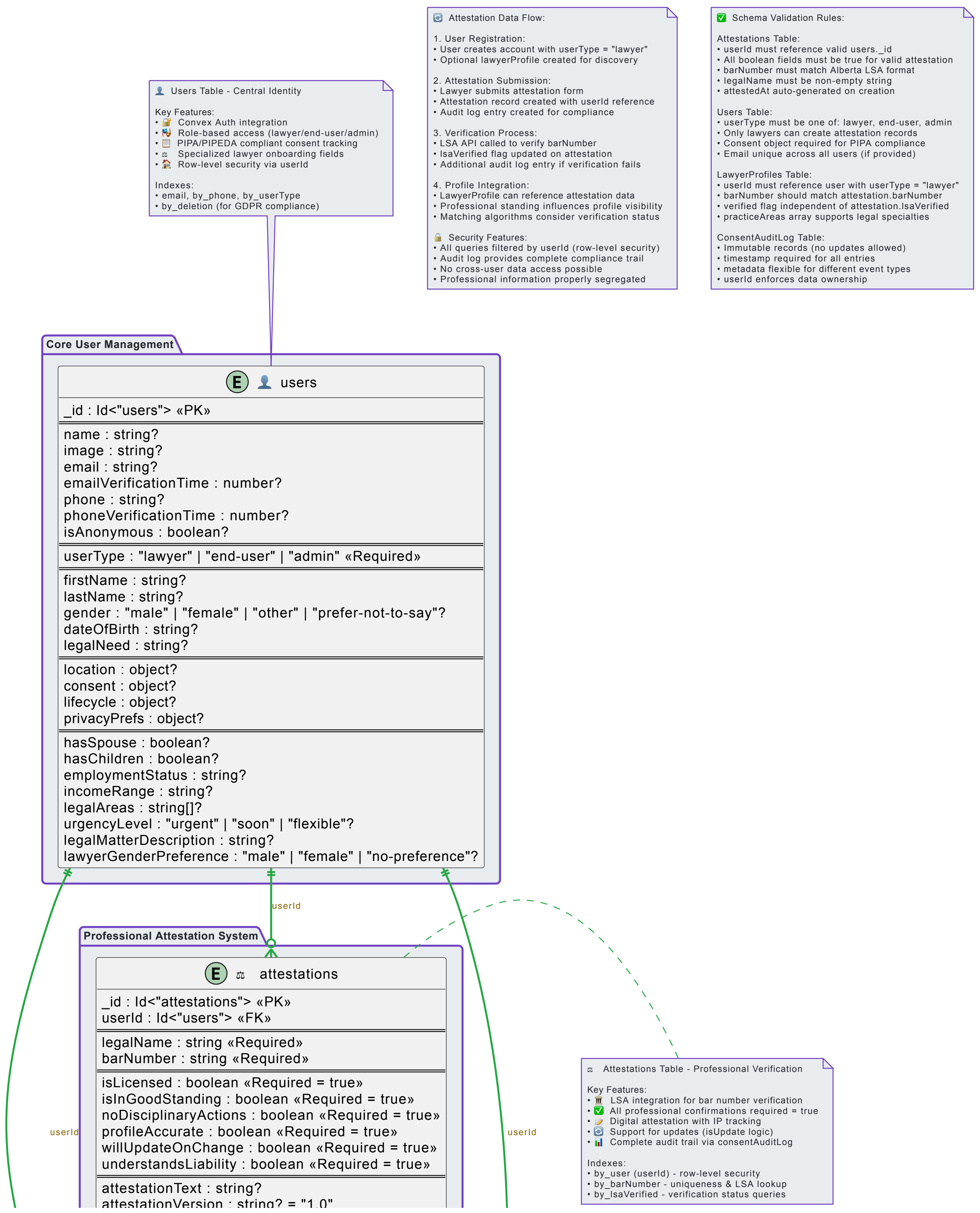
Lawyer Attestation - Visual Diagrams 📊

Think of these diagrams like **maps** that show you how our lawyer ID system works. Just like following directions to get somewhere, these show you the path that data takes through our system! 🗺️

0. Schema

Lawyer Attestation System - Database Schema & Relationships



👤 Users Table - Central Identity

Key Features:
• 🔐 Convex Auth integration
• 🎭 Role-based access (lawyer/end-user/admin)
• 📋 PIPA/PIPEDA compliant consent tracking
• ⚖ Specialized lawyer onboarding fields
• 🏠 Row-level security via userId

Indexes:
• email, by_phone, by_userType
• by_deletion (for GDPR compliance)

🔄 Attestation Data Flow:

1. User Registration:
• User creates account with userType = "lawyer"
• Optional lawyerProfile created for discovery

2. Attestation Submission:
• Lawyer submits attestation form
• Attestation record created with userId reference
• Audit log entry created for compliance

3. Verification Process:
• LSA API called to verify barNumber
• lsaVerified flag updated on attestation
• Additional audit log entry if verification fails

4. Profile Integration:
• LawyerProfile can reference attestation data
• Professional standing influences profile visibility
• Matching algorithms consider verification status

🔒 Security Features:
• All queries filtered by userId (row-level security)
• Audit log provides complete compliance trail
• No cross-user data access possible
• Professional information properly segregated

✅ Schema Validation Rules:

Attestations Table:
• userId must reference valid users._id
• All boolean fields must be true for valid attestation
• barNumber must match Alberta LSA format
• legalName must be non-empty string
• attestedAt auto-generated on creation

Users Table:
• userType must be one of: lawyer, end-user, admin
• Only lawyers can create attestation records
• Consent object required for PIPA compliance
• Email unique across all users (if provided)

LawyerProfiles Table:
• userId must reference user with userType = "lawyer"
• barNumber should match attestation.barNumber
• verified flag independent of attestation.lsaVerified
• practiceAreas array supports legal specialties

ConsentAuditLog Table:
• Immutable records (no updates allowed)
• timestamp required for all entries
• metadata flexible for different event types
• userId enforces data ownership

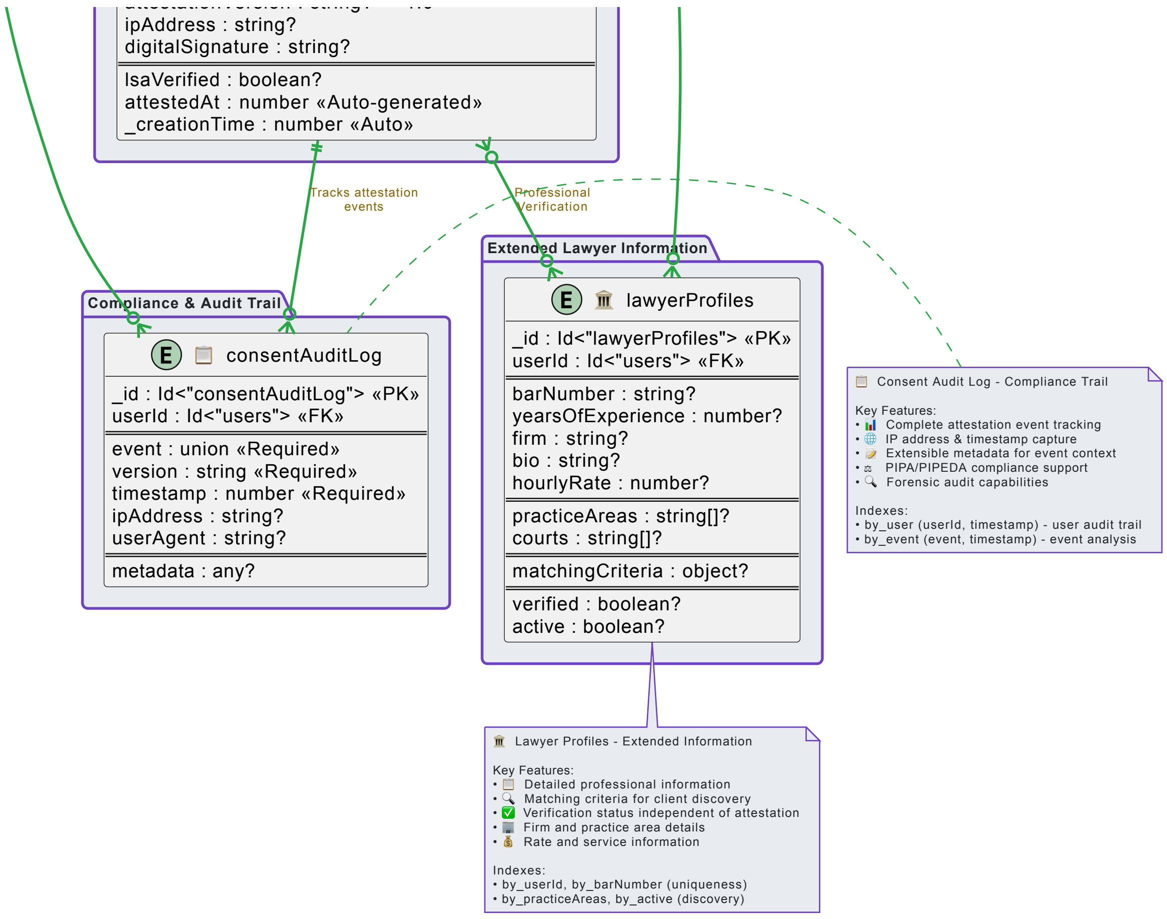
```
(E) 👤 users
_id : Id<"users"> «PK»
----
name : string?
image : string?
email : string?
emailVerificationTime : number?
phone : string?
phoneVerificationTime : number?
isAnonymous : boolean?
----
userType : "lawyer" | "end-user" | "admin" «Required»
----
firstName : string?
lastName : string?
gender : "male" | "female" | "other" | "prefer-not-to-say"?
dateOfBirth : string?
legalNeed : string?
----
location : object?
consent : object?
lifecycle : object?
privacyPrefs : object?
----
hasSpouse : boolean?
hasChildren : boolean?
employmentStatus : string?
incomeRange : string?
legalAreas : string[]?
urgencyLevel : "urgent" | "soon" | "flexible"?
legalMatterDescription : string?
lawyerGenderPreference : "male" | "female" | "no-preference"?
```

```
(E) ⚖ attestations
_id : Id<"attestations"> «PK»
userId : Id<"users"> «FK»
----
legalName : string «Required»
barNumber : string «Required»
----
isLicensed : boolean «Required = true»
isInGoodStanding : boolean «Required = true»
noDisciplinaryActions : boolean «Required = true»
profileAccurate : boolean «Required = true»
willUpdateOnChange : boolean «Required = true»
understandsLiability : boolean «Required = true»
----
attestationText : string?
attestationVersion : string? = "1.0"
```

⚖ Attestations Table - Professional Verification

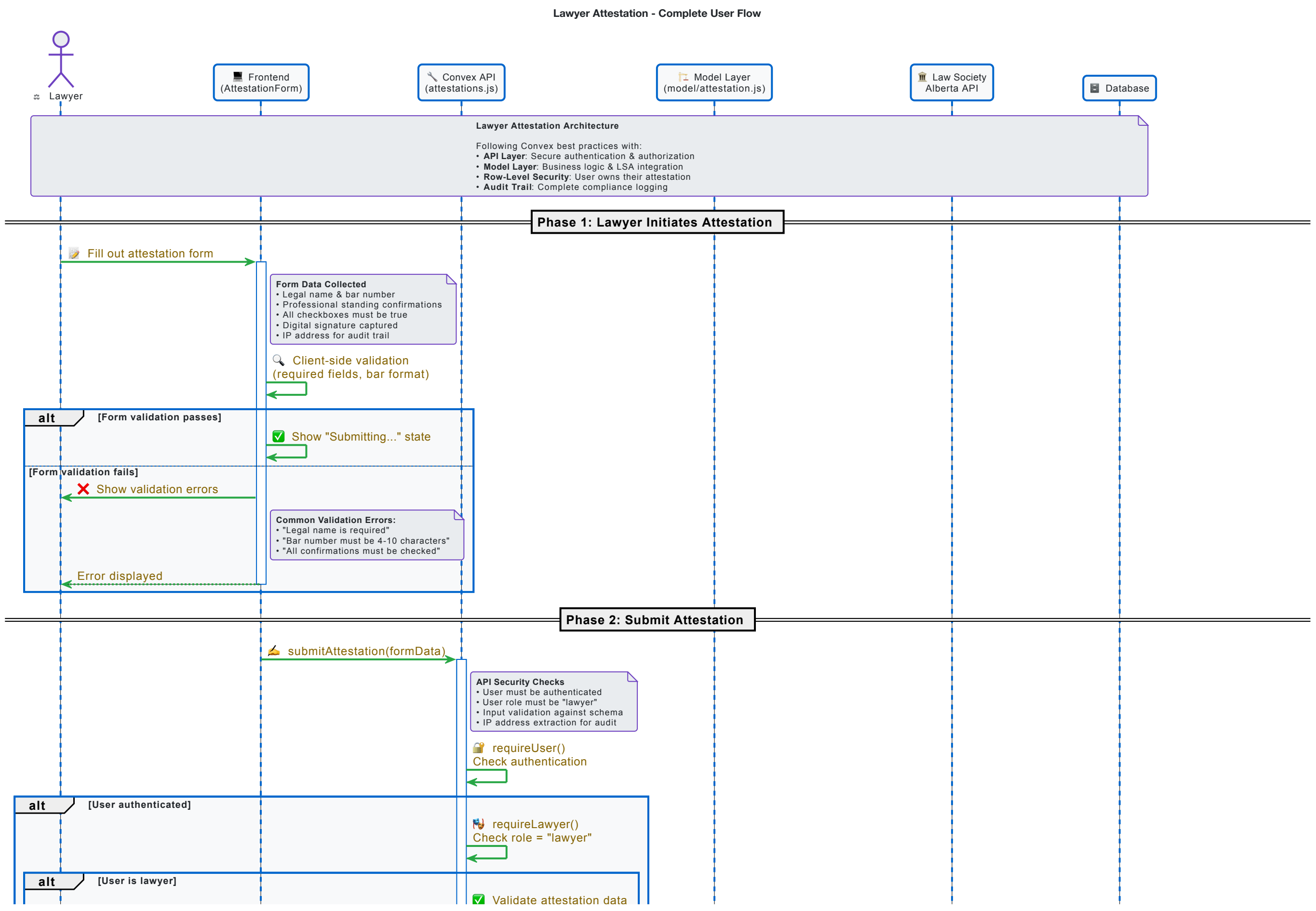
Key Features:
• 🏛 LSA integration for bar number verification
• ✅ All professional confirmations required = true
• 📝 Digital attestation with IP tracking
• 🔄 Support for updates (isUpdate logic)
• 📊 Complete audit trail via consentAuditLog

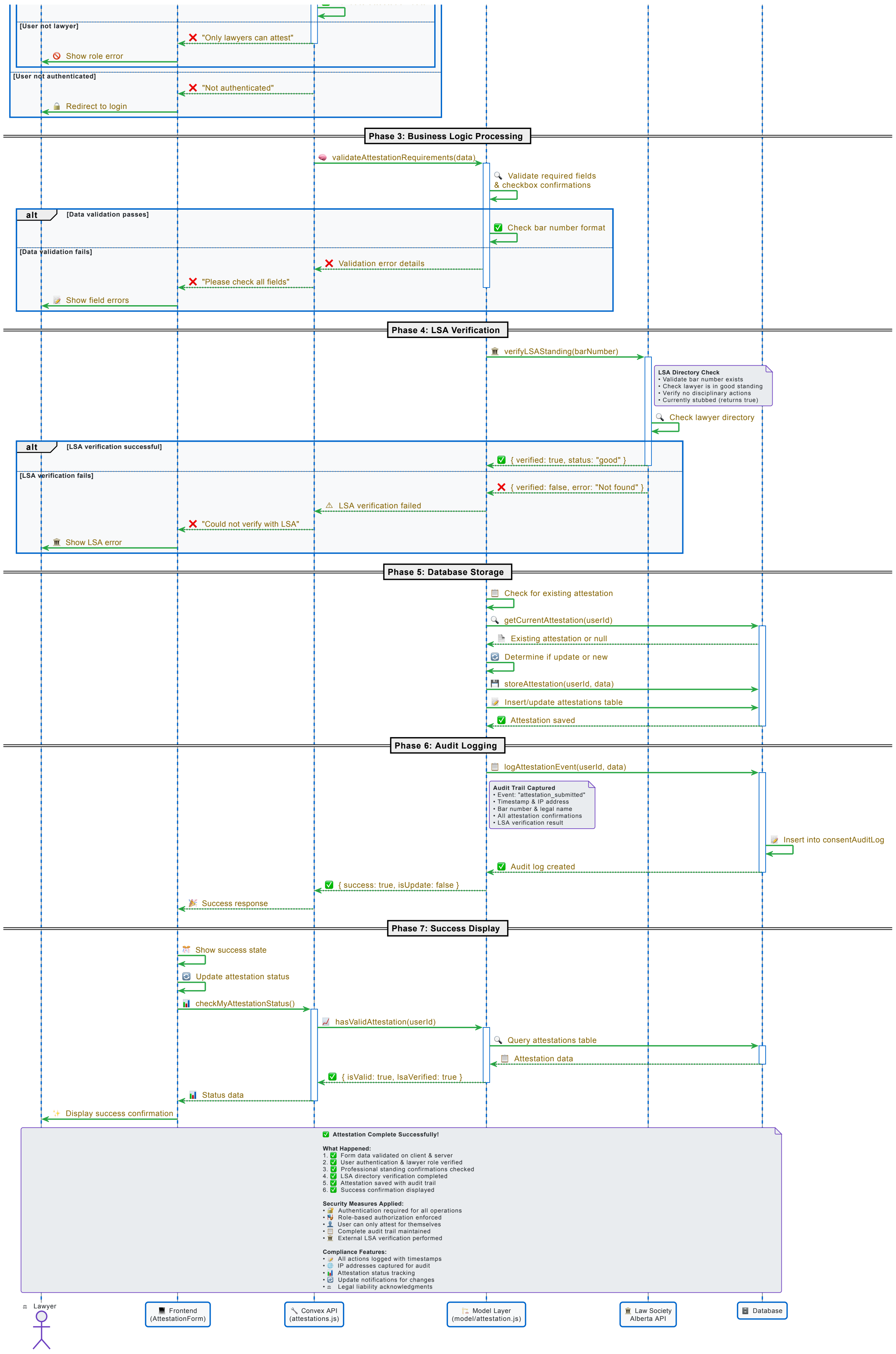
Indexes:
• by_user (userId) - row-level security
• by_barNumber - uniqueness & LSA lookup
• by_lsaVerified - verification status queries


ipAddress : string?
digitalSignature : string?
lsaVerified : boolean?
attestedAt : number «Auto-generated»
_creationTime : number «Auto»
Tracks attestation events
Professional Verification
Compliance & Audit Trail
E 📋 consentAuditLog
_id : Id<"consentAuditLog"> «PK»
userId : Id<"users"> «FK»
event : union «Required»
version : string «Required»
timestamp : number «Required»
ipAddress : string?
userAgent : string?
metadata : any?
Extended Lawyer Information
E 🏛 lawyerProfiles
_id : Id<"lawyerProfiles"> «PK»
userId : Id<"users"> «FK»
barNumber : string?
yearsOfExperience : number?
firm : string?
bio : string?
hourlyRate : number?
practiceAreas : string[]?
courts : string[]?
matchingCriteria : object?
verified : boolean?
active : boolean?
📋 Consent Audit Log - Compliance Trail
Key Features:
• 📊 Complete attestation event tracking
• 🌐 IP address & timestamp capture
• 📝 Extensible metadata for event context
• ⚖ PIPA/PIPEDA compliance support
• 🔍 Forensic audit capabilities
Indexes:
• by_user (userId, timestamp) - user audit trail
• by_event (event, timestamp) - event analysis
🏛 Lawyer Profiles - Extended Information
Key Features:
• 📋 Detailed professional information
• 🔍 Matching criteria for client discovery
• ✅ Verification status independent of attestation
• 🏢 Firm and practice area details
• 💰 Rate and service information
Indexes:
• by_userId, by_barNumber (uniqueness)
• by_practiceAreas, by_active (discovery)

1. The Complete Journey (Sequence Diagram) 🚗

How a lawyer goes from "filling out a form" to "having a verified digital ID"


Lawyer Attestation - Complete User Flow
⚖ Lawyer
💻 Frontend (AttestationForm)
🔧 Convex API (attestations.js)
🏗 Model Layer (model/attestation.js)
🏛 Law Society Alberta API
🗄 Database
Lawyer Attestation Architecture
Following Convex best practices with:
• API Layer: Secure authentication & authorization
• Model Layer: Business logic & LSA integration
• Row-Level Security: User owns their attestation
• Audit Trail: Complete compliance logging
Phase 1: Lawyer Initiates Attestation
📝 Fill out attestation form
Form Data Collected
• Legal name & bar number
• Professional standing confirmations
• All checkboxes must be true
• Digital signature captured
• IP address for audit trail
🔍 Client-side validation (required fields, bar format)
alt [Form validation passes]
✅ Show "Submitting..." state
[Form validation fails]
❌ Show validation errors
Common Validation Errors:
• "Legal name is required"
• "Bar number must be 4-10 characters"
• "All confirmations must be checked"
Error displayed
Phase 2: Submit Attestation
✍ submitAttestation(formData)
API Security Checks
• User must be authenticated
• User role must be "lawyer"
• Input validation against schema
• IP address extraction for audit
🔐 requireUser() Check authentication
alt [User authenticated]
🎭 requireLawyer() Check role = "lawyer"
alt [User is lawyer]
✅ Validate attestation data


[User not lawyer]
❌ "Only lawyers can attest"
🚫 Show role error
[User not authenticated]
❌ "Not authenticated"
🔒 Redirect to login
Phase 3: Business Logic Processing
🧠 validateAttestationRequirements(data)
🔍 Validate required fields & checkbox confirmations
alt [Data validation passes]
✅ Check bar number format
[Data validation fails]
❌ Validation error details
❌ "Please check all fields"
📝 Show field errors
Phase 4: LSA Verification
🏛 verifyLSAStanding(barNumber)
LSA Directory Check
• Validate bar number exists
• Check lawyer is in good standing
• Verify no disciplinary actions
• Currently stubbed (returns true)
🔍 Check lawyer directory
alt [LSA verification successful]
✅ { verified: true, status: "good" }
[LSA verification fails]
❌ { verified: false, error: "Not found" }
⚠ LSA verification failed
❌ "Could not verify with LSA"
🏛 Show LSA error
Phase 5: Database Storage
📋 Check for existing attestation
🔍 getCurrentAttestation(userId)
📄 Existing attestation or null
🔄 Determine if update or new
💾 storeAttestation(userId, data)
📝 Insert/update attestations table
✅ Attestation saved
Phase 6: Audit Logging
📋 logAttestationEvent(userId, data)
Audit Trail Captured
• Event: "attestation_submitted"
• Timestamp & IP address
• Bar number & legal name
• All attestation confirmations
• LSA verification result
📝 Insert into consentAuditLog
✅ Audit log created
✅ { success: true, isUpdate: false }
🎉 Success response
Phase 7: Success Display
🎊 Show success state
🔄 Update attestation status
📊 checkMyAttestationStatus()
📈 hasValidAttestation(userId)
🔍 Query attestations table
📋 Attestation data
✅ { isValid: true, lsaVerified: true }
📊 Status data
✨ Display success confirmation
⚖ Lawyer
💻 Frontend (AttestationForm)
🔧 Convex API (attestations.js)
🏗 Model Layer (model/attestation.js)
🏛 Law Society Alberta API
🗄 Database

✅ **Attestation Complete Successfully!**

What Happened:
1. ✅ Form data validated on client & server
2. ✅ User authentication & lawyer role verified
3. ✅ Professional standing confirmations checked
4. ✅ LSA directory verification completed
5. ✅ Attestation saved with audit trail
6. ✅ Success confirmation displayed

Security Measures Applied:
- 🔐 Authentication required for all operations
- 🎭 Role-based authorization enforced
- 👤 User can only attest for themselves
- 📋 Complete audit trail maintained
- 🏛 External LSA verification performed

Compliance Features:
- 📝 All actions logged with timestamps
- 🌐 IP addresses captured for audit
- 📊 Attestation status tracking
- 🔄 Update notifications for changes
- ⚖ Legal liability acknowledgments

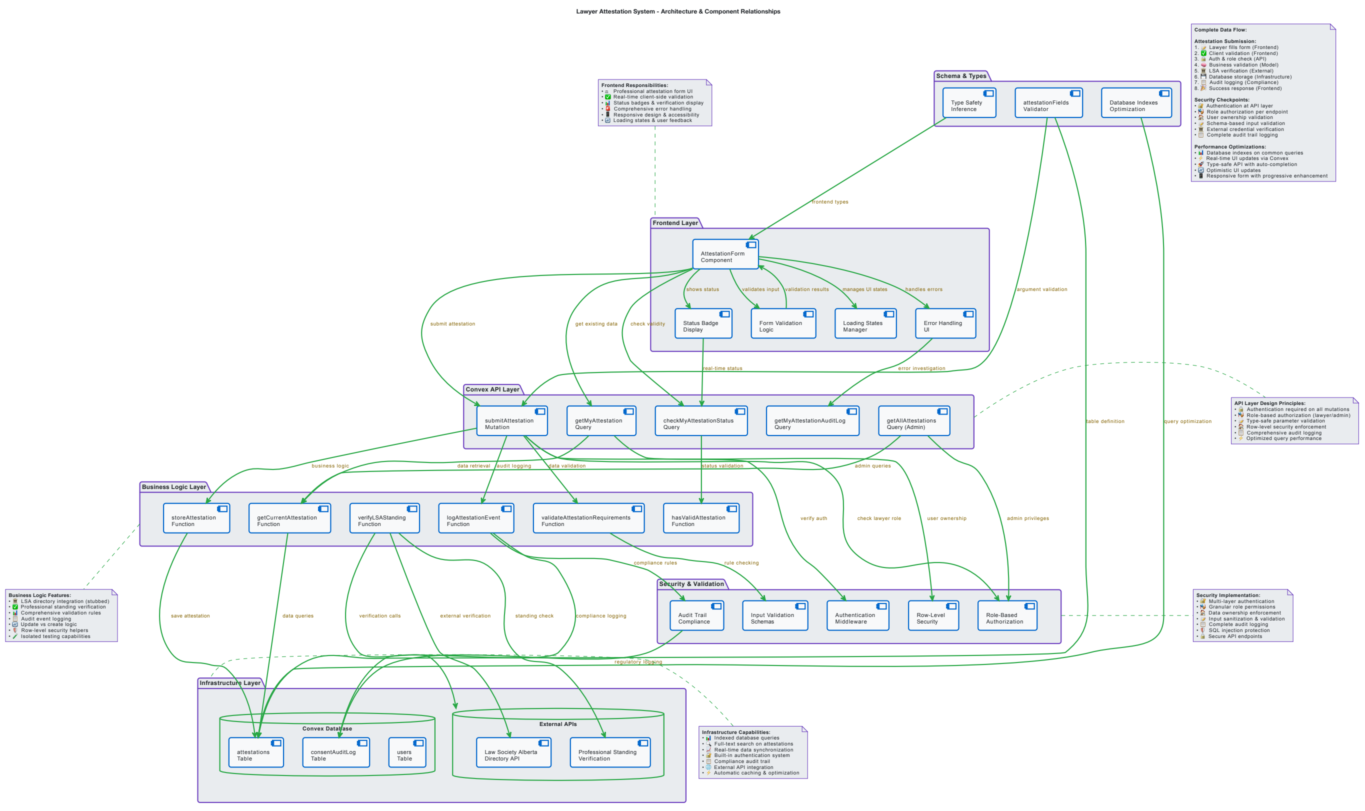
What This Means (Like You're 15):

1. 📝 **Fill out form**: Lawyer types their name, bar number, and checks boxes
2. 🚀 **Hit submit**: Form data flies to our server
3. 🔒 **Security check**: "Are you really logged in as a lawyer?"
4. ✅ **Validate**: "Does this look like real lawyer data?"
5. 📞 **Call LSA**: "Hey Law Society, is lawyer AB12345 legit?"
6. 👍 **LSA responds**: "Yep, that's a real lawyer!"
7. 💾 **Save it**: Store the attestation in our database
8. 📋 **Log it**: Write down "Lawyer X got verified at 2:30pm"
9. 🎉 **Celebrate**: Tell the frontend "It worked!"
10. 😊 **Happy lawyer**: Show success message

---

2. System Architecture (The Big Picture) 🏗️

Think of this like the blueprint for a house - showing all the rooms and how they connect


Lawyer Attestation System - Architecture & Component Relationships
Frontend Responsibilities:
• ⚖ Professional attestation form UI
• ✅ Real-time client-side validation
• 📊 Status badges & verification display
• 🚨 Comprehensive error handling
• 📱 Responsive design & accessibility
• 🔄 Loading states & user feedback
Schema & Types
Type Safety Inference
attestationFields Validator
Database Indexes Optimization
Complete Data Flow:
Attestation Submission:
1. 📝 Lawyer fills form (Frontend)
2. ✅ Client validation (Frontend)
3. 🔒 Auth & role check (API)
4. 🧠 Business validation (Model)
5. 🏛 LSA verification (External)
6. 💾 Database storage (Infrastructure)
7. 📋 Audit logging (Compliance)
8. 🎉 Success response (Frontend)
Security Checkpoints:
• 🔐 Authentication at API layer
• 🎭 Role authorization per endpoint
• 🏠 User ownership validation
• 📝 Schema-based input validation
• 🏛 External credential verification
• 📋 Complete audit trail logging
Performance Optimizations:
• 📊 Database indexes on common queries
• ⚡ Real-time UI updates via Convex
• 🚀 Type-safe API with auto-completion
• 🔄 Optimistic UI updates
• 📱 Responsive form with progressive enhancement
frontend types
Frontend Layer
AttestationForm Component
shows status
validates input
validation results
manages UI states
handles errors
argument validation
Status Badge Display
Form Validation Logic
Loading States Manager
Error Handling UI
submit attestation
get existing data
check validity
real-time status
error investigation
Convex API Layer
submitAttestation Mutation
getMyAttestation Query
checkMyAttestationStatus Query
getMyAttestationAuditLog Query
getAllAttestations Query (Admin)
table definition
query optimization
API Layer Design Principles:
• 🔒 Authentication required on all mutations
• 🎭 Role-based authorization (lawyer/admin)
• 📝 Type-safe parameter validation
• 🏠 Row-level security enforcement
• 📋 Comprehensive audit logging
• ⚡ Optimized query performance
business logic
data retrieval
audit logging
data validation
status validation
admin queries
Business Logic Layer
storeAttestation Function
getCurrentAttestation Function
verifyLSAStanding Function
logAttestationEvent Function
validateAttestationRequirements Function
hasValidAttestation Function
verify auth
check lawyer role
user ownership
admin privileges
compliance rules
rule checking
Business Logic Features:
• 🏛 LSA directory integration (stubbed)
• ✅ Professional standing verification
• 📊 Comprehensive validation rules
• 📋 Audit event logging
• 🔄 Update vs create logic
• 🛡 Row-level security helpers
• 🧪 Isolated testing capabilities
save attestation
data queries
verification calls
external verification
standing check
compliance logging
Security & Validation
Audit Trail Compliance
Input Validation Schemas
Authentication Middleware
Row-Level Security
Role-Based Authorization
Security Implementation:
• 🔐 Multi-layer authentication
• 🎭 Granular role permissions
• 🏠 Data ownership enforcement
• 📝 Input sanitization & validation
• 📋 Complete audit logging
• 🛡 SQL injection protection
• 🔒 Secure API endpoints
regulatory logging
Infrastructure Layer
Convex Database
attestations Table
consentAuditLog Table
users Table
External APIs
Law Society Alberta Directory API
Professional Standing Verification
Infrastructure Capabilities:
• 📊 Indexed database queries
• 🔍 Full-text search on attestations
• 📈 Real-time data synchronization
• 🔐 Built-in authentication system
• 📋 Compliance audit trail
• 🌐 External API integration
• ⚡ Automatic caching & optimization

What This Means (Like You're 15):

- 🌍 **Frontend**: The pretty stuff you see (forms, buttons, messages)
- 🌐 **API Layer**: The security guard who checks IDs and decides who gets in
- 🧠 **Business Logic**: The smart brain that knows the rules and does the work
- 🗄️ **Database**: The filing cabinet where we store everything safely

---

3. Error Handling Flow (When Things Go Wrong) 🚨

Even superheroes have bad days! Here's what happens when something breaks:

Error Types & What They Mean:

🔴 Frontend Errors (You can fix these!)

- **Empty name**: "Please enter your legal name"
- **Bad bar number**: "Bar number must be 4-10 characters"
- **Unchecked boxes**: "Please confirm all statements"

🟠 API Errors (System problems)

- **Not logged in**: "Please sign in to continue"
- **Wrong user type**: "Only lawyers can attest"
- **Network down**: "Connection problem, try again"

🟡 LSA Errors (External problems)

- **LSA API down**: "Verification service temporarily unavailable"
- **Invalid bar number**: "Bar number not found in LSA directory"
- **Suspended lawyer**: "Professional standing issues detected"

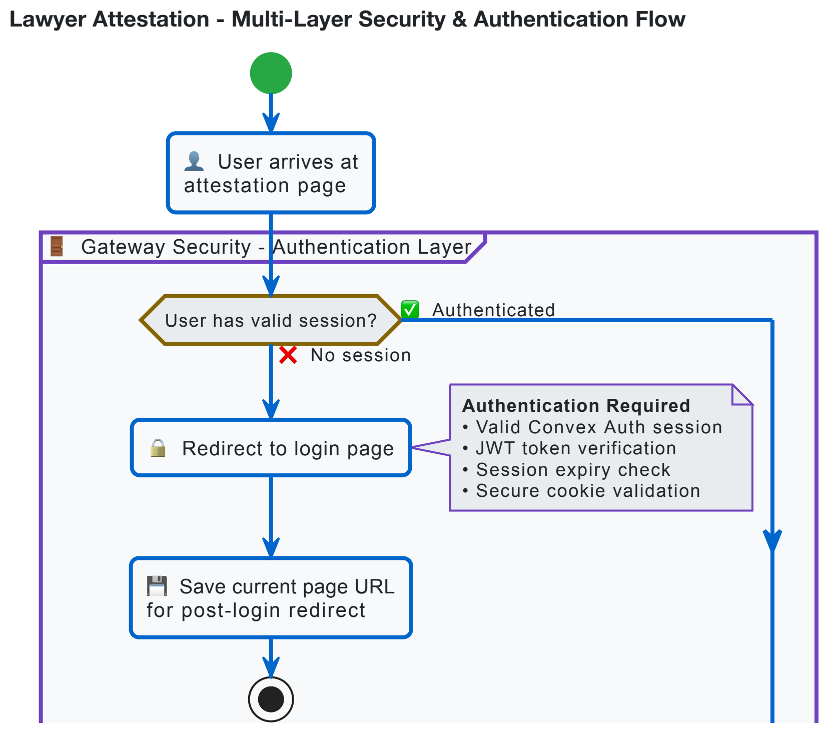
🔵 Database Errors (Rare but possible)

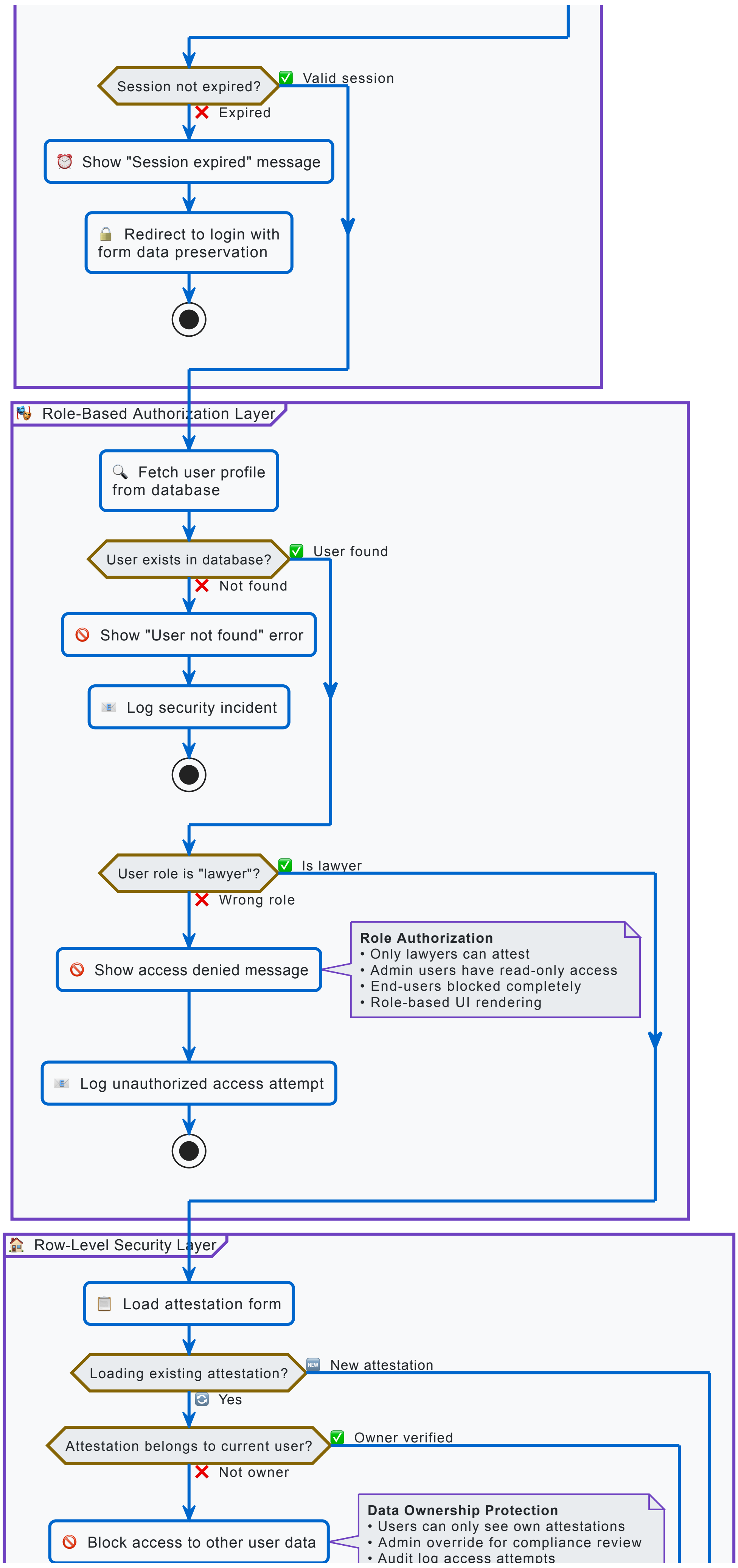
- **Save failed**: "Could not save attestation, please retry"
- **Read failed**: "Could not load your existing data"

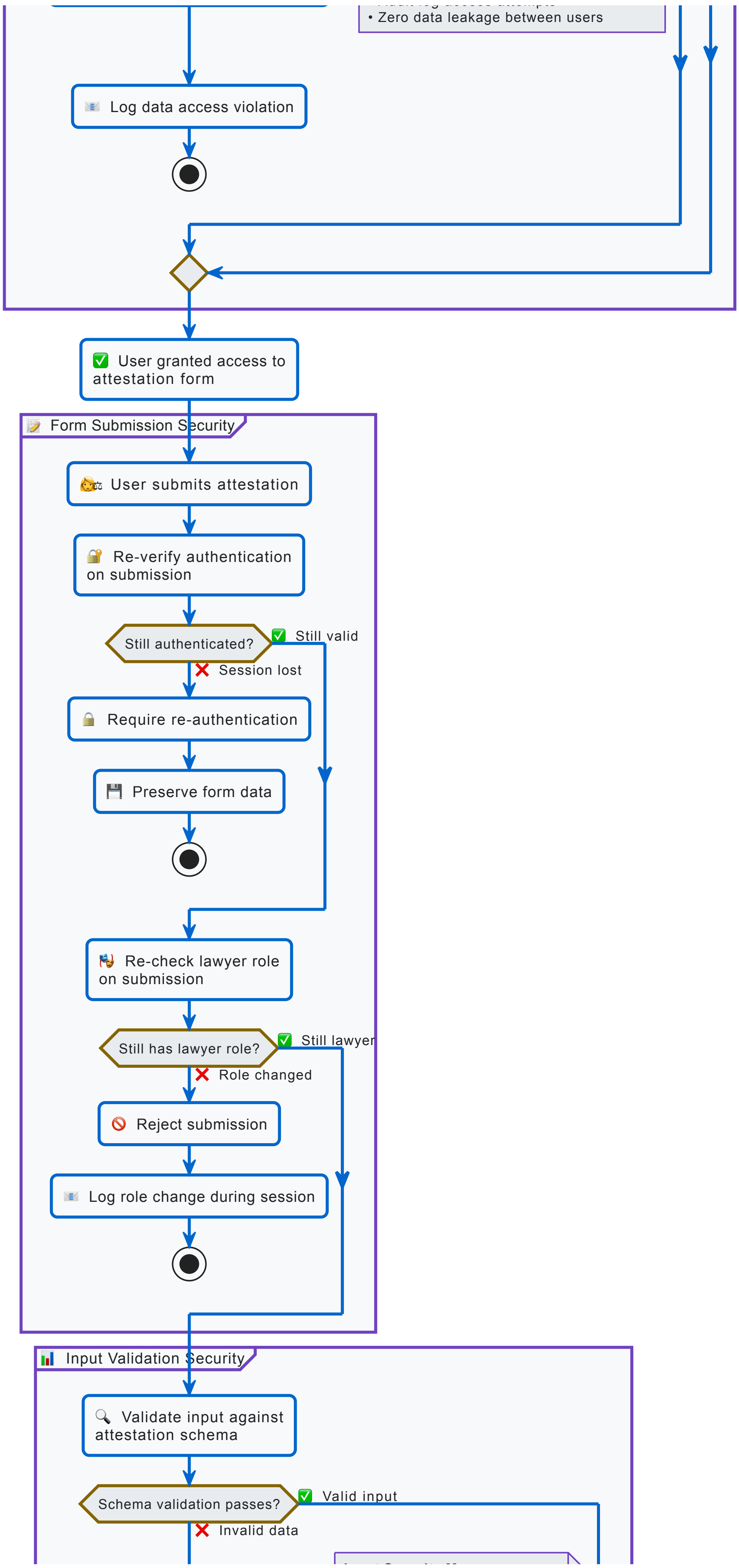
---

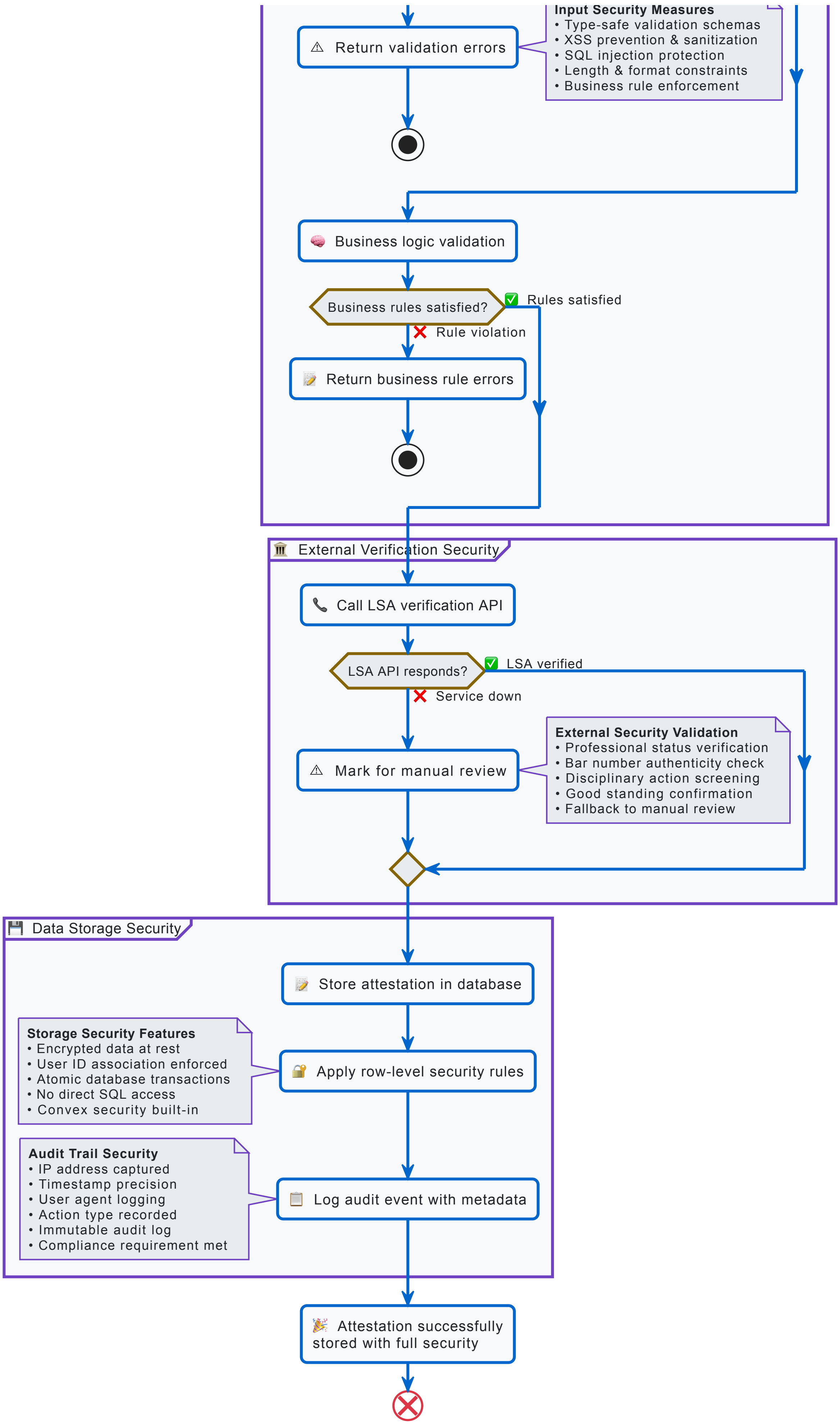
4. Security & Authentication Flow 🛡️

Like having bouncers at a VIP club - multiple checkpoints to keep things safe!


Lawyer Attestation - Multi-Layer Security & Authentication Flow
👤 User arrives at attestation page
🚪 Gateway Security - Authentication Layer
User has valid session?
✅ Authenticated
❌ No session
🔒 Redirect to login page
Authentication Required
• Valid Convex Auth session
• JWT token verification
• Session expiry check
• Secure cookie validation
💾 Save current page URL for post-login redirect


Session not expired?
✅ Valid session
❌ Expired
⏰ Show "Session expired" message
🔒 Redirect to login with form data preservation
🎭 Role-Based Authorization Layer
🔍 Fetch user profile from database
User exists in database?
✅ User found
❌ Not found
🚫 Show "User not found" error
Log security incident
User role is "lawyer"?
✅ Is lawyer
❌ Wrong role
🚫 Show access denied message
Role Authorization
• Only lawyers can attest
• Admin users have read-only access
• End-users blocked completely
• Role-based UI rendering
Log unauthorized access attempt
🏠 Row-Level Security Layer
📋 Load attestation form
Loading existing attestation?
🆕 New attestation
🔄 Yes
Attestation belongs to current user?
✅ Owner verified
❌ Not owner
🚫 Block access to other user data
Data Ownership Protection
• Users can only see own attestations
• Admin override for compliance review
• Audit log access attempts


• Zero data leakage between users
Log data access violation
✅ User granted access to attestation form
Form Submission Security
User submits attestation
Re-verify authentication on submission
Still authenticated?
✅ Still valid
❌ Session lost
Require re-authentication
Preserve form data
Re-check lawyer role on submission
Still has lawyer role?
✅ Still lawyer
❌ Role changed
Reject submission
Log role change during session
Input Validation Security
Validate input against attestation schema
Schema validation passes?
✅ Valid input
❌ Invalid data


⚠ Return validation errors
Input Security Measures
• Type-safe validation schemas
• XSS prevention & sanitization
• SQL injection protection
• Length & format constraints
• Business rule enforcement
🧠 Business logic validation
Business rules satisfied?
✅ Rules satisfied
❌ Rule violation
📝 Return business rule errors
🏛 External Verification Security
📞 Call LSA verification API
LSA API responds?
✅ LSA verified
❌ Service down
⚠ Mark for manual review
External Security Validation
• Professional status verification
• Bar number authenticity check
• Disciplinary action screening
• Good standing confirmation
• Fallback to manual review
💾 Data Storage Security
📝 Store attestation in database
Storage Security Features
• Encrypted data at rest
• User ID association enforced
• Atomic database transactions
• No direct SQL access
• Convex security built-in
🔐 Apply row-level security rules
Audit Trail Security
• IP address captured
• Timestamp precision
• User agent logging
• Action type recorded
• Immutable audit log
• Compliance requirement met
📋 Log audit event with metadata
🎉 Attestation successfully stored with full security

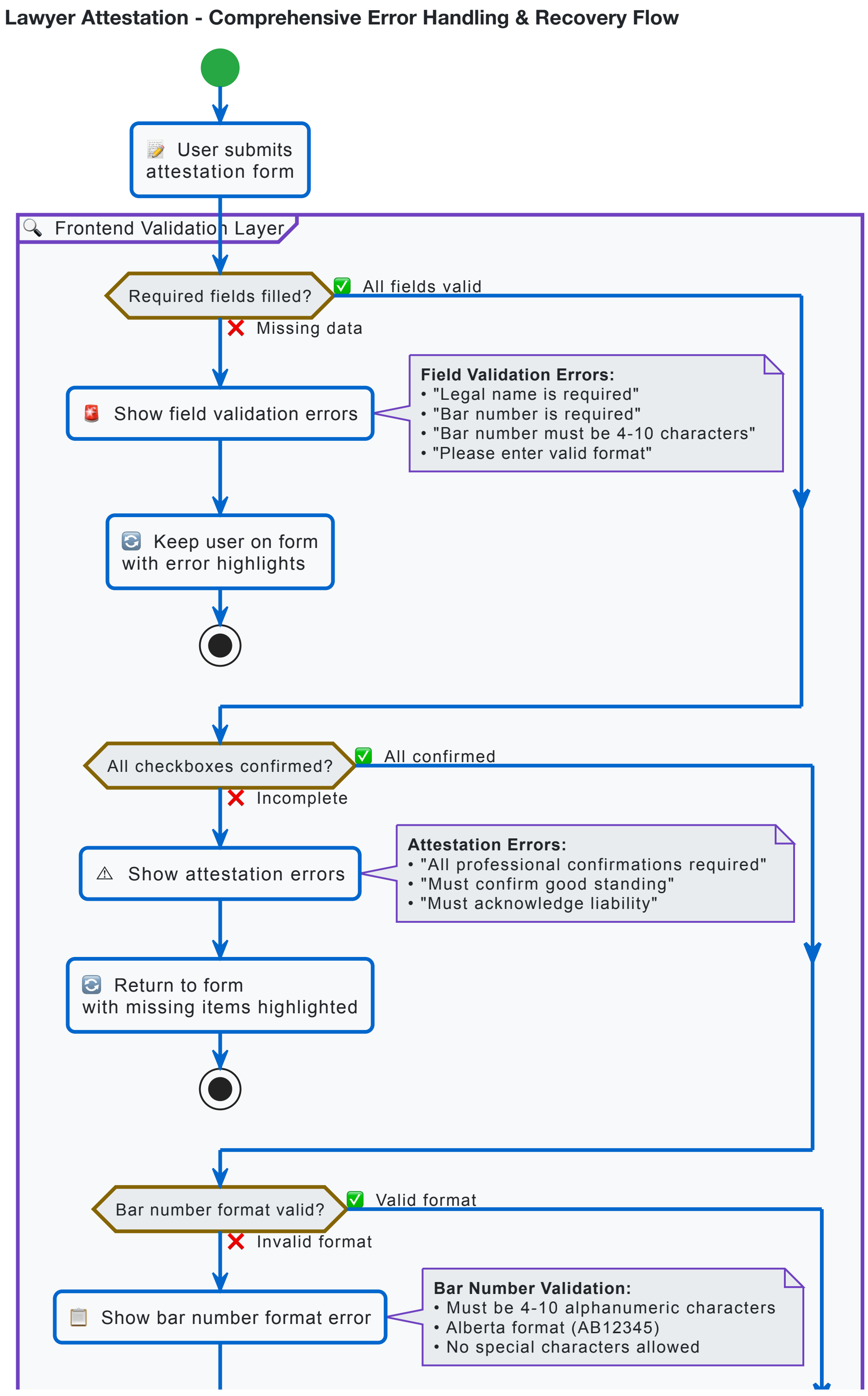
Security Layers Explained:

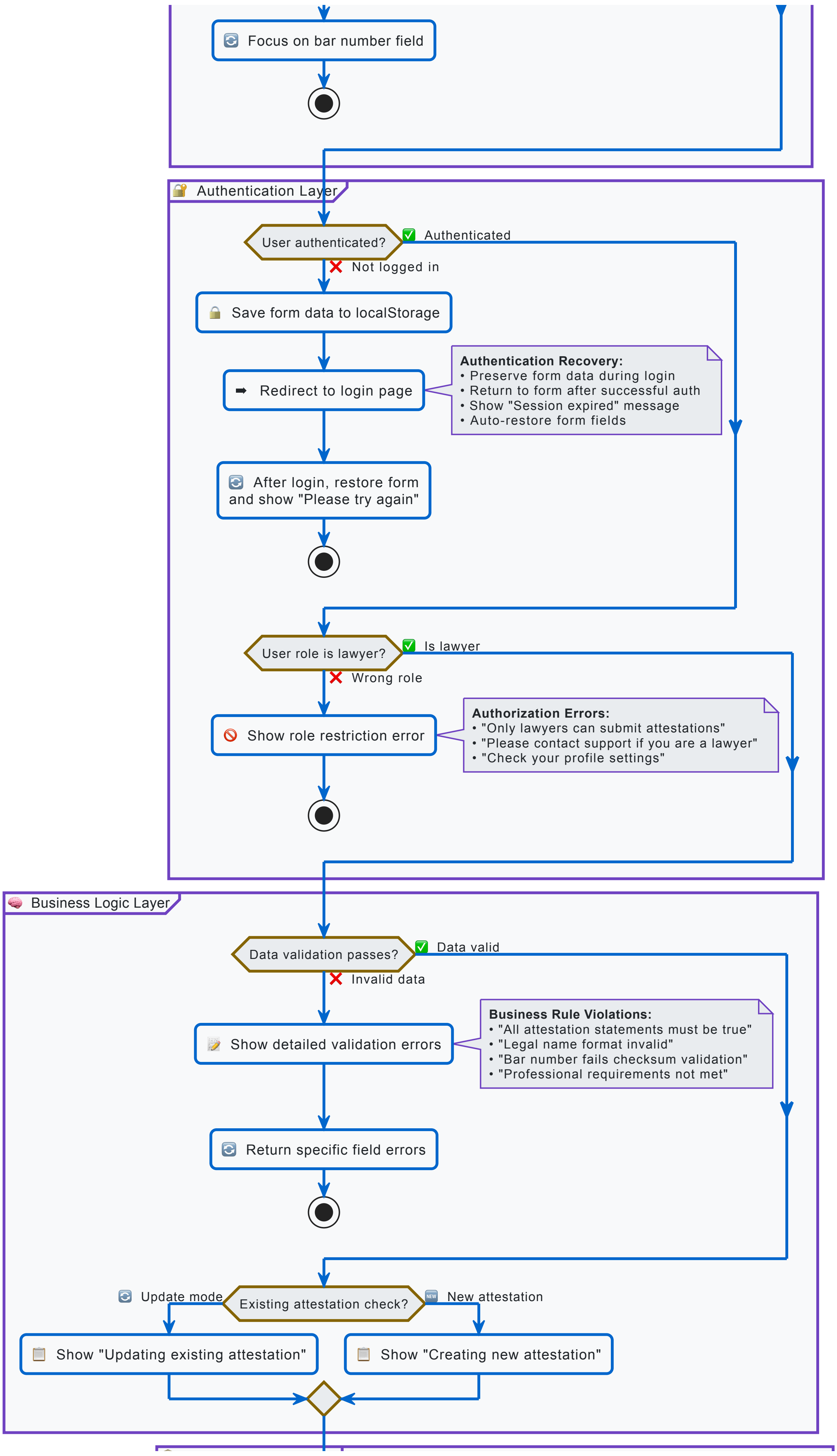
1. 🔐 **Authentication**: "Who are you?" (Login required)
2. 🎭 **Authorization**: "What can you do?" (Role check)
3. 🏠 **Ownership**: "Is this yours?" (Data privacy)
4. ✅ **Validation**: "Is this data good?" (Quality check)
5. 🏛 **Verification**: "Is this real?" (External validation)

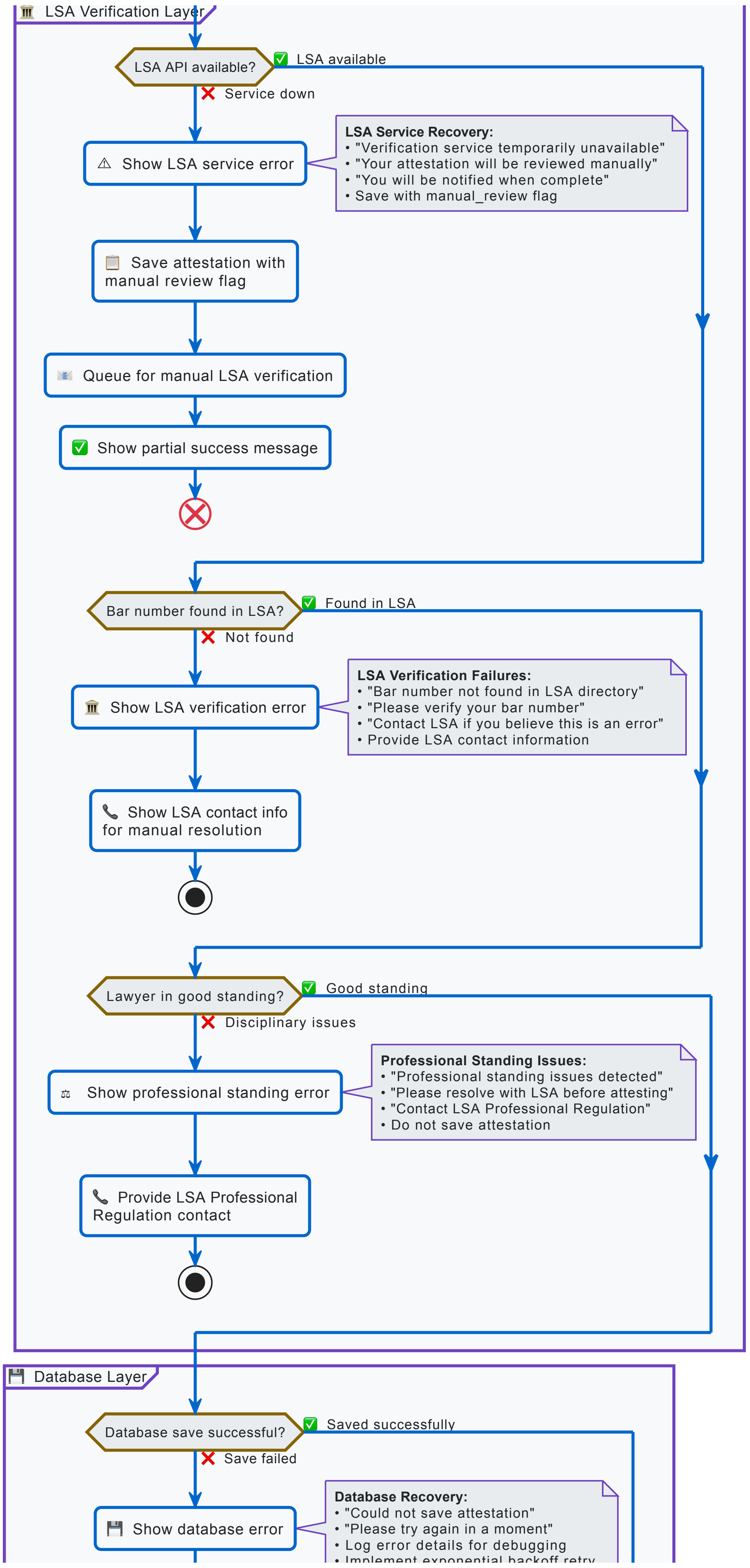
---

5. Data Journey Map 🗺

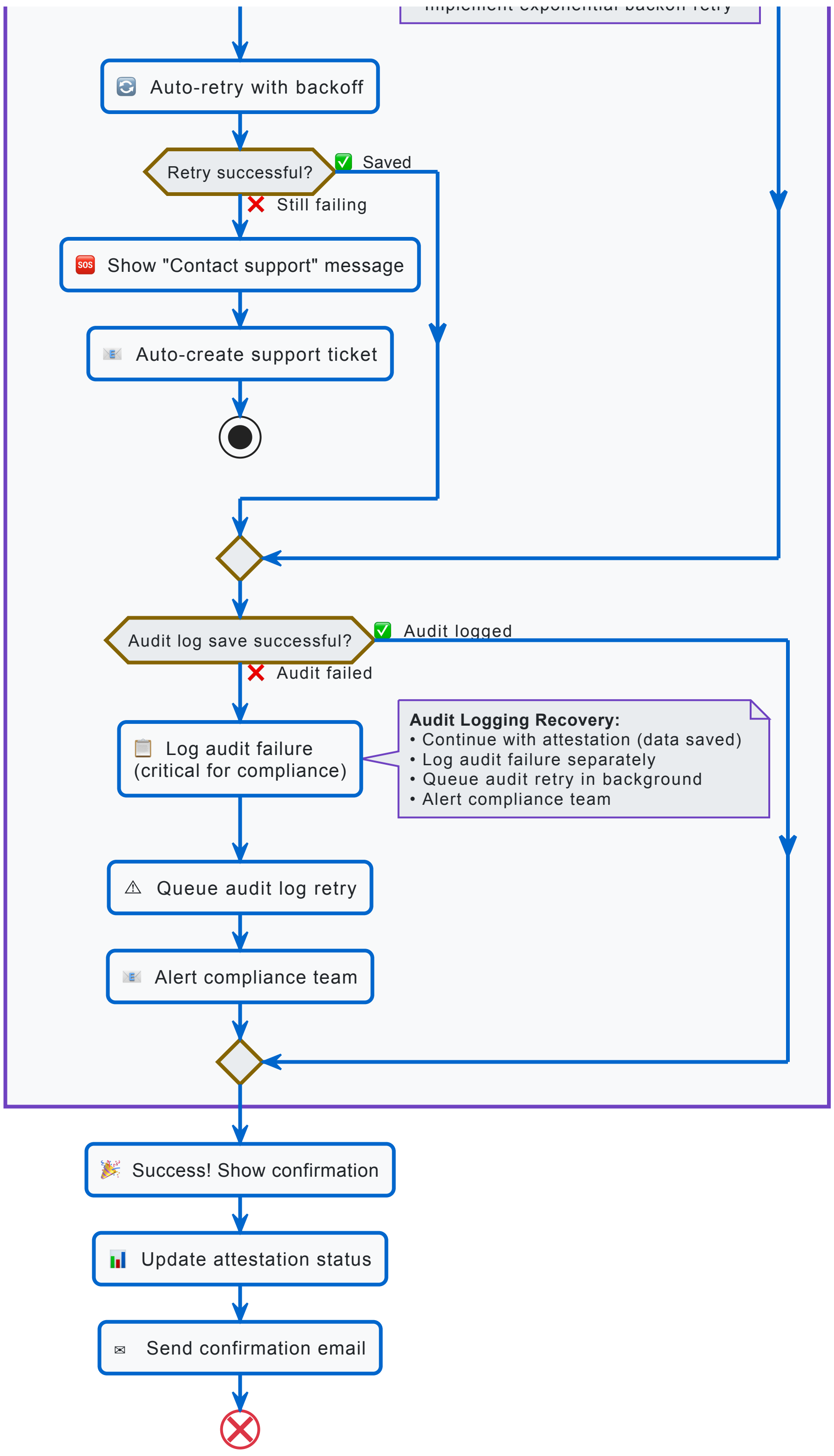
Follow the data from form field to database!


Lawyer Attestation - Comprehensive Error Handling & Recovery Flow
📝 User submits attestation form
🔍 Frontend Validation Layer
Required fields filled?
✅ All fields valid
❌ Missing data
🚨 Show field validation errors
Field Validation Errors:
• "Legal name is required"
• "Bar number is required"
• "Bar number must be 4-10 characters"
• "Please enter valid format"
🔄 Keep user on form with error highlights
All checkboxes confirmed?
✅ All confirmed
❌ Incomplete
⚠ Show attestation errors
Attestation Errors:
• "All professional confirmations required"
• "Must confirm good standing"
• "Must acknowledge liability"
🔄 Return to form with missing items highlighted
Bar number format valid?
✅ Valid format
❌ Invalid format
📋 Show bar number format error
Bar Number Validation:
• Must be 4-10 alphanumeric characters
• Alberta format (AB12345)
• No special characters allowed


Focus on bar number field
Authentication Layer
User authenticated?
Authenticated
Not logged in
Save form data to localStorage
Redirect to login page
Authentication Recovery:
• Preserve form data during login
• Return to form after successful auth
• Show "Session expired" message
• Auto-restore form fields
After login, restore form and show "Please try again"
User role is lawyer?
Is lawyer
Wrong role
Show role restriction error
Authorization Errors:
• "Only lawyers can submit attestations"
• "Please contact support if you are a lawyer"
• "Check your profile settings"
Business Logic Layer
Data validation passes?
Data valid
Invalid data
Show detailed validation errors
Business Rule Violations:
• "All attestation statements must be true"
• "Legal name format invalid"
• "Bar number fails checksum validation"
• "Professional requirements not met"
Return specific field errors
Update mode
Existing attestation check?
New attestation
Show "Updating existing attestation"
Show "Creating new attestation"



🏛 LSA Verification Layer
LSA API available?
✅ LSA available
❌ Service down
⚠ Show LSA service error
LSA Service Recovery:
• "Verification service temporarily unavailable"
• "Your attestation will be reviewed manually"
• "You will be notified when complete"
• Save with manual_review flag
📋 Save attestation with manual review flag
📧 Queue for manual LSA verification
✅ Show partial success message
Bar number found in LSA?
✅ Found in LSA
❌ Not found
🏛 Show LSA verification error
LSA Verification Failures:
• "Bar number not found in LSA directory"
• "Please verify your bar number"
• "Contact LSA if you believe this is an error"
• Provide LSA contact information
📞 Show LSA contact info for manual resolution
Lawyer in good standing?
✅ Good standing
❌ Disciplinary issues
⚖ Show professional standing error
Professional Standing Issues:
• "Professional standing issues detected"
• "Please resolve with LSA before attesting"
• "Contact LSA Professional Regulation"
• Do not save attestation
📞 Provide LSA Professional Regulation contact
💾 Database Layer
Database save successful?
✅ Saved successfully
❌ Save failed
💾 Show database error
Database Recovery:
• "Could not save attestation"
• "Please try again in a moment"
• Log error details for debugging
• Implement exponential backoff retry



Implement exponential backoff retry
🔄 Auto-retry with backoff
Retry successful?
✅ Saved
❌ Still failing
🆘 Show "Contact support" message
📧 Auto-create support ticket
Audit log save successful?
✅ Audit logged
❌ Audit failed
📋 Log audit failure (critical for compliance)
Audit Logging Recovery:
• Continue with attestation (data saved)
• Log audit failure separately
• Queue audit retry in background
• Alert compliance team
⚠️ Queue audit log retry
📧 Alert compliance team
🎉 Success! Show confirmation
📊 Update attestation status
✉️ Send confirmation email

What Gets Stored:

```
🗂️ attestations table:
├── userId: "user_123abc"
├── legalName: "Jane Smith"
├── barNumber: "AB12345"
├── isLicensed: true
├── isInGoodStanding: true
├── ... (all checkboxes)
├── lsaVerified: true
├── attestedAt: 1640995200000

📋 consentAuditLog table:
├── userId: "user_123abc"
├── event: "attestation_submitted"
├── timestamp: 1640995200000
├── ipAddress: "192.168.1.1"
├── metadata: { barNumber: "AB12345", ... }
```

🎯 How to Use These Diagrams

For Frontend Developers:

1. 📊 **Use the sequence diagram** to understand the API flow
2. 🚨 **Check error handling** before building your UI states
3. 🔒 **Follow security patterns** for proper authentication
4. 📝 **Reference data structures** when building forms

For Backend Developers:

1. 🏗️ **Architecture diagram** shows your component boundaries
2. 🔒 **Security flow** guides your middleware and validation
3. 📋 **Audit requirements** are built into the system
4. 🗄️ **Database design** supports all the query patterns

For QA/Testing:

1. 🚨 **Error scenarios** give you a testing matrix
2. 🔒 **Security checkpoints** need individual testing
3. 📊 **Data flow** shows all integration points
4. 🎯 **Happy path** (sequence) should always work

For Product Managers:

1. **Simple flow** = better user experience
2. **Multiple security layers** = compliance friendly
3. **Comprehensive logging** = audit trail complete
4. **Error recovery** = fewer support tickets

🎓 Key Takeaways (TL;DR)

1. 🎯 **Simple for users**: Fill form → Check boxes → Submit → Done!
2. 🔒 **Secure by design**: Multiple authentication & validation layers
3. 📋 **Audit everything**: Every action gets logged for compliance
4. 🏛️ **LSA integrated**: Automatic verification with Law Society
5. 🚨 **Handle errors gracefully**: Clear messages, easy recovery
6. 🏗️ **Clean architecture**: Separated layers, maintainable code

Remember: This is like creating a digital ID card for lawyers - it needs to be **simple to use** but **impossible to fake**! 🎯 ✨