Bilkent University
Department of Computer Engineering

CS 491 Senior Design Project I
T2516
Agreemind

Analysis and Requirements Report

Ata Oğuz, 22202453, ata.oguz@ug.bilkent.edu.tr
Ata Soykal, 22202290, ata.soykal@ug.bilkent.edu.tr
Can Polat Bülbül, 22203369, polat.bulbul@ug.bilkent.edu.tr
Edip Emre Dönger, 22201531, emre.donger@ug.bilkent.edu.tr
Emir Görgülü, 22202834, emir.gorgulu@ug.bilkent.edu.tr

Supervisor: Hamdi Dibeklioğlu
Course Instructors: Mert Bıçakçı, İlker Burak Kurt

Contents

1 Introduction

For an average person, navigating today's agreements, from digital Terms of Service to rental agreements, has become an extremely difficult task. Critical obligations are frequently covered by dense legal jargon, which causes users to unintentionally accept restrictive conditions, ignore significant rights, or miss crucial deadlines. By acting as a personal AI companion to assist people in comprehending, monitoring, and carrying out their agreements, Agreemind minimizes this vulnerability. Agreemind, in contrast to corporate tools, is designed for the individual signer and uses sophisticated Natural Language Processing (NLP) to convert complicated clauses into understandable, straightforward explanations.

The system acts as a protective layer, automatically detecting risk patterns, hidden obligations, and aggressive terms before a user commits. Accessible via multiple platforms, Agreemind supports the entire contract lifecycle. The Browser Extension offers real-time protection by analyzing online terms prior to acceptance, while the Personal Vault securely organizes existing documents.

Beyond simple analysis, Agreemind empowers active management through the Rights Enforcer, which highlights actionable rights (like data access or cancellation) and generates draft templates to help users exercise them. Additional features include a Comparison Service to track version changes and a Notification Service for upcoming renewals. By centralizing document storage and providing actionable AI-driven insights, Agreemind shifts the user’s role from passive acceptance to informed control, ensuring they remain protected throughout the life of an agreement.

2 Current System

The current legal technology environment is overwhelmingly leaned towards enterprise solutions, supplying primarily to law firms and corporations with complex contract lifecycle management needs. While these tools are powerful, they create a significant void in the consumer market, leaving individuals to navigate complex agreements from software Terms of Service to rental leases, without adequate resources or assistance. The default approach for most people is a manual, "sign-and-forget" methodology, which is neither feasible nor safe given the density of legal language and the frequency of unnotified updates. Unlike corporate teams equipped with sophisticated AI platforms, the average user lacks accessible tools to interpret obligations or track changes, often resulting in the passive acceptance of unfavorable terms.

Agreemind addresses this disparity by positioning itself as a "Personal Legal Companion" dedicated to the individual signer rather than the corporate drafter. By integrating comprehensive analysis, storage, and enforcement tools, Agreemind shifts the user from a state of vulnerability to one of active, informed management. The following table outlines the limitations of existing solutions and how Agreemind distinguishes itself:

Table 1: Current Systems and Agreemind's Differentiation.

Category & Examples	**Description and Agreemind's Differentiation**
Enterprise Legal AI (Ironclad, Kira Systems, Luminance)	These systems employ sophisticated machine learning models to automate clause detection and streamline the review process. They are engineered exclusively for corporate legal departments to optimize compliance workflows and manage high-volume commercial contracts [1], [2], [3]. **Differentiation:** Such platforms are typically cost-prohibitive and too intricate for the average consumer. Agreemind is specifically designed for the "non-expert reader," focusing on translating complex legal terminology into plain language and providing an accessible interface rather than a dense corporate dashboard.
Crowdsourced Transparency Project (ToS;DR, Open Terms Archive):	These initiatives rely on community contributions to grade and summarize the Terms of Service for popular websites [4], [5]. **Differentiation:** These platforms suffer from limited coverage and cannot handle arbitrary personal documents (e.g., a specific landlord's lease or a freelance NDA). Agreemind utilizes AI to analyze any document uploaded by the user, providing immediate, personalized analysis rather than relying on a pre-existing database.
Real-time ToS Detector (Termzy AI)	These tools generally function as browser extensions, designed to scan and flag terms and conditions on websites at the exact moment of user interaction [6]. **Differentiation:** While effective during browsing, these tools lack post-acceptance support. Agreemind distinguishes itself by managing the full agreement lifecycle; features like the "Personal Vault" and "Rights Enforcer" allow users to archive contracts, track updates over time, and draft legal requests long after the initial signing.
General-Purpose AI Chatbots	These large language models allow users to paste text or upload documents and ask for summaries or explanations. They are widely accessible and can handle general queries about text.

(ChatGPT, Gemini, Claude)	**Differentiation:** These tools have significant privacy risks (data usage for training), lack specific legal safeguards, and do not provide a secure "Personal Vault" for long-term storage. Agreemind is a purpose-built environment that ensures data privacy, tracks deadlines, and enforces rights long after the chat session ends.
Online Legal Services *(LegalZoom, Rocket Lawyer)*	These platforms function as a bridge between personal legal work and hiring a traditional law firm. Their primary business model revolves around document assembly and human attorney connection. Users can access libraries of pre-drafted templates (such as wills, LLC formation documents, or rental leases) and customize them through a questionnaire-based interface [7], [8]. **Differentiation:** These services are often expensive and slow, relying on human intervention or generic templates. Agreemind provides rapid, automated analysis for contracts at a fraction of the cost, empowering users to understand documents without waiting for a consultation.

3 Proposed System

3.1 Overview

Agreemind is a consumer-facing legal assistance platform that helps everyday users understand and manage online Terms of Service, Privacy Policies, and common consumer contracts by providing plain-language explanations, risk-focused highlights, and deadline/obligation extraction.

The system accepts agreements through multiple entry points (mobile app, web interface, and a browser extension that can forward the current page), then processes the content through a centralized backend pipeline that parses documents, identifies clause boundaries, generates summaries, flags potentially risky terms (e.g., data sharing, auto-renewal, arbitration), and extracts key dates for reminders.

Agreemind is designed to support informed decision-making rather than replace legal professionals; therefore, outputs are presented as informational guidance with clear uncertainty and source traceability to the original text. Users can optionally store agreements in a personal vault for later search and comparison, with privacy and security controls applied across storage and any interactions with external AI services.

3.2 Functional Requirements

3.2.1 Document Ingestion & Preparation

- The system must allow users to input agreements through various channels to ensure ease of access. This includes standard file uploads (PDF, DOCX, text files), direct text pasting, importing from HTML pages, and sharing directly from mobile devices.
- Upon ingestion, the system must be capable of performing Optical Character Recognition (OCR) on scanned documents or images to convert them into machine-readable text.
- The system must parse the raw text to identify and segment the content into a structured format. This involves detecting and tagging logical units such as clauses, section headings, numbering, and spatial positioning to prepare the document for analysis.

3.2.2 Contract Analysis

- The system shall automatically generate a high-level, abstractive summary of the entire agreement. This summary must translate complex legal terminology into plain language that is easily understandable by a non-expert reader.
- The system must categorize each segmented clause into predefined legal categories, such as renewal terms, fee structures, liability limitations, dispute resolution mechanisms, and privacy policies.
- The system shall analyze clauses to detect potentially unfair, risky, or aggressive terms. It must assign a risk level to these clauses using a color-coded representation (e.g., Green, Yellow, Red) to visually alert the user to danger zones.
- For every flagged risk, the system must provide a brief explanation describing specifically why the clause is considered risky in that context.
- The system shall cross-reference clauses against a knowledge base of domain-specific and jurisdiction-specific rules to identify likely compliance issues or violations of consumer protection laws.

3.2.3 Risk, Obligations & Deadlines

- The system must detect specific risk patterns within the text, including general red flags (e.g., unilateral modification clauses, forced arbitration) and domain-specific risks tailored to the document type (e.g., a rental lease vs. a software license).
- The system shall identify specific actions required of the user, such as cancelling a service, opting out of data sharing, or filing a claim. It must further extract the necessary details to perform these actions, including the communication channel (email, web portal), contact information, and the responsible actor.

- The system must identify time-sensitive elements within the text, such as "within 14 days" or "before renewal." It shall convert these relative or absolute references into structured, calendar-ready deadline objects.

3.2.4 Personal Vault & Querying

- The system shall maintain a secure "Personal Vault" where users can save, organize, and retrieve their analyzed agreements. This vault must act as a central repository for the user's legal history.
- Users shall be able to query their stored agreements through a natural-language chatbot.
- Users shall be able to search through all stored agreements using natural language (multi-document query).

3.2.5 On-Demand Proactive Protection

- The system shall provide a native "Share Extension" for both iOS and Android mobile platforms. This integration allows users to manually send content, such as PDF files, website URLs, or selected text, directly from third-party applications (e.g., Chrome, Safari, Gmail, Drive) to Agreemind for immediate analysis.
- Upon selecting "Agreemind" from the system share menu, the mobile app shall automatically launch, ingest the shared content, and present the risk analysis summary without requiring the user to manually save and upload files.
- If a user shares a URL (e.g., a link to a Terms of Service page), the system shall automatically fetch the full HTML content of that page, parse the legal text, and generate a report.
- If the device is offline when content is shared, the system shall queue the request and process the analysis once connectivity is restored.

3.2.6 Version Tracking & Change Detection

- The system shall track and store multiple versions of the same agreement over time, maintaining a complete history of the contract's lifecycle.
- When a new version of a stored agreement is detected or uploaded, the system must automatically compare it against the previous version. It shall highlight specific clauses that have been added, removed, or modified.
- The system shall provide a side-by-side comparison view that includes generated summaries explaining the practical significance of the detected changes, rather than just showing raw text diffs.

3.2.7 Contract Comparison

- Users must be able to select two separate contracts or drafts and compare them side-by-side within the interface.
- The system shall align semantically similar clauses between the two documents (even if they are in different orders) and highlight the differences in text, identified risks, obligations, and deadlines.

3.2.8 Rights Enforcer

- The system must actively identify which legal rights a user is entitled to exercise based on the contract's text and applicable regulations (e.g., GDPR rights, consumer cancellation rights).
- For identified actionable rights, the system shall be capable of generating formal draft requests, such as letters for data access, contract termination, or opting out of specific clauses.
- The system shall automatically locate and present the relevant contact channels extracted from the documents such as email addresses, URL forms, or physical addresses, to facilitate the sending of these requests.

3.2.9 Alerts & Reminders

- The system must continuously monitor stored agreements for upcoming critical events, including renewal dates, cancellation deadlines, claim windows, and payment due dates.
- Based on the extracted deadlines, the system shall schedule and dispatch timely notifications to the user. These reminders must be sent sufficiently in advance to allow the user to take necessary action before the opportunity expires.

3.3 Non-functional Requirements

3.3.1 Usability

- The user interface shall present summaries, clause flags, and risks in clear and readable formats understandable by non-experts.
- Color-coded indicators for risk levels shall follow accessibility guidelines.
- The UI for the mobile app should be intuitive and easy to use.
- The application shall support "Dark Mode" and dynamic text sizing to accommodate user preferences and reduce eye strain during reading.

3.3.2 Portability

- The system shall run on major modern browsers and mobile operating systems.

- The backend shall be deployable on major cloud platforms without major modification.
- The browser extension shall support Chromium-based browsers and Firefox, subject to platform API limits.
- The backend services shall be containerized (Docker) to ensure consistent deployment across different cloud providers (AWS, Azure, GCP) or on-premise servers.
- The mobile application shall be compatible with Android 12+ and iOS 15+, covering the active mobile users.

3.3.3 Maintainability

- The system shall use a modular architecture so that core analysis components (clause classification, risk detection, obligation extraction) are independent from domain-specific logic.
- New domains, rule sets, or knowledge bases shall be addable as separate, self-contained modules without modifying core code.
- Regulatory or industry-specific rules shall be stored in external, versioned configuration files so they can be updated or expanded easily.
- The system shall expose clear internal interfaces that define how domain modules interact with the core engine, enabling low coupling and simple future extension.
- Updating or replacing domain modules, rules, or knowledge bases shall not require system downtime.
- Code shall be consistently structured and documented to support long-term maintainability.

3.3.4 Reliability

- The system shall maintain high availability.
- The vault and document records shall not be lost due to server errors; periodic backups must be maintained.
- Notification services shall reliably trigger reminders before deadlines.
- The system shall implement a Mean Time to Recovery (MTTR) in the event of a critical service crash.
- The notification service shall employ a retry mechanism to ensure delivery of critical deadline reminders in case of temporary network failure.

3.3.5 Scalability

- The analysis pipeline shall scale horizontally to handle multiple simultaneous document uploads.
- The system shall support growth in the number of users and stored documents without significant performance degradation.
- The vector store and search mechanisms shall support large embedding collections efficiently.

3.3.6 Privacy

- Users shall retain full ownership of uploaded contracts and analysis results.
- No contract text or user-generated data shall be used for model training or external sharing without explicit opt-in.
- The system shall provide mechanisms for deleting individual documents from the vault and deleting the entire user account and all associated data.
- The system shall comply with applicable data protection laws.
- All personal data and document text shall be encrypted.
- The system shall implement a logical separation of data, ensuring that a user's document embeddings in the vector store are isolated and cannot be queried by other users.

3.4 Pseudo Requirements

- Agreemind will target users across multiple platforms, allowing access through a Web App, Mobile App, and Browser Extension to ensure contract analysis is available on any device.
- Git and GitHub will be used for version control
- Jira will be used for issue tracking and managing the project
- PostgreSQL will be used as the primary relational database to store structured analysis outputs, clause metadata, user records, and detected deadlines.
- Python will serve as the primary programming language for the backend and the core processing layer, chosen for its extensive support for NLP libraries and AI integration.
- FastAPI will be used to develop the REST API, ensuring stateless and scalable communication between the backend and the web, mobile, and extension clients.

- React Native will be utilized to construct the Web App interface, providing a responsive and user-friendly dashboard for managing the "Personal Vault".
- PyTorch and the Hugging Face Transformers library will be used to implement the Abstractive Summarization and Risk Detection modules using BERT or GPT variants.
- LangChain (or custom pipelines) will be used to orchestrate the Retrieval-Augmented Generation (RAG) flows, connecting the analysis modules with the LLM for the "Chat/Query Service".
- AWS S3 (or compatible Object Store) will be used to securely store the original uploaded documents (PDF, DOCX) in their raw format.
- Docker will be used to containerize the application services, ensuring consistency across development and production environments.
- Zoom will be used for synchronous meetings and real-time project discussions, while WhatsApp will be used for asynchronous communication.
- Automated Clause Classifiers will be trained to detect potentially unfair terms and assign risk levels (color-coded indicators) based on labeled legal datasets.
- Named Entity Recognition (NER) models will be used to extract time-sensitive elements and specific entities, such as renewal dates and cancellation windows, to structure actionable deadlines.
- Chromium and Firefox APIs will be utilized to construct the browser extension, enabling real-time, on-page analysis of online agreements before acceptance.

3.5 System Models

3.5.1 Scenarios

Scenario 1: Sign Up

Primary Actor: End User

Supporting Actors: Agreemind Mobile/Web UI, Backend API, Authentication Serviced

Entry Condition: User is on the Sign Up screen and is not authenticated.

Exit Condition (Success): User account is created; user is authenticated and redirected to the home/dashboard screen.

Main Flow:

1. User selects **Sign Up**.
2. System displays a registration form (e.g., email, password, confirm password).
3. User submits the form.

4. Backend validates input (email format, password policy, uniqueness of email).
5. Backend creates the user account and issues an authentication token/session.
6. Client stores the session securely and navigates the user to the main app screen.

Alternative/Exception Flows:

- **A1 (Email Already Registered):** System informs the user and offers Log In or Reset Password.
- **A2 (Weak Password):** System rejects the password and displays the password requirements.
- **A3 (Network/API Error):** System shows a retry message; no account is created unless confirmation is received.

Scenario 2: Log In

Primary Actor: End User

Supporting Actors: Agreemind Mobile/Web UI, Backend API, Authentication Service

Entry Condition: User is on the Log In screen and is not authenticated.

Exit Condition (Success): User is authenticated and can access the vault and saved reports.

Main Flow:

1. User selects **Log In**.
2. User enters email and password.
3. Client submits credentials to backend.
4. Backend validates credentials and issues an authentication token/session.
5. Client stores the session securely and redirects to the home/dashboard screen.

Alternative/Exception Flows:

- **A1 (Invalid Credentials):** System displays an error message and allows retry without revealing which field was incorrect.
- **A2 (Account Locked/Rate Limited):** After repeated failures, system temporarily blocks attempts and informs the user.
- **A3 (Session Expired):** If an existing session is invalid/expired, system requests login again.

Scenario 3: Log Out

Primary Actor: End User

Supporting Actors: Agreemind UI, Backend API (optional), Authentication Service

Entry Condition: User is authenticated and is in the app.

Exit Condition (Success): Session is cleared on the client (and invalidated server-side if applicable); user returns to the login screen.

Main Flow:

1. User selects **Log Out** from settings/menu.
2. Client clears local session tokens securely.
3. Client navigates to the Log In screen.
4. Backend invalidates the token/session.

Alternative/Exception Flows:

- **A1 (Offline Logout):** Client still clears local session; server invalidation occurs on next connection.

Scenario 4: Forgot Password

Primary Actor: End User

Supporting Actors: Agreemind Mobile/Web UI, Backend API, Authentication Service, Email Service

Entry Condition: User is on the Log In screen and cannot access their account.

Exit Condition (Success): User sets a new password and can log in successfully.

Main Flow:

1. User selects **Forgot Password** on the Log In screen.
2. System prompts for the account email address.
3. User enters email and submits.
4. Backend verifies that the email exists (without revealing account existence explicitly, if you choose to prevent enumeration).
5. Backend generates a time-limited reset token and sends a reset link/code to the email address.
6. User opens the reset link (or enters the code) and sets a new password.
7. Backend validates the new password, updates credentials, and confirms success.
8. User logs in with the new password.

Alternative/Exception Flows:

- **A1 (Invalid/Expired Token):** System rejects the reset attempt and asks the user to request a new reset email.
- **A2 (Weak Password):** System rejects the password and shows password policy requirements.
- **A3 (Email Delivery Failure):** System shows a retry option and suggests checking spam/junk folders.
- **A4 (Rate Limiting):** System limits reset requests to prevent abuse and informs the user to wait before retrying.

Scenario 5: Upload an Agreement PDF and Generate an Analysis Report

Primary Actor: End User

Supporting Actors: Agreemind Mobile/Web UI, Backend API, Document Processing Service, LLM Service

Entry Condition: User is on the “New Analysis / Upload” screen; user is authenticated (or using a guest mode if supported).

Exit Condition (Success): A new Analysis Report is created and shown to the user; the original document is stored temporarily or in the vault depending on user choice.

Main Flow:

1. User selects **Upload PDF** and chooses a file from device storage.
2. Client uploads the file to the backend.
3. Backend validates file type/size and creates an “Analysis Job” with status **Queued**.
4. Document Processing extracts text (OCR if needed) and normalizes formatting.
5. Clause segmentation runs and produces clause boundaries.
6. The LLM pipeline generates: (a) plain-language summary, (b) risk flags per clause, (c) extracted dates/obligations.
7. Backend stores the analysis results and marks the job **Completed**.
8. Client displays the report: summary + risk categories + highlighted clauses + extracted dates.

Alternative/Exception Flows:

- **A1 (Unsupported File):** If the file is not a supported type, backend rejects it and client shows an error with accepted formats.
- **A2 (OCR Failure/Low Confidence):** If OCR fails or confidence is low, system returns partial results and asks the user to re-upload a better scan or paste text.

- **A3 (LLM Unavailable):** If the LLM call fails, system returns extracted text + segmentation (if available) and marks summary/risk fields as "Unavailable."

Scenario 6: Paste Agreement Text and Generate an Analysis Report

Primary Actor: End User

Supporting Actors: Agreemind UI, Backend API, LLM Service

Entry Condition: User is on "Paste Text" screen.

Exit Condition (Success): Report shown with summary, risk flags, and dates.

Main Flow:

1. User pastes agreement text into the text input area.
2. Client performs basic validation (non-empty, length limits) and submits to backend.
3. Backend stores the raw text as a new Analysis Job.
4. Backend segments clauses and runs summarization + risk detection + date extraction.
5. Client displays the resulting report with traceability (each highlight maps to source text).

Alternative/Exception Flows:

- **A1 (Text Too Long):** System prompts user to shorten or upload as PDF; optionally supports chunking if implemented.
- **A2 (Non-English Detected):** System warns that MVP is English-only and may produce unreliable output.

Scenario 7: Use Browser Extension to Analyze the Current Webpage

Primary Actor: End User

Supporting Actors: Browser Extension, Backend API, Agreemind Web/Mobile App

Entry Condition: User is viewing a webpage likely containing ToS/Privacy Policy; extension is installed and enabled.

Exit Condition (Success): A report is created and a badge/result is shown in the extension popup, with a link to open the full report in the app.

Main Flow:

1. User clicks the extension icon on the current page.
2. Extension extracts page content (e.g., visible text or DOM text) and captures the page URL.

3. Extension sends text + URL to backend to start an Analysis Job.
4. Backend processes text (segmentation + summary + risks + dates).
5. Extension shows a **risk badge** (e.g., low/medium/high) and a link to open the detailed report.

Alternative/Exception Flows:

- **A1 (Extraction Blocked):** If the page blocks scripts or extraction fails, extension prompts user to copy/paste text into the app.
- **A2 (Very Large Page):** Extension sends only relevant sections or truncates with warning, or instructs user to open in-app capture.

Scenario 8: Review Risk Highlights and Inspect Supporting Evidence

Primary Actor: End User

Supporting Actors: Agreemind UI, Backend API

Entry Condition: A completed report exists and is opened.

Exit Condition (Success): User views risk highlights and understands why they were flagged.

Main Flow:

1. User opens an Analysis Report.
2. System displays overall summary and risk categories (e.g., data sharing, auto-renewal).
3. User selects a risk category to filter highlights.
4. System scrolls to each relevant clause and highlights the exact text span.
5. User taps “Why flagged?” to view a short explanation and (optionally) a confidence indicator.

Alternative/Exception Flows:

- **A1 (Low Confidence):** If confidence is low, the UI shows a caution label and encourages manual review.

Scenario 9: Extract Dates/Obligations and Set a Reminder

Primary Actor: End User

Supporting Actors: Agreemind UI, Backend API, Notification/Reminder Service

Entry Condition: Report contains extracted dates/obligations.

Exit Condition (Success): A reminder is scheduled and visible in the user’s reminder list.

Main Flow:

1. User opens “Key Dates & Obligations” in the report.
2. System lists extracted items (e.g., cancellation window, renewal date) with source clause links.
3. User selects an item and taps “Set Reminder.”
4. User chooses reminder time (e.g., 7 days before).
5. System stores the reminder and confirms success.

Alternative/Exception Flows:

- **A1 (Ambiguous Date):** If the date is uncertain, system labels it as estimated and asks user to confirm/edit before saving.

Scenario 10: Save an Agreement to the Vault and Search Later

Primary Actor: End User
Supporting Actors: Agreemind UI, Backend API, Storage Service

Entry Condition: User has at least one completed report.

Exit Condition (Success): Agreement is saved and appears in the vault; search returns relevant results.

Main Flow:

1. User chooses “Save to Vault” from the report screen.
2. User optionally enters metadata (service name, category, tags).
3. System stores the agreement + report under the user account.
4. Later, user opens Vault and searches by keyword/tag/service name.
5. System displays matching agreements and user opens one report.

Alternative/Exception Flows:

- **A1 (Storage Limit Reached):** System warns user and offers deletion/upgrade (if applicable) or blocks save.

Scenario 11: Share a Webpage to Agreemind from a Mobile Device

Primary Actor: End User

Supporting Actors: Mobile OS Share Sheet (iOS/Android), Agreemind Mobile App, Backend API, Document Processing Service, LLM Service

Entry Condition: User is viewing a webpage (e.g., ToS/Privacy Policy) in a mobile browser or another app; Agreemind is installed and registered as a share target.

Exit Condition (Success): A new Analysis Report is created and displayed in Agreemind.

Main Flow:

1. User taps **Share** on the current page/app.
2. User selects **Agreemind** from the share sheet.
3. The OS launches Agreemind and passes shared content (typically a **URL**, and optionally selected text if available).
4. Agreemind shows an “Import from Share” screen and asks the user to confirm analysis.
5. Agreemind sends the URL (and any included text) to the backend to create an Analysis Job.
6. Backend retrieves and/or processes the content:
    - If the backend can fetch the URL content, it extracts readable text.
    - If only text was shared, backend analyzes the provided text directly.
7. Backend runs segmentation + summarization + risk detection + date extraction.
8. Agreemind displays the completed report and optionally offers “Save to Vault.”

Alternative/Exception Flows:

- **A1 (URL Fetch Not Allowed / Paywalled / Blocked):** App prompts the user to paste the text manually or open the page in a supported browser mode.
- **A2 (Non-English Detected):** System warns English-only limitation and proceeds only if user confirms.
- **A3 (Very Long Content):** System truncates or analyzes the most relevant sections (with a warning), or requests the user to upload a PDF instead.
- **A4 (No Network):** The app queues the analysis request and submits when connectivity returns (if you implement offline queueing); otherwise it asks user to retry later.

Scenario 12: Compare Two Versions of an Agreement

Primary Actor: End User

Supporting Actors: Agreemind UI, Backend API, Comparison Service

Entry Condition: User has two agreements (or two versions) available.

Exit Condition (Success): A “What changed?” view is shown with added/removed/modified clauses.

Main Flow:

1. User selects “Compare Versions” and chooses Version A and Version B.
2. Backend aligns clauses and computes differences.
3. System presents changes grouped by risk category and highlights new/modified risky clauses.

Alternative/Exception Flows:

- **A1 (Alignment Fails):** System falls back to text-level diff and labels results as coarse.

Scenario 13: Delete an Agreement and Its Analysis Results

Primary Actor: End User

Supporting Actors: Agreemind UI, Backend API, Storage Service

Entry Condition: User is viewing an agreement in the vault.

Exit Condition (Success): Agreement and associated artifacts are removed; vault list updates.

Main Flow:

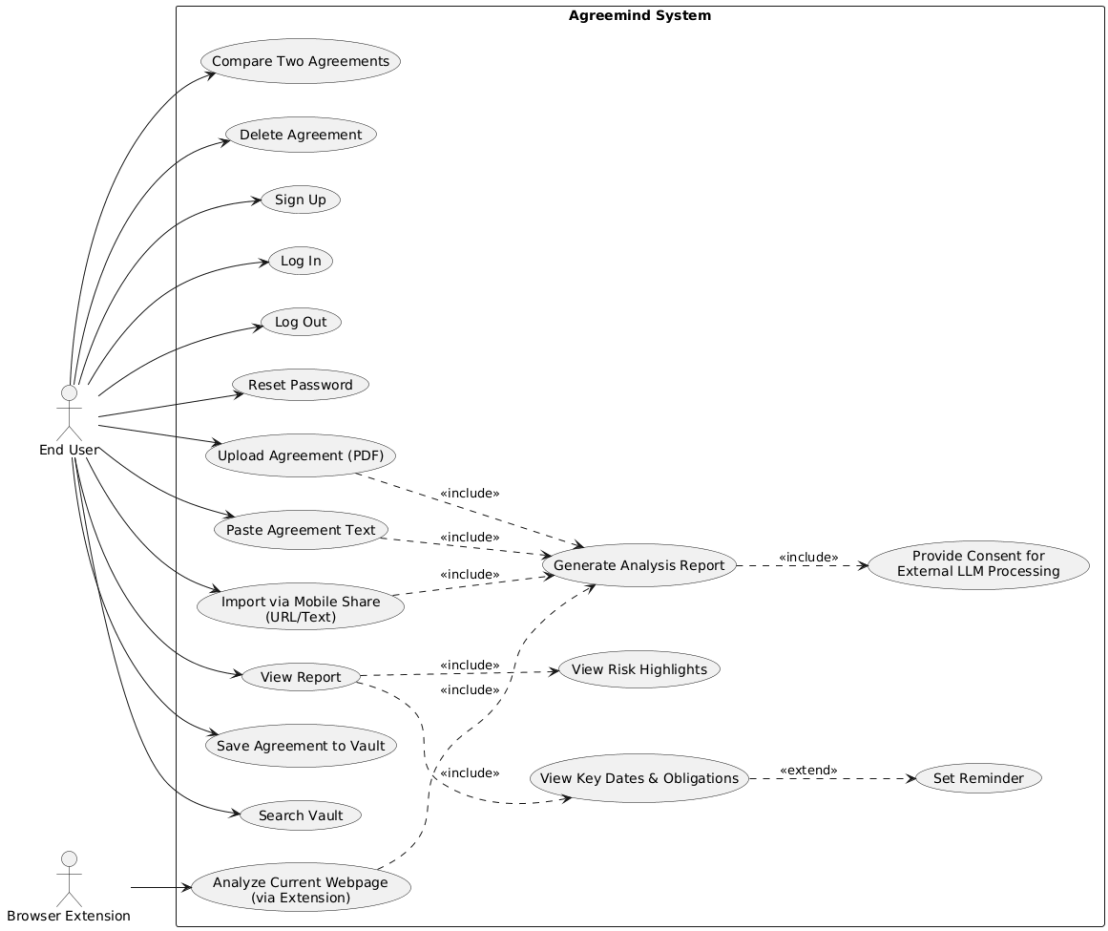
1. User selects “Delete” and confirms.
2. Backend deletes stored document, extracted text, report outputs, and reminders (as applicable).
3. UI confirms deletion and returns to vault list.

Alternative/Exception Flows:

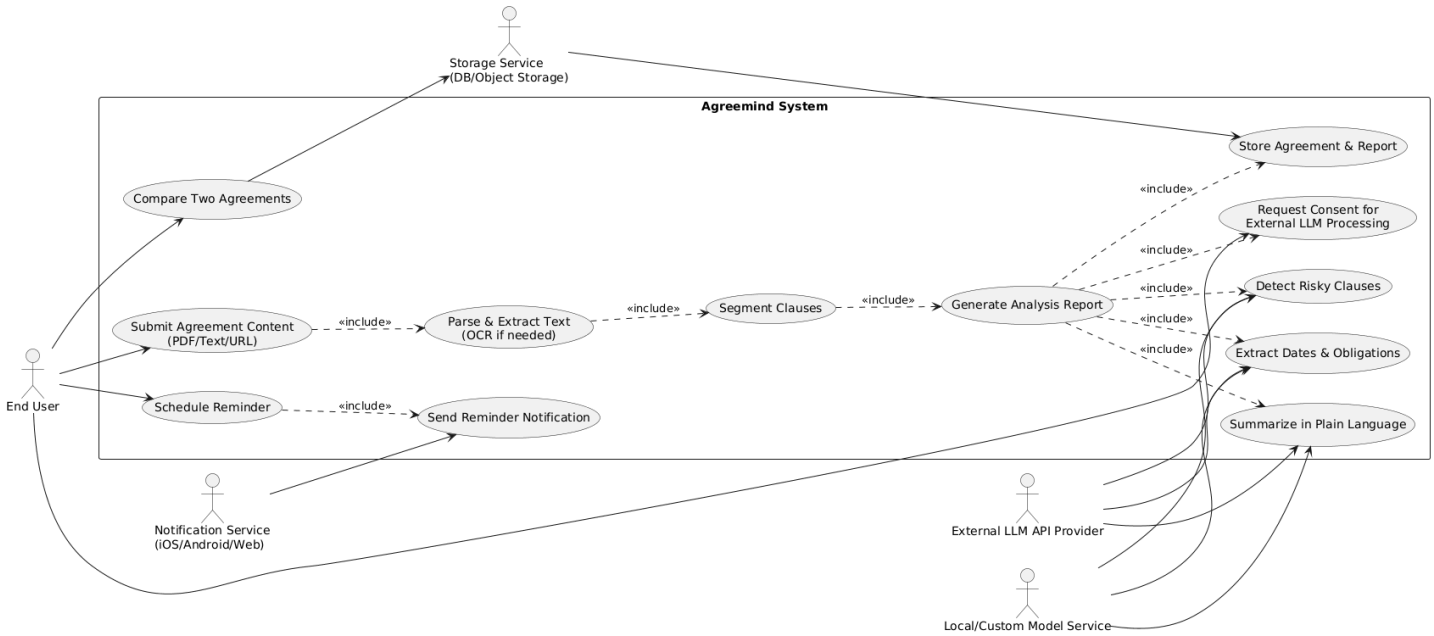
- **A1 (Network Failure):** UI shows “Delete pending” and retries, or asks user to retry.

3.5.2 Use-Case Models

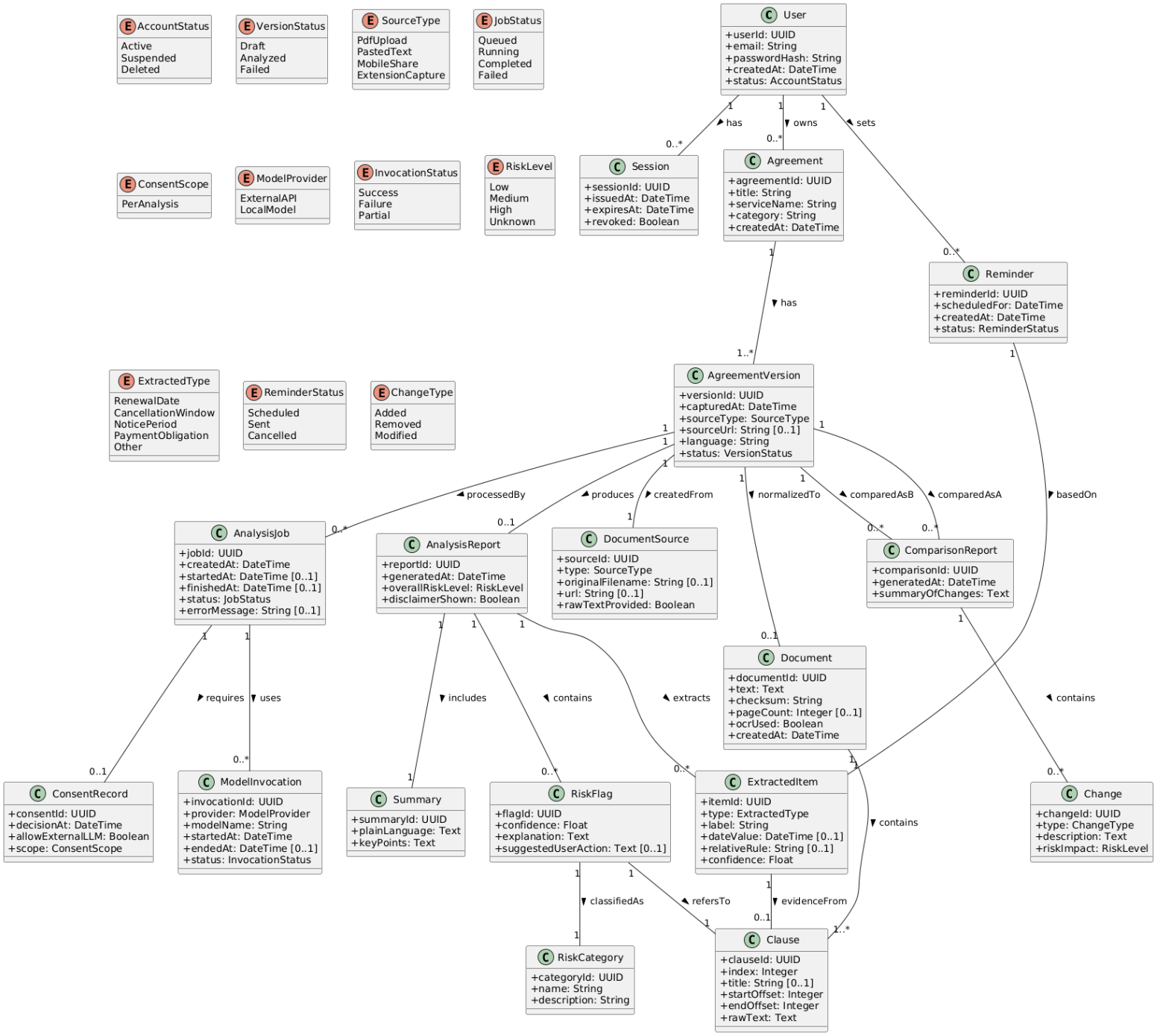
User-facing



System + external services



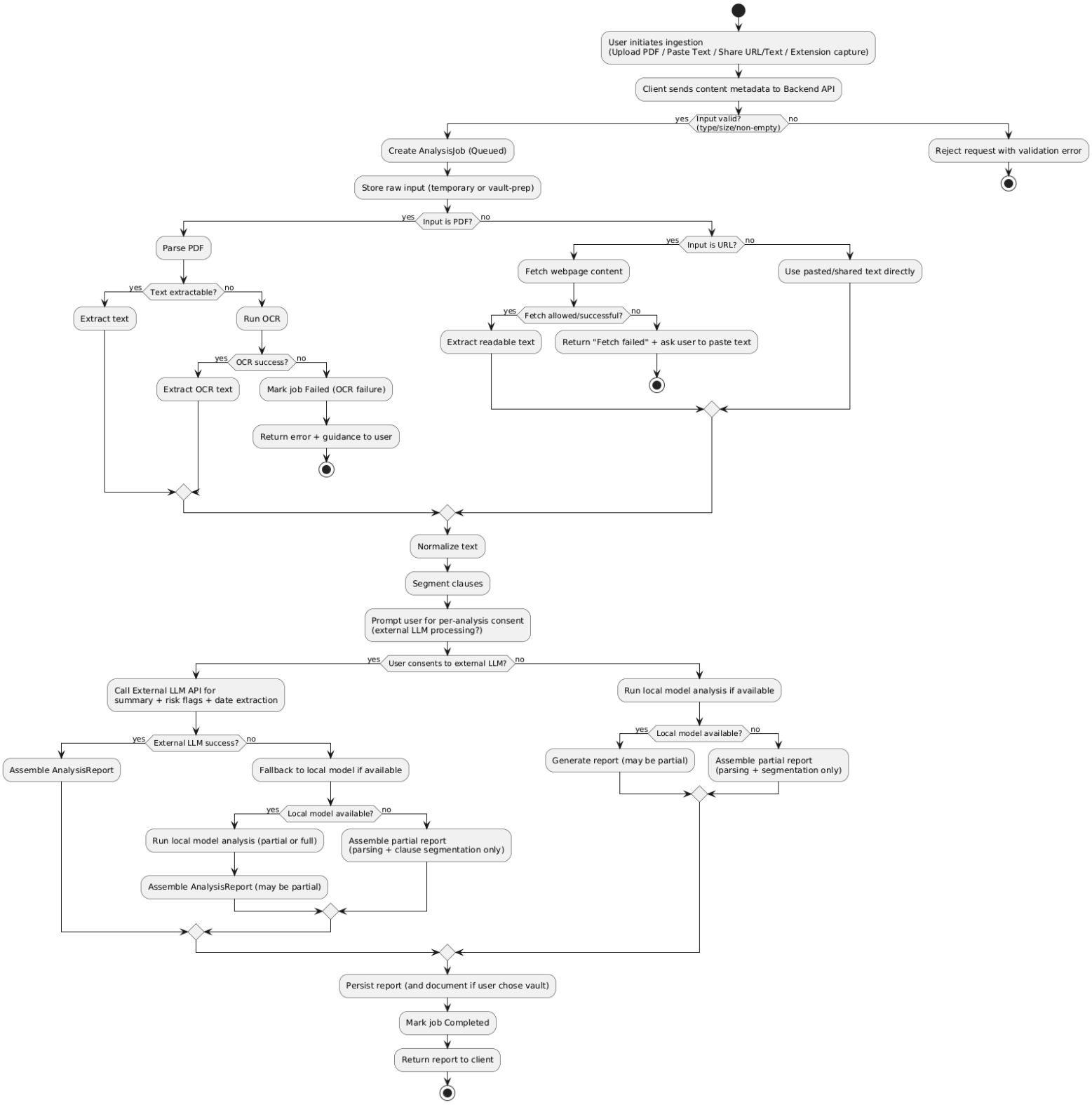
3.5.3 Object and Class Model



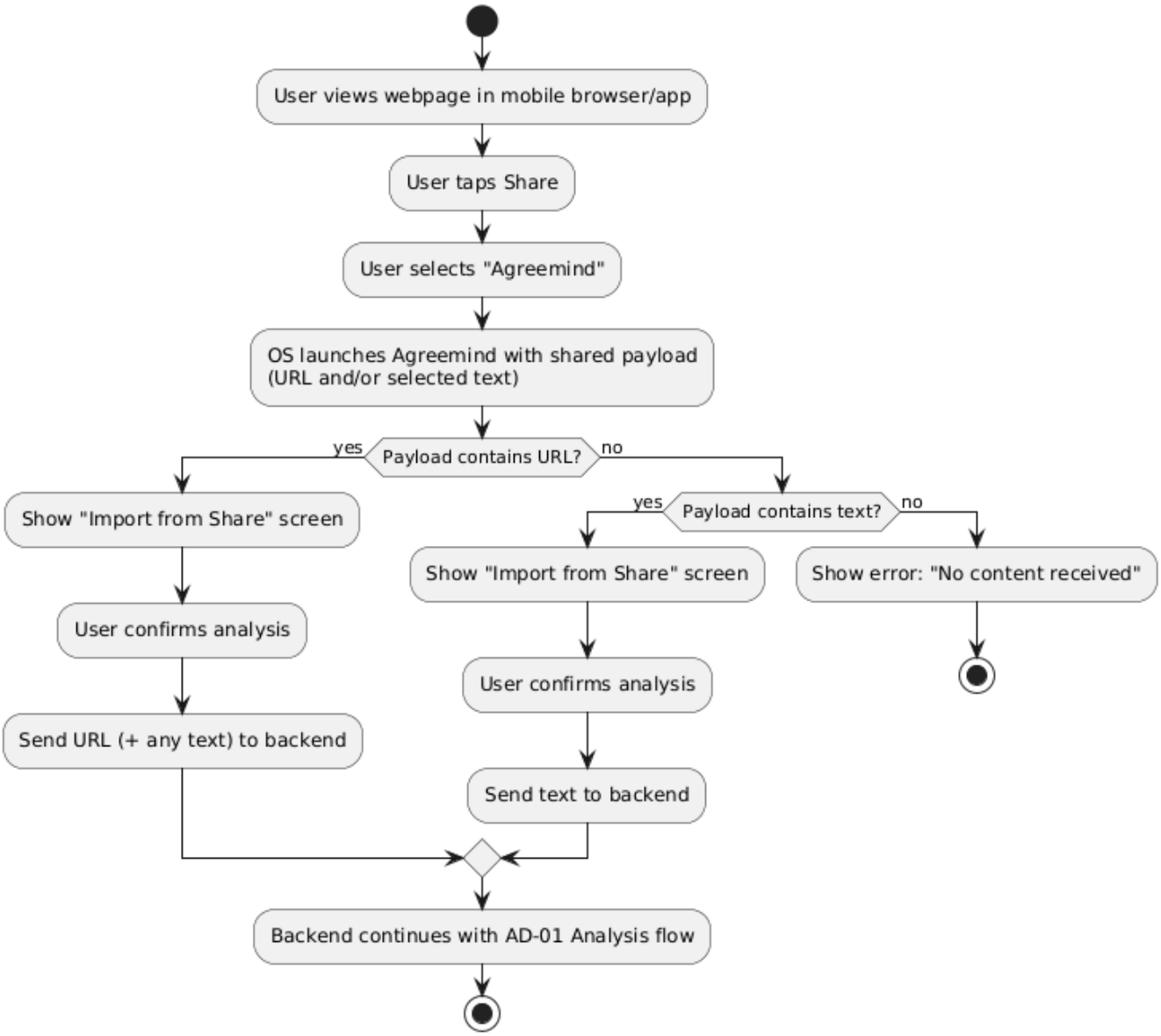
3.5.4 Dynamic Models

3.5.4.1 Activity Diagrams

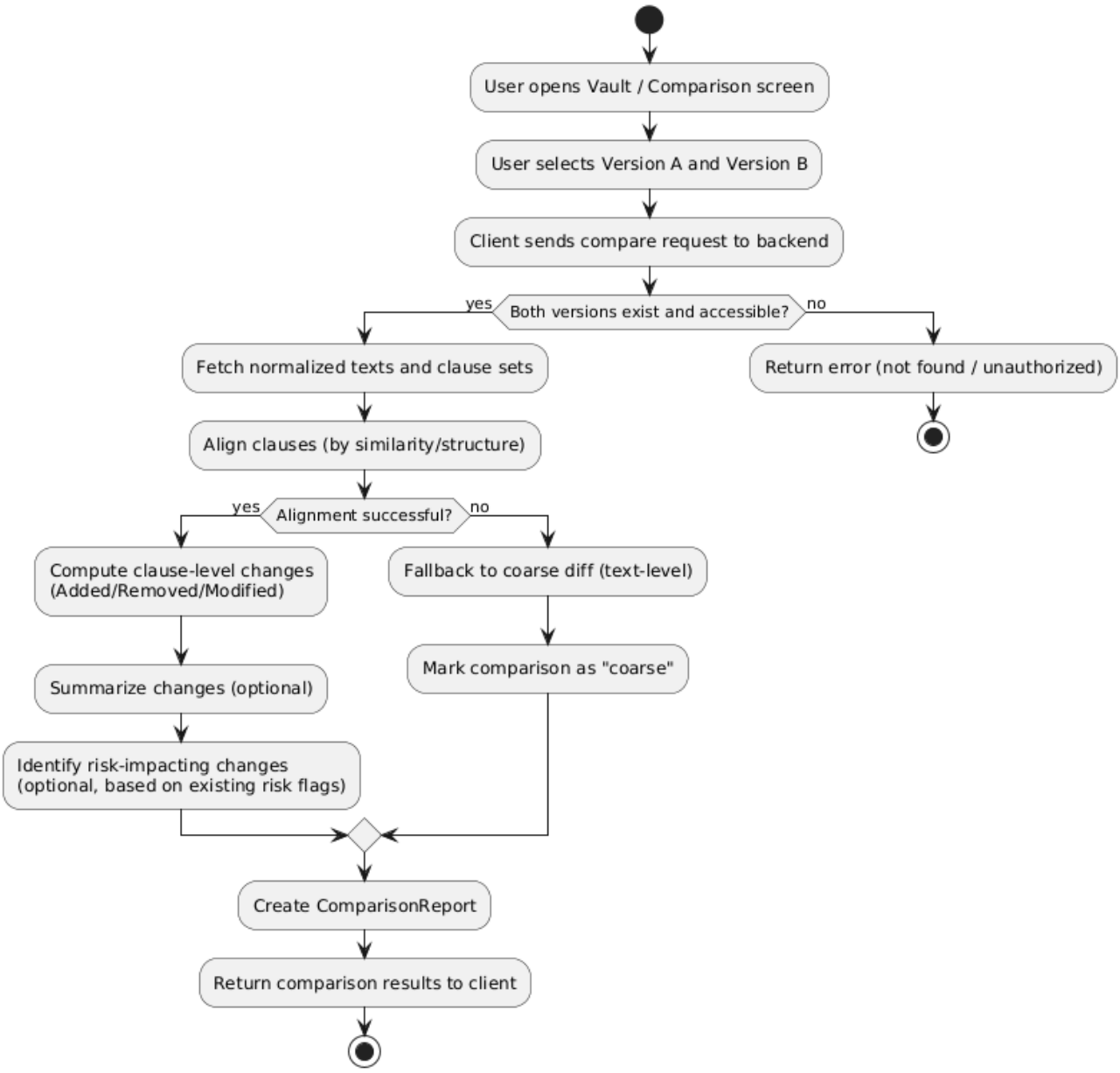
3.5.4.1.1 Agreement Analysis Flow (Upload / Paste / Share / Extension)



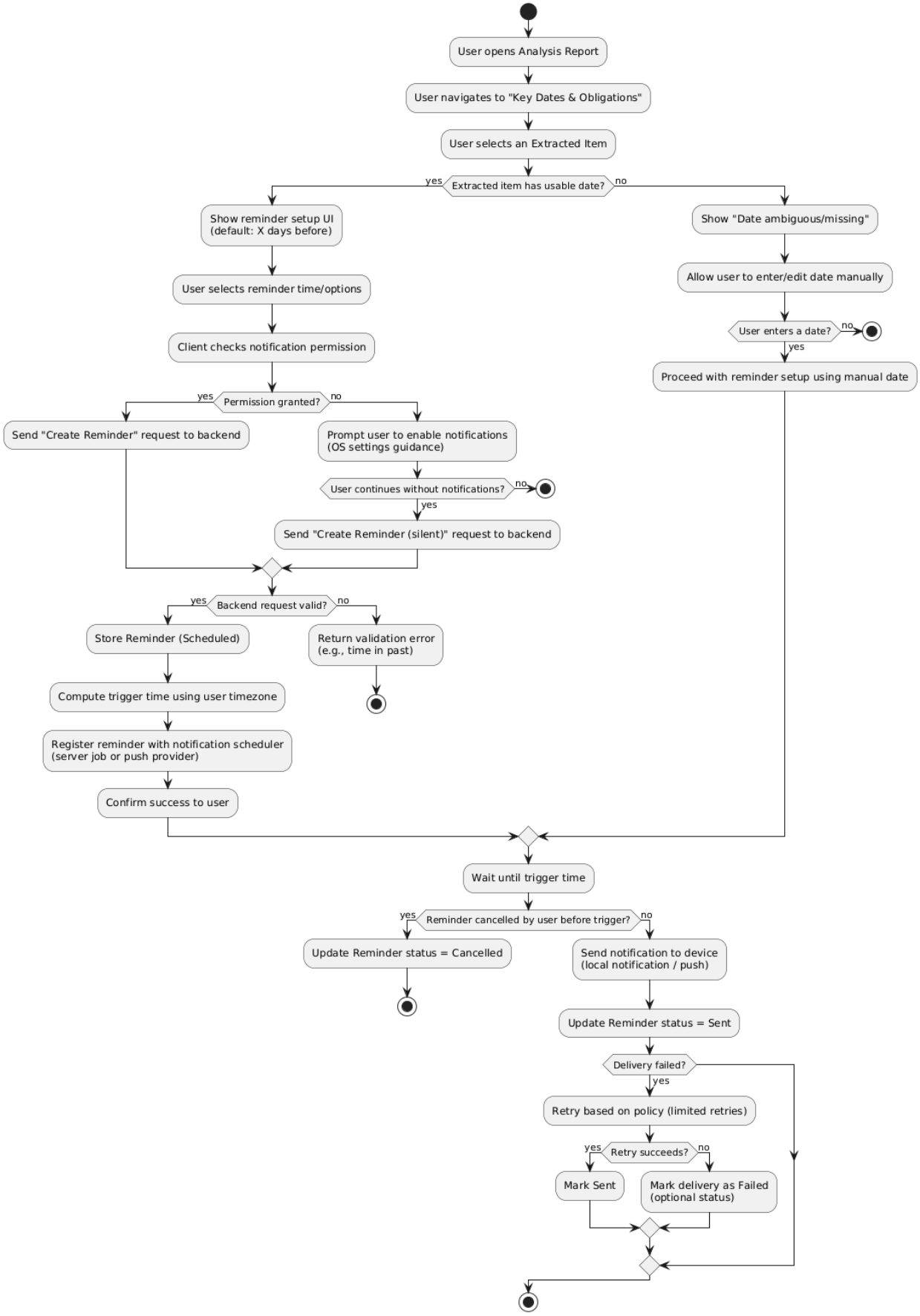
3.5.4.1.2 Mobile Share to Agreemind



3.5.4.1.3 Compare Two Agreements

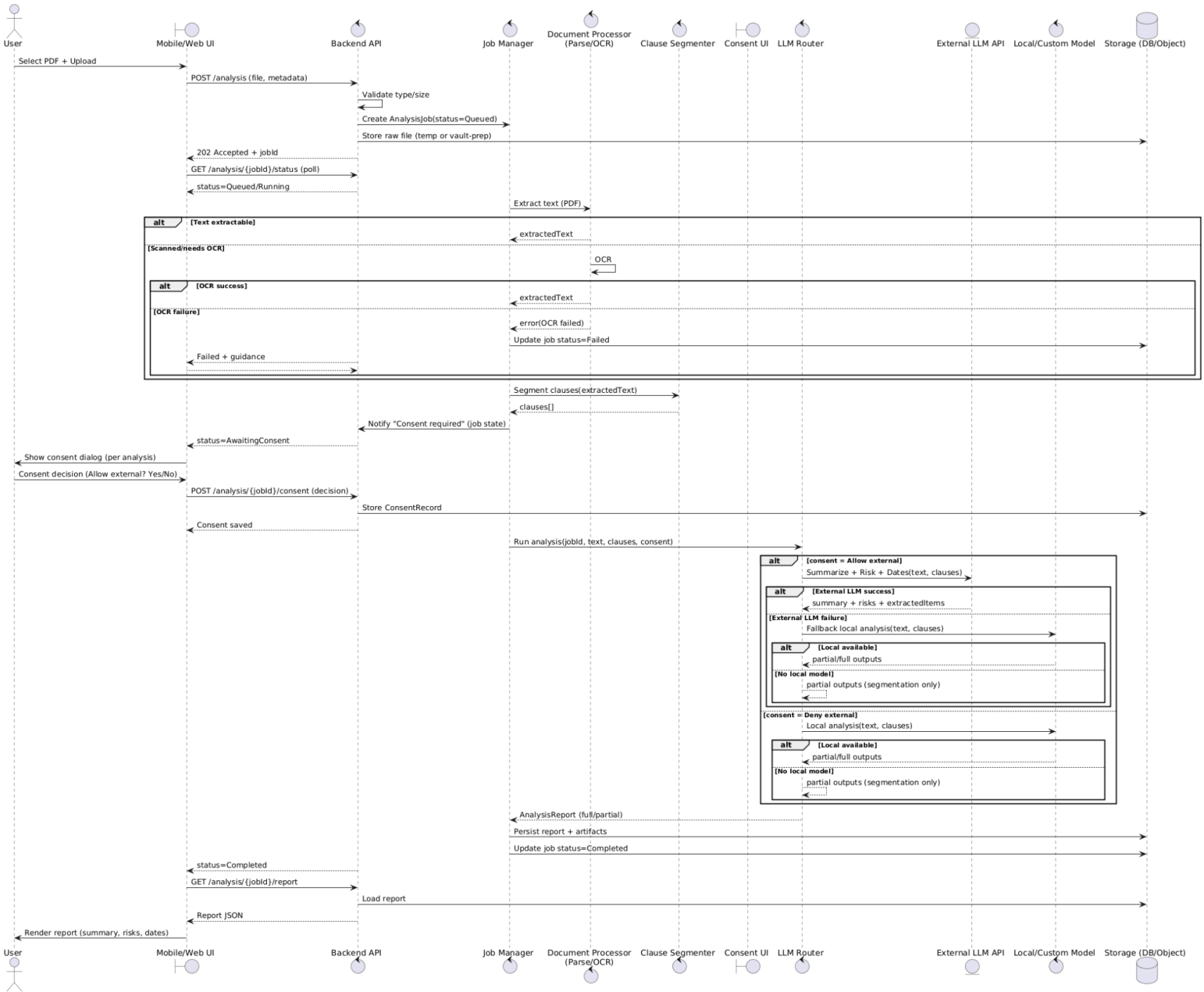


3.5.4.1.4 Reminder Scheduling and Notification Delivery

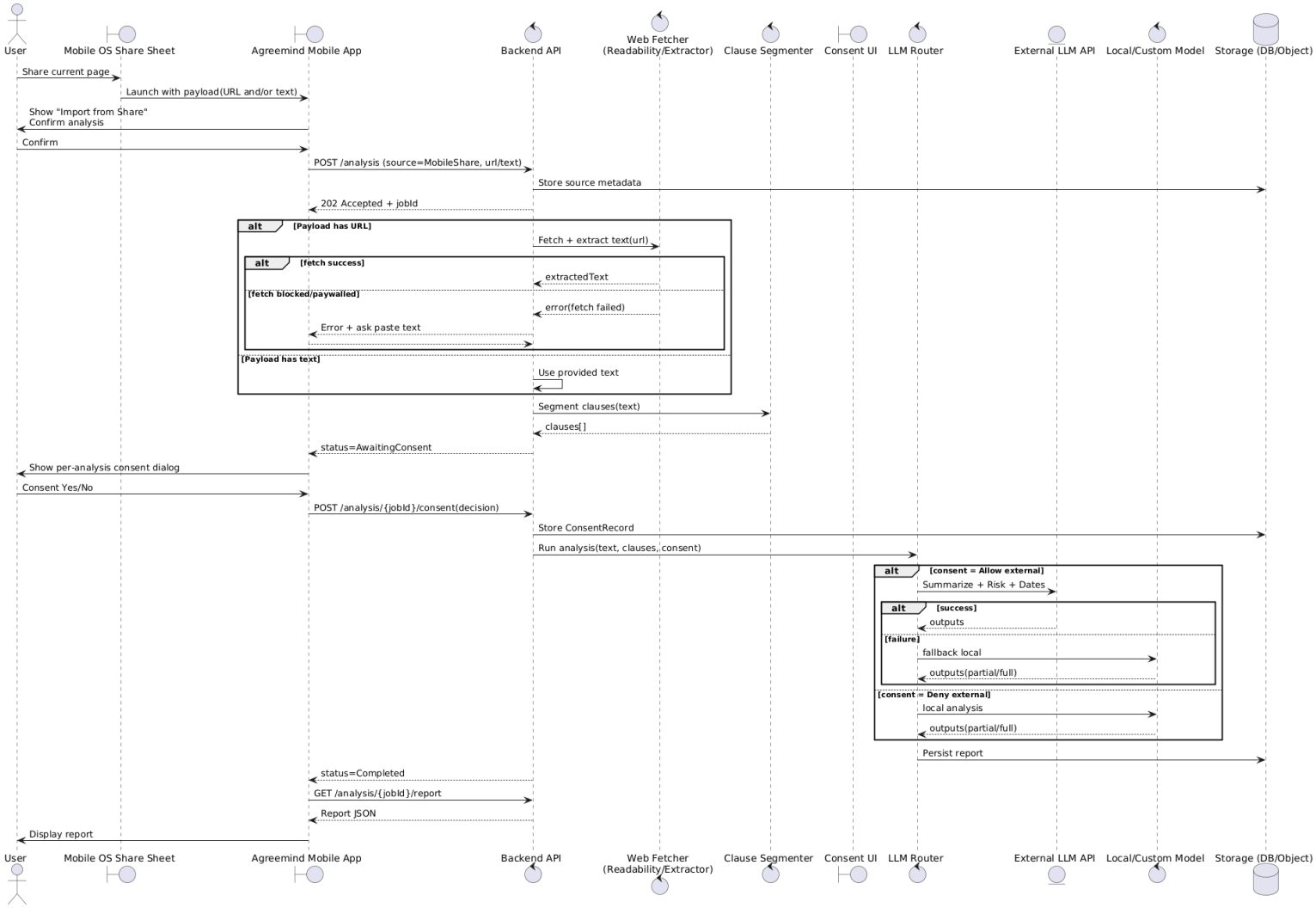


3.5.4.2 Sequence Diagrams

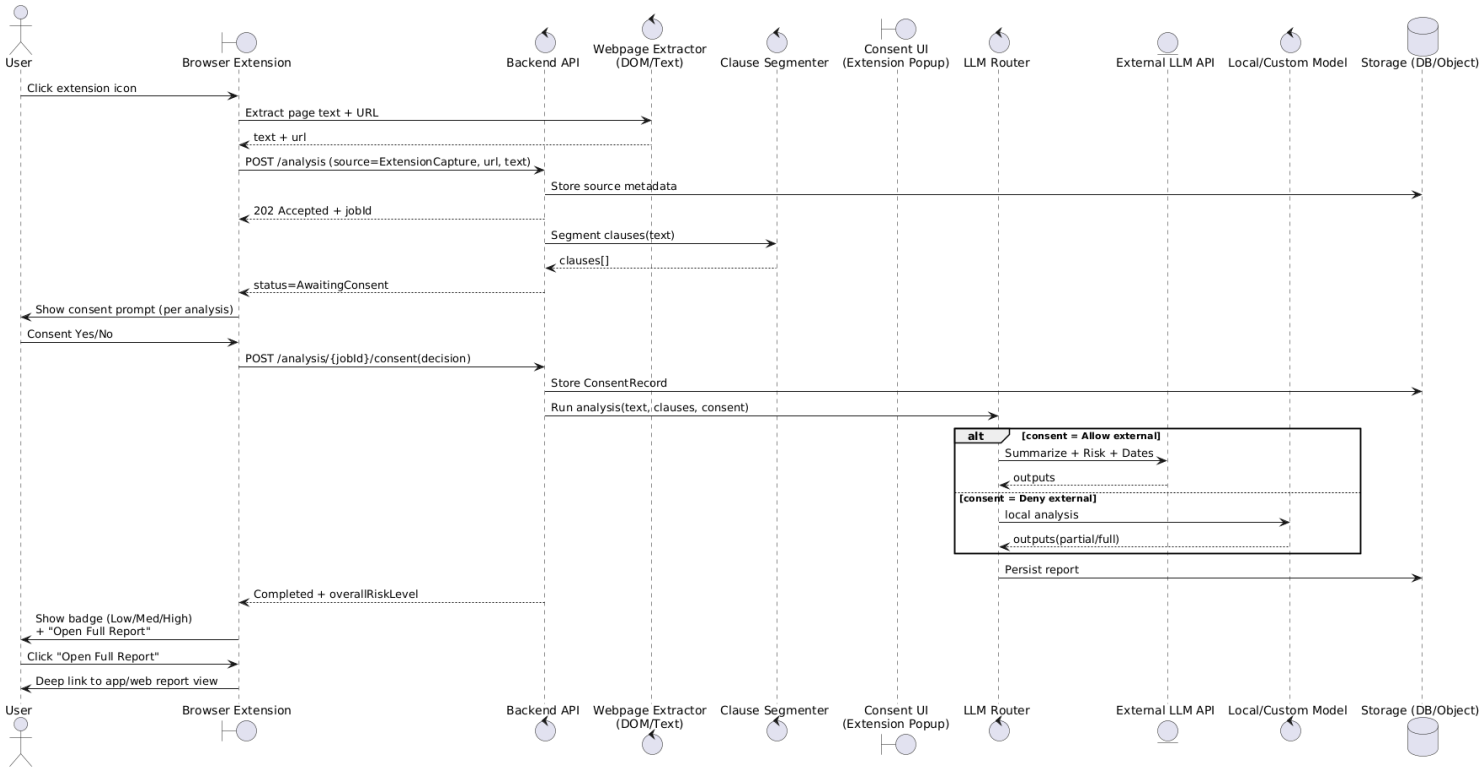
3.5.4.2.1 Upload PDF → Per-Analysis Consent → Report Generation



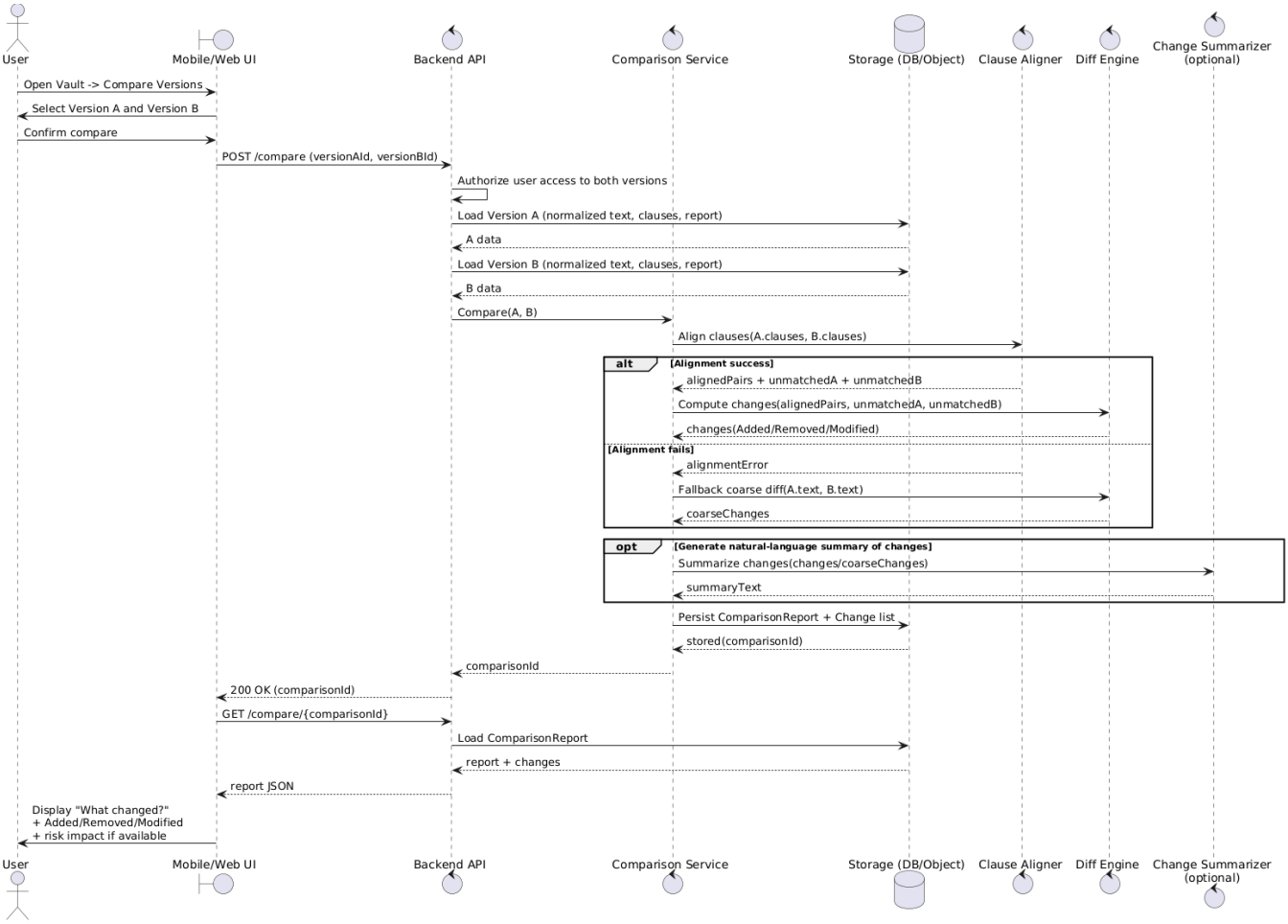
3.5.4.2.2 Mobile Share Sheet (URL/Text) → Backend Fetch → Consent → Report



3.5.4.2.3 Browser Extension → Quick Badge + Deep Link to Full Report

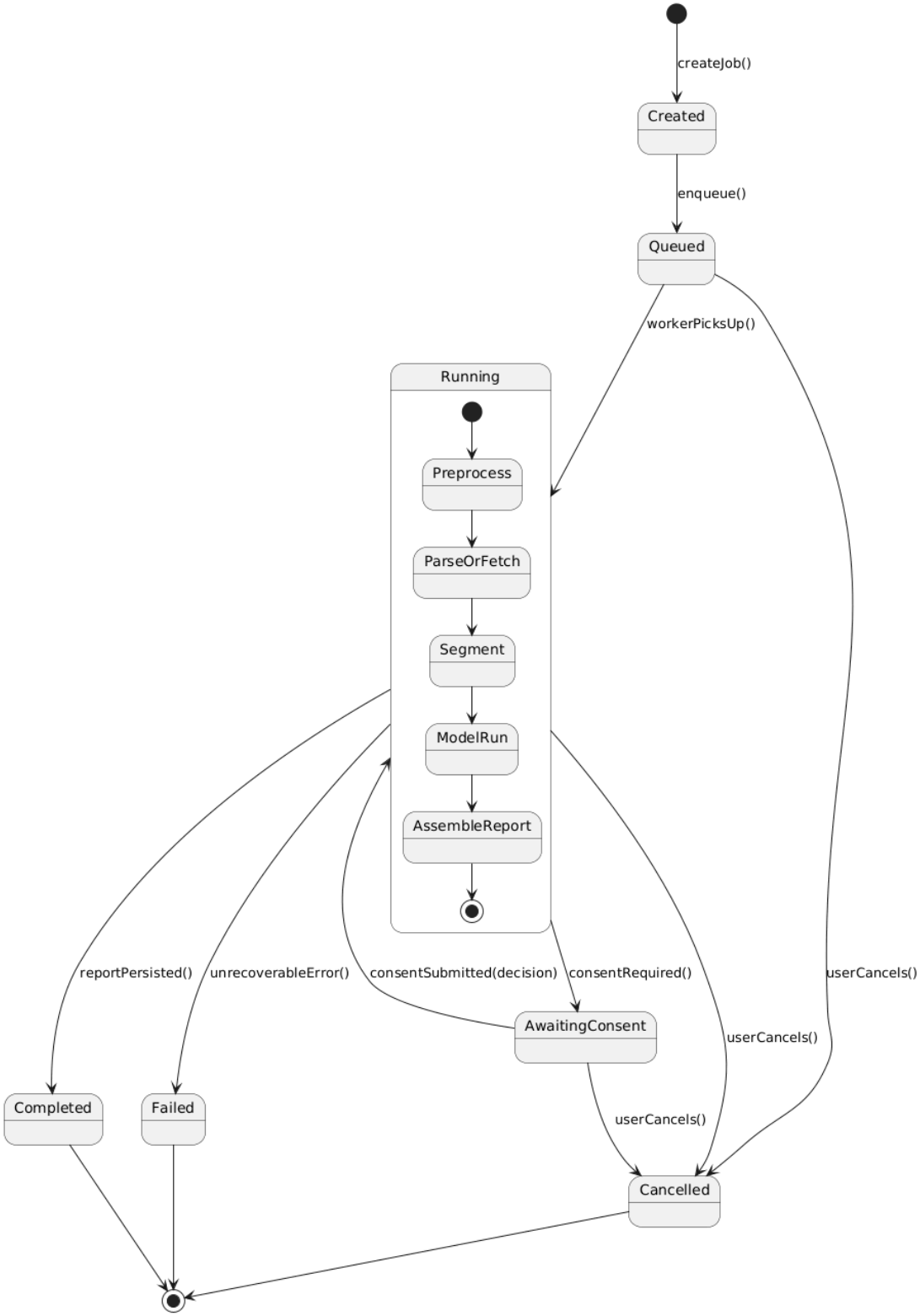


3.5.4.2.4 Compare Two Agreement Versions

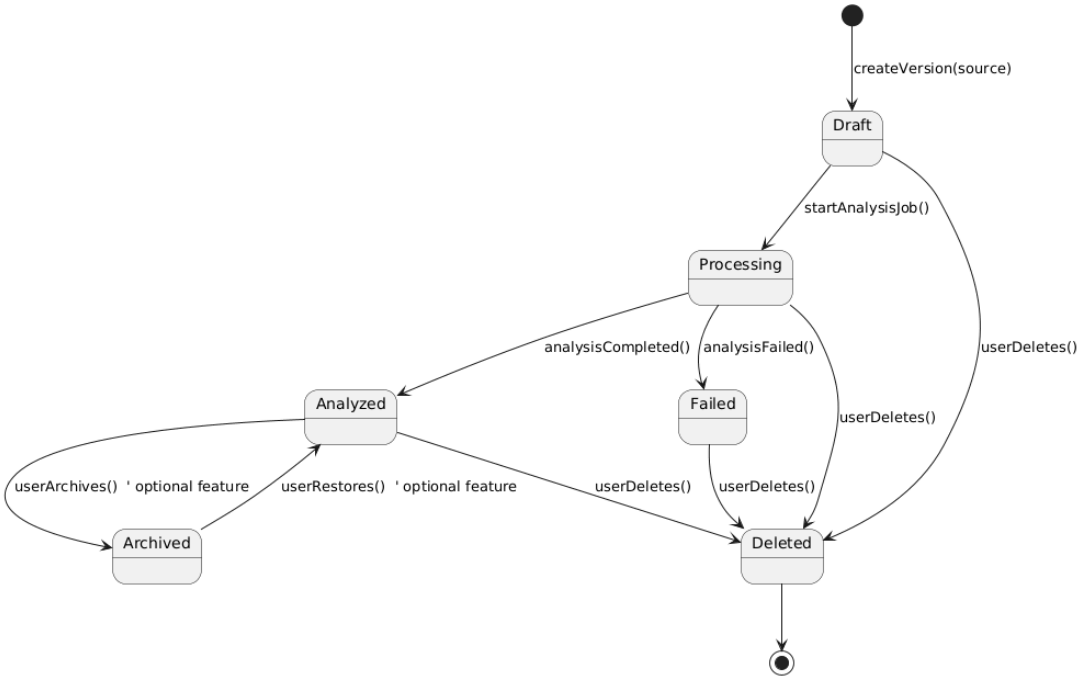


3.5.4.3 State Diagrams

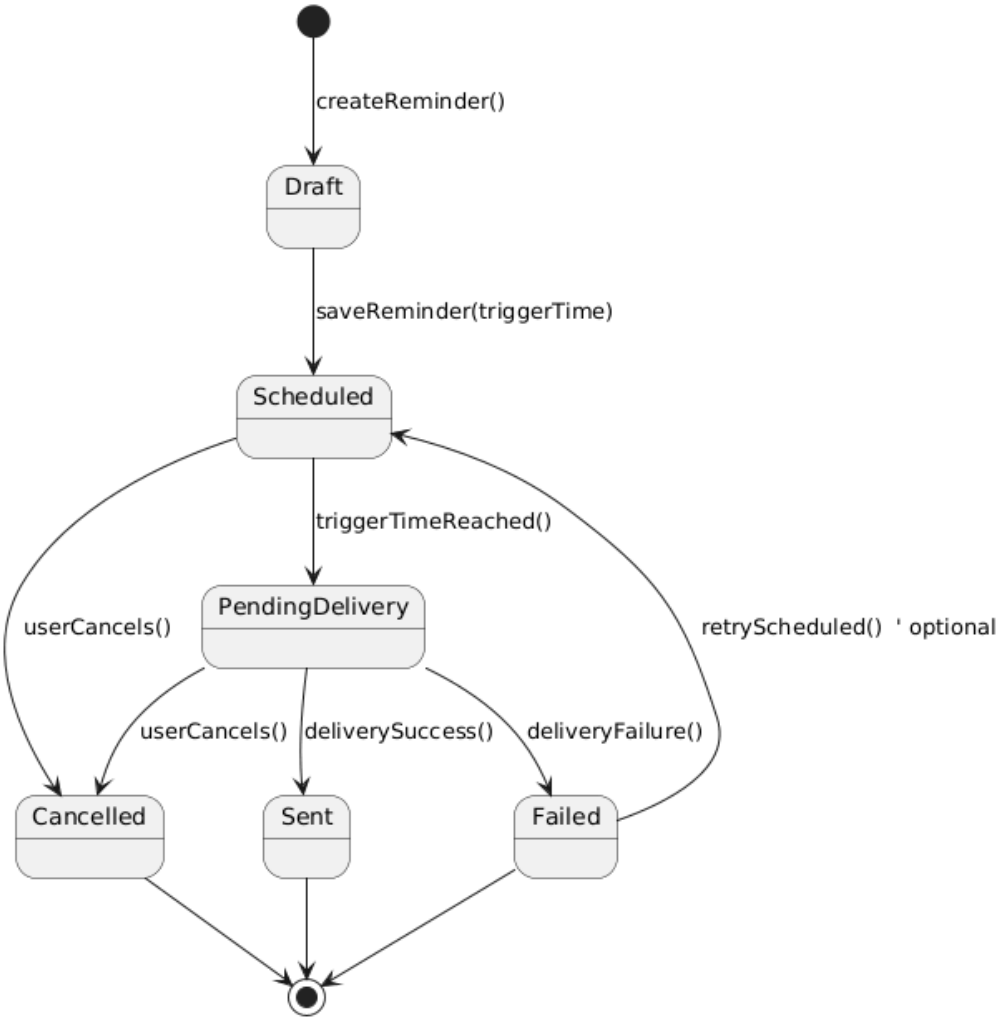
3.5.4.3.1 AnalysisJob State Diagram (core backend pipeline)



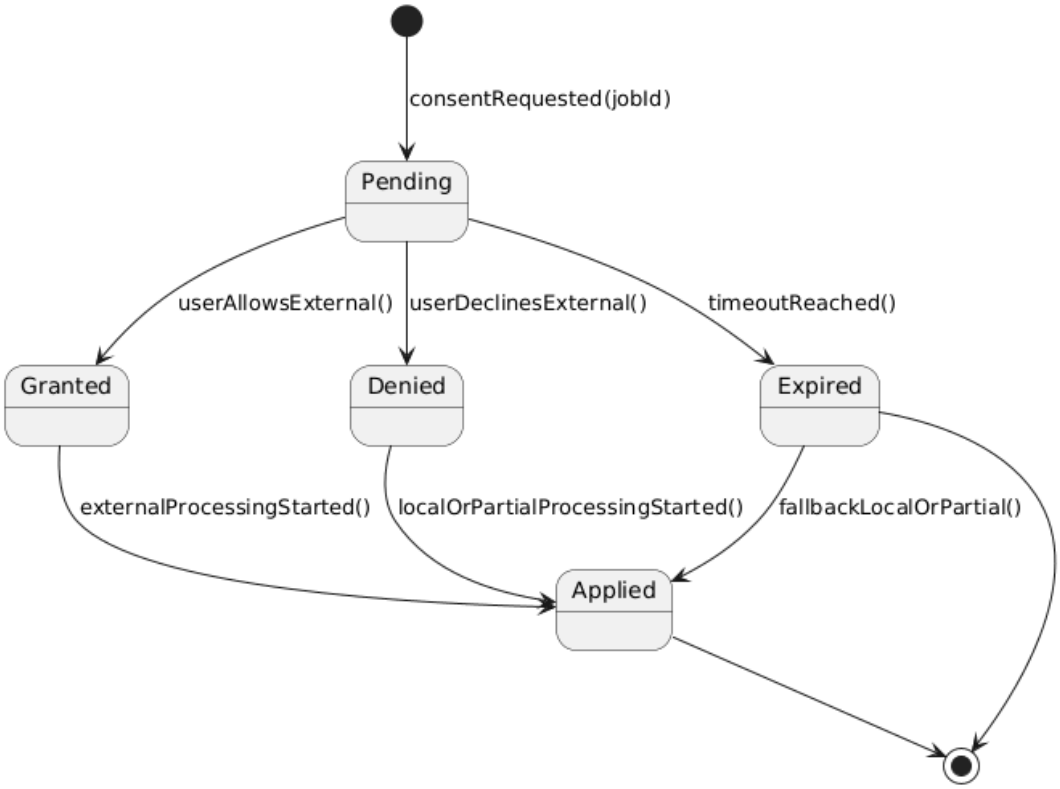
3.5.4.3.2 AgreementVersion State Diagram (vault item lifecycle)



3.5.4.3.3 Reminder State Diagram (notification pipeline)

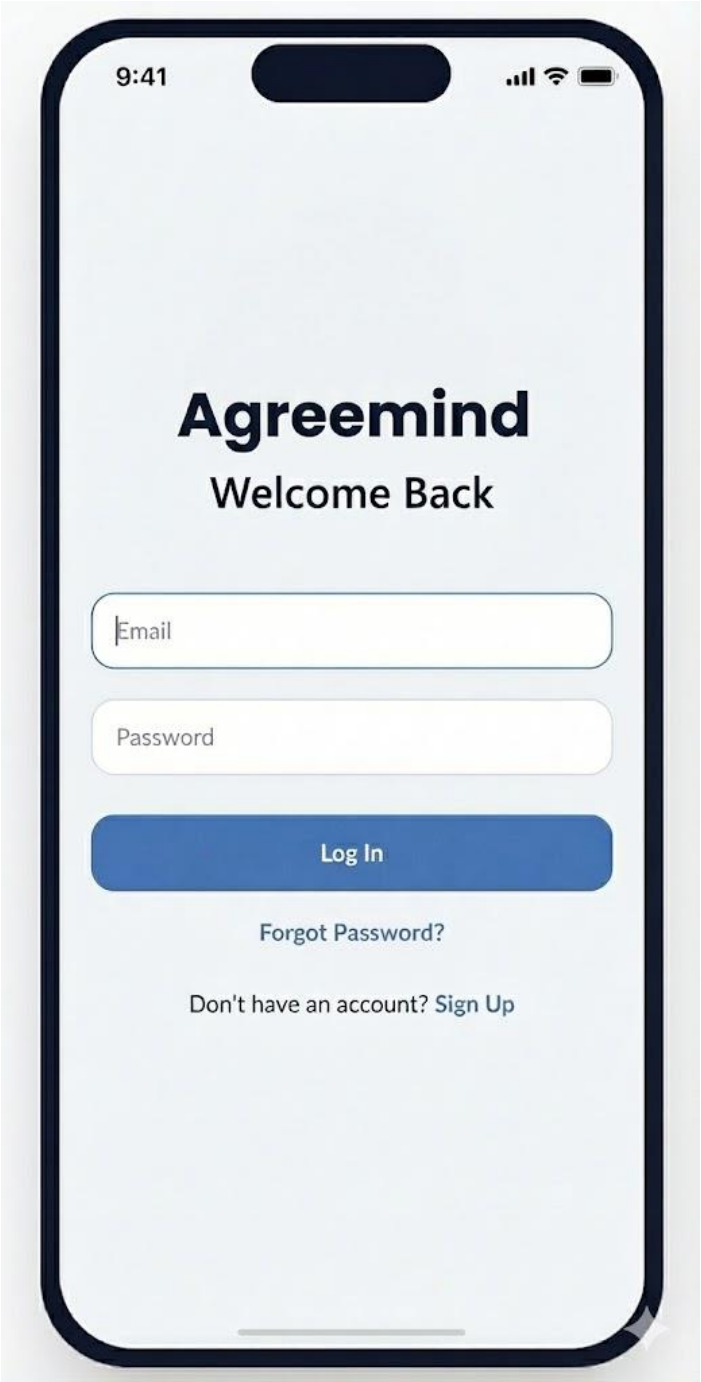


3.5.4.3.4 ConsentRecord State Diagram

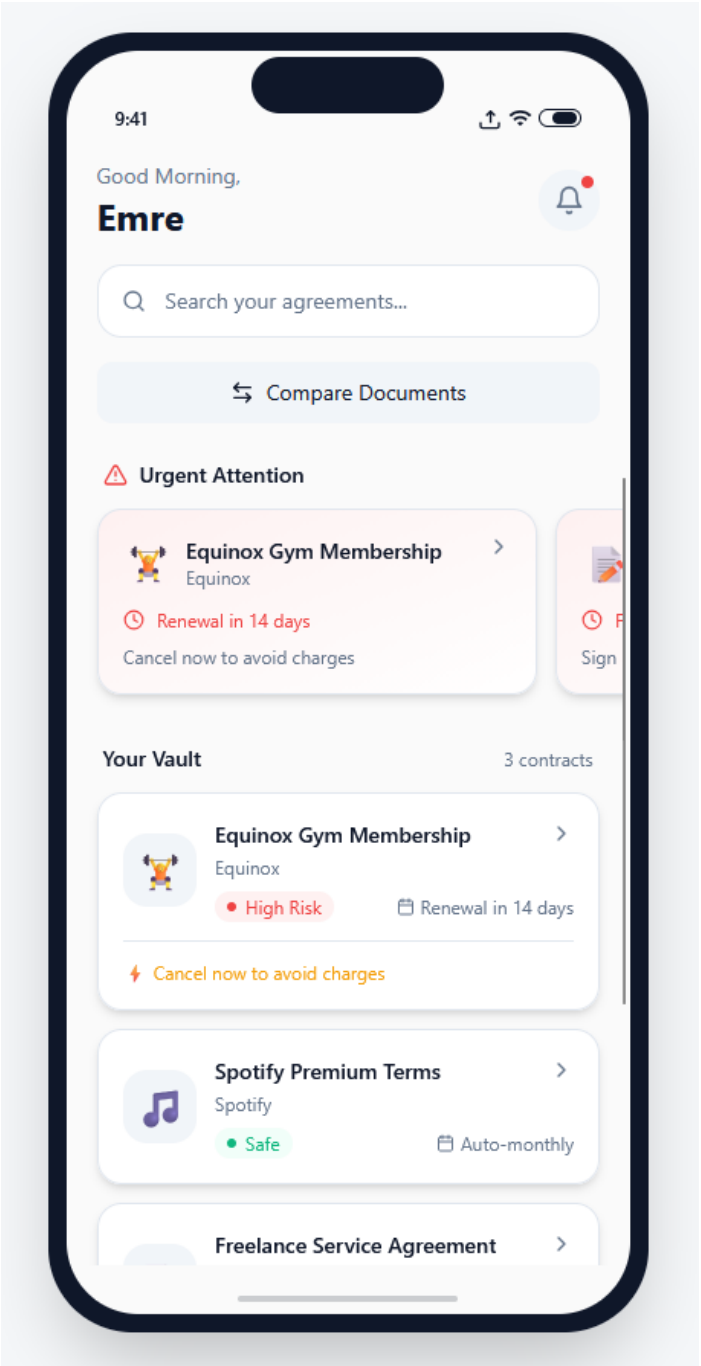


3.5.5 User Interface

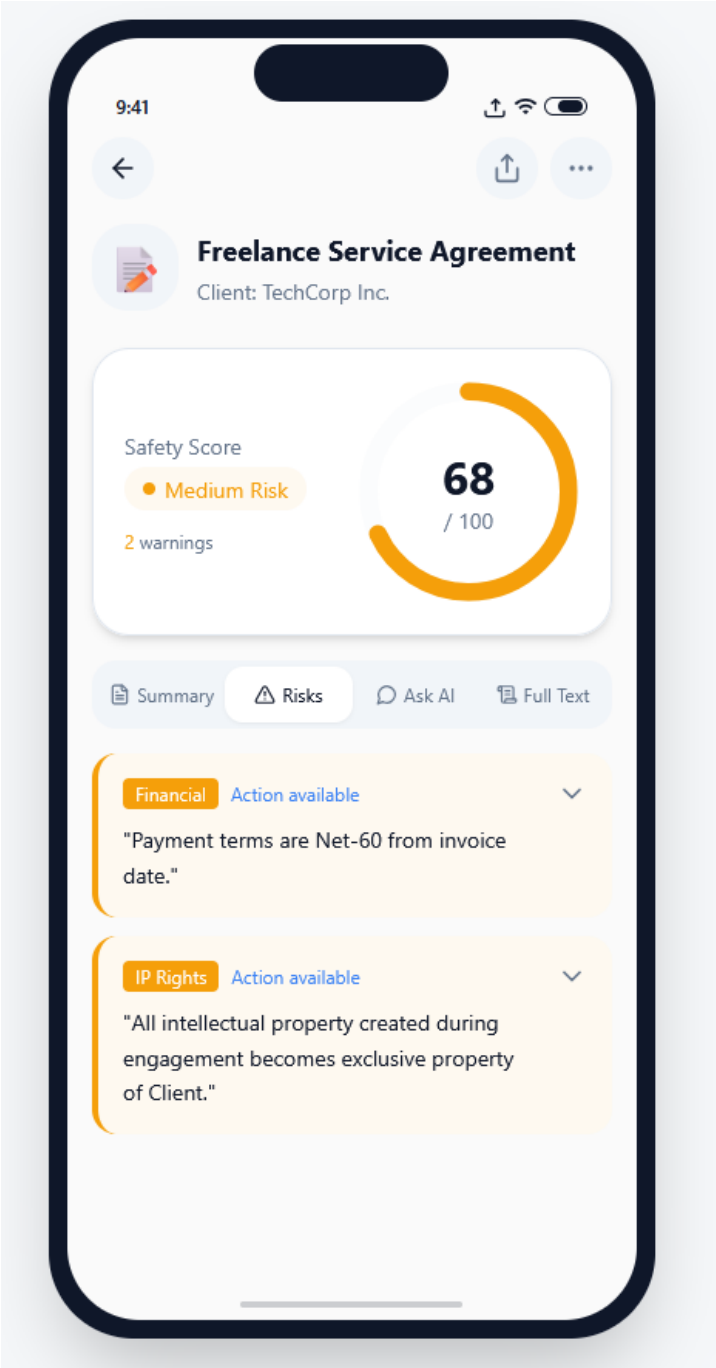
Login Page:



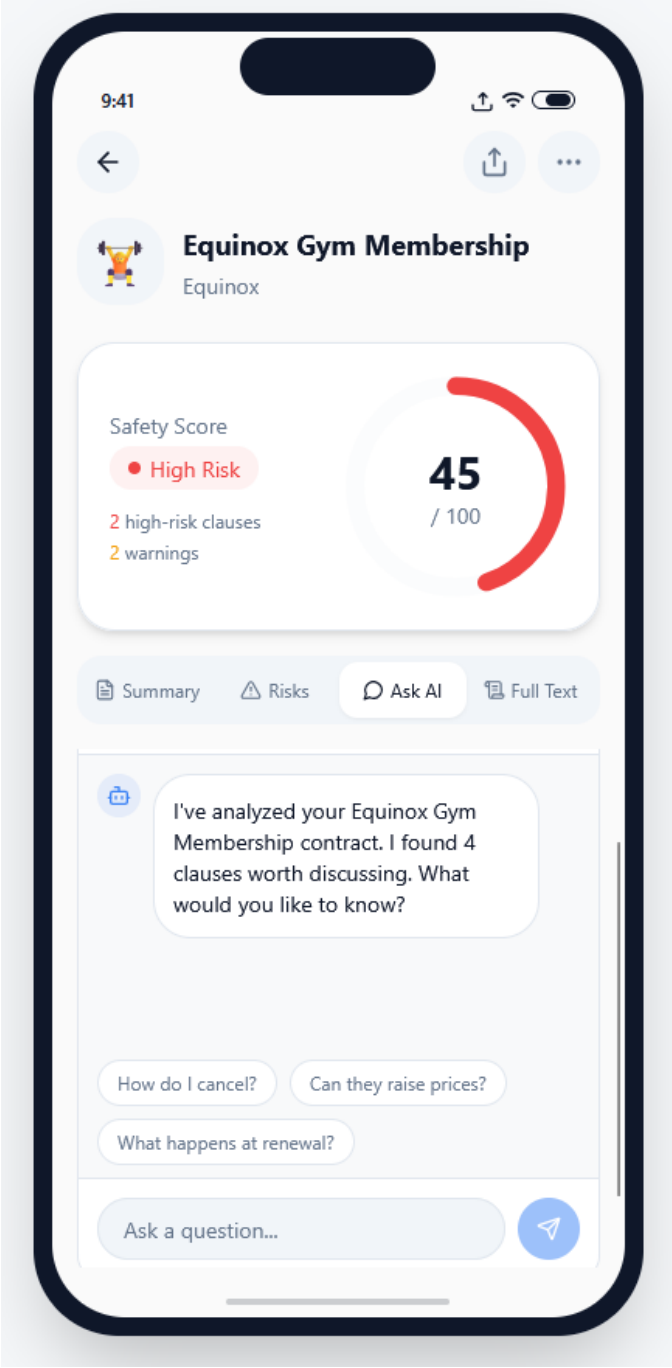
Homepage:


9:41
Good Morning,
Emre
Search your agreements...
Compare Documents
Urgent Attention
Equinox Gym Membership
Equinox
Renewal in 14 days
Cancel now to avoid charges
Sign
Your Vault
3 contracts
Equinox Gym Membership
Equinox
High Risk
Renewal in 14 days
Cancel now to avoid charges
Spotify Premium Terms
Spotify
Safe
Auto-monthly
Freelance Service Agreement

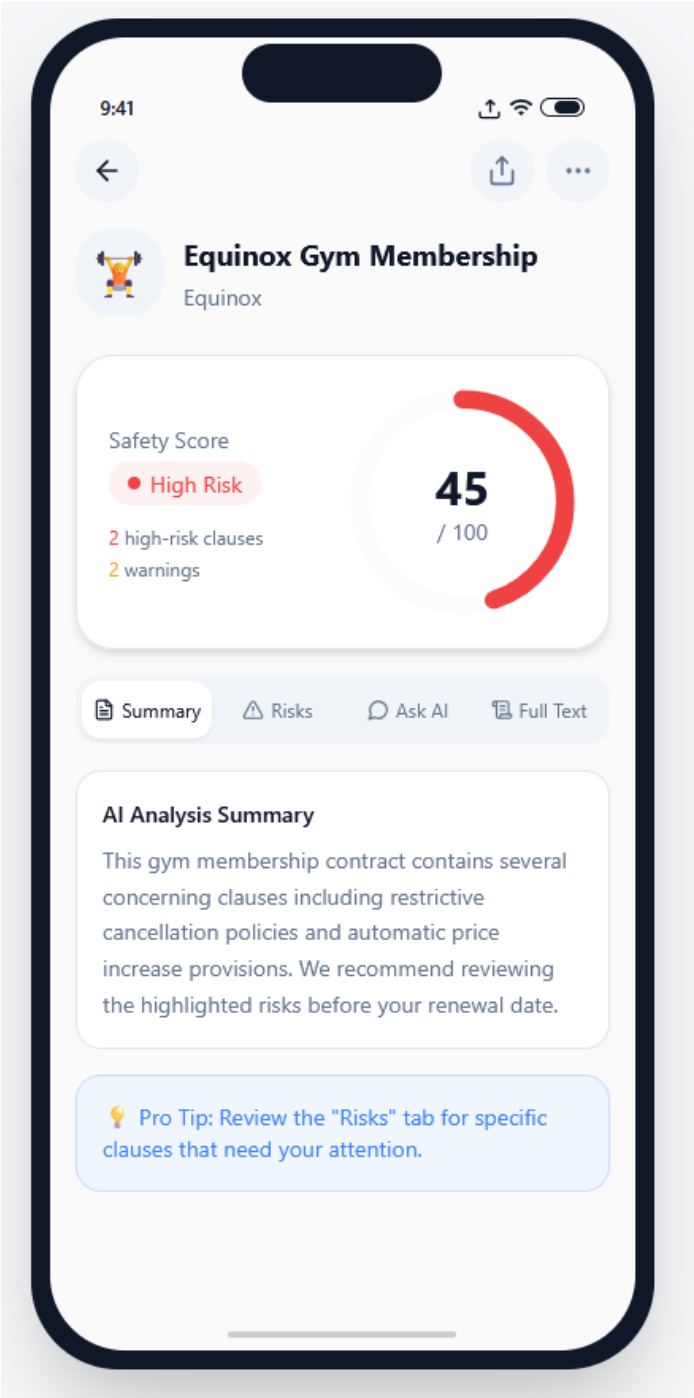
Contract Risk Analysis Page:



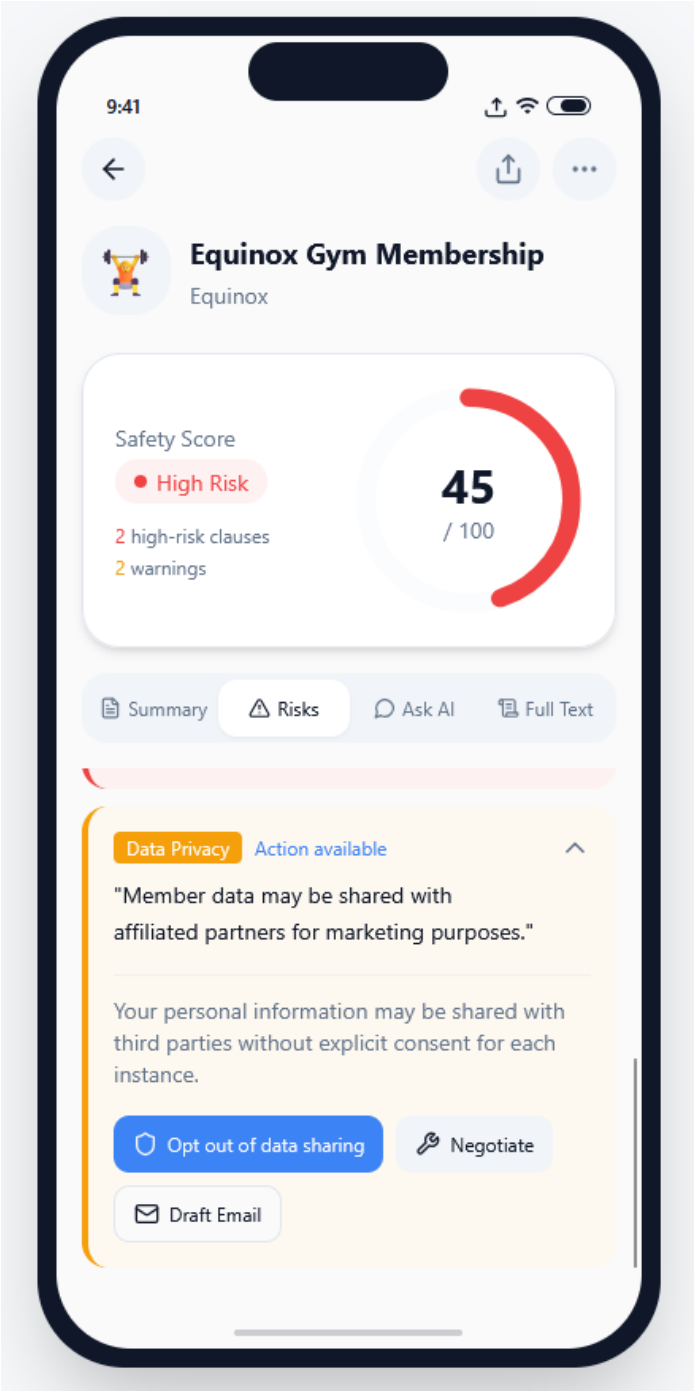
Document Query Page:



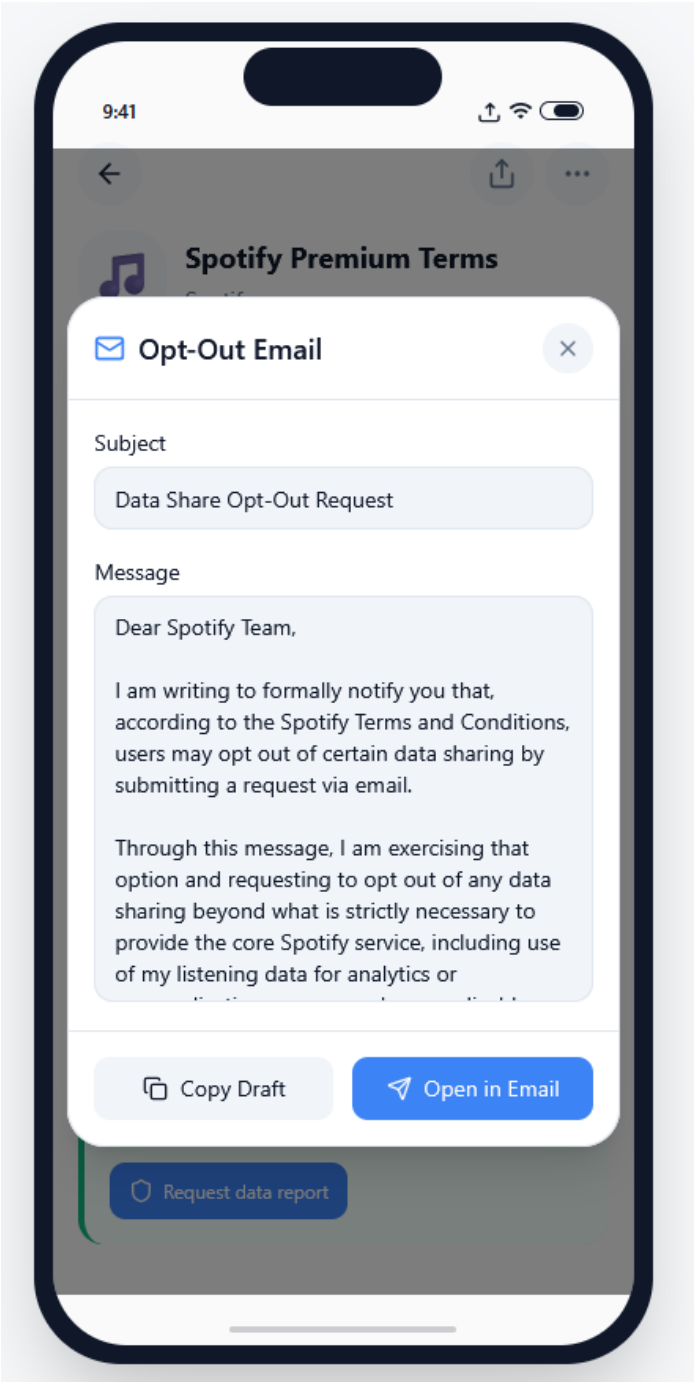
Contract Summary Page:



Actionable Clause Page:

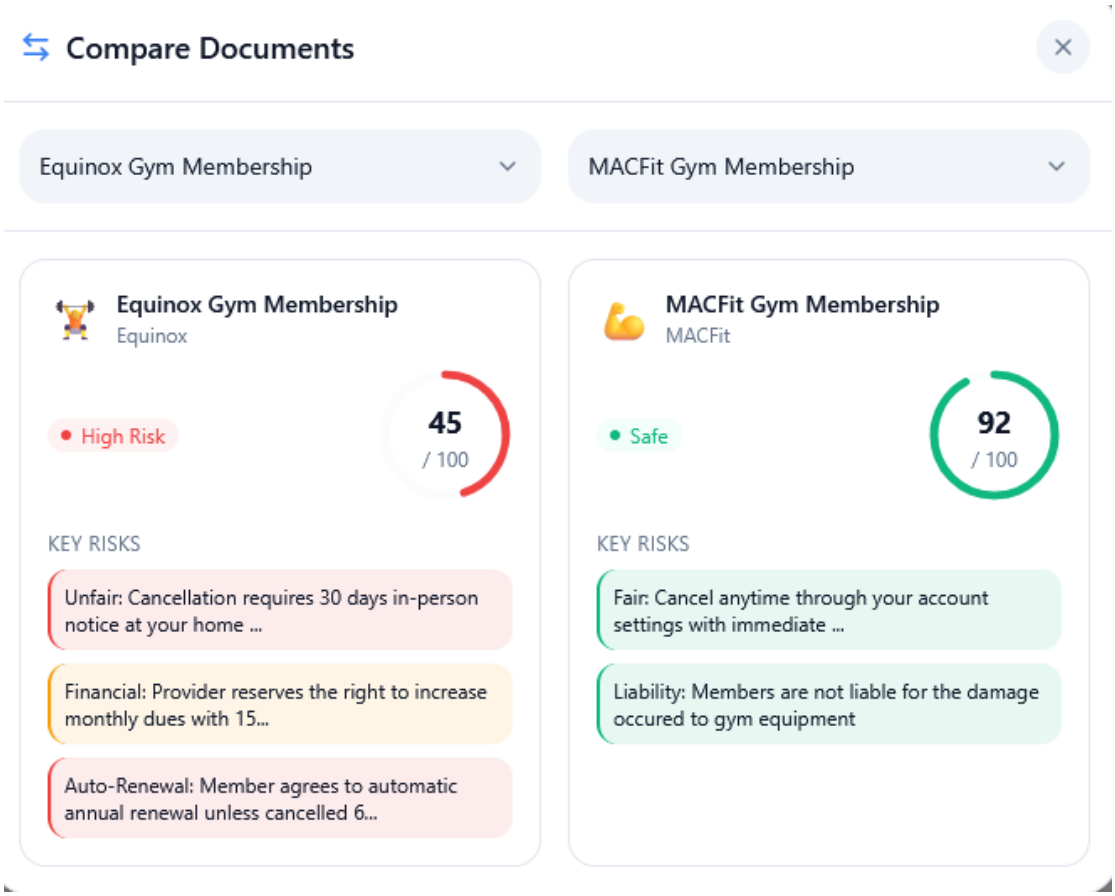


Draft Generation Page:



Document Upload Page:

Contract Comparison Page:



4 Other Analysis Elements

4.1 Consideration of Various Factors in Engineering Design

4.1.1 Constraints

4.1.1.1 Implementation Constraints

- The project is constrained to React Native to allow for a single codebase that deploys to iOS, Android, and Web platforms simultaneously. This limits the use of certain web-specific DOM manipulations and CSS-in-JS libraries that are not compatible with the React Native bridge.

- The backend is restricted to FastAPI (Python). While performant, this constrains the system to synchronous/asynchronous patterns specific to Python's asyncio and requires tight integration with Python-based AI libraries (PyTorch, Transformers) rather than Node.js or Go-based alternatives.

4.1.1.2 Legal & Regulatory Constraints

- The system operates under the strict constraint that it must not provide legal advice. All outputs must be labeled as "informational summaries" or "risk detections." The UI design is constrained to avoid prescriptive language (e.g., "Do not sign this") in favor of neutral analysis (e.g., "This clause limits your liability rights").

- As a platform processing personal contracts, the system must comply with GDPR. This necessitates architectural features for data portability, permanent account deletion ("Right to be Forgotten"), and strict consent logging for data processing.

4.1.1.3 Privacy and Security Constraints

- Users upload sensitive documents. The system is constrained by a "privacy-first" policy where user data must not be used to train the global model without explicit opt-in.

- All data must be encrypted at rest within the Personal Vault and during transmission using industry-standard secure communication protocols. The system implements widely accepted open standards for authentication and authorization to ensure that a user’s legal documents are accessible only to them, even in a shared database environment.

4.1.1.4 Usability Constraints

- Legal documents are notoriously dense. The Web and Mobile interfaces are constrained by internationally recognized accessibility guidelines, requiring high contrast, screen reader compatibility, and scalable text to ensure usability for all citizens, including those with visual impairments.

- Legal contracts are often long and difficult to read. To avoid overwhelming the user, the interface is constrained to use a "progressive disclosure" strategy. This means the system must show summaries and

critical risks first. The full, detailed legal text must remain hidden or collapsed until the user specifically chooses to view it.

4.1.1.5 Social Constraints

- There is a social risk that users may blindly trust the AI and sign a bad contract because the system didn't flag a specific loophole. The interface is constrained to include "Human-in-the-Loop" reminders, encouraging users to verify critical details manually.

- The tool is designed to democratize legal understanding. Therefore, the "Plain Language" summaries must be written with a simpler grammar level to ensure they are accessible to users with different levels of understanding.

4.1.1.6 Technical Constraints

- The "Personal Vault" querying relies on Dense Passage Retrieval (DPR) using dual-encoder networks. The system is constrained by the semantic gap between layman user queries and formal legal terminology. To mitigate hallucinations, the generative model is strictly constrained to answer only using the context retrieved by the vector store, rejecting queries where the similarity score falls below a set threshold.

- Unlike simple extractive methods, the "Sequence-to-Sequence" Transformer models used for abstractive summarization are computationally intensive. Real-time processing is constrained by the inference speed of these models, requiring asynchronous processing queues for large documents to prevent timeout errors in the client application.

- The "Alerts & Reminders" module utilizes Named Entity Recognition (NER) for temporal extraction. The system is constrained by the complexity of resolving relative legal timelines (e.g., "the first business day following the completion of the Audit Period"). The extraction logic is limited to specific recognizable temporal patterns and requires heuristic fallbacks for highly ambiguous date references.

- The Automated Risk Detection classifiers are trained on labeled datasets for specific document types (e.g., Terms of Service). The system is constrained by "domain shift," meaning a model trained on software licenses may fail to accurately detect unfair clauses in a rental lease. Consequently, the system must enforce strict document classification prior to analysis to select the appropriate risk model.

- To support efficient indexing and retrieval for the "Personal Vault," the system is constrained to a fixed vector dimension size (e.g., 768 dimensions for BERT-based embeddings). This imposes a trade-off between semantic precision and query latency, requiring optimized index structures (e.g., HNSW) to maintain sub-second search speeds as the vault grows.

4.1.1.7 Cost Constraints

- Since the system involves training custom models for each module, the project is constrained by limited financial resources for computational power. High-performance hardware accelerators are required for the training and fine-tuning phases. This necessitates strict budget management for cloud computing services and prevents the team from training massive parameter models, limiting the scope to smaller, more efficient architectures that can be trained within a finite budget.

- Hosting multiple custom-trained models requires significant memory and processing availability continuously. The architecture is constrained to use model optimization and compression techniques to reduce hosting overhead. The system must be designed to run on affordable commodity hardware or lower-cost infrastructure tiers to remain financially viable, preventing the reliance on expensive enterprise-grade clusters for daily operations.

Table 2: Factors that can affect analysis and design.

Constraint Type	Effect level	Effect
Implementation	6	React Native + FastAPI constrain UI capabilities, shared code patterns, and backend integration choices; encourages one centralized API and reusable UI components across platforms.
Legal & Regulatory	9	System must avoid legal advice, enforce disclaimers and neutral language, and support consent + deletion/export features aligned with GDPR principles.
Privacy and Security	10	Contracts are sensitive; requires encryption in transit/at rest, strict access control, per-analysis consent for external LLM usage, and data minimization/retention limits.
Usability	7	Dense documents require progressive disclosure, readability-first summaries, accessibility (contrast, scaling, screen readers), and explainability/traceability from outputs back to clauses.

Social	6	Users may over-trust AI; requires uncertainty signaling, “verify yourself” messaging, and careful risk framing to avoid harm from omissions.
Technical	8	Document variability (PDF/OCR/web), long-text processing, async jobs, and model reliability/domain shift constrain what’s feasible; demands fallback behaviors and partial results when needed.
Cost	8	Limited budget/GPU access constrains model training and hosting; pushes toward smaller models, optimized inference, bounded API usage, and phased feature scope.

4.1.2 Consideration of Global, Cultural, Social, Environmental, and Economic Factors in Engineering Design

Agreemind is a consumer-facing legal assistance system intended to improve everyday users’ understanding of terms of service, privacy policies, and common consumer contracts. Because these documents shape user rights and obligations across many contexts, the system’s analysis and design were influenced by global, cultural, social, environmental, and economic factors as described below.

4.1.2.1 Global Factors

- Agreements and privacy practices differ across countries, and legal terminology, consumer protections, and dispute resolution norms are not globally uniform. Therefore, Agreemind was constrained to an English-only MVP and avoided jurisdiction-specific legal conclusions. Outputs are presented as informational summaries and risk indicators, with explicit disclaimers and clause-level traceability.

4.1.2.2 Cultural Factors

- Legal awareness, and expectations about privacy and contractual fairness vary culturally. To reduce misinterpretation and over-trust, Agreemind uses plain-language explanations and avoids culturally loaded or prescriptive language. The user interface is designed to be neutral and respectful, prioritizing clarity and transparency.

4.1.2.3 Social Factors

- A major social risk is that users may rely on the system as a substitute for professional advice or assume that “no warning” means “safe.” This affects design through (1) visible disclaimers that the tool is not a lawyer, (2) uncertainty signaling when the model is not confident or evidence is weak, and (3) human-in-the-loop prompts that encourage users to read critical clauses and verify extracted deadlines.

4.1.2.4 Environmental Factors

- Agreemind uses compute-heavy NLP models, which can increase energy consumption, especially if large documents are processed frequently or models are hosted continuously. This affected design by favoring asynchronous processing (to avoid repeated retries/timeouts), reuse of cached results for previously analyzed documents, and preference for smaller, more efficient models where possible. The system also limits unnecessary processing (e.g., analyzing only when the user explicitly requests it rather than constant monitoring), which reduces compute usage and associated environmental impact.

4.1.2.5 Economic Factors

- Many users affected by unfair terms or complex subscriptions are also sensitive to cost. This influenced the design to keep the MVP feasible with limited budget by using a centralized backend, controlling external API usage through per-analysis consent, and avoiding expensive features. The system is designed so that core value (upload → summary → risk highlights → key dates) can be delivered with bounded compute costs, while advanced features (continuous version tracking, deep legal rule engines) remain optional extensions. Internally, project constraints such as limited GPU access also shaped the choice to prioritize feasible model sizes and incremental improvements over training very large models.

Table 3: Impact of Global/Cultural/Social/Environmental/Economic Factors

Factor	Effect level	How it affected analysis and design
Global	7	Jurisdictional variability and cross-border data processing risks led to English-only MVP, avoidance of legal conclusions, traceability to source clauses, and per-analysis consent before external LLM calls.
Cultural	6	Differences in legal literacy and privacy norms led to plain-language summaries, neutral phrasing, progressive disclosure, and accessibility-focused UI decisions.
Social	8	Over-trust risk led to disclaimers, uncertainty/confidence indicators, human-in-the-loop prompts, and

		conservative wording that avoids prescriptive legal advice.
Environmental	5	Compute/energy concerns led to user-initiated analysis (no background monitoring), asynchronous jobs, caching/reuse of results, and preference for efficient models where possible.
Economic	8	User affordability + team budget constraints led to bounded API usage, feasible model sizes, centralized backend, and MVP scoping that avoids expensive continuous features.

4.1.3 Standards

- **IEEE 830:** Used to define functional and non-functional requirements, ensuring the specification document is complete and verifiable.
- **UML 2.5.1:** Utilized for system modeling. Class Diagrams represent data entities, while Sequence Diagrams map interactions between the Client Layer and API Gateway.
- **REST API Guidelines:** The backend (FastAPI) follows REST principles for stateless communication with client applications, using standard HTTP methods and status codes.
- **WCAG 2.1 Level AA:** Mandates contrast ratios and screen-reader compatibility for the Web and Mobile interfaces, ensuring accessibility for users with visual impairments.
- **TLS 1.3:** Secures data transmission between clients and cloud infrastructure, preventing interception during upload or analysis.
- **ISO/IEC 22989:** Defines AI concepts and lifecycle management terms to ensure consistent terminology across documentation and code.

4.2 Risks and Alternatives

- **AI Misinterpretation of Legal Nuance:** Legal language is highly context-dependent, where a single word can change the entire meaning of a clause. There is a risk that the system might oversimplify a complex provision during summarization, causing the user to miss a subtle but critical liability. If the system fails to flag a specifically worded loophole, the user might sign a harmful agreement. To mitigate this, the system must prioritize "precision over simplicity" for high-risk clauses and always present the original text alongside the summary, encouraging users to verify the source.

- **User Over-Reliance:** Users may develop a habit of blindly trusting the "Green/Safe" indicators without reading the actual contract. This "automation bias" creates a dangerous situation where a user might agree to terms simply because the AI didn't flag them. To address this, the user interface should be designed to prevent "one-click" acceptance, requiring users to interact with or acknowledge specific sections before the analysis is marked as complete.

- **Data Security & Trust:** The platform handles highly sensitive personal documents (e.g., employment contracts, debt agreements). A security breach or even a perceived lack of privacy could destroy user trust and result in significant reputation damage. If users are hesitant to upload documents, the system fails. Mitigation involves minimizing data retention: processing files without permanently storing them where possible, and maintaining transparent data usage policies.

- **Input Quality Issues (OCR Failure):** Since many users will capture contracts using mobile cameras, poor lighting or shaky hands could result in low-quality text extraction. If the underlying text is garbled, the analysis will be flawed. The system must include a quality assurance step that detects illegible inputs immediately and prompts the user to retake the image rather than attempting to analyze bad data.

- **Economic Sustainability:** Processing long legal documents requires significant computational resources, which creates high operational costs. If the cost of analyzing a document exceeds the revenue or budget allocated per user, the project may become financially unsustainable. The alternative plan involves implementing usage limits or tiered service levels to balance the computational load.

Table 4: Risks

	Likelihood	Effect on the project	B Plan Summary
AI Misinterpretation	Medium	High	Side-by-side source verification & disclaimer prompts
User Over Reliance	High	Medium	Mandatory manual review steps for critical flags
Data Security & Trust	Low	High	Data minimization & "process-without-store" options
Input Quality Issues (OCR Failure)	High	Medium	Automated quality checks & retake prompts
Economic Sustainability	Medium	High	Usage quotas & resource optimization strategies

4.3 Project Plan

Below you can see various tables that you will make use of.

The project plan can be reported by list of work packages and their content.

For better readability, a Gantt chart based on work packages can also be added.

Table 5: List of work packages

WP#	Work package title	Leader	Members involved
WP1	Team Formation, Topic Selection, Supervisor Search	Ata Soykal	All members
WP2	Innovation Expert Interviews	Edip Emre Dönger	All members
WP3	Requirements Elicitation & Project Information Form	Can Polat Bülbül	Ata Soykal, Emir Görgülü
WP4	Project Specification Document (Architecture + Scope)	Ata Oğuz	Can Polat Bülbül, Edip Emre Dönger
WP5	Analysis & Requirements Report (CS491 submission)	Emir Görgülü	Emir Görgülü, Can Polat Bülbül
WP6	Web MVP Implementation Sprint (Auth + Upload + Summary + Vault)	Emir Görgülü	Ata Oğuz
WP7	CS491 Demo Preparation (Web-only)	Ata Oğuz	All members
WP8	System Hardening (Jobs, Errors, Logging, Basic Security)	Edip Emre Dönger	Can Polat Bülbül, Ata Soykal
WP9	Mobile App (React Native iOS/Android) + Share-Sheet Ingestion	Ata Soykal	Emir Görgülü, Ata Oğuz
WP10	Browser Extension (Send page → badge → deep link)	Can Polat Bülbül	Ata Soykal, Edip Emre Dönger
WP11	Clause Highlighting + Report Viewer (Cross-platform)	Ata Oğuz	Emir Görgülü, Ata Soykal
WP12	Agreement Vault v2 (Search, Tags, Versions, Compare)	Ata Oğuz	Can Polat Bülbül, Edip Emre Dönger
WP13	Custom/Local Model Prototype + Evaluation Plan	Edip Emre Dönger	Can Polat Bülbül, Ata Soykal
WP14	Testing, QA, Final Demo, and Final Report Package	Can Polat Bülbül	All members

WP 1: *Team Formation, Topic Selection, Supervisor Search*			
Start date: *2025-09-15* **End date:** *2025-10-10*			
Leader:	*Ata Soykal*	**Members involved:**	*All Members*
Objectives: *Establish the team, define a feasible project topic, and secure a supervisor.*			
Tasks: ***Task 1.1*** *Brainstorm project ideas aligned with course expectations.* ***Task 1.2*** *Identify candidate supervisors; schedule and conduct meetings.* ***Task 1.3*** *Refine project scope to match a two-semester deliverable timeline.*			
Deliverables ***D1.1:*** *Topic summary + supervisor confirmation.*			
WP 2: *Innovation Assessment & Stakeholder Interviews*			
Start date: *2025-10-10* **End date:** *2025-10-31*			
Leader:	*Edip Emre Dönger*	**Members involved:**	*All Members*
Objectives: *Validate novelty/value, gather external perspective, and satisfy innovation-form expectations.*			
Tasks: ***Task 2.1*** *Interview 3–4 innovation experts and document feedback.* ***Task 2.2*** *Identify differentiation vs. existing legal-summary tools.* ***Task 2.3*** *Convert feedback into scope boundaries and MVP priorities.* ***Task 2.4*** *Reach an agreement with an innovation expert.*			
Deliverables ***D2.1:*** *Assessment of Innovation Form*			

WP 3: *Requirements Elicitation & Project Information Form*			
Start date: *2025-10-14* **End date:** *2025-10-24*			
Leader:	*Can Polat Bülbül*	**Members involved:**	*Ata Soykal, Emir Görgülü*
Objectives: *Formalize high level requirements and project framing for CS491.*			
Tasks: ***Task 3.1*** *Define target users, key use cases, and non-goals (not a lawyer replacement).* ***Task 3.2*** *Draft initial functional/nonfunctional requirements at high level.* ***Task 3.3*** *Define initial risks and assumptions.*			
Deliverables ***D3.1:*** *Project Information Form*			

WP 4: *Project Specification Document*			
Start date: *2025-11-01* **End date:** *2025-11-28*			
Leader:	*Ata Oğuz*	**Members involved:**	*Can Polat Bülbül, Edip Emre Dönger*
Objectives: *Produce a concrete project spec describing architecture, modules, constraints, and intended features.*			
Tasks: ***Task 4.1*** *Define the backend architecture and module boundaries.*			

Task 4.2 *Specify major subsystems (ingestion, analysis pipeline, vault, reminders, comparison).* ***Task 4.3*** *Document constraints (privacy/security/legal) and standards.*
Deliverables ***D4.1:*** *Project Specification Document*

WP 5: *Analysis & Requirements Report*			
Start date: *2025-11-29* **End date:** *2025-12-19*			
Leader:	*Emir Görgülü*	**Members involved:**	*Can Polat Bülbül*
Objectives: *Produce a detailed analysis model + testable requirements + planning sections required by the department guideline.*			
Tasks: ***Task 5.1*** *Write FR/NFR lists with numbering and testability.* ***Task 5.2*** *Produce UML diagrams (scenarios, use cases, class model, activity/sequence/state).* ***Task 5.3*** *Write constraints, ethics, teamwork strategy, learning plan, risks/alternatives.*			
Deliverables ***D5.1:*** *Analysis & Requirements Report*			

WP 6: *Semantic Core (Segmentation, Obligations, Risk Prototypes)*			
Start date: *2025-12-01* **End date:** *2025-01-15*			
Leader:	*Edip Emre Dönger*	**Members involved:**	*Emir Görgülü*
Objectives: *Establish the semantic backbone of the system.*			
Tasks: ***Task 6.1*** *Clause segmentation with offsets and headings.* ***Task 6.2*** *Obligation and deadline extraction.* ***Task 6.3*** *Initial risk pattern detection.* ***Task 6.4*** *Define internal representations (Clause, RiskFlag, Obligation).*			
Deliverables ***D6.1:*** *Working Semantic Extraction Pipeline*			

WP 7: *Web Application MVP*			
Start date: *2025-12-01* **End date:** *2025-12-22*			
Leader:	*Ata Oğuz*	**Members involved:**	*Emir Görgülü, Can Polat Bülbül, Ata Soykal*
Objectives: *Deliver a functional web MVP for CS491 demo.*			
Tasks: ***Task 7.1*** *Implement web UI, navigation and authentication flow.* ***Task 7.2*** *Upload / paste ingestion.* ***Task 7.3*** *Provide LLM-based summaries.* ***Task 7.4*** *Basic agreement vault.*			
Deliverables ***D7.1:*** *Web MVP build*			

WP 8: *Mobile App MVP*
Start date: *2025-12-15* **End date:** *2026-02-15*

Leader:	*Can Polat Bülbül*	**Members involved:**	*Edip Emre Dönger, Ata Soykal*
Objectives: *Build the mobile client for the app.*			
Tasks: ***Task 8.1*** *Mobile UI for auth, upload, report view, vault.* ***Task 8.2*** *iOS / Android share-sheet ingestion* ***Task 8.3*** *UI stabilization and parity with web*			
Deliverables ***D8.1:*** *Mobile app MVP (iOS + Android).*			

WP 9: *Document Chat and Multi-Document Queries*			
Start date: *2025-12-15* **End date:** *2026-01-31*			
Leader:	*Ata Oğuz*	**Members involved:**	*Emir Görgülü, Ata Soykal*
Objectives: *Deliver mobile clients and a user-initiated ingestion path without background monitoring.*			
Tasks: ***Task 9.1*** *Document chunking & embeddings.* ***Task 9.2*** *Multi-document retrieval.* ***Task 9.3*** *Chat interface over user vault.* ***Task 9.4*** *Answer grounding to source text.*			
Deliverables ***D9.1:*** *Working RAG-based document query system.*			

WP 10: *Custom Model Prototype*			
Start date: *2026-12-10* **End date:** *2026-01-20*			
Leader:	*Edip Emre Dönger*	**Members involved:**	*Ata Oğuz*
Objectives: *Train a preliminary risk classifier to reduce reliance on external APIs.*			
Tasks: ***Task 10.1*** *Find suitable datasets.* ***Task 10.2*** *Train a local model for one subtask.*			
Deliverables ***D10.1:*** *Local model prototype.*			

WP 11: *Document Comparison & Vault Enhancements*			
Start date: *2026-01-10* **End date:** *2026-02-15*			
Leader:	*Ata Oğuz*	**Members involved:**	*Can Polat Bülbül, Ata Soykal, Emir Görgülü*
Objectives: *Enable long-term usefulness and differentiation.*			
Tasks: ***Task 11.1*** *Agreement version grouping.* ***Task 11.2*** *Text / clause-level comparison.* ***Task 11.3*** *Vault metadata, tags, search.*			
Deliverables ***D11.1:*** *Vault v2.*			

WP 12: *Browser Extension & Automatic ToS Detection*
Start date: *2026-02-01* **End date:** *2026-03-15*

Leader:	*Edip Emre Dönger*	**Members involved:**	*Ata Soykal, Can Polat Bülbül*
Objectives: *Support pre-acceptance analysis flows.*			
Tasks: ***Task 12.1*** *Browser extension for page extraction and seamless analysis.* ***Task 12.2*** *Risk badge + deep link.* ***Task 12.3*** *Android ToS screen detection & prompt.*			
Deliverables ***D12.1:*** *Browser extension prototype.* ***D12.2:*** *Mobile ToS detection demo.*			

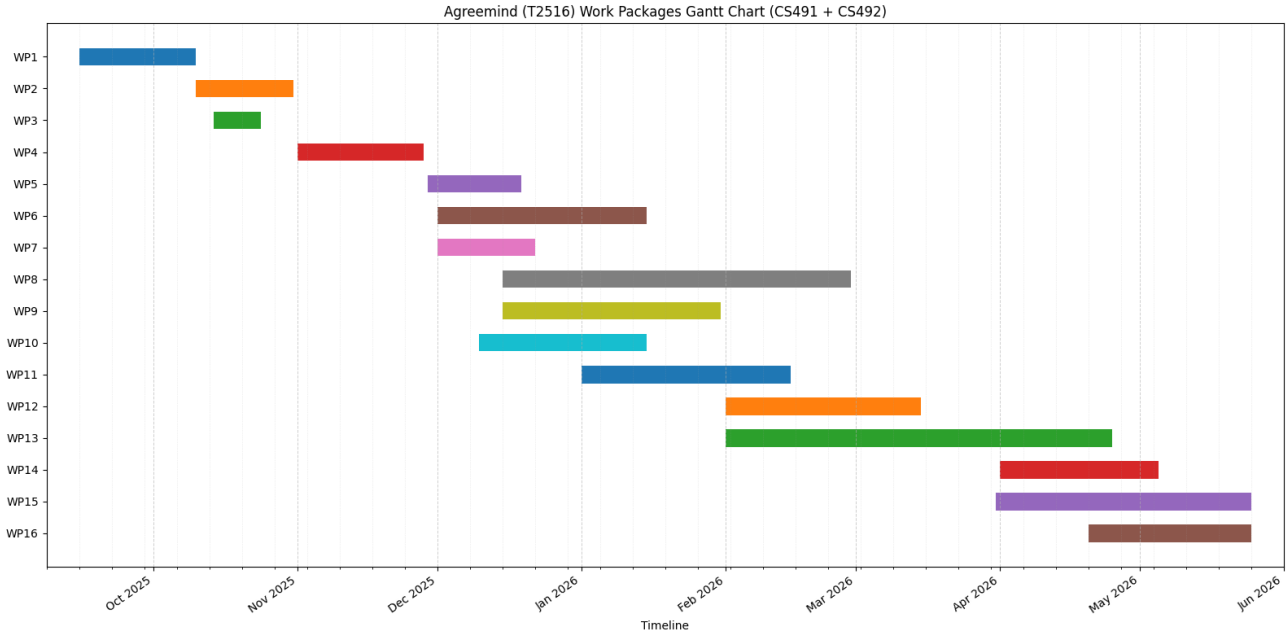
WP 13: *Domain Generalization Beyond ToS*			
Start date: *2026-02-01* **End date:** *2026-04-15*			
Leader:	*Emir Görgülü*	**Members involved:**	*Edip Emre Dönger, Ata Soykal*
Objectives: *Make the system usable in a wider range of contract domains.*			
Tasks: ***Task 13.1*** *Extend taxonomy to new domains (e.g. employment, consumer).* ***Task 13.2*** *Test extraction robustness across domains.* ***Task 13.3*** *Update prompts / models accordingly.*			
Deliverables ***D13.1:*** *Multi-domain analysis report.* ***D13.2:*** *Updated schema & taxonomy.*			

WP 14: *Automatic Version Tracking & Change Detection*			
Start date: *2026-03-15* **End date:** *2026-04-15*			
Leader:	*Ata Soykal*	**Members involved:**	*Ata Oğuz, Can Polat Bülbül*
Objectives: *Track evolving agreements automatically.*			
Tasks: ***Task 13.1*** *Detect new versions.* ***Task 13.2*** *Clause-level diffing.* ***Task 13.3*** *Change significance summaries.*			
Deliverables ***D13.1:*** *Version tracking & change detection module.*			

WP 15: *Legal Grounding & Compliance Checking*			
Start date: *2026-03-15* **End date:** *2026-04-25*			
Leader:	*Edip Emre Dönger*	**Members involved:**	*All Members*
Objectives: *Ground analysis in real regulations and user rights.*			
Tasks: ***Task 13.1*** *Select jurisdictions (e.g. GDPR, EU consumer law).* ***Task 13.2*** *Map clauses to rights and compliance issues.* ***Task 13.3*** *Update prompts / models accordingly.*			
Deliverables ***D13.1:*** *Legal ruleset documentation*			

WP 16: *Testing, QA, Final Demo, Final Report Package*
Start date: *2026-04-15* **End date:** *2026-05-05*

Leader:	*Can Polat Bülbül*	**Members involved:**	*All Members*
Objectives: *Finalize reliability, testability, and documentation required for CS492.*			
Tasks: ***Task 14.1*** *Build a requirements traceability matrix (FR → tests → results).* ***Task 14.2*** *End-to-end tests for golden flows across web + mobile (+ extension if included).* ***Task 14.3*** *Security/privacy checklist verification (consent logs, deletion, access control).* ***Task 14.4*** *Final demo script + final report writing + final architecture diagrams.*			
Deliverables ***D14.1:*** *Final demo build* ***D14.2:*** *Test report* ***D14.3:*** *Final documentation*			



4.4 Ensuring Proper Teamwork

We followed an adapted Scrum workflow that fits our course schedule and the iterative nature of Agreemind. We worked in short 1–2 week sprints, re-prioritizing tasks as requirements and implementation constraints became clearer (e.g., focusing first on a web demo and core backend pipeline). We tracked all tasks in Jira (backlog, assignees, sprint goals, and status) so individual contributions and progress were visible and reviewable.

We coordinated through regular meetings and daily communication on WhatsApp/Discord to resolve potential problems quickly. Key decisions (scope changes, architecture choices, and milestone definitions) were summarized back into Jira to keep an auditable record. Work was organized into work packages with rotating leadership.

Overall everyone contributed to every part of the project, we did not have specific limits for who did which part, we asked each other for help whenever we needed and everyone contributed to every single part of the project.

4.5 Ethics and Professional Responsibilities

The development of Agreemind is governed by a strict ethical framework prioritizing user sovereignty, transparency, and professional integrity. We explicitly define the system as an informational tool rather than a legal advisor to prevent dangerous over-reliance, ensuring all risk assessments are clearly labeled as probabilistic. To protect sensitive legal data, we adhere to "Privacy by Design" principles, enforcing end-to-end encryption and ensuring that no user-uploaded documents are used to train global models without explicit opt-in. Furthermore, our team actively mitigates algorithmic bias through regular model validation and transparency features that explain the rationale behind risk flags, while adhering to ACM and IEEE codes of ethics to maintain honest, responsible engineering practices throughout the project lifecycle.

4.6 Planning for New Knowledge and Learning Strategies

The development of Agreemind requires our team to bridge the gap between advanced software engineering and complex legal theory. To achieve our objectives, we identify specific technical knowledge gaps and implement a targeted learning strategy.

- **Legal NLP & Advanced Models:** To handle the unique complexity of legal texts, we conduct research on specialized architectures capable of processing long documents without losing context. We also study abstractive summarization techniques through academic literature and documentation, and we fine-tune models on open legal datasets to ensure accurate simplification of clauses.
- **Retrieval-Augmented Generation (RAG):** Implementing the "Personal Vault" requires mastering the RAG paradigm. We focus on learning semantic search techniques and vector database management. This allows us to understand optimal text-chunking strategies specifically for legal queries.
- **Cross-Platform & Extension Architecture:** Adopting a unified codebase for web and mobile requires learning to bridge native mobile modules with React Native. Additionally, the browser extension demands a study of modern browser standards to create a solution that complies with strict security restrictions on background processes.
- **Security & Encryption:** Given the sensitivity of user contracts, we engage in self-directed learning regarding client-side encryption and authentication. We review industry security guidelines to ensure our architecture meets the highest standards for encryption at rest and in transit.

5 Glossary

Agreemind: The proposed consumer-facing legal-document assistant that summarizes agreements, highlights risks, and helps users track obligations without providing legal advice.

Agreement: A legal text the user uploads or shares (e.g., Terms of Service, Privacy Policy, rental/subscription contract).

Clause: A meaningful segment of an agreement (sentence/paragraph/section) that expresses a rule, right, limitation, or obligation.

Risk Flag: A detected clause category that may be unfavorable to the user (e.g., data sharing, auto-renewal, unilateral change, arbitration).

Plain-Language Summary: A simplified explanation of an agreement or clause written for non-expert users.

Obligation: An action the user must do (or avoid) according to the agreement (e.g., payment, notice submission, compliance requirement).

Deadline / Notice Period: A time constraint extracted from the agreement (e.g., cancellation window, renewal date, “within 30 days”).

Personal Vault: A secure personal repository where a user’s processed agreements, reports, and metadata are stored for later search and comparison.

Version Comparison: A feature that identifies and presents changes between two versions of the same agreement (“what changed?”).

Analysis Pipeline: The backend processing steps applied to an agreement (ingestion → text extraction → chunking → retrieval/classification → summarization → report).

AnalysisJob: A backend job that represents one analysis request from a user and its processing state (queued/running/completed/failed).

ConsentRecord: A stored record that the user explicitly permitted an agreement to be processed (especially important if external APIs are used).

RAG (Retrieval-Augmented Generation): A method where the system retrieves relevant text passages and constrains the LLM to answer using that context.

Embedding: A numeric vector representation of text used to support semantic search and retrieval in the vault.

Vector Store: A database/index optimized for similarity search over embeddings (used for vault querying and context retrieval).

HNSW: A graph-based approximate nearest neighbor indexing method commonly used for fast vector similarity search.

LLM (Large Language Model): A model used to generate summaries/explanations; in your system it must be constrained to informational output (not legal advice).

Custom Model: A smaller model you train/fine-tune for a specific subtask (e.g., risk classification or date extraction) to reduce cost and dependency on external APIs.

NER (Named Entity Recognition): A technique to detect structured entities in text (e.g., dates, organizations, money amounts).

Share Sheet / Share Intent: Mobile OS functionality that lets the user share a webpage/text into Agreemind for on-demand analysis (instead of background monitoring).

Privacy by Design: Designing the system to minimize data collection, enforce access control, and prevent model training on user data without explicit opt-in.

GDPR / Right to be Forgotten: Data protection requirements that include user deletion/export rights and limits on data retention/processing.

6 References

[1] Ironclad – AI-powered Contract Lifecycle Management Software (2025). Retrieved from https://ironcladapp.com/product/ai-based-contract-management

[2] Kira Systems – AI-powered Contract Analysis Software (2025). Retrieved from https://kira.ai/solutions/legal-workflow

[3] Luminance – Legal-Grade AI Contract & Document Review Software (2025). Retrieved from https://luminance.com/solutions/legal/

[4] ToS;DR – Crowd-sourced ToS & Privacy Policy Ratings (2025). Retrieved from https://tosdr.org

[5] Open Terms Archive – Public Archive of Online Terms & Conditions (2025). Retrieved from https://opentermsarchive.org

[6] Termzy AI – Real-time ToS Detection Software (2025). Retrieved from https://www.termzyai.com/#features

[7] LegalZoom – Online Legal Services & Legal Advice (2025). Retrieved from https://www.legalzoom.com/

[8] Rocket Lawyer – Legal Documents, Advice & Lawyers (2025). Retrieved from https://www.rocketlawyer.com/