ADVYON LEGAL PLATFORM

AI-Powered Legal Case Management System

Final Project Report

Course: Software Engineering Laboratory

Project Title: Advyon - AI-Powered Legal Case Management Platform

Team: ParaDox_C101

Name	Student ID
MD Shahriar Islam	112230187
MD. Abu Bakar	112230200
Ikramul Hasan Moral	112230489
Maknun Ahmed Sifat	112230192
Samiur Rahman Omlan	112230195

GitHub Repository

Client: https://github.com/ihmorol/advyon-client

Server: https://github.com/ihmorol/advyon-server

Submission Date: February 2026

Table of Contents

1. Executive Summary

Advyon Legal Platform is a comprehensive, AI-powered legal case management system designed to modernize how legal professionals manage their practice. The platform integrates artificial intelligence capabilities for document analysis, provides real-time communication features, and offers an intuitive interface for managing legal cases, clients, documents, and schedules.

Built using modern web technologies including React 19, Node.js, Express, and MongoDB, the platform addresses the growing need for digital transformation in the legal industry. Key innovations include AI-powered document categorization and summarization using Gemini 2.0 Flash through OpenRouter, a collaborative community hub for legal professionals, and a comprehensive legal database featuring Bangladeshi laws, acts, and sections.

The platform supports multiple user roles including Lawyers, Clients, Judges, and Administrators, each with tailored interfaces and capabilities designed to enhance productivity and streamline legal workflows. The system successfully demonstrates the integration of modern AI services with traditional case management workflows, providing a foundation for continued development and enhancement.

---

2. Project Overview

2.1 Problem Statement

Legal practitioners face numerous challenges in their daily operations that significantly impact their efficiency and effectiveness:

Document Management Complexity: Legal cases generate extensive documentation including contracts, court filings, evidence, and correspondence. Managing, organizing, and retrieving these documents efficiently is a significant challenge. Traditional filing systems lack intelligent categorization and search capabilities.

Time-Critical Deadline Management: Missing court dates, filing deadlines, or statute of limitations can result in case dismissals, malpractice claims, or adverse judgments. Legal professionals need robust systems for tracking and alerting them to upcoming deadlines across multiple cases.

Client Communication Gaps: Clients often feel disconnected from their legal proceedings due to lack of regular updates. Traditional communication methods create delays and can result in miscommunication.

Research Inefficiency: Legal research requires accessing vast databases of laws, precedents, and regulations. This process is time-consuming and requires specialized skills to navigate effectively.

Collaboration Barriers: Legal teams often work in silos, with limited mechanisms for sharing insights, strategies, or resources across the professional community.

2.2 Objectives

The primary objectives of this project are:

1. To design and implement a web-based legal case management system that streamlines legal practice operations
2. To integrate AI-powered document analysis for automatic categorization, summarization, and entity extraction
3. To provide real-time communication capabilities between lawyers and clients
4. To create a collaborative community platform for legal knowledge sharing
5. To offer a comprehensive searchable database of Bangladeshi laws and legal precedents
6. To implement role-based access control ensuring data security and privacy

2.3 Scope

Included in Scope:

- User authentication and role-based access control
- Case creation, management, and tracking
- AI-powered document analysis and categorization
- Schedule and calendar management
- Client relationship management
- Community discussion forum
- Legal database with search functionality
- Real-time messaging between users

- Analytics and reporting dashboard

Excluded from Scope:

- Payment processing and billing
- Mobile native applications
- Video conferencing integration
- Court filing system integration
- Multi-language interface support

2.4 Target Users

The platform is designed for the following user categories:

User Type	Description	Key Features
Lawyers	Legal professionals managing cases	Full case management, AI analysis, client management
Clients	Individuals seeking legal services	Case status viewing, messaging with lawyers
Judges	Legal authorities reviewing cases	Case review, document access
Administrators	System managers	User management, system configuration

---

3. Technology Stack

3.1 Frontend Technologies

Technology	Version	Purpose
React	19.0.0	User interface library for building component-based UI

Vite	6.0.6	Fast build tool and development server
Tailwind CSS	4.0.3	Utility-first CSS framework for styling
React Router DOM	7.1.1	Client-side routing and navigation
Zustand	5.0.3	Lightweight state management solution
TanStack Query	5.64.1	Server state management and data caching
Clerk React	5.22.3	Authentication UI components
Framer Motion	11.18.0	Animation library for smooth transitions
Socket.io Client	4.8.1	Real-time bidirectional communication
Recharts	2.15.0	Data visualization and charting
React Markdown	9.0.3	Markdown rendering for content display

3.2 Backend Technologies

Technology	Version	Purpose
Node.js	18+	JavaScript runtime environment
Express	4.21.2	Web application framework
TypeScript	5.2.2	Type-safe JavaScript for better maintainability
MongoDB	Latest	NoSQL database for flexible data storage
Mongoose	8.9.5	MongoDB object modeling with schemas
Socket.io	4.8.1	WebSocket implementation for real-time features
Clerk SDK	5.1.5	Authentication service backend integration
Zod	3.24.1	Schema validation for type safety
Cloudinary	2.5.1	Cloud-based file storage and management

3.3 AI and Machine Learning Services

Service	Model	Purpose
OpenRouter	Gemini 2.0 Flash	Document analysis, summarization, and categorization
Groq	LLaMA 3.3 70B	Conversational AI assistant for user queries
Tesseract.js	5.1.1	OCR for extracting text from image-based documents

3.4 Development Tools

Tool	Purpose
ESLint	Code linting and quality assurance
Jest	Unit and integration testing framework
ts-node-dev	TypeScript development with hot reloading
Git	Version control system
npm	Package management

3.5 Technology Justification

The technology choices were driven by the following considerations:

React 19: Chosen for its component-based architecture, extensive ecosystem, and excellent performance with the new concurrent rendering features.

Node.js/Express: Selected for JavaScript consistency across the stack, excellent async handling for I/O operations, and a vast package ecosystem.

MongoDB: Chosen for its flexible schema design, which is ideal for legal documents with varying structures, and excellent scalability characteristics.

TypeScript: Implemented to improve code maintainability, catch errors at compile time, and enhance developer productivity with better IDE support.

Clerk: Selected for secure, production-ready authentication without the complexity of building custom auth systems.

OpenRouter/Groq: Chosen for access to state-of-the-art AI models through simple API integrations, enabling powerful document analysis features.

---

4. System Architecture

4.1 High-Level Architecture

The Advyon Legal Platform follows a modern three-tier architecture pattern:

Presentation Layer (Frontend)

- React-based single-page application
- Component-based UI architecture
- Client-side routing and state management
- Real-time updates via WebSocket

Application Layer (Backend)

- RESTful API server built with Express
- Business logic implementation in service layer
- WebSocket server for real-time features
- Integration with external AI services

Data Layer

- MongoDB database for persistent storage
- Cloudinary for file storage
- Redis for caching (future implementation)

4.2 Directory Structure

Frontend Structure (advyon-client):

```
src/
├── components/     # Reusable UI components
├── pages/          # Page-level components
├── stores/         # Zustand state stores
├── hooks/          # Custom React hooks
├── services/       # API communication layer
├── utils/          # Helper functions
└── lib/            # Third-party integrations
```

Backend Structure (advyon-server):

```
src/
├── app/
│   ├── modules/    # Domain-specific modules
│   │   ├── user/
│   │   ├── case/
│   │   ├── document/
│   │   ├── ai/
│   │   └── ...
│   ├── routes/     # API route definitions
│   └── middleware/ # Request processing middleware
├── config/         # Configuration files
└── utils/          # Shared utilities
```

4.3 Data Flow

1. User interacts with React frontend components
2. Frontend makes API requests to Express backend
3. Authentication tokens validated by Clerk middleware
4. Controllers process requests and call service layer
5. Services implement business logic and interact with database

6. AI services called for document analysis when needed
7. Responses formatted and returned to frontend
8. Real-time updates pushed via WebSocket connections

---

5. Features Breakdown

5.1 Core Features

Feature 1: User Authentication and Management

Description: Secure user authentication with role-based access control supporting multiple user types.

Implementation: Integrated Clerk authentication service providing OAuth, email/password login, and session management. Backend validates JWT tokens for protected routes.

Key Capabilities:

- Email and password authentication
- Google OAuth integration
- Role-based permissions (Super Admin, Admin, Lawyer, Client, Judge)
- Profile management with customizable preferences

Feature 2: Dashboard and Analytics

Description: Central hub providing overview of legal practice activities and AI-generated insights.

Implementation: React dashboard component with data fetching via TanStack Query, displaying statistics, recent activities, and actionable insights.

Key Capabilities:

- Active cases overview with status indicators
- Upcoming hearings and deadlines display
- AI-generated insights and recommendations

- Recent activity feed
- Quick action buttons for common tasks

Feature 3: Case Management System

Description: Comprehensive tools for creating, organizing, and tracking legal cases through their lifecycle.

Implementation: Modular case management with folder-based organization, progress tracking, and status workflows.

Key Capabilities:

- Case creation with classification and urgency levels
- Dedicated workspace for each case
- Hierarchical folder organization (Evidence, Correspondence, Court Filings, Discovery)
- Progress tracking with visual indicators
- Status management (Open, In Progress, Pending Review, Closed, Archived)

Feature 4: AI-Powered Document Analysis

Description: Automatic document processing using AI for categorization, summarization, and entity extraction.

Implementation: OpenRouter integration with Gemini 2.0 Flash for document analysis, with fallback error handling and retry logic.

Key Capabilities:

- Automatic document categorization (Legal Pleading, Contract, Evidence, etc.)
- AI-generated summaries highlighting key points
- Entity extraction (names, dates, amounts, legal references)
- Legal reference detection for cited laws and precedents

Feature 5: Schedule and Calendar Management

Description: Time management tools for tracking hearings, meetings, and important deadlines.

Implementation: Calendar component with event creation, filtering, and case association capabilities.

Key Capabilities:

- Event creation with type categorization
- Multiple view options (All Events, Today, Upcoming, Past)
- Physical location and virtual meeting link support
- Case-linked event organization

Feature 6: Community Hub

Description: Collaborative platform for legal professionals to share knowledge and discuss issues.

Implementation: Discussion forum with categories, Q&A format, and voting system for quality curation.

Key Capabilities:

- Topic-based discussion forums by practice area
- Structured Q&A format
- Top contributor recognition
- Trending topics display
- AI-generated thread summaries

Feature 7: Legal Database

Description: Searchable database of Bangladeshi laws, acts, and legal sections.

Implementation: MongoDB collection with text indexes for full-text search, paginated results, and cross-references.

Key Capabilities:

- Full-text search across legal provisions
- Filtering by Act type and year
- Detailed section views with related content
- Save and cite functionality

5.2 Additional Features

- **AI Assistant:** Contextual chat assistant for case-related queries
- **Client Management:** Tools for managing client relationships and information
- **Real-time Messaging:** Direct communication between lawyers and clients via Socket.io
- **Gamification:** Points and rewards system for platform engagement

---

6. Database Design

6.1 Entity Relationship Diagram

The database consists of the following primary entities and their relationships:

User → creates many → **Cases Case** → contains many → **Documents User** → sends many → **Messages User** → creates many → **Community Posts Legal Acts** → contains many → **Sections**

6.2 Database Schema

Collection: Users

Field	Type	Constraints	Description
_id	ObjectId	Primary Key	Unique identifier
clerkId	String	Unique, Required	Clerk authentication ID
email	String	Unique, Required	User email address
name	Object	Required	First and last name
role	String	Enum	User role (admin, lawyer, client, judge)
preferences	Object	-	User preferences and settings
points	Number	Default: 0	Gamification points
isDeleted	Boolean	Default: false	Soft delete flag

Collection: Cases

Field	Type	Constraints	Description
_id	ObjectId	Primary Key	Unique identifier
caseNumber	String	Unique, Required	Auto-generated case number
title	String	Required	Case title
description	String	-	Case description
status	String	Enum	Case status
urgency	String	Enum	Priority level (low, medium, high)
practiceArea	String	Enum	Legal practice category
createdBy	ObjectId	Ref: User	Case creator
folders	Array	-	Document organization folders
progress	Number	0-100	Case completion percentage
isDeleted	Boolean	Default: false	Soft delete flag

Collection: Documents

Field	Type	Constraints	Description
_id	ObjectId	Primary Key	Unique identifier
caseId	ObjectId	Ref: Case	Associated case
fileName	String	Required	Original file name
fileUrl	String	Required	Cloudinary storage URL
fileType	String	Required	MIME type
aiAnalysis	Object	-	AI analysis results
category	String	Enum	Document category

uploadedBy	ObjectId	Ref: User	Uploader reference
isDeleted	Boolean	Default: false	Soft delete flag

6.3 Relationships

- **One-to-Many:** User to Cases (a user can create multiple cases)
- **One-to-Many:** Case to Documents (a case contains multiple documents)
- **One-to-Many:** User to Messages (a user can send multiple messages)
- **Many-to-Many:** Cases to Users (through role associations - lawyers, clients assigned to cases)
- **Embedded:** Folder structures within Cases, AI analysis results within Documents

---

7. API Documentation

7.1 API Overview

Property	Value
Base URL	http://localhost:5000/api/v1
Authentication	Clerk JWT Tokens
Content Type	application/json
Response Format	JSON

7.2 Endpoints

Authentication Endpoints

Method	Endpoint	Description	Auth Required
POST	/auth/clerk-webhook	Handle Clerk authentication events	No

GET	/auth/verify	Verify authentication token	Yes

User Endpoints

Method	Endpoint	Description	Auth Required
GET	/users/me	Get current user profile	Yes
PATCH	/users/me	Update user profile	Yes
GET	/users	Get all users (admin)	Yes
DELETE	/users/:id	Delete user account	Yes

Case Endpoints

Method	Endpoint	Description	Auth Required
POST	/cases	Create new case	Yes
GET	/cases	Get user's cases	Yes
GET	/cases/:id	Get case details	Yes
PATCH	/cases/:id	Update case	Yes
DELETE	/cases/:id	Delete case	Yes

Document Endpoints

Method	Endpoint	Description	Auth Required
POST	/documents/upload	Upload document	Yes
GET	/documents	Get user's documents	Yes
GET	/documents/:id	Get document details	Yes
DELETE	/documents/:id	Delete document	Yes

AI Endpoints

Method	Endpoint	Description	Auth Required
POST	/ai/analyze	Analyze document with AI	Yes
POST	/ai/chat	Chat with AI assistant	Yes

Community Endpoints

Method	Endpoint	Description	Auth Required
GET	/community	Get discussions	Yes
POST	/community	Create discussion	Yes
POST	/community/:id/answer	Add answer	Yes
POST	/community/:id/vote	Vote on discussion	Yes

Legal Database Endpoints

Method	Endpoint	Description	Auth Required
GET	/legal-acts	Search legal acts	Yes
GET	/legal-acts/:id	Get act details	Yes
GET	/sections	Search sections	Yes
GET	/sections/:id	Get section details	Yes

---

8. Security Considerations

8.1 Authentication Security

- **Third-Party Authentication:** Clerk handles all authentication flows, eliminating storage of sensitive credentials
- **JWT Token Validation:** All API requests validated against Clerk's JWT tokens with proper signature verification

- **Session Management:** Secure session handling with automatic token refresh and expiration
- **OAuth Integration:** Support for Google OAuth with proper scope limitations

8.2 Authorization

- **Role-Based Access Control (RBAC):** Five distinct user roles with granular permissions
- **Route-Level Protection:** Both frontend and backend implement route guards
- **Resource Authorization:** Users can only access resources they own or are assigned to

8.3 Data Protection

- **Input Validation:** All user inputs validated using Zod schema validation before processing
- **Data Sanitization:** Prevention of injection attacks through proper data handling and escaping
- **Encrypted Transmission:** All data transmitted over HTTPS in production
- **Database Security:** MongoDB authentication enabled with access control lists

8.4 Security Best Practices Implemented

Security Measure	Implementation
Rate Limiting	Express rate-limit middleware preventing abuse
CORS Configuration	Explicit origin whitelisting for cross-origin requests
Security Headers	Helmet.js implementation for HTTP security headers
Error Handling	Secure error messages without system detail exposure
File Upload Security	File type validation and size limitations
Soft Deletes	Data preserved for audit trails

---

9. User Interface

The following screenshots demonstrate the key interfaces of the Advyon Legal Platform:

9.1 Dashboard Overview

The main dashboard provides a comprehensive view of legal practice activities.

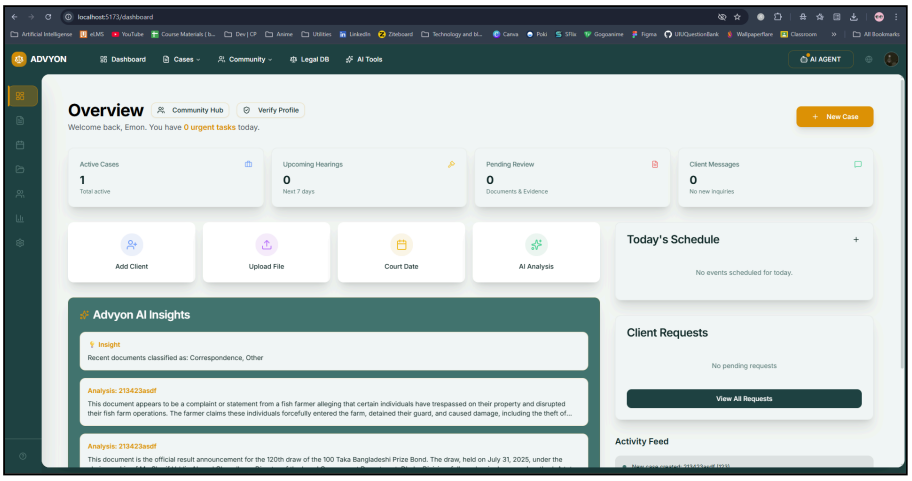

Figure 9.1: Dashboard showing active cases, upcoming hearings, AI insights, and schedule

9.2 Case Workspace

The workspace displays active cases with status and activity tracking.

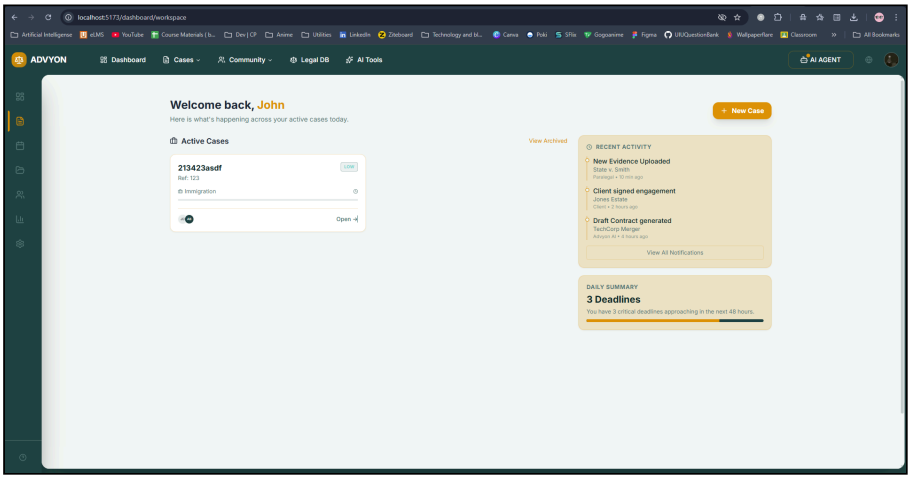

Figure 9.2: Workspace view with active cases and deadline summary

9.3 New Case Creation

The case creation interface with AI-powered workspace preview.

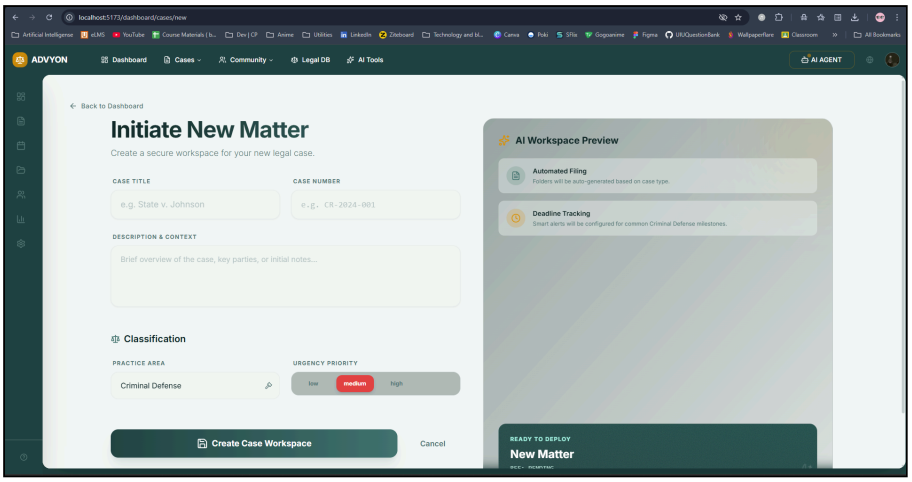

Figure 9.3: New case form with classification and urgency options

9.4 Document Management

Centralized document hub with AI analysis status.

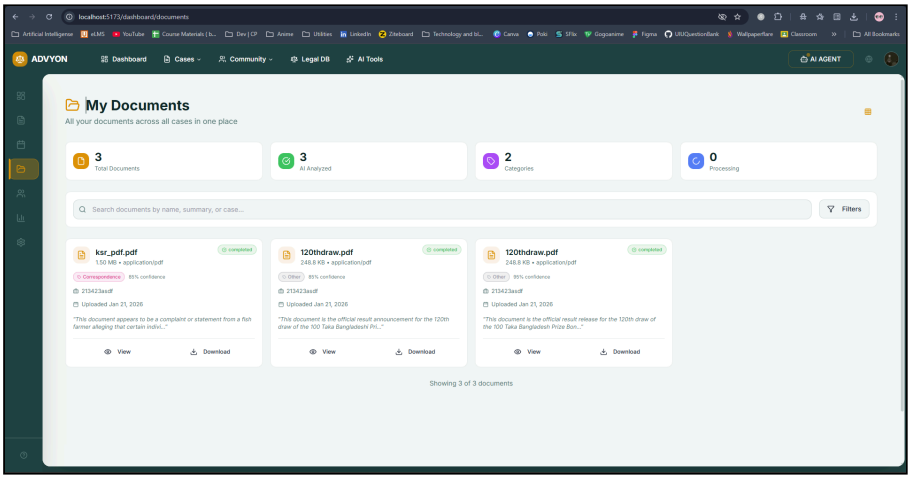

Figure 9.4: Document management with AI analysis indicators

9.5 Archived Cases

Archive section for completed cases.

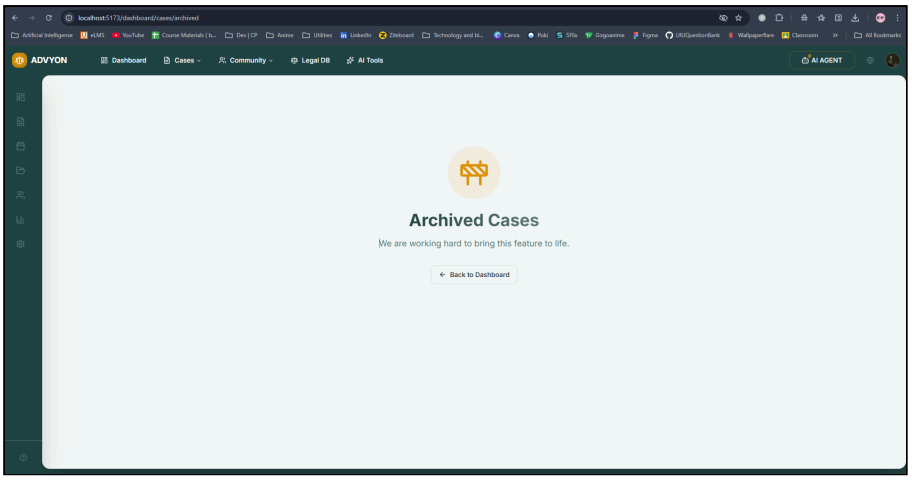

Figure 9.5: Archived cases section

9.6 Case Workspace with AI

Individual case workspace with integrated AI assistant.

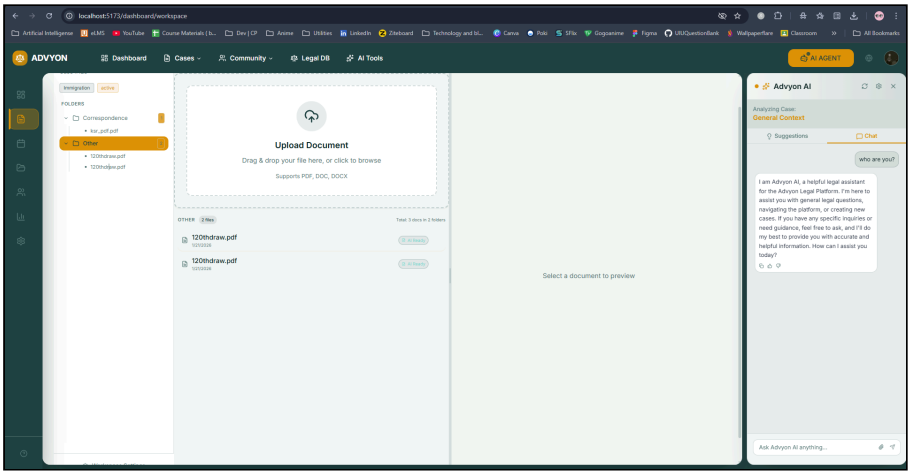

Figure 9.6: Case workspace with folder structure and AI assistant

9.7 Schedule Management

Calendar view for managing hearings and deadlines.

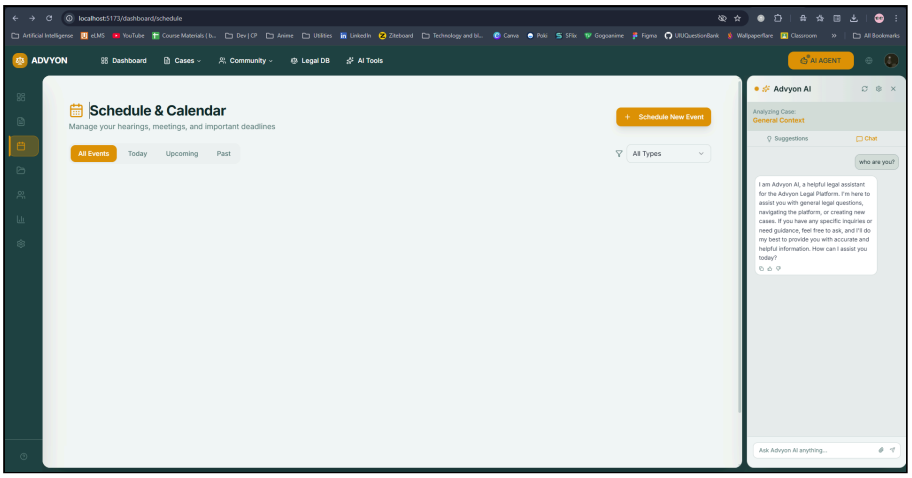

Figure 9.7: Schedule and calendar interface

9.8 Event Creation

Event scheduling form with case linking.

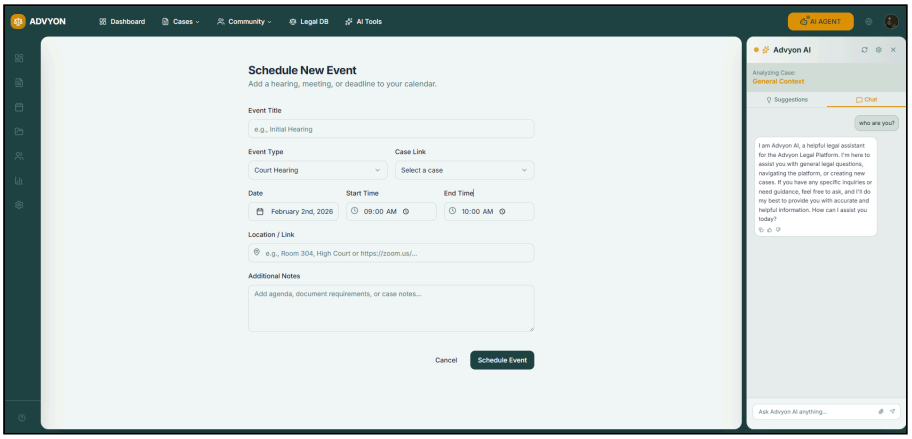

Figure 9.8: New event creation form

9.9 Client Management

Client directory and relationship management.

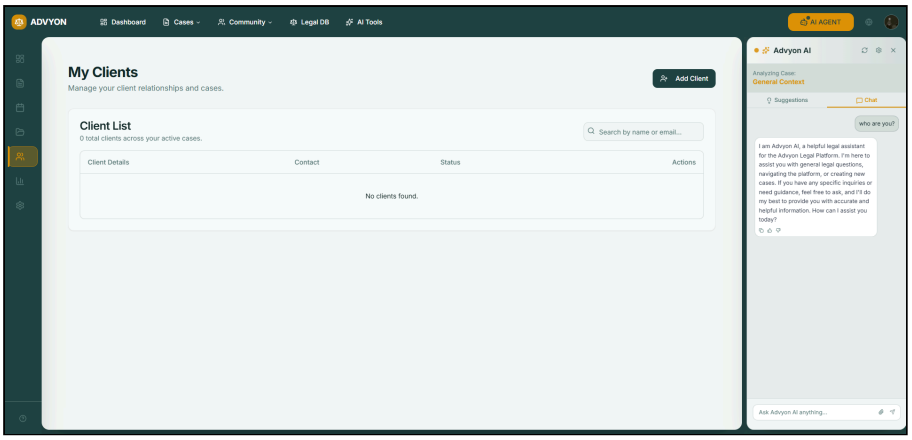

Figure 9.9: Client management interface

9.10 Analytics Dashboard

Performance metrics and case distribution analytics.

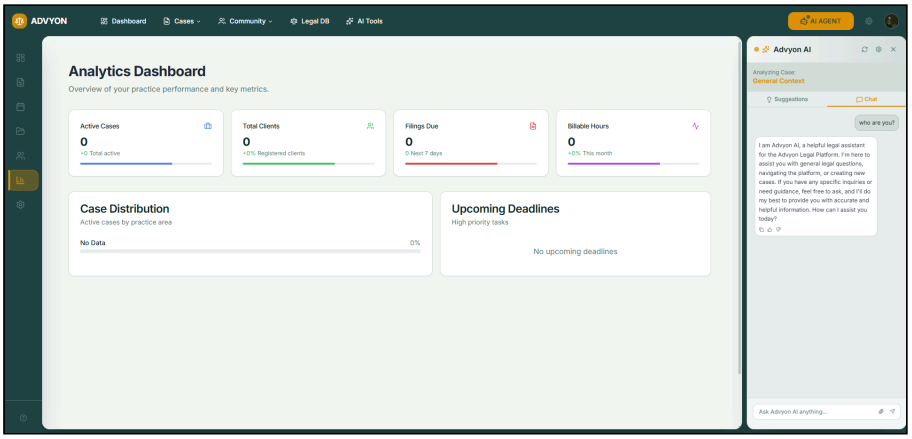

Figure 9.10: Analytics dashboard with metrics

9.11 User Profile

Profile management with preferences and settings.

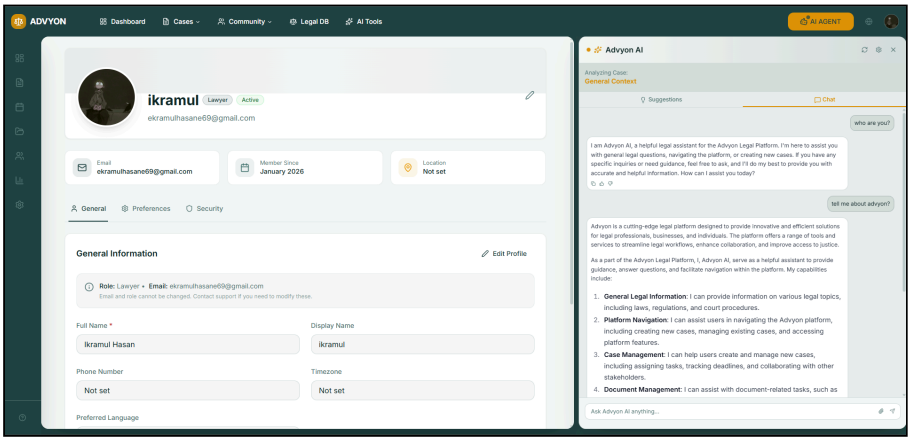

Figure 9.11: User profile settings

9.12 Community Hub

Discussion forum for legal professionals.

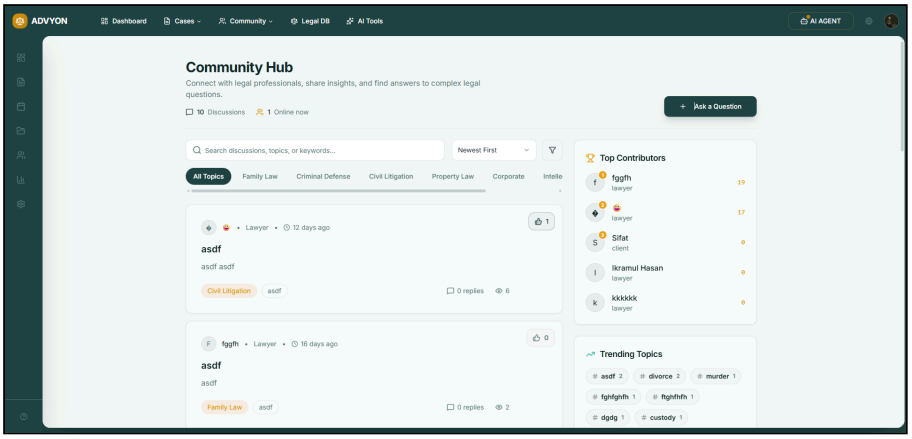

Figure 9.12: Community hub with discussions and trending topics

9.13 Ask a Question

Question submission interface.

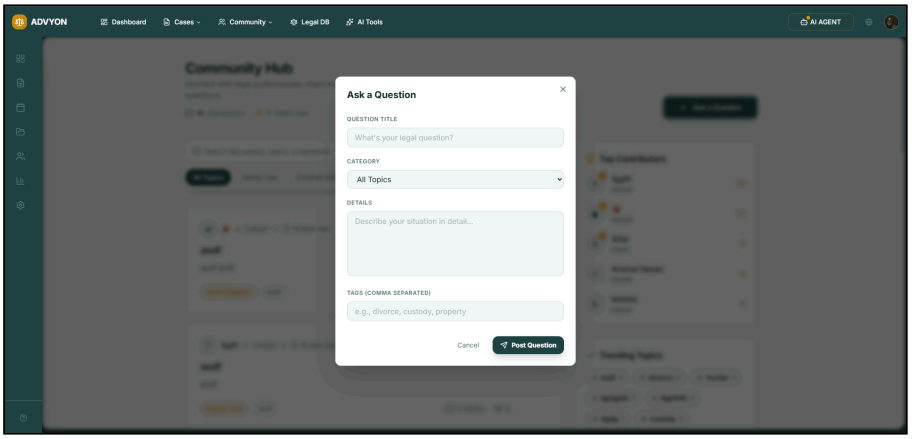

Figure 9.13: Question submission modal

9.14 Discussion Thread

Thread view with AI-generated summary.

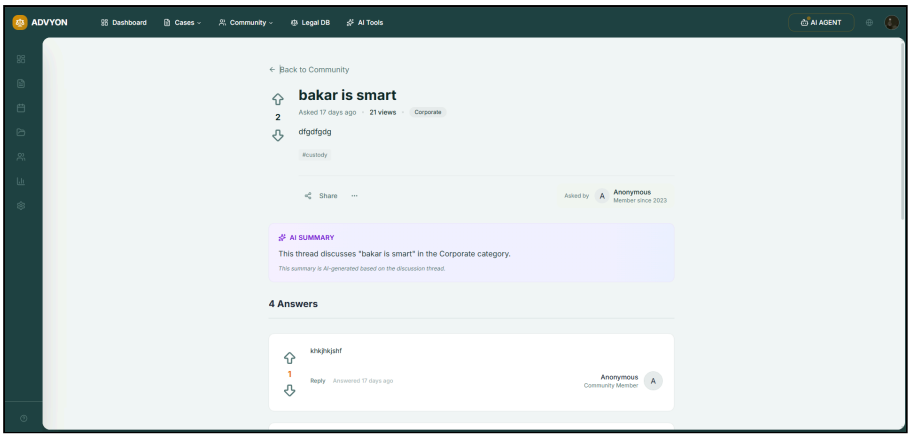

Figure 9.14: Discussion thread with AI summary

9.15 Legal Database Search

Search interface for laws and acts.

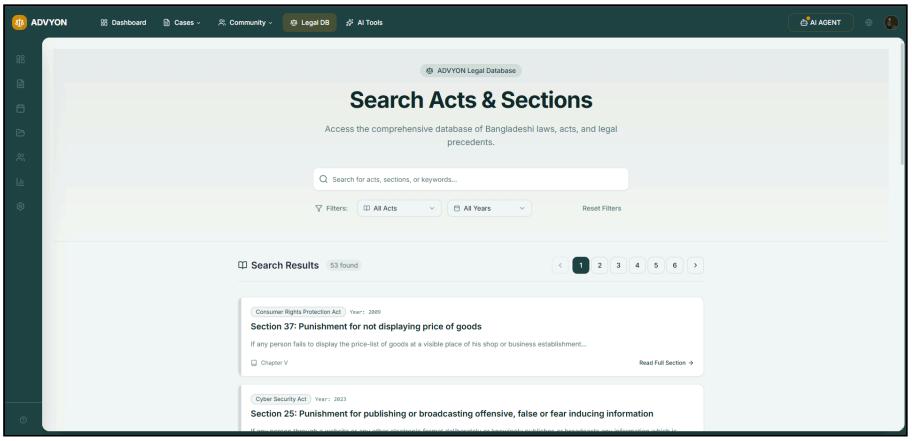

Figure 9.15: Legal database search results

9.16 Section Detail

Detailed view of legal section with references.

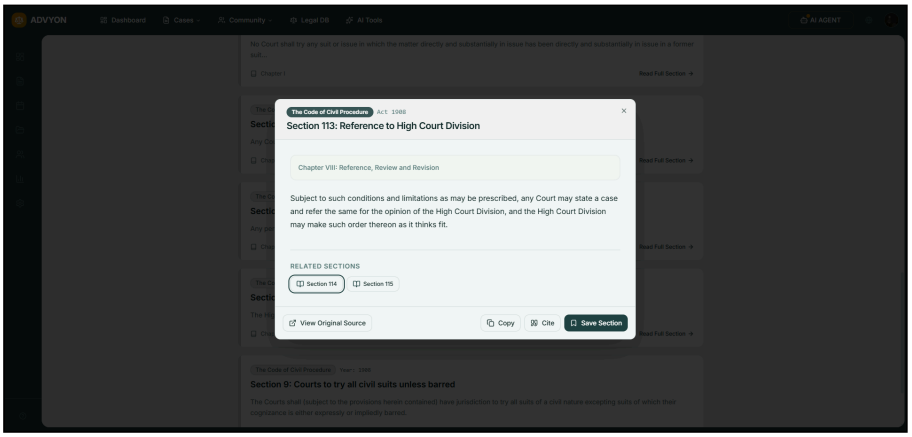

Figure 9.16: Section detail with related sections

9.17 AI Tools

Advanced AI features section.

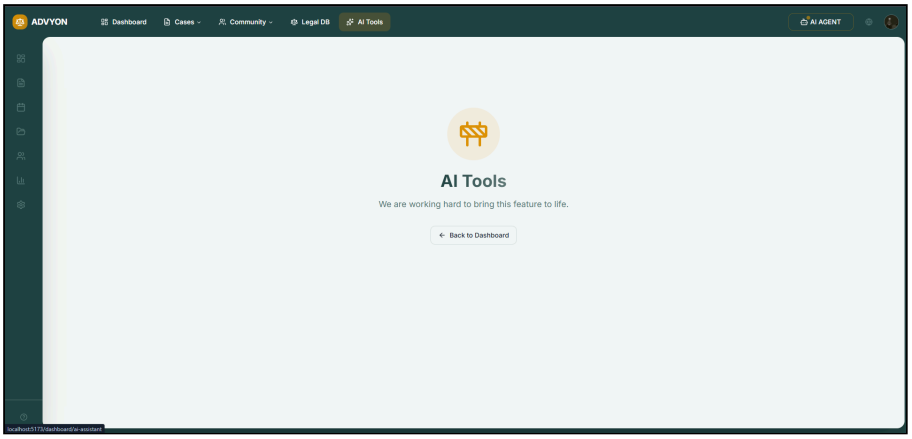

Figure 9.17: AI Tools interface

---

10. Installation & Setup

10.1 Prerequisites

- Node.js version 18 or higher
- MongoDB database (local installation or MongoDB Atlas cloud)
- npm or yarn package manager
- Git for version control
- Clerk account for authentication services
- Cloudinary account for file storage
- OpenRouter API key for AI document analysis
- Groq API key for conversational AI

10.2 Environment Setup

Backend Environment Variables:

Variable	Description
NODE_ENV	Environment (development/production)
PORT	Server port (default: 5000)
DATABASE_URL	MongoDB connection string
CLERK_SECRET_KEY	Clerk authentication secret
CLOUDINARY_CLOUD_NAME	Cloudinary cloud identifier
CLOUDINARY_API_KEY	Cloudinary API key
CLOUDINARY_API_SECRET	Cloudinary API secret
OPENROUTER_API_KEY	OpenRouter API key for AI services
GROQ_API_KEY	Groq API key for chat AI

Frontend Environment Variables:

Variable	Description
VITE_CLERK_PUBLISHABLE_KEY	Clerk frontend key
VITE_API_URL	Backend API base URL
VITE_SOCKET_URL	WebSocket server URL

10.3 Installation Steps

Step 1: Clone the repository from version control system

Step 2: Install backend dependencies by navigating to advyon-server directory and running npm install

Step 3: Install frontend dependencies by navigating to advyon-client directory and running npm install

Step 4: Create environment files (.env) in both directories with required variables

Step 5: Ensure MongoDB is running and accessible

Step 6: Start backend server with npm run dev command

Step 7: Start frontend development server with npm run dev command

Step 8: Access application at http://localhost:5173

10.4 Running the Application

Command	Location	Purpose
npm run dev	advyon-server	Start backend development server
npm run dev	advyon-client	Start frontend development server
npm run build	advyon-client	Build production bundle
npm run lint	Both	Run code linting

---

11. Future Improvements

11.1 Short-term Improvements

- Complete implementation of Archived Cases feature
- Enhance AI Tools module with additional analysis capabilities
- Add email notification system for deadlines and updates
- Implement document version history tracking
- Add bulk document upload functionality

11.2 Long-term Roadmap

- Native mobile applications for iOS and Android platforms
- Video conferencing integration for virtual client meetings
- Integrated billing and invoicing system
- Court filing system integration
- Multi-language interface support

- Advanced reporting with export capabilities

11.3 Scalability Considerations

- Implement Redis caching layer for improved performance
- Migrate to microservices architecture for better scalability
- Add horizontal scaling support with load balancing
- Implement database sharding for large document collections
- Deploy on containerized infrastructure using Docker and Kubernetes

---

12. Challenges & Solutions

Challenge	Solution
Handling large document uploads for AI analysis	Implemented chunked uploads with Cloudinary integration and asynchronous processing with status indicators
Real-time communication across multiple clients	Utilized Socket.io with proper room management and connection handling for scalable real-time features
AI service rate limiting from providers	Implemented request queuing, retry logic with exponential backoff, and fallback responses
Complex role-based access control	Created middleware layer with Clerk integration and modular permission checking per route
Managing state across complex UI components	Adopted Zustand for global state and TanStack Query for server state with proper cache invalidation
Search performance on large legal database	Implemented MongoDB text indexes and optimized queries with pagination
Handling varying document formats	Integrated Tesseract.js for OCR on images and standardized text extraction pipeline

Maintaining type safety across full stack	Strict TypeScript configuration with shared type definitions and Zod validation schemas

---

13. Conclusion

The Advyon Legal Platform successfully demonstrates a modern approach to legal practice management, integrating artificial intelligence capabilities with traditional case management workflows. The project achieved all primary objectives:

Key Achievements:

1. **Comprehensive Case Management:** Successfully implemented a full-featured case management system with folder organization, progress tracking, and status workflows that address real legal practice needs.
2. **AI Integration:** Integrated Gemini 2.0 Flash and LLaMA 3.3 70B models for document analysis and conversational AI, demonstrating practical application of modern AI services in legal technology.
3. **Collaborative Platform:** Created a community hub enabling legal professionals to share knowledge and discuss complex issues, fostering professional collaboration.
4. **Legal Research Tools:** Developed a searchable legal database of Bangladeshi laws, providing valuable research capabilities for legal practitioners.
5. **Secure Architecture:** Implemented robust security measures including Clerk authentication, role-based access control, and data protection best practices.

Learning Outcomes:

- Gained practical experience with modern full-stack development using React 19 and Node.js
- Learned to integrate and orchestrate multiple AI services effectively
- Developed understanding of security best practices in web applications
- Experienced real-world challenges of building production-ready software

The platform provides a solid foundation for future enhancements and demonstrates the potential for AI-assisted legal practice management.

---

14. References

1. React Documentation. React 19 Features and API Reference. https://react.dev/
2. MongoDB Documentation. MongoDB 7.0 Manual. https://www.mongodb.com/docs/
3. Clerk Documentation. Authentication and User Management. https://clerk.com/docs
4. OpenRouter Documentation. AI Model API Reference. https://openrouter.ai/docs
5. Express.js Documentation. Web Application Framework. https://expressjs.com/
6. Socket.io Documentation. Real-time Communication. https://socket.io/docs/
7. Tailwind CSS Documentation. Utility-First CSS Framework. https://tailwindcss.com/docs
8. TypeScript Handbook. TypeScript Language Reference. https://www.typescriptlang.org/docs/
9. Vite Documentation. Next Generation Frontend Tooling. https://vitejs.dev/guide/
10. Cloudinary Documentation. Media Management Platform. https://cloudinary.com/documentation

---

15. Acknowledgements

We would like to express our sincere gratitude to the following individuals and organizations who contributed to the success of this project:

Academic Guidance:

- Our supervisor for continuous guidance, valuable feedback, and encouragement throughout the development process
- Faculty members of the Department for their support and providing necessary resources

Technical Resources:

- Clerk for providing authentication services
- OpenRouter and Groq for AI API access
- Cloudinary for file storage services
- MongoDB Atlas for database hosting
- The open-source community for the libraries and frameworks used in this project

Team Collaboration:

- All team members for their dedication, collaborative spirit, and hard work in completing this project successfully

---

End of Report

This report was prepared as part of the Software Engineering Laboratory coursework, demonstrating the design, development, and implementation of a full-stack web application with AI integration.