

# Lawyer Attestation - Frontend Implementation Guide



## What is this? (Explain Like I'm 15)

Think of lawyer attestation like getting a **digital ID card** for lawyers. Just like how you need to prove you're a student to get student discounts, lawyers need to prove they're real lawyers to use our platform.

### Here's what happens:

1. 🧑 A lawyer fills out a form saying "I'm a real lawyer"
2. ✅ They check boxes confirming they're legit
3. 🗄️ Our system saves this info securely
4. 📅 Later, we can check "Is this person really a lawyer?"



## Quick Start (Copy-Paste Ready!)

### Step 1: Import the Hooks

```
import { useMutation, useQuery } from "convex/react";
import { Authenticated, Unauthenticated } from "convex/react";
import { api } from "../convex/_generated/api";
```

### Step 2: Basic Component Template

```
import { useState } from "react";

function LawyerAttestationForm() {
  // 🐟 These are your "fishing hooks" to talk to the backend
  const submitAttestation =
    useMutation(api.attestations.submitAttestation);
  const currentAttestation = useQuery(api.attestations.getMyAttestation);
  const attestationStatus =
    useQuery(api.attestations.checkMyAttestationStatus);

  // 📝 Form data state
  const [formData, setFormData] = useState({
    legalName: "",
    barNumber: "",
    isLicensed: false,
    isInGoodStanding: false,
    noDisciplinaryActions: false,
    profileAccurate: false,
    willUpdateOnChange: false,
    understandsLiability: false,
  });
```

```
const [isSubmitting, setIsSubmitting] = useState(false);




return (
  <Authenticated>
    { /* Your form goes here! */ }
    <AttestationFormContent />
  </Authenticated>
);
}
```

## Available APIs (Your Toolkit)

### 1. Submit Attestation 📝

**What it does:** Saves the lawyer's attestation form




```
const result = await submitAttestation({
  legalName: "Jane Smith",
  barNumber: "AB12345",
  isLicensed: true,
  isInGoodStanding: true,
  noDisciplinaryActions: true,
  profileAccurate: true,
  willUpdateOnChange: true,
  understandsLiability: true,
});
// Returns: { success: true, isUpdate: false }
```

-  **Auth Required:** Yes (only lawyers can attest)
-  **Updates:** If lawyer already has attestation, it updates it
-  **Audit Trail:** Automatically logs everything for compliance

### 2. Get My Attestation 📄

**What it does:** Gets the current lawyer's existing attestation




```
const attestation = useQuery(api.attestations.getMyAttestation);
// Returns: { legalName: "Jane Smith", barNumber: "AB12345", ... } or null
```

-  **Auth Required:** Yes (only your own data)
-  **Real-time:** Updates automatically when data changes
-  **Fast:** Use "skip" if user isn't a lawyer

### 3. Check Attestation Status 🔍

**What it does:** Quick check if lawyer's attestation is valid

```
const status = useQuery(api.attestations.checkMyAttestationStatus);
// Returns: {
//   isValid: true,
//   hasAttestation: true,
//   attestedAt: 1640995200000,
//   lsaVerified: true
// }
```

-  **Status Light:** Use for showing green/red status indicators
-  **Badges:** Show "LSA Verified" badges when `lsaVerified: true`
-  **Timestamps:** Show "Attested on [date]" with `attestedAt`

## Complete Form Component (Copy This!)

```
import { useState } from "react";
import { useMutation, useQuery } from "convex/react";
import { Authenticated, Unauthenticated } from "convex/react";
import { api } from "../convex/_generated/api";

export function AttestationForm() {
  // 🔄 Backend connections
  const submitAttestation =
    useMutation(api.attestations.submitAttestation);
  const currentAttestation = useQuery(api.attestations.getMyAttestation);
  const attestationStatus =
    useQuery(api.attestations.checkMyAttestationStatus);

  // 📝 Form state
  const [formData, setFormData] = useState({
    legalName: "",
    barNumber: "",
    isLicensed: false,
    isInGoodStanding: false,
    noDisciplinaryActions: false,
    profileAccurate: false,
    willUpdateOnChange: false,
    understandsLiability: false,
  });

  const [isSubmitting, setIsSubmitting] = useState(false);
  const [submitSuccess, setSubmitSuccess] = useState(false);
  const [submitError, setSubmitError] = useState(null);

  // 🔄 Pre-fill form with existing data
  useState(() => {
    if (currentAttestation) {
      setFormData(prev => ({
        ...prev,
        legalName: currentAttestation.legalName || "",
        barNumber: currentAttestation.barNumber || "",
      }));
    }
  });
}
```

```

        isLicensed: currentAttestation.isLicensed || false,
        isInGoodStanding: currentAttestation.isInGoodStanding || false,
        noDisciplinaryActions: currentAttestation.noDisciplinaryActions ||
false,
        profileAccurate: currentAttestation.profileAccurate || false,
        willUpdateOnChange: currentAttestation.willUpdateOnChange ||
false,
        understandsLiability: currentAttestation.understandsLiability ||
false,
      }));
    }
  }, [currentAttestation]);

// 🚧 Handle form submission
const handleSubmit = async (e) => {
  e.preventDefault();
  setIsSubmitting(true);
  setSubmitError(null);

  try {
    const result = await submitAttestation(formData);
    setSubmitSuccess(true);
    console.log("Success!", result);
  } catch (error) {
    setSubmitError(error.message);
    console.error("Error:", error);
  } finally {
    setIsSubmitting(false);
  }
};

// 🚧 Form validation
const isFormValid = () => {
  return (
    formData.legalName.trim() &&
    formData.barNumber.trim() &&
    formData.isLicensed &&
    formData.isInGoodStanding &&
    formData.noDisciplinaryActions &&
    formData.profileAccurate &&
    formData.willUpdateOnChange &&
    formData.understandsLiability
  );
};

return (
  <>
    {/* 🚫 Not logged in */}
    <Unauthenticated>
      <div className="text-center p-6">
        <h2 className="text-2xl font-bold mb-4">Sign In Required</h2>
        <p>Please sign in to complete your lawyer attestation.</p>
      </div>
    </Unauthenticated>
  </>
);

```

```

    { /* ✅ Logged in */}
    <Authenticated>
      <div className="max-w-2xl mx-auto p-6 bg-white rounded-lg shadow-
md">

        { /* 🏠 Status Badge */}
        {attestationStatus?.hasAttestation && (
          <div className="mb-6 flex items-center gap-2">
            <span className={`w-3 h-3 rounded-full ${
              attestationStatus.isValid ? 'bg-green-500' : 'bg-yellow-
500'
            }`} ></span>
            <span>
              {attestationStatus.isValid ? '✅ Valid Attestation' : '⚠️
Needs Update'}
            </span>
            {attestationStatus.lsaVerified && (
              <span className="bg-blue-100 text-blue-800 px-2 py-1
rounded text-sm">
                🏛️ LSA Verified
              </span>
            )}
            </div>
          )}

          <h2 className="text-2xl font-bold mb-6">Professional
Attestation</h2>

          { /* 🎉 Success State */}
          {submitSuccess ? (
            <div className="text-center bg-green-50 p-6 rounded-lg">
              <div className="text-6xl mb-4">🎉</div>
              <h3 className="text-xl font-bold text-green-800 mb-2">
                Attestation Submitted!
              </h3>
              <p className="text-green-600">
                Your professional credentials have been recorded.
              </p>
              <button
                onClick={() => setSubmitSuccess(false)}
                className="mt-4 bg-green-600 text-white px-4 py-2 rounded
hover:bg-green-700"
              >
                Back to Form
              </button>
            </div>
          ) : (
            { /* 📝 Form */}
            <form onSubmit={handleSubmit} className="space-y-6">

              { /* Basic Info */}
              <div className="space-y-4">
                <h3 className="text-lg font-semibold border-b pb-2">

```

### Basic Information

### 

**Legal Name** \*

**LSA Bar Number** \*

4-10

characters

*/\* Attestation Checkboxes \*/*

### Professional Confirmations

Please confirm all statements below:

{ key: 'isLicensed', label: '🏛️ I am currently licensed to practice law in Alberta' },

{ key: 'isInGoodStanding', label: '⭐ I am in good

```

standing with the Law Society of Alberta' },
      { key: 'noDisciplinaryActions', label: '🛡️ I have no
pending or active disciplinary actions' },
      { key: 'profileAccurate', label: '📄 My profile
information is accurate and current' },
      { key: 'willUpdateOnChange', label: '🔄 I will update my
info if my status changes' },
      { key: 'understandsLiability', label: '⚖️ I understand my
professional responsibilities on this platform' }
    ].map(({ key, label }) => (
      <label key={key} className="flex items-start gap-3 p-3
bg-gray-50 rounded cursor-pointer hover:bg-gray-100">
        <input
          type="checkbox"
          checked={formData[key]}
          onChange={(e) => setFormData(prev => ({ ...prev,
[key]: e.target.checked })))}
          className="mt-1 h-4 w-4 text-blue-600 rounded"
          required
        />
        <span className="text-sm">{label}</span>
      </label>
    ))}
  </div>

  {/* Disclaimer */}
  <div className="bg-yellow-50 border border-yellow-200
rounded-lg p-4">
    <div className="flex items-start gap-3">
      <span className="text-2xl">⚠️</span>
      <div>
        <h4 className="font-medium text-yellow-800 mb-
1">Important Notice</h4>
        <p className="text-sm text-yellow-700">
          <strong>Lawyer status is self-reported.</strong> Our
platform performs automated
            checks against the Law Society of Alberta directory,
but users should independently
              verify credentials when required.
            </p>
          </div>
        </div>
      </div>
    </div>

    {/* Submit Button */}
    {submitError && (
      <div className="bg-red-50 border border-red-200 rounded p-
3">
        <span className="text-red-700 text-sm">❌ {submitError}</span>
      </div>
    )}

    <button

```

```

        type="submit"
        disabled={!isFormValid() || isSubmitting}
        className={`w-full py-3 px-4 rounded-lg font-medium
transition ${
        isFormValid() && !isSubmitting
        ? 'bg-blue-600 text-white hover:bg-blue-700'
        : 'bg-gray-300 text-gray-500 cursor-not-allowed'
        }}
      >
        {isSubmitting ? (
          <span className="flex items-center justify-center gap-2">
            <span className="animate-spin">⌚ </span>
            Submitting...
          </span>
        ) : currentAttestation ? '🔄 Update Attestation' : '✅
Submit Attestation'}
      </button>
    </form>
  )}
</div>
</Authenticated>
</>
);
}

```

## Error Handling Made Simple

### Common Errors & What They Mean

```

const handleSubmit = async (e) => {
  e.preventDefault();
  setIsSubmitting(true);

  try {
    await submitAttestation(formData);
    setSubmitSuccess(true);
  } catch (error) {
    // 🔍 Figure out what went wrong
    if (error.message.includes("Not authenticated")) {
      setSubmitError("Please sign in to continue");
      // Maybe redirect to login?
    } else if (error.message.includes("lawyer")) {
      setSubmitError("Only lawyers can submit attestations");
    } else if (error.message.includes("validation")) {
      setSubmitError("Please check all required fields");
    } else {
      setSubmitError("Something went wrong. Please try again.");
    }
  } finally {
    setIsSubmitting(false);
  }
}

```



```

    }
  };

```

## Validation Helpers

```

// ✅ Check if bar number looks right
const validateBarNumber = (barNumber) => {
  const cleaned = barNumber.trim().toUpperCase();
  if (!cleaned) return "Bar number is required";
  if (!/^[A-Z0-9]{4,10}$/i.test(cleaned)) {
    return "Bar number must be 4-10 alphanumeric characters";
  }
  return null; // Valid!
};

// ✅ Check if all checkboxes are checked
const validateAttestations = (formData) => {
  const required = [
    'isLicensed', 'isInGoodStanding', 'noDisciplinaryActions',
    'profileAccurate', 'willUpdateOnChange', 'understandsLiability'
  ];

  for (const field of required) {
    if (!formData[field]) {
      return `Please confirm: ${field}`;
    }
  }
  return null; // All good!
};

```

## 🎨 UI States & Loading

### Loading State

```

// Show different states based on what's happening
const LoadingSpinner = () => (
  <div className="flex items-center gap-2">
    <span className="animate-spin">⌚ </span>
    Submitting your attestation...
  </div>
);

```

### Status Indicators

```

// Show lawyer's current status
const StatusBadge = ({ attestationStatus }) => {

```

```

if (!attestationStatus?.hasAttestation) {
  return <span className="text-gray-500">🚧 No attestation yet</span>;
}

if (attestationStatus.isValid) {
  return (
    <div className="flex items-center gap-2">
      <span className="text-green-600">✅ Valid Attestation</span>
      {attestationStatus.lsaVerified && (
        <span className="bg-blue-100 text-blue-800 px-2 py-1 rounded
text-sm">
          🏛️ LSA Verified
        </span>
      )}
    </div>
  );
}

return <span className="text-yellow-600">⚠️ Needs Update</span>;
};

```

## 🔧 Testing Your Component

### Manual Testing Checklist

```

// ✅ Things to check manually:
const testingChecklist = [
  "Form loads without errors",
  "All input fields work",
  "Validation prevents invalid submissions",
  "Success state shows after submission",
  "Error messages display clearly",
  "Loading states work",
  "Pre-fills existing attestation data",
  "Works on mobile devices",
  "Authentication flow works",
  "Status badges show correctly"
];

```

### Testing Different Scenarios

```

// 🤖 Fake different user states for testing
const TestScenarios = () => {
  const [testMode, setTestMode] = useState('new-lawyer');

  const mockAttestationStatus = {
    'new-lawyer': null,
    'valid-lawyer': { isValid: true, hasAttestation: true, lsaVerified:
true },

```

```

    'needs-update': { isValid: false, hasAttestation: true, lsaVerified:
false }
  };

  return (
    <div>
      <select onChange={(e) => setTestMode(e.target.value)}>
        <option value="new-lawyer">🆕 New Lawyer</option>
        <option value="valid-lawyer">✅ Valid Lawyer</option>
        <option value="needs-update">⚠ Needs Update</option>
      </select>

      <AttestationForm
        testStatus={mockAttestationStatus[testMode]}
      />
    </div>
  );
};

```

## 🚀 Integration Tips

### Using in Different Pages

```

// 📄 Dedicated attestation page
function AttestationPage() {
  return (
    <div className="min-h-screen bg-gray-50 py-8">
      <div className="max-w-4xl mx-auto px-4">
        <h1 className="text-3xl font-bold text-center mb-8">
          Professional Attestation
        </h1>
        <AttestationForm />
      </div>
    </div>
  );
}

// 🏠 Part of lawyer dashboard
function LawyerDashboard() {
  const attestationStatus =
    useQuery(api.attestations.checkMyAttestationStatus);

  return (
    <div>
      {!attestationStatus?.isValid && (
        <div className="bg-yellow-50 border border-yellow-200 rounded p-4
mb-6">
          <h3 className="font-medium mb-2">⚠ Attestation Required</h3>
          <p className="text-sm mb-3">Complete your professional
attestation to access all features.</p>
          <Link href="/attestation" className="text-blue-600 underline">

```

```

        Complete Attestation →
      </Link>
    </div>
  )}

  {/* Rest of dashboard */}
</div>
);
}

```

## Responsive Design






```

// 📱 Make it work on all screen sizes
const ResponsiveAttestationForm = () => (
  <div className="
    max-w-2xl mx-auto
    p-4 md:p-6
    bg-white
    rounded-lg shadow-md
    m-4 md:m-0
  ">
    <h2 className="text-xl md:text-2xl font-bold mb-4 md:mb-6">
      Professional Attestation
    </h2>
    {/* Form content */}
  </div>
);






```

## Security Notes

### What's Handled Automatically

-  **Authentication:** Only logged-in lawyers can access
-  **Authorization:** Users can only see/edit their own data
-  **Validation:** Backend validates all data again
-  **Audit Trail:** Every submission is logged
-  **LSA Verification:** Automatic checks against official directory

### What You Need to Remember

-  Always wrap in `<Authenticated>` component
-  Never show sensitive data to wrong users
-  Validate on frontend AND backend
-  Show clear error messages
-  Handle network failures gracefully

## Pro Tips for Frontend Developers

1. 🧑🔧 **Use Query Skipping:** `useQuery(..., user?.userType === "lawyer" ? {} : "skip")`
  2. ⚡ **Optimize Renders:** Memoize expensive operations
  3. 🎨 **Loading States:** Always show what's happening
  4. 📱 **Mobile First:** Test on small screens early
  5. 🐛 **Error Boundaries:** Wrap components to catch crashes
  6. ♿ **Accessibility:** Add proper labels and ARIA attributes
  7. 🪄 **Test Edge Cases:** What if API is down? Network is slow?
- 

**Need Help?** Check the diagrams or ask the backend team! Remember: this is like a digital ID system - keep it simple, secure, and user-friendly! 🎉