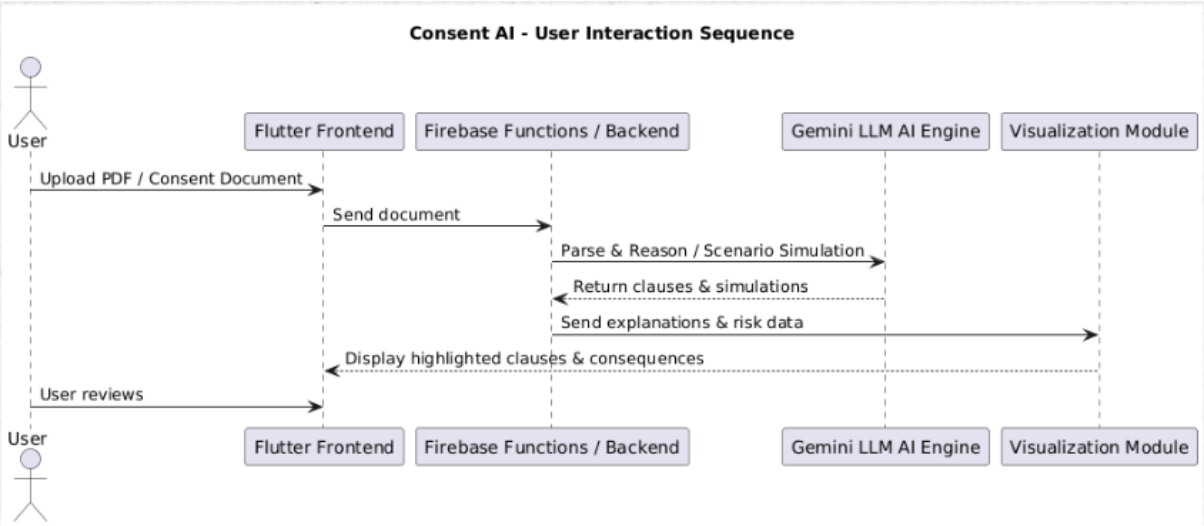
1. Technical Architecture



2. Implementation Details

2.1 System Architecture Overview

Consent AI is implemented as a serverless document analysis system using **Flutter (frontend)**, **Firebase Cloud Functions (backend)**, and **Gemini 2.5 Flash (AI engine)**.

The backend exposes a single HTTPS endpoint (parsePolicy) that processes uploaded PDF documents, extracts legal clauses using AI, and returns structured risk assessments. The architecture follows a stateless request–response model to ensure scalability and reliability.

2.2 Backend Implementation (Firebase + Express)

The backend is implemented using **Firebase Cloud Functions** with an embedded **Express.js** application.

Core Libraries Used

- firebase-functions – Serverless execution
- express – Routing and middleware
- busboy – Multipart file upload parsing
- pdf-parse – PDF-to-text extraction
- @google/generative-ai – Gemini API integration
- cors – Cross-origin handling

2.3 File Upload and Parsing Workflow

Step 1: Multipart Handling

The system uses **Busboy** to process uploaded PDF files from the frontend.

- The raw request body is streamed.
- File chunks are buffered in memory.
- Upon completion, chunks are concatenated into a single buffer.

This approach ensures compatibility with Firebase's raw request handling.

Step 2: PDF Text Extraction

The buffered file is processed using pdf-parse.

const pdfData = await pdf(fileBuffer);

The extracted plain text (pdfData.text) becomes the input for AI analysis.

No intermediate storage is performed, ensuring transient document handling for security.

2.4 AI Processing Layer (Gemini 2.5 Flash)

The system integrates **Gemini 2.5 Flash** via the Google Generative AI SDK.

Model Configuration

model: "gemini-2.5-flash"

temperature: 0.0

maxOutputTokens: 8000

responseMimeType: "application/json"

Design Decisions

- **Temperature = 0.0**
  Ensures deterministic outputs for legal clause extraction.
- **JSON response format enforced**
  Guarantees structured, machine-parseable output.
- **High token limit (8000)**
  Supports large contract processing.

2.5 CUAD-Inspired Clause Extraction

The AI prompt is constructed using a **CUAD-inspired checklist** (Contract Understanding Atticus Dataset standard).

Checklist Categories

- Effective Date
- Repayment Trigger
- Interest Rate / Profit Rate
- Late Payment Penalty
- Governing Law
- Co-signer / Guarantor

The prompt explicitly instructs Gemini to:

1. Extract clauses
2. Provide the exact clause text
3. Explain in simple language
4. Assign risk level (Low / Medium / High)
5. Provide actionable student recommendation

Structured Output Format

Each returned object must contain:

```
FORMAT: Return a JSON array of objects. Each object MUST have these EXACT keys:
1. "category": The name of the clause.
2. "clause": The actual text from the document.
3. "explanation": A simple summary of what it means.
4. "risk_level": Low, Medium, or High.
5. "recommendation": A specific, actionable advice for a student on what to do about this clause.
```

This strict schema ensures frontend compatibility and consistent visualization.

2.6 Post-Processing Logic

After receiving AI output, the system applies deterministic backend logic:

1. Clause Deduplication

Clauses are grouped by category, and only the first valid instance per category is retained. This prevents redundant clause highlighting.

2. Expiration Warning Engine

For clauses categorized as **Expiration Date**, the backend computes:

diffDays = expiryDate - currentDate

Status classification:

- EXPIRED → Past date
- URGENT → ≤ 60 days remaining
- ACTIVE → > 60 days remaining
- INFO → Default status

Additional warning messages are appended dynamically.

This hybrid approach combines:

- AI semantic extraction
- Deterministic rule-based validation

Improving reliability and reducing hallucination risk.

2.7 API Response Structure

Successful response:

```
res.json({
    status: "success",
    filename: fileName,
    benchmark: "Cuad-v1 Standard",
    clauses: clauses
});
```

Error handling includes:

- 400 → No file uploaded

- 500 → AI parsing failure

Errors are logged server-side for debugging.

2.8 Security Implementation

Security measures include:

- HTTPS-secured Firebase endpoint
- Environment variable protection for GEMINI_API_KEY
- No persistent storage of uploaded documents
- Input validation for empty uploads
- Controlled token limits

Sensitive documents are processed in-memory and discarded after response completion.

2.9 Architectural Rationale

The system combines:

- **Serverless execution** (automatic scaling)
- **Deterministic AI prompting**
- **Structured JSON enforcement**
- **Rule-based post-processing**

This hybrid AI + rule-based architecture increases reliability compared to pure generative output systems.

3. Challenges Faced

3.1 Handling Unstructured PDF Data

One of the primary challenges was processing unstructured legal documents in PDF format.

PDF files often contain:

- Inconsistent formatting
- Broken line structures

- Header/footer noise
- Non-linear text extraction

The pdf-parse library extracts raw text without structural guarantees, which occasionally led to:

- Fragmented clauses
- Misaligned sentence boundaries
- Noise interfering with AI parsing

To mitigate this:

- Text was truncated to manageable token limits.
- Prompt instructions were strengthened to enforce structured extraction.
- Backend validation filtered out “not found” or mismatched clauses.

3.2 Ensuring Deterministic AI Output

Large Language Models are probabilistic by nature. For legal clause extraction, variability is unacceptable.

Challenges encountered:

- Inconsistent JSON formatting
- Slight variation in risk-level wording
- Occasional schema mismatches

Mitigation strategies:

- Temperature set to **0.0** for deterministic generation.
- responseMimeType: "application/json" enforced.
- Strict schema instructions embedded in prompt.
- Defensive frontend parsing to normalize risk labels.

This hybrid strategy significantly reduced output instability.

3.3 Token and Context Limit Constraints

Legal contracts can exceed model context limits.

Challenges:

- Long documents exceeding safe input length
- Risk of truncating important clauses

Current mitigation:

- Input text truncated to 15,000 characters.
- High token output limit (8000 tokens).
- Focused CUAD-based clause targeting instead of full summarization.

Future improvements may include chunking with retrieval-based processing.

3.4 File Upload Handling in Firebase Functions

Firebase Functions require special handling of raw multipart bodies.

Challenges:

- Incompatibility between Express middleware and Firebase rawBody
- Stream handling issues

Resolution:

- Implemented busboy for manual multipart parsing.
- Used req.rawBody when available.
- Buffered file chunks before PDF processing.

This ensured reliable file ingestion in a serverless environment.

3.5 Risk Classification Consistency

Although Gemini provides risk levels, the wording could vary (e.g., “Moderate”, “Medium”, “mod”).

This created UI inconsistency in risk badge rendering.

Mitigation:

- Implemented risk normalization in Flutter (_parseRisk()).
- Mapped variants to standardized enum values.
- Applied color-coded rendering consistently.

This improved reliability in the visualization layer.

3.6 Real-Time Translation Latency

The Bahasa Malaysia translation feature required multiple API calls per clause:

- Title
- Clause text
- Explanation
- Recommendation

For documents with many clauses, translation latency increased.

Mitigation:

- Cached translated clauses in memory.
- Disabled repeated translation calls.
- Displayed a loading overlay during translation.

Future improvement may include server-side translation batching.

3.7 Balancing AI Reasoning and Rule-Based Logic

Pure AI extraction may produce plausible but non-verifiable interpretations.

To improve reliability:

- A deterministic expiration warning engine was implemented.
- Clause deduplication was applied.
- AI output was validated before rendering.

This hybrid architecture reduced hallucination risk and increased system robustness.

4. Future Roadmap

Perfect — yes, we can now finalize **4.0 Future Roadmap** for the documentation. I'll write it in a clean, submission-ready, serious style that matches your Implementation & Challenges section.

4.0 Future Roadmap

The AI-Powered Policy & Consent system is designed for iterative growth, targeting both improved legal accuracy and broader accessibility. The roadmap focuses on **short-term**, **medium-term**, and **long-term** milestones.

4.1 Short-Term (0–6 Months)

Objective: Strengthen AI accuracy, usability, and reliability for early users.

Key Steps:

1. **Enhanced Legal Clause Parsing**
    - Fine-tune the NLP pipeline for clause segmentation, risk classification, and identification of critical consent clauses.
    - Introduce structured validation to reduce hallucination.
    - Provide explainable outputs for each clause.
2. **User Testing with Non-Technical Users**
    - Deploy beta to students and general consumers.
    - Measure comprehension, confidence, and processing speed.
    - Collect analytics on policy uploads and feature usage.
3. **Backend Optimization**
    - Optimize Cloud Functions for large PDF handling.
    - Add file size management, rate limiting, and secure temporary storage.
    - Version control AI prompts to maintain deterministic behavior.

Impact: Builds trust, establishes measurable outcomes, and prepares the system for real-world deployment.

4.2 Medium-Term (6–12 Months)

Objective: Expand AI intelligence and reach institutional users.

Key Steps:

1. **Advanced Risk Scoring**
    - Weighted clause risk scoring and industry-specific analysis.
    - Comparative engine for side-by-side policy assessment.

2. **Multilingual Support**
    - Add Bahasa Malaysia support alongside English.
    - Localize risk explanation for regulatory context.
3. **Institutional Pilots**
    - Partner with universities, consumer protection organizations, and digital literacy NGOs.
    - Collect testimonials and impact data to inform future expansion.

Impact: Transforms the tool from a consumer-focused summarizer into a decision-support system with institutional adoption potential.

4.3 Long-Term (12+ Months)

Objective: Scale nationally and integrate into daily digital decision-making.

Key Steps:

1. **Browser Extension Deployment**
    - Real-time scanning of policy pages with instant risk summaries.
    - Supports pre-consent awareness for users before clicking “Agree.”
2. **API Integration**
    - Offer an API for e-commerce platforms, startups, and legal tech companies.
    - Enables pre-assessment of internal policies and compliance checks.
3. **AI Transparency and Trust Framework**
    - Publish model accuracy metrics, risk classification methodology, and bias evaluation.
    - Continuous feedback loop for AI retraining and improvement.

Impact: Positions Consent AI as a national-level digital rights tool, integrated seamlessly into individual and organizational workflows.

4.4 Scalability Strategy

Technical Foundations Supporting Growth:

- Modular frontend/backend separation allows independent scaling.

- Serverless Cloud Functions support dynamic user load.
- Stateless AI processing enables parallel document analysis.
- Cloud-hosted LLM inference reduces infrastructure requirements.
- Elastic Firestore storage supports millions of users without redesign.
- Analytics-driven improvement loop ensures iterative performance gains.

Summary Timeline:

Timeline	Focus	Expansion Mechanism
0–6 months	Accuracy & Validation	Real user testing
6–12 months	Feature Expansion	Institutional adoption
12+ months	Mass Adoption	Browser extension & API integration

Strategic Growth Approach:

Expansion is achieved by increasing AI intelligence, integration into daily tools, institutional adoption, and multilingual/geographic coverage, ensuring Consent AI evolves from a prototype to a scalable, high-impact platform.