 3
}
```

Side Effects:

- All `refresh_tokens` rows for `user_id`: `revoked_at` set to NOW()
- Clears cookies

POST `/auth/refresh`

Purpose: Issues a new access token using a valid refresh token.

Authentication: Public (uses refresh token cookie)

Request Body: *(none — reads refresh_token cookie)*

Response (200):

```
{
  "success": true,
  "data": {
    "expiresAt": "2026-02-22T22:32:00.000Z"
  }
}
```

Sets new `access_token` cookie.

Errors:

- 401 `TOKEN_EXPIRED` — Refresh token past expiry
- 401 `TOKEN_REVOKED` — `revoked_at` is set
- 401 `TOKEN_NOT_FOUND` — No matching hash in `refresh_tokens`
- 403 `ACCOUNT_INACTIVE` — User `is_active=false`

GET `/auth/saml`

Purpose: Initiates SAML 2.0 SSO redirect to the firm's configured IdP.

Authentication: Public

Query Params:

- `firmSlug` (required) — identifies which IdP to redirect to

Response: HTTP 302 redirect to IdP

Errors:

- 404 FIRM_NOT_FOUND — No firm with that slug
 - 400 SSO_NOT_CONFIGURED — Firm exists but has no SAML metadata
-

POST /auth/saml/callback

Purpose: Handles SAML assertion from IdP. Provisions user if first login. Issues JWT.

Authentication: Public (validated via SAML assertion signature)

Request Body: SAML XML payload (application/x-www-form-urlencoded from IdP)

Response: HTTP 302 redirect to /dashboard

Side Effects:

- If user doesn't exist: provisioned with role=ASSOCIATE (admin can upgrade)
 - last_login_at updated
 - JWT access + refresh tokens issued
-

POST /auth/password/reset-request

Purpose: Sends password reset email. Rate-limited to prevent account enumeration.

Authentication: Public

Request Body:

```
{
  "email": "sarah.chen@kirklandellis.com"
}
```

Response (200): Always returns success (prevents email enumeration)

```
{
  "success": true,
  "message": "If an account exists, a reset email has been sent."
}
```

Side Effects:

- If user found AND firm allows password login: generates reset token, stores SHA-256 hash in password_reset_token, sets password_reset_expires_at = NOW() + 1h
- Sends password reset email via Resend

Rate Limit: 5 requests per email per hour

POST /auth/password/reset

Purpose: Completes password reset using the token from email.

Authentication: Public

Request Body:

```
{
  "token": "abc123def456...",
  "newPassword": "NewSecure!Pass#2026"
}
```

Validation Rules:

- `token`: required, 64 char hex string
- `newPassword`: min 10 chars, must contain uppercase, lowercase, digit, and special char

Response (200):

```
{
  "success": true,
  "message": "Password updated. Please log in."
}
```

Errors:

- 400 `TOKEN_INVALID` — No matching hashed token
- 400 `TOKEN_EXPIRED` — `password_reset_expires_at` in past
- 400 `VALIDATION_ERROR` — Password doesn't meet requirements

Side Effects:

- New `password_hash` written (bcrypt cost 12)
- `password_reset_token` and `password_reset_expires_at` cleared
- All existing refresh tokens revoked (forces re-login)

FIRM ONBOARDING MODULE

POST `/onboarding/activate`

Purpose: Step 1 of firm setup. Validates the firm invitation token and saves firm configuration.

Authentication: Public (requires firm invitation token in request body)

Request Body:

```
{
  "invitationToken": "abc123...",
  "firmName": "Kirkland & Ellis LLP",
  "adminName": "Sarah Chen",
  "adminEmail": "sarah.chen@kirklandellis.com",
  "authMethod": "sso",
  "ssoMetadataUrl": "https://kirkland.okta.com/app/saml/metadata"
}
```

Response (200):

```
{
  "success": true,
  "data": {
    "firmId": "clxyz_firm",
    "slug": "kirkland-ellis",
    "setupStep": 2
  }
}
```

```
}  
}
```

POST /onboarding/users

Purpose: Step 2 — Bulk provision users via CSV data or individual entries.

Authentication: Requires firm admin session (JWT issued after Step 1)

Request Body:

```
{  
  "users": [  
    { "name": "Marcus Webb", "email": "m.webb@kirklandellis.com", "role": "ASSOCIATE" },  
    { "name": "Lisa Park", "email": "l.park@kirklandellis.com", "role": "PARALEGAL" }  
  ]  
}
```

Response (200):

```
{  
  "success": true,  
  "data": {  
    "provisioned": 2,  
    "skipped": 0,  
    "invitationsSent": 2  
  }  
}
```

PATCH /onboarding/security

Purpose: Step 3 — Saves firm security settings and marks onboarding complete.

Authentication: Required (firm admin)

Request Body:

```
{  
  "retentionDays": 90,  
  "sentinelEnabled": false  
}
```

Response (200):

```
{  
  "success": true,  
  "data": { "setupComplete": true }  
}
```

Side Effects:

- `firms.setup_complete` set to TRUE
- Firm record created in Databricks Delta Lake via background job

CASES MODULE

GET /cases

Purpose: Lists all cases for the authenticated attorney's firm.

Authentication: Required

Query Params:

- `archived` (boolean, default: false)
- `sort` ('deposition_date_asc', 'created_at_desc' — default: 'deposition_date_asc')
- `page` (integer, default: 1)
- `limit` (integer, default: 20, max: 50)

Response (200):

```
{
  "success": true,
  "data": {
    "cases": [
      {
        "id": "clxyz_case",
        "name": "Chen v. Metropolitan Hospital",
        "caseType": "MEDICAL_MALPRACTICE",
        "depositionDate": "2026-03-15",
        "documentsReady": 3,
        "documentsTotal": 3,
        "witnessCount": 2,
        "latestSessionScore": 79,
        "createdAt": "2026-02-01T09:00:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 20,
      "total": 7
    }
  }
}
```

Caching: Redis key `cases:firm:{firmId}:list` — TTL 5 minutes. Invalidated on: case create, update, archive.

POST /cases

Purpose: Creates a new case.

Authentication: Required

Request Body:

```
{
  "name": "Chen v. Metropolitan Hospital",
  "caseType": "MEDICAL_MALPRACTICE",
  "opposingFirm": "Defense Partners LLP",
  "depositionDate": "2026-03-15"
}
```

Validation Rules:

- `name` : required, 3–255 chars, sanitized for XSS

- `caseType` : required, must be valid enum value
- `caseTypeCustom` : required when `caseType = "OTHER"`, max 255 chars
- `depositionDate` : optional, must be in future if provided

Response (201):

```
{
  "success": true,
  "data": {
    "id": "clxyz_case",
    "name": "Chen v. Metropolitan Hospital",
    "caseType": "MEDICAL_MALPRACTICE",
    "depositionDate": "2026-03-15",
    "createdAt": "2026-02-21T14:00:00.000Z"
  }
}
```

Side Effects:

- Invalidates `cases:firm:{firmId}:list` cache

GET `/cases/:caseId`

Purpose: Fetches full case detail including document and witness summary.

Authentication: Required (case must belong to authenticated user's firm)

Response (200):

```
{
  "success": true,
  "data": {
    "id": "clxyz_case",
    "name": "Chen v. Metropolitan Hospital",
    "caseType": "MEDICAL_MALPRACTICE",
    "opposingFirm": "Defense Partners LLP",
    "depositionDate": "2026-03-15",
    "documentsIngestionStatus": "READY",
    "documents": [
      {
        "id": "clxyz_doc",
        "filename": "chen_depo_2024.pdf",
        "docType": "PRIOR_DEPOSITION",
        "ingestionStatus": "READY",
        "pageCount": 247,
        "factsConfirmedAt": "2026-02-10T10:30:00.000Z"
      }
    ],
    "witnesses": [
      {
        "id": "clxyz_witness",
        "name": "Dr. Emily Chen",
        "role": "DEFENDANT",
        "sessionCount": 3,
        "latestScore": 79,
        "plateauDetected": false
      }
    ]
  }
}
```

Errors:

- 403 FORBIDDEN — Case belongs to different firm
 - 404 NOT_FOUND — No case with that ID
-

PATCH /cases/:caseId

Purpose: Updates case name, type, opposing firm, or deposition date.

Authentication: Required

Request Body:

```
{
  "name": "Chen v. Metropolitan Health System",
  "depositionDate": "2026-03-22"
}
```

Response (200): Updated case object (same shape as GET)

Side Effects: Cache invalidation for this case and the firm's case list.

DELETE /cases/:caseId

Purpose: Soft-archives a case (`is_archived = true`).

Authentication: Required (case owner or firm admin)

Response (200):

```
{
  "success": true,
  "message": "Case archived"
}
```

DOCUMENTS MODULE

POST /cases/:caseId/documents/presign

Purpose: Generates an S3 pre-signed URL for direct browser-to-S3 upload. Called BEFORE the file upload.

Authentication: Required

Request Body:

```
{
  "filename": "chen_depo_2024.pdf",
  "mimeType": "application/pdf",
  "fileSizeBytes": 15728640
}
```

Validation Rules:

- `filename` : required, max 500 chars

- `mimeType`: must be one of `application/pdf`, `application/vnd.openxmlformats-officedocument.wordprocessingml.document`, `text/plain`
- `fileSizeBytes`: required, max 209,715,200 (200MB)

Response (200):

```
{
  "success": true,
  "data": {
    "documentId": "clxyz_doc",
    "uploadUrl": "https://s3.amazonaws.com/verdict-docs/...",
    "uploadFields": {},
    "s3Key": "clxyz_firm/clxyz_case/clxyz_doc/chen_depo_2024.pdf",
    "expiresAt": "2026-02-21T15:00:00.000Z"
  }
}
```

Side Effects:

- Document record created with `ingestion_status = 'UPLOADING'`
- Redis: SET `ingestion:{documentId}` → `{ status: 'UPLOADING', progress: 0 }` TTL 600s

POST `/cases/:caseId/documents/:documentId/confirm-upload`

Purpose: Called by frontend after S3 upload completes. Triggers Nia ingestion pipeline.

Authentication: Required

Request Body: *(none)*

Response (200):

```
{
  "success": true,
  "data": {
    "documentId": "clxyz_doc",
    "ingestionStatus": "INDEXING"
  }
}
```

Side Effects:

- Document status updated: `UPLOADING` → `INDEXING`
- `ingestion_started_at` set to `NOW()`
- Background job queued: pre-screen with `pdf-parse` / `mammoth`, then send to Nia API
- Redis: `ingestion:{documentId}` updated to `{ status: 'INDEXING', progress: 0, pageCount: null }`
- Duplicate hash check: if SHA-256 matches existing doc → return early with `409 DUPLICATE_DOCUMENT`

GET `/cases/:caseId/documents/:documentId/ingestion-status`

Purpose: Polls ingestion progress. Frontend polls every 3 seconds while status is `INDEXING`.

Authentication: Required

Response (200):

```
{
  "success": true,
  "data": {
    "documentId": "clxyz_doc",
    "ingestionStatus": "INDEXING",
    "pagesProcessed": 127,
    "pageCount": 247,
    "progressPercent": 51,
    "eta": "1m 15s"
  }
}
```

Caching: Read from Redis `ingestion:{documentId}` first; fall back to PostgreSQL.

GET `/cases/:caseId/documents/facts`

Purpose: Returns all extracted facts across all READY documents for attorney review.

Authentication: Required

Response (200):

```
{
  "success": true,
  "data": {
    "documentsReady": 3,
    "allConfirmed": false,
    "parties": [
      { "id": "p1", "name": "Dr. Emily Chen", "role": "Defendant", "sourceDocId": "clxyz_doc",
    ],
    "keyDates": [
      { "id": "d1", "date": "2024-03-15", "event": "Initial consultation", "sourceDocId": "clx
    ],
    "disputedFacts": [
      { "id": "f1", "fact": "Dosage administered was $217", "sourceDocId": "clxyz_doc", "sourc
    ],
    "priorStatements": [
      { "id": "s1", "quote": "The dosage was exactly $217", "speaker": "Dr. Emily Chen", "sour
    ]
  }
}
```

PATCH `/cases/:caseId/documents/facts`

Purpose: Saves attorney edits to extracted facts and/or marks a section as confirmed.

Authentication: Required

Request Body:

```
{
  "updates": [
    { "type": "keyDate", "id": "d1", "changes": { "event": "Initial consultation regarding dos
    { "type": "priorStatement", "id": "s1", "changes": { "sourcePage": 48 } }
  ],
  "confirmSections": ["keyDates", "priorStatements"]
}
```

Response (200): Updated facts object (same shape as GET)

POST `/cases/:caseId/documents/:documentId/reingest`

Purpose: Retries ingestion on a FAILED document.

Authentication: Required

Request Body: *(none)*

Response (200):

```
{
  "success": true,
  "data": { "ingestionStatus": "INDEXING" }
}
```

WITNESSES MODULE

GET `/cases/:caseId/witnesses`

Purpose: Lists all witnesses for a case with session summary.

Authentication: Required

Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": "clxyz_witness",
      "name": "Dr. Emily Chen",
      "role": "DEFENDANT",
      "email": "emily.chen@metro-hospital.com",
      "sessionCount": 3,
      "latestScore": 79,
      "baselineScore": 44,
      "improvementDelta": 35,
      "plateauDetected": false,
      "latestSessionId": "clxyz_session3",
      "latestBriefId": "clxyz_brief3"
    }
  ]
}
```

POST `/cases/:caseId/witnesses`

Purpose: Adds a new witness to a case.

Authentication: Required

Request Body:

```
{
  "name": "Dr. Emily Chen",
  "email": "emily.chen@metro-hospital.com",
  "role": "DEFENDANT",
}
```

```
{
  "notes": "Known weakness: medication dosage timeline",
  "linkedDocumentIds": ["clxyz_doc1", "clxyz_doc2"]
}
```

Validation Rules:

- `name` : required, 2–255 chars, sanitized
- `email` : required, valid RFC 5321, max 255 chars
- `role` : required, valid enum
- `notes` : optional, max 5000 chars
- `linkedDocumentIds` : must all belong to this case

Response (201): Created witness object

GET `/cases/:caseId/witnesses/:witnessId`

Purpose: Full witness profile — all sessions, scores, inconsistency log, weakness evolution.

Authentication: Required

Response (200):

```
{
  "success": true,
  "data": {
    "id": "clxyz_witness",
    "name": "Dr. Emily Chen",
    "role": "DEFENDANT",
    "sessionCount": 3,
    "sessions": [
      {
        "id": "clxyz_session1",
        "sessionNumber": 1,
        "score": 44,
        "consistencyRate": 0.61,
        "endedAt": "2026-02-07T18:30:00.000Z",
        "briefId": "clxyz_brief1"
      }
    ],
    "scoreTrend": [44, 61, 79],
    "plateauDetected": false,
    "weaknessMapEvolution": {
      "session1": { "timeline": 40, "financial": 28, "communications": 60, "relationships": 75 },
      "sessionLatest": { "timeline": 72, "financial": 34, "communications": 80, "relationships": 80 }
    },
    "persistingInconsistencies": [
      {
        "alertId": "clxyz_alert",
        "priorQuote": "The dosage was exactly $217",
        "firstDetectedSession": 1,
        "detectedCount": 2,
        "impeachmentRisk": "HIGH"
      }
    ]
  }
}
```

PATCH `/cases/:caseId/witnesses/:witnessId`

Purpose: Updates witness name, email, role, notes, or linked documents.

Authentication: Required

DELETE /cases/:caseId/witnesses/:witnessId

Purpose: Deletes a witness and all associated sessions, alerts, and briefs.

Authentication: Required (case owner or admin only)

Response (200):

```
{
  "success": true,
  "message": "Witness and all associated session data deleted."
}
```

SESSIONS MODULE

POST /cases/:caseId/witnesses/:witnessId/sessions

Purpose: Creates and configures a new deposition simulation session.

Authentication: Required

Request Body:

```
{
  "durationMinutes": 45,
  "focusAreas": ["TIMELINE_CHRONOLOGY", "FINANCIAL_DETAILS", "PRIOR_STATEMENTS"],
  "aggressionLevel": "ELEVATED",
  "objectionCopilot": true,
  "behavioralSentinel": false
}
```

Validation Rules:

- durationMinutes : required, must be 15, 30, 45, or 60
- focusAreas : required, 1–6 items, all valid enum values
- aggressionLevel : required, valid enum
- objectionCopilot : required, boolean
- behavioralSentinel : required, boolean — if true, firms.sentinel_enabled must be true

Response (201):

```
{
  "success": true,
  "data": {
    "id": "clxyz_session4",
    "sessionNumber": 4,
    "status": "CONFIGURED",
    "witnessToken": null,
    "priorWeakAreas": {
      "lowestAxes": ["financial_details"],
      "persistingInconsistencies": ["dosage timeline"]
    }
  }
}
```

Side Effects:

- Nia pre-briefing job queued: loads case documents + witness's prior session weak areas into session context
-

POST `/sessions/:sessionId/witness-token`

Purpose: Generates a time-limited access token for the witness practice link.

Authentication: Required

Request Body: *(none)*

Response (200):

```
{
  "success": true,
  "data": {
    "witnessToken": "abc123def456ghi789jkl",
    "witnessUrl": "https://verdict.law/witness/session/clxyz_session4?token=abc123...",
    "expiresAt": "2026-02-24T14:00:00.000Z"
  }
}
```

Side Effects:

- `witness_token` and `witness_token_expires_at` set on session
 - Redis: SET `witness:{token}` → `{ sessionId }` EX 259200 (72h)
 - Witness invitation email sent via Resend
-

POST `/sessions/:sessionId/start`

Purpose: Begins the live session. Transitions status from LOBBY → ACTIVE.

Authentication: Required (attorney JWT)

Request Body: *(none)*

Response (200):

```
{
  "success": true,
  "data": {
    "sessionId": "clxyz_session4",
    "status": "ACTIVE",
    "startedAt": "2026-02-21T15:00:00.000Z"
  }
}
```

Side Effects:

- `sessions.status` → 'ACTIVE'; `started_at` set
 - WebSocket event broadcast to session room: `{ type: 'SESSION_START', sessionId }`
 - `session_events` record inserted: type `SESSION_START`
 - Interrogator Agent starts question generation pipeline
-

POST `/sessions/:sessionId/pause`

Purpose: Pauses session timer and halts Interrogator.

Authentication: Required

Response (200):

```
{
  "success": true,
  "data": { "status": "PAUSED", "pausedAt": "2026-02-21T15:22:00.000Z" }
}
```

Side Effects:

- WebSocket: { type: 'SESSION_PAUSE' } broadcast to room
- session_events record: type SESSION_PAUSE

POST /sessions/:sessionId/resume

Purpose: Resumes paused session.

Authentication: Required

Response (200): { status: 'ACTIVE' }

POST /sessions/:sessionId/end

Purpose: Ends session and triggers coaching brief generation.

Authentication: Required

Request Body:

```
{
  "reason": "TIMER_EXPIRED"
}
```

reason must be "TIMER_EXPIRED" or "ATTORNEY_ENDED"

Response (200):

```
{
  "success": true,
  "data": {
    "status": "COMPLETE",
    "endedAt": "2026-02-21T15:45:00.000Z",
    "briefId": "clxyz_brief4",
    "briefStatus": "GENERATING"
  }
}
```

Side Effects:

- sessions.status → 'COMPLETE'; ended_at set
- Transcript finalized and written to sessions.transcript_raw
- All Redis session event buffer flushed to session_events table
- Brief record created (generation_status = 'GENERATING')

- Background job started: Review Orchestrator pipeline
 - Witness token invalidated in Redis
 - WebSocket: `SESSION_END` event + `BRIEF_GENERATING` event
 - `witnesses.session_count` incremented; `witnesses.latest_score` updated after brief completes
-

GET `/sessions/:sessionId`

Purpose: Returns session status, configuration, and live alert summary for the attorney.

Authentication: Required

Response (200):

```
{
  "success": true,
  "data": {
    "id": "clxyz_session4",
    "status": "ACTIVE",
    "sessionNumber": 4,
    "durationMinutes": 45,
    "startedAt": "2026-02-21T15:00:00.000Z",
    "questionCount": 7,
    "alerts": {
      "objections": 3,
      "inconsistencies": 2,
      "composure": 1
    },
    "agentStatus": {
      "interrogator": "ACTIVE",
      "objectionCopilot": "ACTIVE",
      "inconsistencyDetector": "ACTIVE",
      "behavioralSentinel": "INACTIVE"
    }
  }
}
```

GET `/witness/sessions/:sessionId` (*witness-facing*)

Purpose: Validates witness token and returns session metadata for the witness view.

Authentication: Witness token (query param `?token=`)

Response (200):

```
{
  "success": true,
  "data": {
    "sessionId": "clxyz_session4",
    "status": "LOBBY",
    "caseName": "Confidential Legal Matter",
    "durationMinutes": 45,
    "attorneyName": "Sarah Chen"
  }
}
```

Errors:

- `401 TOKEN_INVALID` — No matching token
- `401 TOKEN_EXPIRED` — `witness_token_expires_at` in past

- 410 SESSION_COMPLETE — Session already finished

AI AGENTS MODULE

POST /sessions/:sessionId/agents/question

Purpose: Generates the next Interrogator question. Called by the session engine after each answer.

Authentication: Required (attorney JWT or internal service call)

Request Body:

```
{
  "questionNumber": 8,
  "priorAnswer": "I always consult before adjusting doses, approximately in the $200 range",
  "hesitationDetected": false,
  "behavioralSignal": null,
  "recentInconsistencyFlag": true,
  "currentTopic": "FINANCIAL_DETAILS"
}
```

Response (200) — Server-Sent Events stream:

```
data: {"type":"QUESTION START","questionNumber":8}
data: {"type":"QUESTION CHUNK","text":"Isn't it your sworn testimony"}
data: {"type":"QUESTION CHUNK","text":" that the dosage was exactly"}
data: {"type":"QUESTION CHUNK","text":" $217?"}
data: {"type":"QUESTION_END","fullText":"Isn't it your sworn testimony that the dosage was exa"}
```

Side Effects:

- sessions.question_count incremented
- session_events record: type QUESTION_DELIVERED, audio_latency_ms set after TTS completes

POST /sessions/:sessionId/agents/objection

Purpose: Classifies a question for FRE objection types. Must complete ≤ 1.5 s from question delivery.

Authentication: Required

Request Body:

```
{
  "questionNumber": 7,
  "questionText": "Isn't it true you had forgotten about the dosage by then?",
  "questionTimestamp": "2026-02-21T15:14:22.000Z"
}
```

Response (200):

```
{
  "success": true,
  "data": {
    "isObjectionable": true,
    "alertId": "clxyz_alert_7",
  }
}
```

```

    "category": "LEADING",
    "freRule": "FRE 611(c)",
    "explanation": "Question suggests a specific answer (that the witness had 'forgotten'). Wi",
    "confidence": 0.94,
    "processingMs": 1180
  }
}

```

Side Effects:

- If `isObjectionable`: `alerts` record inserted; WebSocket: `objection_alert` event to attorney's socket room

POST `/sessions/:sessionId/agents/inconsistency`

Purpose: Checks a witness answer for contradictions against prior sworn statements. Must complete ≤ 4 s.

Authentication: Required

Request Body:

```

{
  "questionNumber": 8,
  "questionText": "What was the exact dosage you administered?",
  "answerText": "Approximately $200, in that range",
  "answerTimestamp": "2026-02-21T15:15:03.000Z",
  "behavioralCorroboration": {
    "emotionCategory": "FEAR",
    "durationMs": 1200,
    "auVectors": { "au4": 0.82, "au20": 0.71 }
  }
}

```

Response (200):

```

{
  "success": true,
  "data": {
    "flagFound": true,
    "alertId": "clxyz_alert_8",
    "isLiveFired": true,
    "contradictionConfidence": 0.91,
    "priorQuote": "The dosage was exactly $217.",
    "priorDocumentId": "clxyz_doc1",
    "priorDocumentPage": 47,
    "priorDocumentLine": 12,
    "impeachmentRisk": "HIGH",
    "behavioralCorroborated": true,
    "processingMs": 3420
  }
}

```

Decision logic (server-side):

- `confidence >= 0.75` + behavioral corroboration \rightarrow `impeachmentRisk = "HIGH"`, live fired
- `confidence >= 0.75` + no corroboration \rightarrow `impeachmentRisk = "STANDARD"`, live fired
- `confidence 0.50-0.74` \rightarrow `isLiveFired = false`, secondary queue
- `confidence < 0.50` \rightarrow `flagFound = false`, nothing stored

Side Effects:

- `alerts` record inserted; WebSocket: `inconsistency_alert` event if `isLiveFired`
-

POST `/sessions/:sessionId/agents/behavioral`

Purpose: Receives FACS AU vectors from client-side MediaPipe and classifies emotional state.

Authentication: Required

Request Body:

```
{
  "questionNumber": 8,
  "auVectors": { "au4": 0.82, "au6": 0.11, "au12": 0.05, "au20": 0.71, "au45": 0.02 },
  "durationMs": 1200,
  "timestamp": "2026-02-21T15:15:01.000Z"
}
```

Validation: Raw video never accepted — only numeric AU vector JSON.

Response (200):

```
{
  "success": true,
  "data": {
    "alertFired": true,
    "alertId": "clxyz_alert_8b",
    "emotionCategory": "FEAR",
    "confidence": 0.87,
    "meetsThreshold": true,
    "thresholdMs": 800
  }
}
```

Side Effects:

- If `meetsThreshold`: `alerts` record inserted (type `COMPOSURE`); WebSocket: `composure_alert` event
 - `session_events` record: type `COMPOSURE_ALERT` with `emotion_vectors` and `facs_au_vector`
 - Event streamed to Databricks Delta Live Tables via background flush
-

ALERTS MODULE

PATCH `/sessions/:sessionId/alerts/:alertId`

Purpose: Attorney confirms, rejects, or annotates an alert. Used during session and from the coaching brief.

Authentication: Required

Request Body:

```
{
  "decision": "CONFIRMED",
  "note": "This is the key impeachment moment — drill on this in Session 5"
}
```

`decision` must be `"CONFIRMED"`, `"REJECTED"`, or `"ANNOTATED"`

Response (200):

```
{
  "success": true,
  "data": {
    "alertId": "clxyz_alert_8",
    "decision": "CONFIRMED",
    "impeachmentRisk": "HIGH",
    "inBrief": true
  }
}
```

Side Effects:

- alerts.attorney_decision, attorney_note, decision_at updated
- If brief already generated: briefs.confirmed_flags count updated
- in_brief flag set based on decision (CONFIRMED → true, REJECTED → false)

BRIEFS MODULE

GET /briefs/:briefId

Purpose: Returns the full coaching brief. Attorney gets full access; share-token access gets read-only subset.

Authentication: Required (JWT) OR share token query param ?token=

Response (200) — attorney view:

```
{
  "success": true,
  "data": {
    "id": "clxyz_brief4",
    "generationStatus": "COMPLETE",
    "sessionScore": 79,
    "consistencyRate": 0.87,
    "improvementDelta": 35,
    "confirmedFlags": 2,
    "objectionCount": 4,
    "composureAlertCount": 1,
    "topRecommendations": [
      "Address the $200 vs $217 dosage discrepancy — prepare a precise, consistent explanation",
      "Practice a 3-second deliberate pause before answering any leading question.",
      "Focus next session on Financial Details (weakest axis: 34/100).",
    ],
    "narrativeText": "Session 4 showed marked improvement overall...",
    "weaknessMapScores": {
      "timeline": 72, "financial": 34, "communications": 80,
      "relationships": 85, "actions": 78, "prior_statements": 65
    },
    "alerts": [
      {
        "id": "clxyz_alert_8",
        "alertType": "INCONSISTENCY",
        "questionNumber": 8,
        "questionText": "What was the exact dosage you administered?",
        "answerText": "Approximately $200, in that range",
        "priorQuote": "The dosage was exactly $217.",
        "priorDocumentPage": 47,
        "priorDocumentLine": 12,
        "contradictionConfidence": 0.91,
        "impeachmentRisk": "HIGH",
      }
    ]
  }
}
```

```

        "behavioralCorroborated": true,
        "attorneyDecision": "CONFIRMED",
        "audioClipS3Key": "firms/clxyz_firm/briefs/clxyz_brief4/alert_8.mp3"
    }
],
"pdfS3Key": "firms/clxyz_firm/briefs/clxyz_brief4/brief.pdf",
"shareToken": "abc123...",
"shareTokenExpiresAt": "2026-02-28T14:00:00.000Z"
}
}

```

Response (200) — share token (witness/client) view: Same structure but excludes: `narrativeText` in full (truncated to summary), `alerts[].attorneyDecision`, `alerts[].attorneyNote`, `pdfS3Key` (replaced with public download URL if attorney has shared)

Errors:

- 401 `TOKEN_INVALID` — Invalid share token
- 401 `TOKEN_EXPIRED` — Share token past `share_token_expires_at`
- 404 `NOT_FOUND`
- 403 `BRIEF_GENERATING` — `generation_status` is 'GENERATING' (poll again in 15s)

POST `/briefs/:briefId/pdf`

Purpose: Generates or re-generates the PDF for a brief.

Authentication: Required

Request Body: *(none)*

Response (202):

```

{
  "success": true,
  "message": "PDF generation started",
  "estimatedSeconds": 30
}

```

Side Effects:

- Background job: Puppeteer renders brief HTML → PDF → uploads to S3
- On completion: `briefs.pdf_s3_key` and `pdf_generated_at` set; WebSocket: `BRIEF_PDF_READY`

POST `/briefs/:briefId/share`

Purpose: Generates a 7-day share token for the brief.

Authentication: Required

Request Body: *(none)*

Response (200):

```

{
  "success": true,
  "data": {
    "shareUrl": "https://verdict.law/briefs/clxyz_brief4?token=abc123...",
    "expiresAt": "2026-02-28T14:00:00.000Z"
  }
}

```

```
}  
}
```

Side Effects:

- `share_token` (nanoid 24) and `share_token_expires_at` set on brief
- Optional: sends share email to witness if email provided

DELETE `/briefs/:briefId/share`

Purpose: Revokes the active share token.

Authentication: Required

Response (200): { "success": true, "message": "Share link revoked" }

Side Effects: `share_token` set to NULL; `share_token_expires_at` cleared.

ADMIN MODULE

GET `/admin/users`

Purpose: Lists all users in the firm for the admin panel.

Authentication: Required + Admin role

Query Params: `page`, `limit`, `role` filter, `isActive` filter

Response (200): Paginated user list with: id, name, email, role, isActive, lastLoginAt, createdAt

POST `/admin/users`

Purpose: Manually adds a single user to the firm.

Authentication: Required + Admin role

Request Body:

```
{  
  "name": "Marcus Webb",  
  "email": "m.webb@kirklandellis.com",  
  "role": "ASSOCIATE"  
}
```

Side Effects: User record created; onboarding email sent via Resend.

PATCH `/admin/users/:userId`

Purpose: Updates a user's role or active status.

Authentication: Required + Admin role

Request Body:

```
{
  "role": "PARTNER",
  "isActive": true
}
```

Errors:

- 409 SEAT_LIMIT_EXCEEDED — Activating a user would exceed `firms.seats`
-

DELETE `/admin/users/:userId`

Purpose: Deactivates a user (soft delete — sets `is_active = false`).

Authentication: Required + Admin role

Side Effects: All refresh tokens for that user revoked; cannot log in again until re-activated.

GET `/admin/stats`

Purpose: Firm-level usage analytics for the admin dashboard.

Authentication: Required + Admin role

Response (200):

```
{
  "success": true,
  "data": {
    "seatsUsed": 18,
    "seatsTotal": 25,
    "sessionsThisMonth": 47,
    "activeCases": 12,
    "briefs": { "generated": 43, "pdfsDownloaded": 28 },
    "averageScoreImprovement": 22,
    "topPerformingAttorney": { "name": "Marcus Webb", "sessionsRun": 14 }
  }
}
```

Caching: Redis key `admin:stats:firm:{firmId}` — TTL 5 minutes.

PATCH `/admin/settings`

Purpose: Updates firm-level settings (retention, sentinel toggle, SSO config).

Authentication: Required + Admin role

Request Body:

```
{
  "retentionDays": 60,
  "sentinelEnabled": true,
  "passwordLoginEnabled": false
}
```

USER SETTINGS MODULE

GET /users/me

Purpose: Returns the authenticated user's profile and preferences.

Authentication: Required

Response (200):

```
{
  "success": true,
  "data": {
    "id": "clxyz_user",
    "email": "sarah.chen@kirklandellis.com",
    "name": "Sarah Chen",
    "role": "PARTNER",
    "firm": { "id": "clxyz_firm", "name": "Kirkland & Ellis LLP" },
    "notifications": {
      "briefReady": true,
      "plateauAlert": true,
      "sessionReminder": true,
      "ingestionComplete": true
    },
    "activeSessions": [
      { "id": "clxyz_rt1", "deviceHint": "Chrome / Mac", "createdAt": "2026-02-20T08:00:00.000" }
    ]
  }
}
```

PATCH /users/me

Purpose: Updates name or notification preferences.

Authentication: Required

Request Body:

```
{
  "name": "Sarah Chen-Williams",
  "notifications": {
    "briefReady": true,
    "plateauAlert": false
  }
}
```

DELETE /users/me/sessions/:tokenId

Purpose: Revokes a single refresh token (log out one device from active sessions list).

Authentication: Required

WEBSOCKET EVENTS

Connection: wss://prod.verdict.law/ws/session/:sessionId

Authentication: JWT must be present in connection handshake cookie. Witness connections include `?token=` query param.

Server → Client Events:

Event Name	Payload	Recipient
session_start	{ sessionId, startedAt }	Both
session_pause	{ pausedAt }	Both
session_resume	{ resumedAt }	Both
session_end	{ endedAt, briefId }	Both
interrogator_question	{ questionNumber, text, audioUrl }	Witness
answer_received	{ questionNumber, transcribedText }	Attorney
objection_alert	{ alertId, questionNumber, category, freRule, explanation }	Attorney only
inconsistency_alert	{ alertId, questionNumber, priorQuote, page, line, confidence, impeachmentRisk }	Attorney only
composure_alert	{ alertId, questionNumber, emotionCategory, durationMs }	Attorney only
witness_connected	{ witnessName }	Attorney only
witness_disconnected	{ questionNumber }	Attorney only
agent_degraded	{ agent, fallbackMode }	Attorney only
brief_generating	{ briefId, estimatedSeconds }	Attorney only
brief_ready	{ briefId }	Attorney only
brief_pdf_ready	{ briefId, pdfUrl }	Attorney only

Client → Server Events:

Event Name	Payload	Sender
answer_audio	{ audioBlob, questionNumber }	Witness
pause_request	{}	Both
resume_request	{}	Attorney
end_request	{ reason }	Attorney

annotation_add	{ questionNumber, text }	Attorney
topic_skip	{}	Attorney
behavioral_vectors	{ auVectors, durationMs, questionNumber }	Witness (Sentinel)

4. AUTHENTICATION & AUTHORIZATION

JWT Token Structure

Access Token Payload:

```
{
  "sub": "clxyz_user",
  "firmId": "clxyz_firm",
  "role": "PARTNER",
  "email": "sarah.chen@kirklandellis.com",
  "iat": 1740139200,
  "exp": 1740168000,
  "jti": "clxyz_jti_access"
}
```

Refresh Token Payload:

```
{
  "sub": "clxyz_user",
  "firmId": "clxyz_firm",
  "jti": "clxyz_jti_refresh",
  "iat": 1740139200,
  "exp": 1742731200
}
```

Refresh token only has `sub`, `firmId`, `jti` — no role (fetched fresh on each refresh to pick up role changes).

Authorization Levels

Level	Routes	Check
Public	/auth/*, /onboarding/*, /witness/sessions/*, /briefs/:id?token=	None
Authenticated	/cases/*, /sessions/*, /agents/*, /alerts/*, /briefs/:id (no token), /users/me	Valid JWT, is_active=true
Firm-scoped	All authenticated routes	resource.firm_id === jwt.firmId — DB-level check
Admin	/admin/*	jwt.role === 'ADMIN'

Fastify Auth Middleware

```
// Prehandler on all /api/v1 routes except public
async function requireAuth(request: FastifyRequest, reply: FastifyReply) {
  const token = request.cookies.access_token;
  if (!token) throw new UnauthorizedException('TOKEN_MISSING');
```

```

const payload = fastify.jwt.verify(token);
const user = await db.user.findUnique({
  where: { id: payload.sub },
  select: { id: true, firmId: true, role: true, isActive: true }
});

if (!user) throw new UnauthorizedException('USER_NOT_FOUND');
if (!user.isActive) throw new ForbiddenException('ACCOUNT_INACTIVE');

request.user = user; // Attach to request context
}

// Resource ownership check (call after requireAuth)
function requireFirmOwnership(resourceFirmId: string, userFirmId: string) {
  if (resourceFirmId !== userFirmId) throw new ForbiddenException('FORBIDDEN');
}

```

Password Security

Algorithm: bcrypt (node.bcryptjs 5.1.1)
 Cost factor: 12 (~350ms hash time)
 Min rounds: Never go below 10 in any environment
 Salt: Auto-generated per hash (not reused)

Password Requirements:

- Minimum 10 characters
- At least one uppercase letter (A-Z)
- At least one lowercase letter (a-z)
- At least one digit (0-9)
- At least one special character (!@#\$\$%^&*() +=)
- Maximum 128 characters (bcrypt truncates beyond 72 bytes – enforce max)

Reset Flow:

1. User requests reset → nanoid(32) token generated
2. SHA-256 hash stored in password reset token (never store raw token)
3. Email sent with raw token in URL: /auth/password/reset?token={rawToken}
4. On submit: SHA-256 hash of submitted token compared to stored hash
5. On success: new hash stored, reset token cleared, all refresh tokens revoked
6. Token TTL: 1 hour

5. DATA VALIDATION RULES

Zod Schemas (shared between frontend React Hook Form and backend Fastify)

```

import { z } from 'zod';

// Email
export const emailSchema = z
  .string()
  .email('Must be a valid email address')
  .max(255, 'Email must be under 255 characters')
  .toLowerCase()
  .trim();

// Password (create/reset)
export const passwordSchema = z
  .string()
  .min(10, 'Password must be at least 10 characters')
  .max(128, 'Password must be under 128 characters')
  .regex(/[A-Z]/, 'Must contain at least one uppercase letter')
  .regex(/[a-z]/, 'Must contain at least one lowercase letter')
  .regex(/[0-9]/, 'Must contain at least one number')
  .regex(/![@#$$%^&*() _+\-=]/, 'Must contain at least one special character');

```

```

// Case name
export const caseNameSchema = z
  .string()
  .min(3, 'Case name must be at least 3 characters')
  .max(255, 'Case name must be under 255 characters')
  .transform(s => s.replace(/<[>]*>/g, '').trim()); // XSS strip

// Witness name
export const witnessNameSchema = z
  .string()
  .min(2, 'Witness name must be at least 2 characters')
  .max(255, 'Witness name must be under 255 characters')
  .transform(s => s.replace(/<[>]*>/g, '').trim());

// Attorney note / free text (XSS stripped, max length enforced)
export const noteTextSchema = z
  .string()
  .max(5000, 'Note must be under 5000 characters')
  .transform(s => s.replace(/<script\b[<]*(?:?!<\script><[<]*)*<\script>/gi, '').trim())

// File upload validation (pre-sign step)
export const fileUploadSchema = z.object({
  filename: z.string().min(1).max(500),
  mimeType: z.enum([
    'application/pdf',
    'application/vnd.openxmlformats-officedocument.wordprocessingml.document',
    'text/plain'
  ]),
  fileSizeBytes: z.number().int().positive().max(209_715_200, 'File must be under 200MB'),
});

// Session configuration
export const sessionConfigSchema = z.object({
  durationMinutes: z.union([z.literal(15), z.literal(30), z.literal(45), z.literal(60)]),
  focusAreas: z.array(z.enum([
    'TIMELINE_CHRONOLOGY', 'FINANCIAL_DETAILS', 'COMMUNICATIONS',
    'RELATIONSHIPS', 'ACTIONS_TAKEN', 'PRIOR_STATEMENTS'
  ])).min(1).max(6),
  aggressionLevel: z.enum(['STANDARD', 'ELEVATED', 'HIGH_STAKES']),
  objectionCopilot: z.boolean(),
  behavioralSentinel: z.boolean(),
});

// Behavioral Sentinel AU vectors
export const auVectorSchema = z.object({
  au4: z.number().min(0).max(1),
  au6: z.number().min(0).max(1),
  au12: z.number().min(0).max(1),
  au20: z.number().min(0).max(1),
  au45: z.number().min(0).max(1),
}).strict(); // Reject any unknown keys

```

Input Length Limits

Field	Min	Max	Notes
Email	5	255	RFC 5321 format validation
Password	10	128	bcrypt 72-byte effective limit
Case name	3	255	XSS stripped
Witness name	2	255	XSS stripped

Attorney notes	0	5,000	XSS stripped
Coaching recommendation	0	1,000	Per recommendation string
Filename	1	500	Sanitized: no path traversal chars
Narrative text (brief)	0	50,000	Claude output
Transcript (raw)	0	500,000	Full session transcript
Prior quote (alert)	1	5,000	Nia-retrieved
AU vector values	0.0	1.0	Float range enforcement

Content Sanitization

All user-controlled string inputs passed through:

- 1. **Trim whitespace** — leading/trailing removed
- 2. **XSS strip** — HTML tags removed via regex before DB write
- 3. **SQL injection** — Prisma parameterized queries (no raw SQL with interpolation)
- 4. **Path traversal** — filenames validated: no `../`, `./`, `/`, `\`, or null bytes
- 5. **AI prompt injection** — User-supplied text inserted into Claude prompts inside XML-escaped wrappers:

```
<witness_answer>{ESCAPED_TEXT}</witness_answer>
```

6. ERROR HANDLING

Standard Error Response Format

```
{
  "success": false,
  "error": {
    "code": "INCONSISTENCY_DETECTOR_TIMEOUT",
    "message": "The Inconsistency Detector is taking longer than expected. Falling back to Claude",
    "statusCode": 503,
    "details": {
      "agent": "inconsistency_detector",
      "fallback": "claude_only",
      "thresholdRaised": true
    },
    "requestId": "req_clxyz_1234",
    "timestamp": "2026-02-21T15:15:03.000Z"
  }
}
```

Error Code → HTTP Status Mapping

Error Code	HTTP Status	Trigger
VALIDATION_ERROR	400	Zod schema validation failed
MISSING_REQUIRED_FIELD	400	Required field absent
INVALID_ENUM_VALUE	400	Field value not in allowed enum
FILE_TOO_LARGE	400	Upload exceeds 200MB

UNSUPPORTED_FILE_TYPE	400	MIME type not in allowlist
DUPLICATE_DOCUMENT	409	Same file hash already in case
PASSWORD_REQUIREMENTS	400	Password doesn't meet complexity
TOKEN_MISSING	401	No auth token in request
TOKEN_EXPIRED	401	JWT or refresh token past expiry
TOKEN_INVALID	401	Cannot verify signature
TOKEN_REVOKED	401	Refresh token manually revoked
INVALID_CREDENTIALS	401	Login email/password mismatch
EMAIL_NOT_VERIFIED	401	Account not yet verified
ACCOUNT_LOCKED	403	Too many failed login attempts
ACCOUNT_INACTIVE	403	User <code>is_active = false</code>
SSO_REQUIRED	403	Firm requires SSO, password login disabled
FORBIDDEN	403	Resource belongs to different firm
ADMIN_REQUIRED	403	Route requires Admin role
NOT_FOUND	404	Resource doesn't exist
SEAT_LIMIT_EXCEEDED	409	Adding user exceeds firm seat count
SESSION_ALREADY_ACTIVE	409	Witness already has active session
SESSION_COMPLETE	410	Witness token for completed session
BRIEF_GENERATING	202	Brief not yet ready — poll again
RATE_LIMIT_EXCEEDED	429	Too many requests
NEMOTRON_UNAVAILABLE	503	Nemotron API down — fallback active
NIA_UNAVAILABLE	503	Nia API down — fallback active
ELEVENLABS_UNAVAILABLE	503	ElevenLabs API down — text-only mode
INTERNAL_ERROR	500	Unhandled exception

Fastify Global Error Handler

```
fastify.setErrorHandler((error, request, reply) => {
  const requestId = request.id;

  // Log all errors to Pino + Sentry
  fastify.log.error({ error, requestId }, 'Request error');
  if (error.statusCode >= 500) {
    Sentry.captureException(error, { extra: { requestId } });
  }
});
```

```

// Zod validation errors → format nicely
if (error instanceof ZodError) {
  return reply.code(400).send({
    success: false,
    error: {
      code: 'VALIDATION_ERROR',
      message: 'Request validation failed',
      statusCode: 400,
      details: error.flatten().fieldErrors,
      requestId,
      timestamp: new Date().toISOString()
    }
  });
}

// Known app errors (thrown explicitly)
if (error instanceof VerdictError) {
  return reply.code(error.statusCode).send({
    success: false,
    error: {
      code: error.code,
      message: error.message,
      statusCode: error.statusCode,
      details: error.details ?? undefined,
      requestId,
      timestamp: new Date().toISOString()
    }
  });
}

// Fallback 500
return reply.code(500).send({
  success: false,
  error: {
    code: 'INTERNAL_ERROR',
    message: 'An unexpected error occurred.',
    statusCode: 500,
    requestId,
    timestamp: new Date().toISOString()
  }
});
});

```

7. CACHING STRATEGY

Cache Layers & Keys

Cache Key Pattern	Data	TTL	Invalidation Trigger
<code>cases:firm:{firmId}:list</code>	Case list (without full details)	5 min	Case create, update, archive
<code>case:{caseId}:detail</code>	Full case detail + documents + witnesses	2 min	Document upload, witness add/update, session end
<code>ingestion:{documentId}</code>	Ingestion status + progress %	10 min	Nia status webhook; set on READY/FAILED
<code>session:{sessionId}:events</code>	Buffered session events list	5 min	Flushed to DB every 60 seconds

session: {sessionId}:state	Current session state object	5 min	Every state transition
witness:{token}	{ sessionId } for witness token validation	72 hours (259200s)	DEL on session complete or manual revoke
brief:{briefId}:status	Generation status polling	30 sec	Set to COMPLETE on generation finish
admin:stats:firm: {firmId}	Aggregated firm stats	5 min	Any session complete, brief generated
ratelimit:{endpoint}: {identifier}	Request count	60 sec (window-based)	Natural TTL expiry
refresh:{userId}:{jti}	Refresh token validity flag	30 days	DEL on logout

Session Event Buffer (60-Second Auto-Flush)

```
// Redis list-based event buffer
// Key: session:{sessionId}:events
// Type: Redis List (RPUSH per event, LRange + DEL on flush)

const BUFFER_FLUSH_INTERVAL_MS = 60_000;
const EVENT_BUFFER_TTL_S = 300; // 5 min - failsafe TTL

async function bufferSessionEvent(sessionId: string, event: SessionEvent) {
  const key = `session:${sessionId}:events`;
  await redis.rpush(key, JSON.stringify(event));
  await redis.expire(key, EVENT_BUFFER_TTL_S);
}

// Cron-like flush - runs every 60s per active session
async function flushSessionEvents(sessionId: string) {
  const key = `session:${sessionId}:events`;
  const events = await redis.lrange(key, 0, -1);
  if (events.length === 0) return;

  await db.sessionEvent.createMany({
    data: events.map(e => JSON.parse(e))
  });
  await redis.del(key);

  // Also stream to Databricks Delta Live Tables
  await databricks.streamEvents(events.map(e => JSON.parse(e)));
}

// Force-flush on session end (ensures zero data loss)
await flushSessionEvents(sessionId);
```

Cache Invalidation Triggers

```
// Centralized invalidation - called after any mutation
const cacheInvalidation = {
  onCaseCreate: (firmId: string) => redis.del(`cases:firm:${firmId}:list`),
  onCaseUpdate: (firmId: string, caseId: string) => Promise.all([
    redis.del(`cases:firm:${firmId}:list`),
    redis.del(`case:${caseId}:detail`),
  ]),
}
```

```

onDocumentIngested: (caseId: string, docId: string) => Promise.all([
  redis.del(`case:${caseId}:detail`),
  redis.del(`ingestion:${docId}`),
]),
onSessionEnd: (firmId: string, caseId: string) => Promise.all([
  redis.del(`case:${caseId}:detail`),
  redis.del(`admin:stats:firm:${firmId}`),
]),
onBriefComplete: (firmId: string) => redis.del(`admin:stats:firm:${firmId}`),
};

```

8. RATE LIMITING

Limits by Endpoint

Route	Limit	Window	Scope	Store
POST /auth/login	10 requests	1 minute	Per IP	Redis
POST /auth/login	5 requests	1 minute	Per email	Redis
POST /auth/refresh	30 requests	1 minute	Per user	Redis
POST /auth/password/reset-request	5 requests	1 hour	Per email	Redis
POST /cases	10 requests	1 minute	Per user	Redis
POST /cases/:id/documents/presign	50 requests	1 hour	Per user	Redis
POST /sessions/:id/agents/question	120 requests	1 minute	Per session	Redis
POST /sessions/:id/agents/objection	120 requests	1 minute	Per session	Redis
POST /sessions/:id/agents/inconsistency	60 requests	1 minute	Per session	Redis
POST /sessions/:id/agents/behavioral	300 requests	1 minute	Per session	Redis
POST /briefs/:id/pdf	10 requests	1 minute	Per user	Redis
GET /api/v1/* (general)	200 requests	1 minute	Per user	Redis

Rate Limit Response (429)

```

{
  "success": false,
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Too many requests. Please wait before trying again.",
    "statusCode": 429,
    "details": {
      "retryAfterSeconds": 42,
      "limit": 10,
      "window": "1 minute"
    },
    "requestId": "req_clxyz_1234",
    "timestamp": "2026-02-21T15:15:03.000Z"
  }
}

```

Response Headers:

```
X-RateLimit-Limit: 10
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1740142323
Retry-After: 42
```

Fastify Rate Limit Configuration

```
await fastify.register(rateLimit, {
  global: true,
  max: 200,
  timeWindow: '1 minute',
  keyGenerator: (request) => request.user?.id ?? request.ip,
  redis: redisClient,
  errorHandler: (request, context) => ({
    success: false,
    error: {
      code: 'RATE_LIMIT_EXCEEDED',
      message: 'Too many requests.',
      statusCode: 429,
      details: {
        retryAfterSeconds: Math.ceil(context.ttl / 1000),
        limit: context.max,
        window: context.timeWindow,
      },
    },
  })
});

// Route-specific overrides
fastify.post('/auth/login', {
  config: { rateLimit: { max: 10, timeWindow: '1 minute', keyGenerator: r => r.body.email } },
  loginHandler);
```

9. DATABASE MIGRATIONS

Migration Tool: Prisma Migrate

Migration Naming Convention

```
{YYYYMMDD}_{HHMMSS}_{verb}_{entity}_{field}
Examples:
20260221 140000 create initial schema
20260222 090000 add behavioral sentinel columns
20260225 110000 add session events emotion vectors
20260301_150000_drop_legacy_alert_status_field (DESTRUCTIVE — two-phase)
```

Migration Workflow

Development — create new migration `npx prisma migrate dev --name "add_behavioral_sentinel_columns"` # → Generates SQL diff + applies to dev DB # Review the generated SQL before committing: `cat prisma/migrations/20260222_090000_add_behavioral_sentinel_columns/migration.sql` # Staging — apply without interactive prompts `npx prisma migrate deploy` # Production — same command; runs in CI before server restart `npx prisma migrate deploy` # Validate schema integrity `npx prisma validate`

Migration Process (dev → staging → production)

1. Engineer creates migration on feature branch:
npx prisma migrate dev --name "add x"
→ migration.sql generated in prisma/migrations/
2. PR opened → CI runs:
 - npx prisma validate (schema syntax check)
 - npx prisma migrate deploy on test DB
 - Full test suite must pass
3. Merge to develop → auto-deploy to staging:
 - npx prisma migrate deploy
 - Staging smoke tests
4. Merge to main → production deploy sequence:
 - a. npx prisma migrate deploy (runs BEFORE server restart)
 - b. New Fastify server starts (new code works with migrated DB)
 - c. Old server drains connections and shuts down (zero-downtime)

Destructive Change Policy (Two-Phase Deployment)

Problem: Dropping a column crashes old server that still reads it.

Phase 1 (deploy first):

- Add new column OR keep old column but stop writing to it
- Application code reads new column with fallback to old
- Old server remains happy reading old column

Phase 2 (deploy after Phase 1 is stable — next release):

- Drop old column
- Remove fallback read code

Example — renaming alerts.alert type to alerts.category:

Phase 1 migration: ADD COLUMN category VARCHAR(50);
UPDATE alerts SET category = alert_type;

Phase 1 code: Read category, fallback to alert_type

Phase 2 migration: DROP COLUMN alert_type;

Phase 2 code: Read category only

Rollback Plan

Prisma does NOT auto-rollback data migrations. # Prevention: ALL migrations must be additive in v1.0 (add columns/tables). # No DROP COLUMN or DROP TABLE without the two-phase process. # If a bad migration is deployed and must be undone: 1. Mark migration as rolled back (metadata only — does not undo SQL): npx prisma migrate resolve --rolled-back "20260222_090000_add_x" 2. Manually write + run reverse SQL: ALTER TABLE sessions DROP COLUMN IF EXISTS behavioral_sentinel; 3. Restore from RDS snapshot (nuclear option — last resort): See §10 Backup & Recovery # Docker image pinning for backend rollback: flyctl deploy --image registry.fly.io/verdict-api:20260221-stable

Connection Pooling

Development: Prisma built-in pool (connection_limit=5 in DATABASE_URL)

Production: PgBouncer in transaction mode

Fastify connects to PgBouncer (port 6432), not Postgres directly

Pool size: 20 connections per Fastify instance

Max PostgreSQL connections: 100

PgBouncer config: pool_mode=transaction, max_client_conn=200

DATABASE URL (production, via PgBouncer):

postgres://verdict:pass@pgbouncer:6432/verdict_prod?connection_limit=20&pool_timeout=30

10. BACKUP & RECOVERY

Backup Schedule

Environment	Type	Frequency	Retention	Storage
Hackathon (Supabase)	Managed automated backup	Daily	7 days	Supabase internal
Production (RDS)	Automated RDS snapshot	Daily (2 AM UTC)	35 days	AWS S3 (RDS-managed)
Production (RDS)	Point-in-time recovery (PITR)	Continuous WAL archiving	7 days	AWS S3 (RDS-managed)
Production (S3 docs)	S3 Versioning	On every write	90 days (lifecycle)	S3 versioning
Production (Redis)	ElastiCache snapshot	Daily	7 days	S3
Pre-deploy snapshot	Manual RDS snapshot	Before every production deploy	5 snapshots retained	AWS S3

Recovery Procedure

Scenario 1: Accidental Data Deletion (Table-Level)

1. Identify when deletion occurred from CloudWatch logs / Sentry # 2. Calculate recovery point: $T = \text{now} - \text{minutes_since_deletion}$ # 3. Restore to a read-only RDS instance from PITR: `aws rds restore-db-instance-to-point-in-time \ --source-db-instance-identifier verdict-production \ --target-db-instance-identifier verdict-recovery-20260221 \ --restore-time 2026-02-21T14:00:00Z` # 4. Connect to recovery instance and export affected rows: `pg_dump -h verdict-recovery-20260221.us-east-1.rds.amazonaws.com \ -U verdict -d verdict_prod -t alerts --where "session_id='clxyz'" > alerts_recovery.sql` # 5. Import recovered rows into production: `psql -h verdict-production.us-east-1.rds.amazonaws.com \ -U verdict -d verdict_prod < alerts_recovery.sql` # 6. Terminate recovery instance `aws rds delete-db-instance --db-instance-identifier verdict-recovery-20260221 --skip-final-snapshot`

Scenario 2: Full Database Corruption (Catastrophic)

RTO target: < 4 hours | RPO target: < 1 hour (PITR) # 1. Stop all application traffic (Fly.io scale to 0) `flyctl scale count 0 --app verdict-api` # 2. Restore from most recent RDS snapshot (last nightly or pre-deploy): `aws rds restore-db-instance-from-db-snapshot \ --db-instance-identifier verdict-production-restored \ --db-snapshot-identifier rds:verdict-production-2026-02-21-02-00` # OR use PITR for more recent recovery point: `aws rds restore-db-instance-to-point-in-time \ --source-db-instance-identifier verdict-production \ --target-db-instance-identifier verdict-production-restored \ --restore-time 2026-02-21T14:55:00Z` # 3. Update DATABASE_URL to point to restored instance # 4. Run prisma migrate deploy (verify schema is current) # 5. Resume traffic `flyctl scale count 2 --app verdict-api`

Scenario 3: Redis Data Loss (Session Events)

Impact: Up to 60 seconds of session events lost (acceptable per PRD §8.4) # session_events table has all flushed events (flushed every 60s) # Restore Redis from ElastiCache snapshot: `aws elasticache create-snapshot \ --replication-group-id verdict-redis \ --snapshot-name verdict-redis-recovery`

S3 Document Recovery

Documents protected by S3 versioning — restore deleted file: aws s3api restore-object \ --bucket verdict-documents-prod \ --key "clxyz_firm/clxyz_case/clxyz_doc/chen_depo_2024.pdf" \ --version-id "PREVIOUS_VERSION_ID"

11. API VERSIONING

Current Version: **v1**

URL Structure

```
https://prod.verdict.law/api/v1/cases
https://prod.verdict.law/api/v1/sessions/:id
https://prod.verdict.law/api/v1/briefs/:id
```

Versioning Strategy

URL-path versioning is used (not header-based). The version is the second segment: `/api/v1/` , `/api/v2/` (future).

Rationale: URL-path versioning is unambiguous for enterprise law firm integrations — it doesn't require clients to manage custom headers, making it simpler for firm IT teams and CLI tooling.

Backwards Compatibility Policy

Change Type	Policy
Add new optional response field	Non-breaking — deploy without version bump
Add new optional request field	Non-breaking — deploy without version bump
Add new endpoint	Non-breaking — deploy without version bump
Change response field name	Breaking → v2 required
Remove response field	Breaking → v2 required
Change field type	Breaking → v2 required
Change HTTP method	Breaking → v2 required
Change auth requirement	Breaking → v2 required

Deprecation Process

```
Phase 1 — Announce:
  Add Deprecation header to all v1 responses when v2 is released:
  Deprecation: true
  Sunset: Sat, 22 Feb 2027 00:00:00 GMT
  Link: <https://docs.verdict.law/api/v2>; rel="successor-version"

Phase 2 — 6-month window:
  Both v1 and v2 run simultaneously
  v1 responses include: X-API-Warning: "This version is deprecated..."

Phase 3 — Sunset:
```

v1 routes return 410 Gone with migration guide URL

- Deprecation Notice Channels:
- 1. HTTP response headers (above)
 - 2. Developer docs changelog at docs.verdict.law
 - 3. Email to all firm technical contacts on record

Version Discovery Endpoint

GET /api/versions

Authentication: Public

Response (200):

```
{
  "success": true,
  "data": {
    "current": "v1",
    "supported": [
      {
        "version": "v1",
        "status": "current",
        "basePath": "/api/v1",
        "docsUrl": "https://docs.verdict.law/api/v1",
        "sunsetAt": null
      }
    ]
  }
}
```

QUICK REFERENCE

Database Tables

Table	Rows (typical firm)	Key Index
firms	1	slug
users	10–25	firm_id, email
refresh_tokens	20–75	user_id, token_hash
cases	15–50	firm_id, deposition_date
documents	50–200	case_id, ingestion_status
witnesses	30–100	case_id, firm_id
sessions	100–500	witness_id, status
session_events	50,000–500,000	session_id, event_type
alerts	1,000–10,000	session_id, alert_type
briefs	100–500	session_id (1:1), witness_id
attorney_annotations	500–5,000	session_id

--	--	--

AI Agent Latency SLAs

Agent	SLA	Fallback
Interrogator (question gen)	< 2s to audio start	Text-mode if ElevenLabs TTS down
Objection Copilot	≤ 1.5s from question	Claude-only if Nia down
Inconsistency Detector	≤ 4s from answer	Claude-only, threshold 0.85 if Nemotron down
Behavioral Sentinel	≤ 2s FACS → alert	Client-side degradation if camera denied
Review Orchestrator (brief)	≤ 3 min post-session	Email when ready if > 3 min

Module File Structure

```
apps/backend/src/
├── index.ts # Fastify app bootstrap
├── plugins/
│   ├── auth.ts # JWT + SAML passport setup
│   ├── cors.ts # CORS config
│   ├── rate-limit.ts # Rate limiter registration
│   ├── socket.ts # Socket.io setup + Redis adapter
│   └── prisma.ts # Prisma client singleton
├── modules/
│   ├── auth/
│   │   ├── auth.routes.ts
│   │   ├── auth.service.ts
│   │   └── auth.schema.ts # Zod schemas
│   ├── cases/
│   │   ├── cases.routes.ts
│   │   ├── cases.service.ts
│   │   └── cases.schema.ts
│   ├── documents/
│   │   ├── documents.routes.ts
│   │   ├── documents.service.ts
│   │   ├── documents.schema.ts
│   │   └── ingestion.worker.ts # Background ingestion pipeline
│   ├── witnesses/
│   ├── sessions/
│   │   ├── sessions.routes.ts
│   │   ├── sessions.service.ts
│   │   ├── sessions.websocket.ts
│   │   └── sessions.buffer.ts # Redis event buffer + flush
│   ├── agents/
│   │   ├── interrogator.agent.ts
│   │   ├── objection.agent.ts
│   │   ├── inconsistency.agent.ts
│   │   ├── behavioral.agent.ts
│   │   └── orchestrator.agent.ts # Review Orchestrator (brief gen)
│   ├── briefs/
│   │   ├── briefs.routes.ts
│   │   ├── briefs.service.ts
│   │   └── briefs.pdf.ts # Puppeteer render
│   ├── alerts/
│   ├── admin/
│   └── users/
└── lib/
    ├── redis.ts # ioredis singleton
    ├── s3.ts # AWS S3 client
    ├── databricks.ts # Databricks SQL client + streaming
    └── email.ts # Resend client + templates
```



```
|   └─ errors.ts                # VerdictError class definitions
└─ middleware/
   └─ require-auth.ts
   └─ require-admin.ts
   └─ require-firm-ownership.ts
└─ types/
   └─ fastify.d.ts              # Augment FastifyRequest with request.user
   └─ socket.d.ts               # Typed socket events
```

BACKEND_STRUCTURE.md — VERDICT v1.0.0 — Hackathon Edition

B2B Deposition Coaching Platform — Team VoiceFlow Intelligence

NYU Startup Week Buildathon 2026 | AI Automation — August.law Sponsor Track