Bilkent University
Department of Computer Science

CS 491 - Senior Design Project I
Project Specification Document

Agreemind (T2516)

Group Members:
Ata Oğuz - 22202453
Ata Soykal - 22202290
Can Polat Bülbül - 22203369
Edip Emre Dönger - 22201531
Emir Görgülü - 22202834

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

1. Introduction

1.1. Description

People regularly enter into agreements of all kinds: rental contracts, subscription services, insurance policies, warranty policies, loan documents, employment terms, freelance contracts, and many others. These documents form the basis of how individuals interact with companies, service providers, landlords, financial institutions, and digital platforms. Although these agreements shape important aspects of people's financial, personal, and professional lives, they are often difficult to understand. The language is typically dense and technical, key details are buried deep within long clauses, and the implications of certain terms are not always obvious to a non-expert reader. As a result, individuals may unknowingly accept conditions that impose restrictive obligations, and they may overlook important deadlines or rights.

The challenge does not end once a contract is signed. Many agreements are revised over time, sometimes without proper notification or clear explanations of how the changes affect the user. Companies frequently update Terms of Service or privacy policies, and users have no practical way to see what has been added or removed. Managing agreements becomes even more difficult when people juggle multiple subscriptions, insurance plans, and recurring services across different platforms. Users struggle not only to interpret the agreements they encounter but also to keep track of them and maintain awareness of their rights and responsibilities throughout the lifecycle of the contract.

Agreemind is designed to address these challenges by serving as an intelligent companion for understanding, tracking, and acting upon personal agreements. It analyzes documents in natural language, translates complex clauses into clear explanations, highlights terms that may be risky or unusually strict, and identifies obligations or deadlines the user must keep in mind. It stores processed agreements in a secure personal vault, making it possible to revisit them, search across them, or compare different versions or drafts. When companies update their terms, Agreemind identifies and explains the changes, ensuring that users stay informed.

Beyond interpretation, the system also helps users act on their rights by identifying possible actions and generating drafts or templates tailored to the relevant clause. By centralizing documents, offering meaningful analysis, and providing practical tools, Agreemind enables individuals to make informed decisions, avoid unfavorable conditions, and maintain control over their responsibilities and protections throughout the life of an agreement.

1.2. High Level System Architecture & Components of Proposed Solution

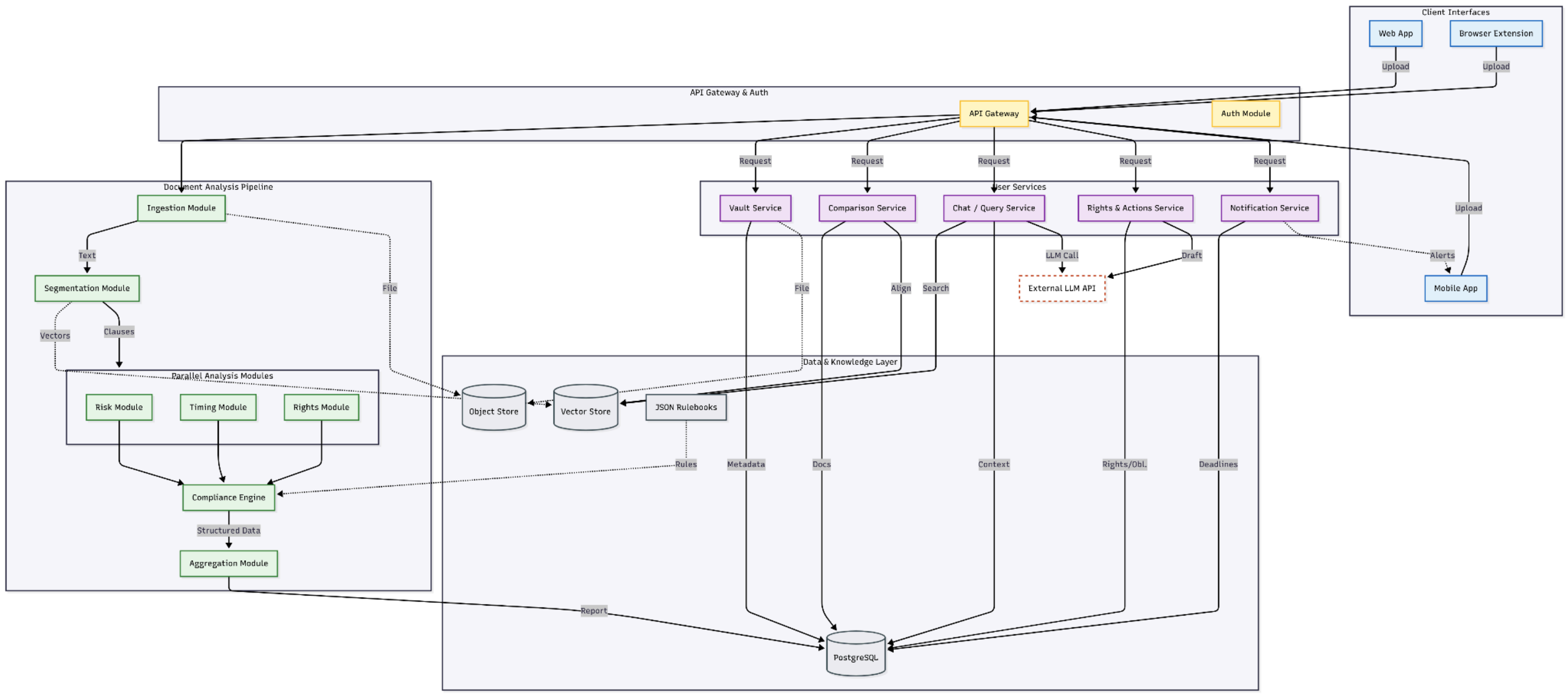

Diagram also viewable from here.

1.2.1. Client Layer

- **Web App:** Provides the main user interface for uploading documents, viewing summaries, comparing agreements, querying the system, and managing stored contracts.
- **Mobile App:** Allows users to upload documents from their device, share files directly into the system, and receive notifications or reminders on mobile.
- **Browser Extension:** Enables instant, on-page analysis of Terms of Service or online agreements before acceptance. Sends extracted page text to the API for quick risk assessment.

1.2.2. Gateway Layer

- **API Gateway:** Acts as the unified entry point for all client requests. Routes each request to the correct backend service or pipeline module and ensures consistent request handling.

- **Auth Module:** Handles authentication and authorization, protecting access to user-specific documents and data.

1.2.3. Core Processing Layer

- **Ingestion Module:** Extracts text from uploaded documents, including OCR for scanned files. Normalizes the input and saves the original file to the Object Store.

- **Segmentation Module:** Breaks the raw text into structured clauses and logical sections. Produces the clause list that the analysis modules operate on.

- **Analysis Modules (Parallel Execution):**
  - **Risk Module:** Detects risky, unusual, or unfavorable clauses using the core risk-scoring model.
  - **Timing Module:** Extracts dates, renewal cycles, windows, and other time-related constraints.
  - **Rights Module:** Identifies obligations, actions, opt-outs, and user rights embedded in clause text.

- **Compliance Engine:** Applies rules from the knowledge base to validate clauses and enrich the extracted features with compliance insights.

- **Aggregation Module:** Combines all outputs from the analysis and compliance stages into a unified, human-readable report. Uses an LLM for generating summaries, explanations, and natural-language interpretations.

1.2.4. User Services Layer

- **Vault Service:** Manages the storage, retrieval, and organization of user agreements. Fetches structured metadata and links to corresponding document files.

- **Comparison Service:** Compares two documents or versions by aligning semantically related clauses and highlighting textual and structural differences.

- **Chat / Query Service:** Provides natural-language querying across all stored agreements. Performs semantic search using clause embeddings and uses an external LLM to generate responses based on retrieved context.

- **Rights & Actions Service:** Displays actionable rights identified in the agreement (e.g., terminate, opt-out, request data). Can generate drafts and templates for executing these actions through an LLM.

- **Notification Service:** Periodically checks deadlines and renewal dates stored in the database. Sends reminders and alerts to users ahead of important contractual events.

1.2.5. Storage & Knowledge Layer

- **PostgreSQL:** Stores structured analysis outputs, summaries, clause metadata, detected deadlines, user records, and all processed results.

- **Object Store:** Holds the original uploaded documents (PDFs, DOCX, images, webpage snapshots) in their raw form.

- **Vector Store:** Contains clause embeddings for semantic retrieval, document comparison, and chat-based querying.

- **Rulebooks:** Houses domain and jurisdiction-specific rule definitions used by the Compliance Engine. Structured as versioned JSON files for easy updates and extensibility.

1.3. Constraints

1.3.1. Legal Constraints

The system cannot provide legally binding advice. All outputs must remain informational, explanatory, or suggestive in tone, without prescribing actions that could be interpreted as legal counsel.

1.3.2. Document Variability Constraints

Agreements may come in diverse formats (PDFs, scans, images, URLs, raw text) with inconsistent structure or quality. OCR errors, missing headings, and formatting issues can affect the accuracy of clause extraction and analysis.

1.3.3. Domain and Jurisdiction Constraints

Legal rules and industry practices vary significantly. Early versions of the system can only support a limited number of domains and jurisdictions. Expanding coverage requires adding new rule sets and knowledge bases over time. Legal requirements may also evolve, requiring continuous updates to rulebooks and domain packs to maintain accuracy.

1.3.4. Language Constraints

The system's models may not support all languages. Agreements written in unsupported or mixed languages may lead to reduced accuracy or incomplete analysis.

1.3.5. Data Privacy and Security Constraints

Agreements often contain sensitive personal or financial information. Strict privacy constraints require encrypted storage, secure transmission, minimal data retention, and restricted use of third-party services for processing.

1.3.6. Performance Constraints

Real-time features such as the browser extension require fast inference and lightweight processing. The Android background monitoring service must minimize battery usage and respect OS-imposed background execution limits. Real-time detection on Android must rely on lightweight heuristics to avoid excessive CPU or memory consumption.

1.3.7. Economic Constraints

The system must be developed and maintained within limited financial resources. Storage, compute, and external model usage should be optimized to reduce ongoing operational costs. Where possible, open-source tools, lightweight models, and cost-efficient deployment options shall be used to ensure the system remains financially sustainable.

1.3.8. Model and Rule Interpretation Constraints

AI models and rule-based systems have inherent limitations. They may miss subtle legal nuances or misclassify complex clauses. System outputs must account for these uncertainties and communicate that interpretations are probabilistic.

1.3.9. Version Monitoring Constraints

Detecting updates to external agreements (such as Terms of Service pages) depends on the availability, accessibility, and stability of source URLs, which the system cannot fully control.

1.3.10. User Device Constraints

Browser extension capabilities and mobile protection integrations depend on platform APIs and OS permissions, which may restrict certain functionalities.

1.4. Professional and Ethical Issues

The development of Agreemind involves several professional and ethical considerations due to the sensitive nature of legal documents and the use of AI-generated interpretations. A key issue is ensuring that all summaries, risk flags, and explanations remain strictly informational and do not present themselves as legal advice. The system clearly informs users of these limitations so that they understand the scope of the tool and do not rely on it as a substitute for professional legal counsel.

Data privacy and confidentiality are central concerns. Users upload documents that may contain personal, financial, or contractual information, and the system handles this material with strict security measures. All data is stored and transmitted securely, with controlled access to prevent unauthorized use. Ethical development also includes minimizing data collection, allowing users to delete their information, and maintaining transparency about how documents are processed and used.

Because the system relies on automated classification and analysis, fairness and accuracy represent additional ethical considerations. The models and rulebooks undergo regular validation to reduce misclassifications, biases, or misleading interpretations. Users are informed about the limitations of automated analysis so they understand that the outputs support decision-making but do not represent definitive judgments.

On the professional conduct side, the development team maintains responsible engineering practices throughout the project. Team members communicate regularly, document design decisions, and collaborate to ensure consistent implementation across modules. Code changes go through review to maintain quality and ensure proper handling of sensitive data. The team shares collective responsibility for addressing ethical concerns and for ensuring that user trust and transparency guide the project's development.

1.5. Standards

1.5.1. IEEE 830 - Software Requirements Specifications

The IEEE 830 standard is utilized to rigorously define the functional and non-functional requirements of the system, ensuring clarity and completeness in the specification document.

1.5.2. UML 2.5.1 - Unified Modeling Language

UML 2.5.1 is employed as the standard for visualizing the system's architecture, ensuring that the interactions between the diverse client applications and the backend are clearly mapped. Specifically, Sequence and Class diagrams will be used to model the modular architecture.

1.5.3. REST API Guidelines

The system adheres to REST architectural principles to ensure stateless and scalable communication between the backend and its various clients, including the web dashboard, mobile app, and browser extension. This standardization allows for efficient resource management.

2. Design Requirements

2.1. Functional Requirements

2.1.1. Document Ingestion & Preparation

- The system shall allow users to add agreements via file upload (PDF, DOCX, text), pasting text, sharing from mobile, or importing HTML pages.
- The system shall extract text from documents (including OCR for scanned files) and segment content into structured clauses with headings, numbering, and positions.

2.1.2. Contract Analysis

- The system shall generate a high-level plain-language summary for each uploaded agreement.
- The system shall classify each clause by type (renewal, fees, liability, dispute resolution, privacy).
- The system shall detect potentially risky or unfair clauses and assign a risk level using a color-coded representation.
- The system shall provide short explanations describing why each flagged clause may pose a risk.
- The system shall check clauses against domain- and jurisdiction-specific rules to identify likely compliance issues.

2.1.3. Risk, Obligations & Deadlines

- The system shall detect risk patterns, both general (unilateral changes, arbitration clauses, etc.) and domain-specific.
- The system shall identify actionable obligations within clauses (canceling, opting out, filing a claim) and extract action details such as channel (email/portal), contact information, and actor.
- The system shall extract time-sensitive elements (e.g., “within 14 days”, “before renewal”, fixed dates) and convert them into structured deadlines.

2.1.4. Personal Vault & Querying

- The system shall maintain a secure personal vault where users can save, store, and revisit analyzed agreements.
- 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).

2.1.5. Real-Time Proactive Protection

- A browser extension and mobile share-sheet integration shall allow users to analyze contracts before accepting them.
- The system shall display a quick risk overview and highlight severe clauses directly during pre-acceptance flows.
- The Android client shall include a background monitoring service that detects when a Terms of Service or similar agreement screen appears and prompts the user to run an analysis.

2.1.6. Version Tracking & Change Detection

- The system shall track multiple versions of the same agreement and store them as part of the contract’s history.
- When a new version is detected, the system shall compare it against previous versions, highlighting added, removed, or modified clauses.
- The system shall provide a side-by-side comparison with summaries explaining the significance of changes.

2.1.7. Contract Comparison

- Users shall be able to compare two separate contracts or drafts side-by-side.
- The system shall align semantically similar clauses and highlight differences in text, risks, obligations, and deadlines.

2.1.8. Rights Enforcer

- The system shall identify which legal rights a user may exercise based on the contract and applicable regulations.
- For actionable rights, the system shall generate a draft formal request (e.g., data access, contract termination, opt-out).
- The system shall locate and present relevant contact channels (emails, URLs, forms) to send such requests.

2.1.9. Alerts & Reminders

- The system shall detect upcoming contract-related events such as renewal dates, cancellation deadlines, claim windows, and payment obligations.
- The system shall schedule timely reminders based on extracted deadlines and notify the user before key events occur.

2.2. Non-Functional Requirements

2.2.1. 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.

2.2.2. 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.

2.2.3. 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.

2.2.4. 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.

2.2.5. 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.

2.2.6. 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.

2.2.7. Legal and Ethical Requirements

- The system shall clearly communicate that it does not provide legal advice and should be used as an informational tool.
- Explanations, warnings, and summaries shall be generated in a neutral, non-directive tone.

- The system shall ensure transparency regarding data usage, automated decision-making, and third-party integrations.

3. Feasibility Discussions

3.1. Market & Competitive Analysis

The current legal technology market is predominantly focused on enterprise solutions, serving law firms and large corporations with complex contract lifecycle management tools. While these tools are powerful, they leave a significant gap in the consumer market. Individuals are frequently exposed to complex agreements, ranging from software Terms of Service to rental leases and freelance contracts, but lack accessible tools to understand or manage them.

Agreemind addresses this gap by positioning itself as a "Personal Legal Companion" rather than a corporate tool. Unlike existing solutions that focus on drafting or B2B compliance, Agreemind focuses on the reception of contracts, empowering the individual signer. Below is an analysis of similar tools and how Agreemind distinguishes itself:

- **Enterprise Legal AI (Ironclad, Kira Systems, Luminance):** These platforms utilize advanced machine learning for clause detection and contract review. They are designed for corporate legal teams to streamline workflows and ensure compliance [1], [2], [3].

Differentiation: These tools are cost-prohibitive and overly complex for individual users. Agreemind is built for the "non-expert reader," prioritizing the translation of "legalese" into plain language and offering a user-centric interface rather than a corporate dashboard.

- **Crowdsourced Transparency Projects (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 Detectors (Termzy AI):** These tools often exist as browser extensions that analyze terms while a user browses the web [6].

Differentiation: While helpful for web browsing, these tools lack lifecycle management. Agreemind differentiates itself through its "Personal Vault" and "Rights Enforcer," which allow users to store agreements, track changes over time, and actively generate legal request drafts long after the document is signed.

In summary, while the market contains high-end corporate tools and basic web-summarizers, Agreemind stands out by combining the analytical power of enterprise AI with the accessibility required for consumer use. It shifts the user's role from passive acceptance to active management.

3.2. Academic Analysis

The development of Agreemind relies on established and emerging research in Natural Language Processing (NLP) and Information Retrieval, specifically focusing on the legal domain (Legal NLP). By integrating methodologies from these fields, the platform ensures that its simplification and risk detection mechanisms are computationally sound.

- **Legal Text Simplification and Abstractive Summarization:** Academic research in text simplification often utilizes Transformer-based architectures (such as BERT or GPT variants) to transform complex syntactic structures into simpler forms while retaining semantic meaning. Agreemind leverages these "Sequence-to-Sequence" models to perform abstractive summarization. This approach allows the system to generate natural language explanations of dense clauses, moving beyond simple extractive methods that only highlight existing sentences [7], [8].
- **Automated Risk Detection and Classification:** The classification of legal clauses involves identifying specific categories (e.g., "Liability," "Termination," "Arbitration") and assessing their sentiment or risk profile. Research in "Legal Judgment Prediction" and "Unfair Clause Detection" demonstrates that training classifiers on labeled legal datasets allows for the probabilistic identification of potentially unfair terms [9]. Agreemind incorporates these classification techniques to assign risk levels (color-coded indicators) to specific clauses, alerting users to non-standard or aggressive obligations.
- **Retrieval-Augmented Generation (RAG) for Document Querying:** To enable the "Personal Vault & Querying" feature, Agreemind utilizes Retrieval-Augmented Generation (RAG) enhanced by Dense Passage Retrieval (DPR) [10]. Unlike traditional keyword search, DPR utilizes dual-encoder networks to map both user queries and document passages into a shared dense vector space, enabling the system to retrieve semantically relevant clauses even when exact keywords do not match. By feeding these precise, context-rich chunks into the generation model, Agreemind allows users to ask natural language questions (e.g., "What is the notice period?") and receive accurate

answers grounded strictly in the document's text, minimizing hallucinations common in pure generative models.

- **Temporal Information Extraction:** Research in Named Entity Recognition (NER) focuses on extracting specific entities like dates, organizations, and monetary values [11]. Agreemind applies temporal extraction techniques to identify "time-sensitive elements" such as renewal dates and cancellation windows. This aligns with academic work on "Temporal Relation Extraction," enabling the system to structure unstructured text into actionable deadlines for the "Alerts & Reminders" module [12], [13].

By grounding its architecture in these academic principles: Abstractive Summarization, Text Classification, RAG, and Temporal Extraction, Agreemind ensures a robust, scalable, and scientifically validated approach to legal document analysis.

Glossary

API – Application Programming Interface

OCR – Optical Character Recognition

LLM – Large Language Model

RAG – Retrieval-Augmented Generation

DPR – Dense Passage Retrieval

NLP – Natural Language Processing

NER – Named Entity Recognition

UML – Unified Modeling Language

SRS – Software Requirements Specification

JSON – JavaScript Object Notation

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] Kornilova, A., & Eidelman, V. (2019). BillSum: A Corpus for Automatic Summarization of US Legislation. *In Proceedings of the 2nd Workshop on New Frontiers in Summarization* (pp. 48-56).

[8] Manor, L., & Li, J. J. (2019). Plain English Summarization of Contracts. *In Proceedings of the Natural Legal Language Processing Workshop 2019* (pp. 1-11).

[9] Lippi, M., Palka, P., Contissa, G., Lagioia, F., Hanser, H., Khoroshavin, Y., ... & Sartor, G. (2019). *CLAUDETTE: an automated detector of potentially unfair clauses in online terms of service.* Artificial Intelligence and Law, 27(2), 117-139.

[10] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 6769-6781).

[11] K. Pakhale, "Comprehensive Overview of Named Entity Recognition: Models, Domain-Specific Applications and Challenges," *arXiv preprint arXiv:2309.14084*, 2023.

[12] Chalkidis, I., Androutsopoulos, I., & Michos, A. (2017). *Extracting Contract Elements.* In Proceedings of the 16th International Conference on Artificial Intelligence and Law (ICAIL '17), 19–28.

[13] B. Jehangir, S. Radhakrishnan, and R. Agarwal, "A survey on Named Entity Recognition – datasets, tools, and methodologies," *Natural Language Processing Journal*, vol. 3, p. 100017, 2023.