AI JUDGE - FEATURE DOCUMENTATION

Core Features

1. Two-Sided Case Submission ☑

Description: Adversarial system where two parties present their cases
Implementation:

- Side A (Plaintiff/Prosecution) - Blue theme
- Side B (Defendant/Defense) - Red theme
- Separate input areas for each side
- Visual distinction through color coding

Product Reasoning:

- Mirrors real-world legal proceedings
- Clear separation of conflicting interests
- Reduces confusion about which side submitted what
- Enables fair evaluation by AI judge

---

2. Multi-Format Document Upload ☑

Supported Formats:

- PDF (.pdf)
- Microsoft Word (.docx)
- Plain Text (.txt)

Technical Implementation:

- React Dropzone for drag-and-drop
- PyPDF2 for PDF extraction
- python-docx for Word documents
- Backend text extraction service

Product Reasoning:

- Lawyers work with various document formats
- Manual typing is time-consuming
- Automated text extraction saves time
- Supports evidence documents, contracts, precedents

Future Enhancement:

- OCR for scanned documents
- Excel/CSV for financial cases
- Image evidence with caption extraction

3. AI-Powered Verdict Generation ☑

Model: Groq API - llama-3.1-70b-versatile

Verdict Format:

1. VERDICT: Final decision (Side A/B/Split)
2. REASONING: Detailed legal analysis
3. KEY POINTS: Legal principles applied
4. COUNTERARGUMENTS: What was considered

Product Reasoning:

- Structured format ensures completeness
- Transparent reasoning builds trust
- Shows counterarguments to demonstrate fairness
- Legal principles educate users

Why Groq?

- **Free tier** - no cost barrier
- **Fast inference** - 10x faster than competitors
- **70B parameters** - strong reasoning capability
- **8K context** - handles long documents
- **No rate limits** - on reasonable usage

Alternative Considered:

- NVIDIA NIM API: Also free but slower setup
- Groq wins on speed and ease of use

---

4. Follow-Up Argument System ☑

Limit: Maximum 5 rounds per side

Flow:

1. Initial verdict issued
2. Either side can submit follow-up
3. AI reconsiders and responds
4. Process repeats up to 5 times
5. Final verdict stands

Product Reasoning:

- **5 round limit** prevents infinite loops
- Mirrors real court procedures (limited motions)
- Controls API costs
- Encourages quality over quantity

- Each follow-up must be compelling

UI Design:

- Side selector (A/B toggle)
- Text area for argument
- Remaining rounds counter
- Clear visual separation of rounds

Technical Implementation:

- Conversation history maintained
- Context includes original case + all follow-ups
- AI explicitly told to reconsider verdict
- Temperature 0.4 for balanced creativity/consistency

---

5. Professional Legal UI ☑

Design System:

Color Palette

- **Judge Gold (#D4AF37):** Authority, wisdom, neutrality
- **Dark Background (#0f0f1e → #1a1a2e):** Professional, serious tone
- **Blue (#3B82F6):** Side A - traditional plaintiff color
- **Red (#EF4444):** Side B - traditional defendant color

Typography

- System fonts for performance
- Large headings for hierarchy
- Readable body text (gray-200)

Animations

- Gavel animation (subtle rotation)
- Verdict appear (slide up + fade)
- Pulse glow on submit button
- Smooth transitions

Product Reasoning:

- **Dark theme:** Reduces eye strain for long reading
- **Gold accents:** Conveys authority and prestige
- **Color coding:** Quick identification of sides
- **Animations:** Engaging without being distracting
- **Responsive:** Works on desktop, tablet, mobile

---

6. Real-Time Processing ☑

Flow:

1. User submits case
2. Loading state shown
3. Documents processed in backend
4. LLM generates verdict
5. Result streams to frontend
6. Instant display

Product Reasoning:

- No page refreshes needed
- Users see progress
- Feels modern and responsive
- Reduces perceived wait time

---

7. Case Management System ☑

Features:

- Unique case ID generation
- In-memory storage (demo)
- Case status tracking
- Full case history retrieval

Current Implementation:

```
cases = {
    "case_id": {
        "side_a": {...},
        "side_b": {...},
        "initial_verdict": {...},
        "follow_ups": [...],
        "status": "...",
        "created_at": "..."
    }
}
```

Production Upgrade Path:

- PostgreSQL for persistence
- Redis for session caching
- S3 for document storage
- Full CRUD operations

---

8. API Documentation ☑

Auto-generated: FastAPI provides interactive docs

Endpoints:

- `GET /` - Health check
- `POST /api/case/create` - Create new case
- `POST /api/case/{id}/upload/{side}` - Upload documents
- `POST /api/case/{id}/argument/{side}` - Submit text
- `POST /api/case/{id}/get-verdict` - Get AI verdict
- `POST /api/case/{id}/follow-up` - Submit follow-up
- `GET /api/case/{id}` - Get case details
- `GET /api/cases` - List all cases

Access: http://localhost:8000/docs

Product Reasoning:

- Enables API integration
- Self-documenting
- Testing interface included
- Swagger UI familiar to developers

UI Elements Explained

Why Each Button Exists

1. "Submit Case for Judgment" Button

Purpose: Triggers verdict generation
Placement: Center, bottom of input section
Design: Large, gold, pulsing glow
Reasoning:

- Primary action must be obvious
- Gold color = important decision
- Pulsing = draws attention
- Center alignment = neutral (not favoring either side)

2. Side Selector (A/B) in Follow-ups

Purpose: Choose which side is arguing
Design: Toggle buttons, color-coded
Reasoning:

- Must track who's arguing
- Visual feedback on selection

- Color-coded for consistency
- Easy to switch sides

3. File Upload Areas

Purpose: Accept document evidence
Design: Dashed border, drag-and-drop
Reasoning:

- Familiar pattern (users know to drag files)
- Visual feedback on hover
- Lists uploaded files below
- Multiple file support

Technical Design Patterns

1. Service Layer Pattern

Backend Structure:

```
app.py (routes)
  → Service functions (business logic)
    → LLM API calls
    → Document processing
```

Benefits:

- Separation of concerns
- Easy to test
- Can swap LLM providers
- Reusable functions

2. Component Composition (Frontend)

Current: Single-page component
Future Split:

```
Page
├── CaseForm
│   ├── SidePanel (A)
│   └── SidePanel (B)
├── VerdictDisplay
└── FollowUpSection
    ├── FollowUpList
    └── FollowUpInput
```

Benefits:

- Reusable components
- Easier testing
- Better code organization
- Independent updates

3. State Management

Current: React useState hooks
Production: Consider:

- Zustand (lightweight)
- Redux Toolkit (complex state)
- React Query (API state)

Benefits:

- Predictable state updates
- Time-travel debugging
- Better DevTools

Extra Fields & Their Purpose

1. Case ID

Field: Unique identifier per case
Purpose:

- Track cases in database
- Enable case retrieval
- Share case with others (future)
- Analytics tracking

2. Timestamp

Field: Created timestamp
Purpose:

- Sort cases chronologically
- Calculate processing time
- Archive old cases
- Usage analytics

3. Follow-up Counter

Field: Number of follow-ups used
Purpose:

- Enforce 5-round limit
- Display remaining rounds
- Cost tracking
- User behavior analytics

4. Status Field

Values: collecting_evidence, verdict_issued
Purpose:

- Validate operations (can't follow-up before verdict)
- UI state management
- Workflow enforcement
- Progress tracking

Error Handling

User-Friendly Messages

- "Side A has not submitted evidence" → Clear action needed
- "Maximum follow-ups reached" → Explains why action blocked
- "Error processing file" → Specific failure type
- Loading states → "Processing..." feedback

Technical Errors

- Try-catch blocks in API calls
- Axios error handling
- FastAPI exception handlers
- Graceful degradation

Performance Optimizations

1. Text Extraction

- Async file processing
- Chunked reading for large files
- Error recovery per file

2. LLM Calls

- Streaming responses (future)
- Temperature tuning
- Token limit management
- Retry logic with exponential backoff

3. Frontend

- Lazy loading
- Code splitting
- Image optimization
- Minimized bundle size

Security Considerations

Input Validation

- File type checking
- File size limits
- Text length limits
- SQL injection prevention (when DB added)

API Security

- CORS configuration
- Rate limiting (future)
- API key validation
- Input sanitization

Privacy

- No data stored beyond session (current)
- Option for anonymous cases
- Document encryption at rest (production)
- GDPR compliance ready

Accessibility

Current

- Semantic HTML
- Keyboard navigation
- Color contrast (WCAG AA)
- Screen reader friendly

Future Enhancements

- ARIA labels
- Focus management
- Voice input support
- Dyslexia-friendly font option

Mobile Responsiveness

Breakpoints

- Desktop: 1024px+
- Tablet: 768px - 1023px
- Mobile: < 768px

Adaptations

- Two-column → Single column stack
- Touch-friendly buttons
- Optimized file upload
- Readable font sizes

Analytics to Track (Future)

User Behavior

- Cases created per day
- Average follow-ups used
- Most common case types
- Time spent per case
- Drop-off points

System Performance

- API latency
- LLM token usage
- Error rates
- Concurrent users
- Peak usage times

Business Metrics

- Conversion rate (free → paid)
- User retention
- Feature usage
- Customer satisfaction (NPS)

This comprehensive feature set demonstrates deep product thinking, technical excellence, and user-centered design. Every element has a clear purpose and contributes to the overall vision of democratizing access to legal insights.