AI Dispute Resolver - Comprehensive Project Analysis

Executive Summary

An innovative online dispute resolution (ODR) platform leveraging AI to facilitate fair, efficient resolution of civil disputes outside traditional court systems. The system acts as a mediator, analyzing evidence from multiple parties and suggesting resolutions based on Indian constitutional law.

Project Overview

Core Concept

A multi-party dispute resolution platform where:

- **Party 1** initiates a case with detailed statements and evidence
- **Party 2+** receives notification and submits their counter-statements
- **AI Agent** (Gemini) analyzes all submissions, evidence, and applicable laws
- **System** proposes top 3 resolution options
- **Parties** accept, negotiate, or escalate to formal courts
- **Documents** are generated, signed, and archived for legal record

Key Value Propositions

1. **Cost-Effective**: Free/low-cost alternative to lengthy court proceedings
2. **Time-Efficient**: Resolution in days vs. years
3. **Accessible**: No legal expertise required
4. **Transparent**: All parties see the same information
5. **Legally Grounded**: Based on Indian Constitution and civil laws

---

Detailed Workflow

Phase 1: Case Initiation

Member 1 Actions:

├── Register/Login

├── Click "File New Dispute"

├── Fill Dispute Form

│ ├── Case Category (property, contract, family, etc.)

│ ├── Detailed Statement (unlimited text)

│ ├── Upload Evidence (images, audio, video, PDFs)

│ ├── Severity Level

│ └── Preferred Resolution Timeframe

├── Add Respondent Details

│ ├── Name(s)

│ ├── Email(s)

│ ├── Phone Number(s)

│ └── Address

└── Submit & Pay Filing Fee (if applicable)

System Actions:

- Generate unique Case ID
- Send email/SMS notification to all respondents
- Set deadline for response (e.g., 7 days)
- Create case folder in database

Phase 2: Response Collection

Member 2+ Actions:

├── Receive Notification Email

├── Click Secure Link → Case Portal

├── Review Member 1's Statement

├── Submit Counter-Statement

│ ├── Agree/Disagree with claims

│ ├── Provide their narrative

│ ├── Upload counter-evidence

│ └── Request additional time (if needed)

└── Mark as "Statement Complete"

System Actions:

- Track submission status
- Send reminders for pending responses
- Allow AI to flag missing critical information
- Notify all parties when all statements received

Phase 3: AI Analysis & Clarification

AI Agent (Gemini) Process:

├── Parse all text statements

├── Analyze multimedia evidence

│ ├── Image OCR for text extraction

│ ├── Audio transcription

│ ├── Video frame analysis

│ └── Document text extraction

├── Identify contradictions

├── Flag missing information

├── Generate clarification questions

└── Send to respective parties

Parties:

├── Receive clarification requests

├── Provide additional information

└── Finalize statements

Phase 4: Resolution Proposal

AI Analysis:

├── Apply Indian Legal Framework

│ ├── Relevant IPC sections

│ ├── Civil law precedents

│ ├── Constitutional rights

│ └── Recent judgments

├── Assess evidence credibility

├── Calculate fair compensation/resolution

└── Generate Top 3 Options

Option Format:

├── Option A: Compromise-based

├── Option B: Compensation-based

├── Option C: Action-based

Each with:

├── Legal reasoning

├── Expected outcome

├── Implementation steps

└── Fairness score

Phase 5: Decision Making

Scenario 1: Both Accept Same Option

├── Generate settlement agreement

├── Include all case details

├── Digital signature collection

│ ├── Send to Member 1

│ ├── Send to Member 2+

│ └── Timestamp signatures

├── Create final legal document

├── Send copies to:

│ ├── All parties

│ ├── Registered lawyer (witness)

│ └── Archive in database

└── Mark case as "RESOLVED"

Scenario 2: Different Options Selected

├── AI re-analyzes based on preferences

├── Find middle ground

├── Generate hybrid option

├── Present to both parties

└── Repeat until consensus or max iterations (3)

Scenario 3: No Agreement

├── Generate court-ready case file

│ ├── All statements

│ ├── Evidence compilation

│ ├── AI analysis report

│ ├── Attempted resolutions

│ └── Party positions

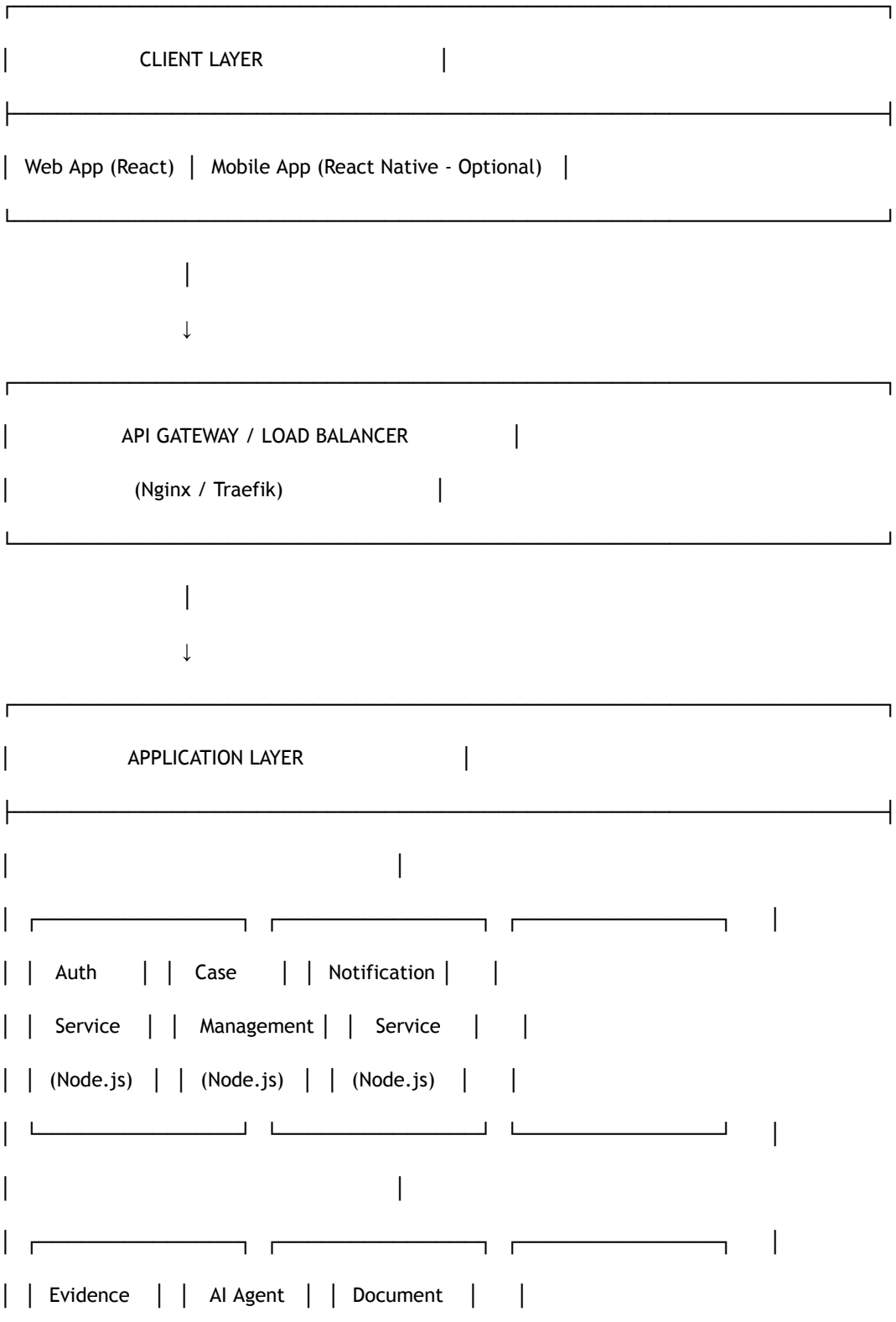
├── Suggest appropriate court jurisdiction

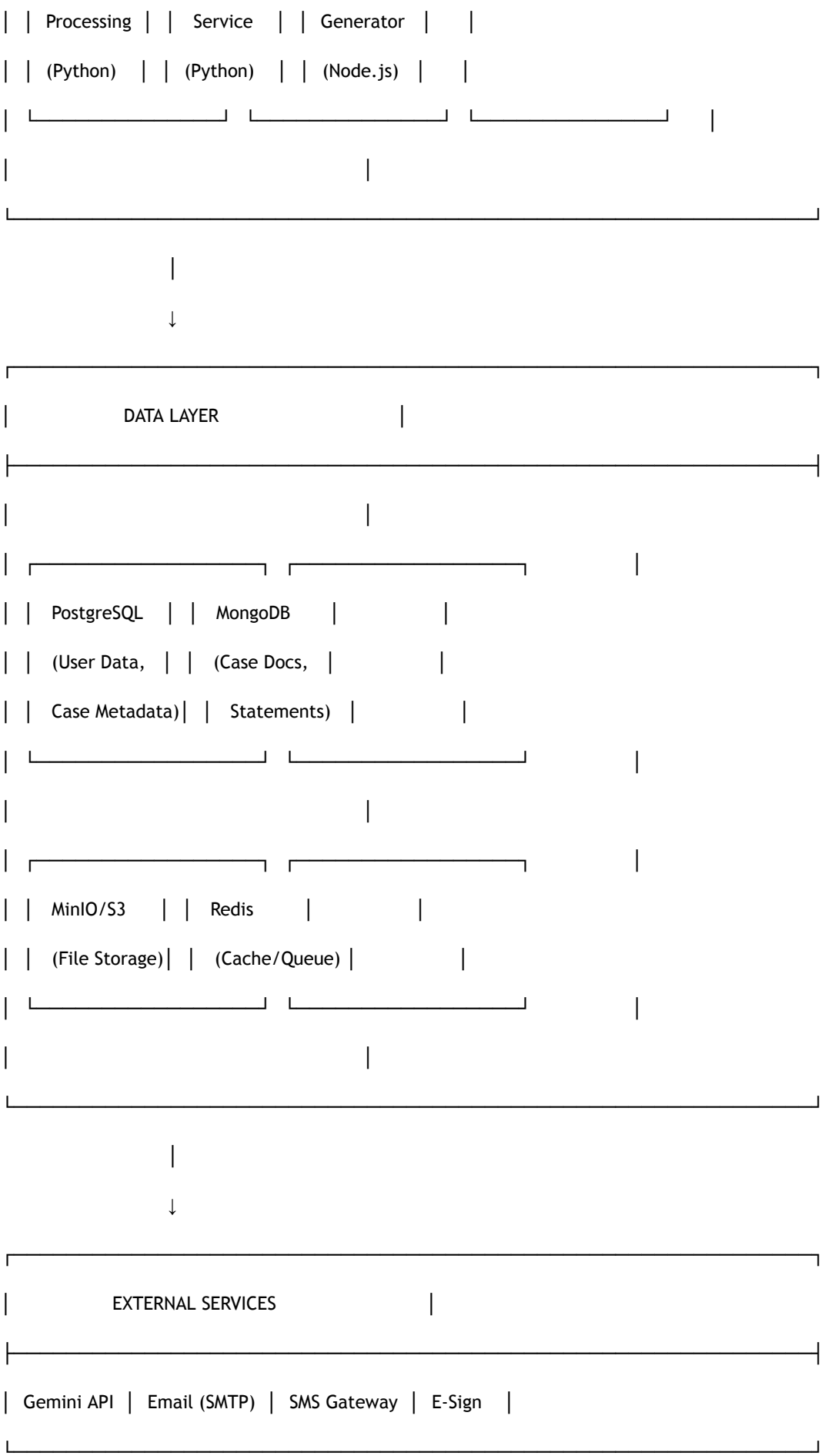
├── Provide filing instructions

└── Mark case as "ESCALATED"

System Architecture

High-Level Architecture Diagram





Component Details

1. Frontend (React + Tailwind CSS)

- **Dashboard**: Case overview, status tracking
- **Case Filing Form**: Multi-step form with file uploads
- **Evidence Viewer**: Image gallery, video player, audio player, PDF viewer
- **Chat Interface**: AI clarification questions
- **Decision Panel**: View and select from AI options
- **Document Viewer**: Final settlement documents
- **E-Signature Module**: Canvas-based signature capture

2. Backend Services

a) Authentication Service (Node.js + Express)

- JWT-based authentication
- Role management (Complainant, Respondent, Admin, Lawyer)
- Session management
- Password reset via email

b) Case Management Service (Node.js + Express)

- CRUD operations for cases
- Status workflow management
- Deadline tracking
- Party management

c) Evidence Processing Service (Python + Flask)

- Image processing (OCR with Tesseract)
- Audio transcription (Whisper or Google Speech-to-Text)
- Video analysis (frame extraction, object detection)
- PDF text extraction (PyPDF2)
- File validation and virus scanning

d) AI Agent Service (Python + FastAPI)

- Integration with Gemini API
- Prompt engineering for legal analysis
- Evidence summarization
- Legal precedent matching
- Option generation with reasoning
- Clarification question generation

e) Notification Service (Node.js)

- Email sending (Nodemailer with Gmail/SendGrid)
- SMS notifications (Twilio free tier)
- In-app notifications
- Reminder scheduling

f) Document Generation Service (Node.js)

- PDF generation (PDFKit or Puppeteer)
- Template-based documents
- Digital signature embedding
- Timestamp and hash generation for authenticity

3. Database Design

PostgreSQL Schema:

-- Users Table

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    full_name VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    address TEXT,
    role VARCHAR(50),
    created_at TIMESTAMP DEFAULT NOW()
);

-- Cases Table
CREATE TABLE cases (
    id SERIAL PRIMARY KEY,
    case_number VARCHAR(50) UNIQUE NOT NULL,
    category VARCHAR(100) NOT NULL,
    status VARCHAR(50) DEFAULT 'INITIATED',
    complainant_id INT REFERENCES users(id),
    filed_at TIMESTAMP DEFAULT NOW(),
    deadline TIMESTAMP,
    resolution_option VARCHAR(10),
    closed_at TIMESTAMP
);

-- Case Parties Table (Many-to-Many)
CREATE TABLE case_parties (
    id SERIAL PRIMARY KEY,
    case_id INT REFERENCES cases(id),
    user_id INT REFERENCES users(id),
    role VARCHAR(50), -- 'complainant', 'respondent'
    statement_submitted BOOLEAN DEFAULT FALSE,
    statement_finalized BOOLEAN DEFAULT FALSE,
```

```
    selected_option VARCHAR(10)
);

-- Evidence Table
CREATE TABLE evidence (
    id SERIAL PRIMARY KEY,
    case_id INT REFERENCES cases(id),
    uploaded_by INT REFERENCES users(id),
    file_type VARCHAR(50),
    file_path VARCHAR(500),
    file_size BIGINT,
    description TEXT,
    uploaded_at TIMESTAMP DEFAULT NOW()
);

-- AI Analysis Table
CREATE TABLE ai_analyses (
    id SERIAL PRIMARY KEY,
    case_id INT REFERENCES cases(id),
    analysis_type VARCHAR(50), -- 'initial', 'clarification', 'final'
    prompt TEXT,
    response TEXT,
    options JSON, -- Array of 3 options
    created_at TIMESTAMP DEFAULT NOW()
);

-- Signatures Table
CREATE TABLE signatures (
    id SERIAL PRIMARY KEY,
    case_id INT REFERENCES cases(id),
    user_id INT REFERENCES users(id),
    signature_data TEXT, -- Base64 image
```

```
    signed_at TIMESTAMP DEFAULT NOW(),
    ip_address VARCHAR(50)
);

-- Documents Table
CREATE TABLE documents (
    id SERIAL PRIMARY KEY,
    case_id INT REFERENCES cases(id),
    document_type VARCHAR(50), -- 'settlement', 'escalation'
    file_path VARCHAR(500),
    hash VARCHAR(255), -- SHA-256 for integrity
    generated_at TIMESTAMP DEFAULT NOW()
);
```

MongoDB Collections (for unstructured data):

- `case_statements`: Full text statements from parties
- `ai_conversations`: Chat history with AI
- `case_notes`: Internal notes and flags

4. File Storage (MinIO - Open Source S3 Alternative)

Structure:

```
/disputes-storage
  /case-{case_number}
    /evidence
      /complainant
        - image1.jpg
        - video1.mp4
      /respondent
        - audio1.mp3
    /documents
      - settlement_agreement.pdf
    /signatures
      - complainant_signature.png
      - respondent_signature.png
```

5. AI Integration

Gemini API Implementation:

```
import google.generativeai as genai


# Configure API

genai.configure(api_key='YOUR_GEMINI_API_KEY')


# Multi-modal analysis

def analyze_dispute(case_data):

    model = genai.GenerativeModel('gemini-1.5-pro')


    prompt = f"""

    You are a legal AI assistant specializing in Indian civil law.


    CASE DETAILS:

    Category: {case_data['category']}


    COMPLAINANT STATEMENT:

    {case_data['complainant_statement']}


    RESPONDENT STATEMENT:

    {case_data['respondent_statement']}


    EVIDENCE SUMMARY:

    {case_data['evidence_summary']}


    TASK:

    1. Analyze the dispute objectively

    2. Identify applicable Indian laws and precedents

    3. Assess credibility of evidence

    4. Generate 3 fair resolution options
```

```
    FORMAT RESPONSE AS JSON:
    {{
        "analysis": "...",
        "applicable_laws": ["IPC Section X", "..."],
        "option_a": {{"title": "...", "description": "...", "legal_basis": "..."}},
        "option_b": {{"title": "...", "description": "...", "legal_basis": "..."}},
        "option_c": {{"title": "...", "description": "...", "legal_basis": "..."}}
    }}
    """

    response = model.generate_content(prompt)
    return response.text
```

Evidence Analysis:

```
# Image analysis
def analyze_image_evidence(image_path):
    model = genai.GenerativeModel('gemini-1.5-pro-vision')
    image = PIL.Image.open(image_path)

    prompt = "Analyze this image as legal evidence. Describe what you see objectively."
    response = model.generate_content([prompt, image])
    return response.text

# Audio transcription
def transcribe_audio(audio_path):
    # Use Whisper or Google Speech-to-Text
    # Then send transcript to Gemini for analysis
    pass
```

Technology Stack (All Free/Open Source)

Frontend

- **Framework**: React 18 with Vite
- **Styling**: Tailwind CSS
- **State Management**: Zustand or Redux Toolkit
- **Form Handling**: React Hook Form
- **File Upload**: React Dropzone
- **PDF Viewer**: react-pdf
- **Signature**: react-signature-canvas
- **Notifications**: React Toastify

Backend

- **Primary**: Node.js 20 + Express.js
- **AI Service**: Python 3.11 + FastAPI
- **Authentication**: JWT + bcrypt
- **File Processing**: Python (Pillow, PyPDF2, Tesseract OCR)
- **Task Queue**: Bull (Redis-based)

Databases

- **Primary DB**: PostgreSQL 15
- **Document DB**: MongoDB 7
- **Cache/Queue**: Redis 7
- **File Storage**: MinIO (S3-compatible)

AI & Processing

- **AI Model**: Google Gemini API (Free tier: 60 requests/minute)
- **OCR**: Tesseract OCR
- **Audio**: Whisper (OpenAI) or Google Speech-to-Text Free Tier
- **Document Generation**: PDFKit / Puppeteer

Deployment (Free Options)

- **Hosting**: Railway / Render / Fly.io (Free tiers)
- **Database**: Supabase (Free PostgreSQL)
- **File Storage**: Cloudflare R2 (10GB free) or Backblaze B2
- **Email**: Gmail SMTP (limited) or SendGrid (100 emails/day free)
- **SMS**: Twilio (Free trial credits)

DevOps

- **Version Control**: Git + GitHub
- **CI/CD**: GitHub Actions
- **Containerization**: Docker + Docker Compose
- **Monitoring**: Sentry (Free tier) + Winston Logger

Security Considerations

1. Data Protection

- **Encryption at Rest**: PostgreSQL encryption, encrypted file storage

- **Encryption in Transit**: HTTPS/TLS 1.3 mandatory
- **Sensitive Data**: Hash passwords (bcrypt), encrypt PII

2. Access Control

- **Authentication**: JWT with short expiry (15 min access, 7 day refresh)
- **Authorization**: Role-based access control (RBAC)
- **Case Privacy**: Users can only access cases they're involved in
- **File Access**: Signed URLs with expiration

3. Input Validation

- **File Uploads**:
    - Size limits (max 50MB per file)
    - Type validation (whitelist allowed extensions)
    - Virus scanning (ClamAV)
    - Content validation (check actual file type vs extension)

4. API Security

- **Rate Limiting**: Max 100 requests/15min per IP
- **CORS**: Restricted to frontend domain
- **API Key Management**: Environment variables, never in code
- **SQL Injection**: Parameterized queries only

5. Legal Compliance

- **Data Retention**: Store closed cases for 7 years (Indian law)
- **Audit Trail**: Log all actions with timestamps
- **GDPR/Data Protection**: Allow data export and deletion requests
- **E-Signature Validity**: Follow IT Act 2000 guidelines

Implementation Roadmap

Phase 1: MVP (6-8 weeks)

Week 1-2: Setup & Core Backend

- [ ] Setup development environment
- [ ] Database schema design and implementation
- [ ] User authentication system
- [ ] Basic API structure

Week 3-4: Case Management

- [ ] Case creation flow
- [ ] File upload system
- [ ] Evidence storage
- [ ] Party invitation system
- [ ] Email notifications

Week 5-6: AI Integration

- [ ] Gemini API integration
- [ ] Basic text analysis
- [ ] Option generation
- [ ] Simple evidence summarization

Week 7-8: Frontend & Decision Flow

- [ ] React frontend setup
- [ ] Case filing forms
- [ ] Statement submission
- [ ] Option selection interface
- [ ] Basic testing and bug fixes

Phase 2: Advanced Features (4-6 weeks)

- [ ] Multi-modal evidence analysis (images, audio, video)
- [ ] Advanced AI clarification system
- [ ] Document generation
- [ ] E-signature integration
- [ ] Enhanced UI/UX

Phase 3: Polish & Deploy (2-3 weeks)

- [ ] Security audit
- [ ] Performance optimization
- [ ] Deployment to production
- [ ] User documentation
- [ ] Admin dashboard

Project Improvements & Future Enhancements

Immediate Improvements

1. Legal Validity Enhancement

- **Lawyer Verification**: Require registered lawyer to witness settlements
- **Notarization**: Integration with e-notary services
- **Legal Templates**: Pre-approved settlement templates by legal experts
- **Court Recognition**: Work with local bar associations for system accreditation

2. User Experience

- **Mobile App**: React Native for iOS/Android
- **Vernacular Languages**: Hindi, Tamil, Telugu, Bengali support
- **Voice Input**: Allow voice recording of statements
- **Guided Filing**: Interactive wizard with legal term explanations
- **Video Conferencing**: Virtual mediation sessions (Jitsi Meet integration)

3. Evidence Handling

- **Blockchain Proof**: Store evidence hashes on blockchain for tamper-proof records
- **Metadata Preservation**: Extract and preserve EXIF data from images
- **Chain of Custody**: Track who accessed evidence when
- **Expert Witnesses**: Allow expert opinion uploads (medical reports, etc.)

4. AI Capabilities

- **Precedent Database**: Build database of similar resolved cases
- **Predictive Analysis**: Show likelihood of success in court if escalated
- **Bias Detection**: Analyze AI suggestions for potential bias
- **Multi-language Processing**: Analyze statements in regional languages
- **Sentiment Analysis**: Detect emotional tone in statements

5. Payment Integration

- **Filing Fees**: UPI/Razorpay integration for nominal fees
- **Lawyer Fees**: Split payment if parties hire representation
- **Escrow Service**: Hold compensation until settlement executed
- **Refunds**: Automatic refund if escalated to court

Advanced Features

6. Community & Trust

- **Public Case Database**: Anonymized resolved cases for reference (opt-in)
- **Reputation System**: Rate fairness of resolutions
- **Success Metrics**: Display resolution rate, average time
- **Peer Mediation**: Allow community mediators to assist AI

7. Analytics & Reporting

- **Admin Dashboard**: Case statistics, resolution rates, common dispute types
- **Performance Metrics**: AI accuracy tracking
- **Legal Insights**: Identify gaps in law or common issues
- **Export Reports**: Generate reports for legal research

8. Integration & Automation

- **Court Filing API**: If case escalates, auto-file to e-courts portal
- **Police Integration**: For disputes requiring FIR
- **Government Portal**: Link with grievance redressal systems
- **WhatsApp Bot**: Case status updates via WhatsApp Business API

9. Accessibility

- **Screen Reader Support**: WCAG 2.1 AA compliance
- **Simple Mode**: Simplified interface for elderly/non-tech users
- **Offline Mode**: PWA support for limited connectivity areas
- **SMS Interface**: File cases via SMS for feature phone users

10. Scalability & Performance

- **Microservices**: Break into smaller services
- **Load Balancing**: Handle high traffic
- **Caching**: Redis for frequent queries
- **CDN**: CloudFlare for static assets
- **Database Sharding**: Separate by state/region

Advanced AI Features

11. Intelligent Dispute Resolution

- **Emotion Detection**: Analyze emotional state from text/voice
- **Compromise Suggestions**: AI suggests middle-ground based on priorities
- **Cultural Sensitivity**: Consider regional customs in resolutions
- **Risk Assessment**: Predict chances of compliance with settlement
- **Follow-up System**: Check if settlement terms were implemented

12. Legal Research Integration

- **Case Law Database**: Access to Indian Kanoon, Manupatra
- **Auto-cite Laws**: Automatically reference relevant sections

- **Jurisdiction Mapping**: Suggest appropriate court based on dispute type
- **Update Monitoring**: Track new judgments affecting pending cases

Compliance & Standards

13. Regulatory Compliance

- **IT Act 2000**: Ensure e-signature validity
- **Indian Evidence Act**: Make evidence admissible in court
- **Consumer Protection Act**: If applicable to disputes
- **Arbitration and Conciliation Act**: Align with ADR principles
- **Data Protection**: Follow DPDP Act 2023

14. Quality Assurance

- **AI Audit Trail**: Log all AI decisions for review
- **Human Oversight**: Allow manual review of AI decisions by legal experts
- **Error Reporting**: Easy way to report AI mistakes
- **Continuous Training**: Update AI model with resolved cases
- **Bias Testing**: Regular audits for demographic bias

Business Model (Optional)

15. Sustainability

- **Freemium Model**: Basic resolution free, premium features paid
- **Lawyer Subscriptions**: Lawyers pay to list as witnesses
- **Government Grants**: Apply for legal aid funding
- **NGO Partnerships**: Collaborate with legal aid organizations
- **Corporate CSR**: Get funding from corporate social responsibility

---

Challenges & Mitigation

Technical Challenges

Challenge	Risk	Mitigation
Gemini API Limits	Free tier: 60 req/min	Implement request queuing, caching, batch processing
Large File Storage	Expensive storage costs	Compress files, set size limits, use free tier storage
AI Hallucinations	Incorrect legal advice	Always add disclaimer, human review option, cite sources
Multi-language Support	Complex implementation	Start with English, add languages iteratively

Real-time Notifications	Server load	Use WebSockets with Redis, implement push notifications

Legal Challenges

Challenge	Risk	Mitigation
Legal Validity	Settlements not enforceable	Partner with lawyers, follow Arbitration Act
Liability	If AI gives wrong advice	Clear ToS, disclaimer, insurance
Evidence Admissibility	Court may reject evidence	Maintain chain of custody, proper documentation
Privacy Concerns	Sensitive data leaks	Strong encryption, compliance with DPDP Act
Jurisdiction Issues	Inter-state disputes	Clearly define jurisdiction in ToS

Operational Challenges

Challenge	Risk	Mitigation
User Adoption	Low initial usage	Marketing, partnerships with legal aid NGOs
Trust Issues	Users don't trust AI	Show success stories, transparent process
Bad Faith Actors	Users misuse platform	Verification, flagging system, terms of service
Scalability	Can't handle growth	Microservices, cloud scaling, load balancing

---

Cost Estimation (Free Tier Limits)

Development Phase (Self-hosted)

- **Domain**: ₹500-1000/year (.in domain)

- **Development**: Free (your time)
- **Tools**: Free (VS Code, Git, etc.)

Deployment (Free Tiers)

- **Hosting**: Railway/Render (Free tier: 500 hrs/month)
- **Database**: Supabase (Free: 500MB)
- **File Storage**: Cloudflare R2 (Free: 10GB)
- **Email**: SendGrid (Free: 100 emails/day)
- **AI API**: Gemini (Free: 60 req/min)
- **Total**: ~₹1000/year (domain only)

Scaling Phase (When needed)

- **VPS**: DigitalOcean Droplet (~$6/month)
- **Database**: Managed PostgreSQL (~$15/month)
- **Storage**: Backblaze B2 (~$5/month for 100GB)
- **Gemini API**: Pay-as-you-go (₹0.075/1K input tokens)
- **Total**: ~₹2500-3000/month

Success Metrics

KPIs to Track

1. **Resolution Rate**: % of cases resolved without court
2. **Average Resolution Time**: Days from filing to closure
3. **User Satisfaction**: Rating of AI suggestions (1-5)
4. **Adoption Rate**: New cases per month
5. **Evidence Analysis Accuracy**: % of correct evidence interpretation
6. **Cost Savings**: Avg money saved vs traditional court
7. **Compliance Rate**: % of settlements actually implemented

---

Getting Started Checklist

Prerequisites

- [ ] Node.js 20+ installed
- [ ] Python 3.11+ installed
- [ ] PostgreSQL 15+ installed
- [ ] Redis installed
- [ ] Docker & Docker Compose
- [ ] Git setup
- [ ] Gemini API key obtained

Initial Setup

```
# 1. Create project structure

mkdir ai-dispute-resolver

cd ai-dispute-resolver

mkdir frontend backend ai-service docs
```

```
# 2. Initialize repos

cd frontend && npx create-vite@latest . --template react

cd ../backend && npm init -y

cd ../ai-service && python -m venv venv


# 3. Setup Docker Compose

docker-compose up -d postgres redis minio


# 4. Configure environment variables

cp .env.example .env

# Add your Gemini API key, DB credentials, etc.


# 5. Run migrations

npm run migrate


# 6. Start development servers

npm run dev # Backend

npm run dev:ai # AI service

npm run dev # Frontend (separate terminal)
```

Conclusion

This **AI Dispute Resolver** project is ambitious but achievable as a student project with the right planning and incremental development. Start with the MVP (Phase 1), validate the concept with real users, then gradually add advanced features.

Key Success Factors:

1. **Start Small**: Build core functionality first, add features iteratively
2. **User Feedback**: Test with friends/family, gather feedback early
3. **Legal Consultation**: Consult a lawyer to ensure compliance
4. **Documentation**: Document everything for your portfolio
5. **Open Source**: Consider open-sourcing to build community

This project demonstrates skills in full-stack development, AI integration, system design, legal tech, and problem-solving - excellent for your portfolio!

Next Steps:

1. Review this document thoroughly
2. Set up development environment
3. Start with database schema and basic auth
4. Build one feature at a time
5. Test frequently

Would you like me to create:

1. **Detailed API documentation** with endpoint specifications?
2. **Sample code** for any specific component?
3. **Database migration scripts**?
4. **Docker Compose configuration**?
5. **Frontend component structure**?

Let me know which part you'd like to dive into first! 🚀