

ClauseEase

AI based Contract Language Simplifier

ClauseEase

Transforming Legal Documents through Intelligent Automation

- ✓ AI-Driven Contract Understanding
- ✓ Multi-Format Uploads (PDF, DOCX, TXT)
- ✓ Simplify Complex Legal Language
- ✓ Clear Contracts. Confident Decisions

What is ClauseEase?

Project Statement:

Legal contracts are often written in dense, complex language that is difficult for non-lawyers to understand.

This project, **ClauseEase**, aims to build an **AI-powered system** that automatically extracts, analyses, and simplifies legal contract clauses without losing legal meaning or intent.

The goal is to make contracts more **readable, accessible, and understandable** for all users.

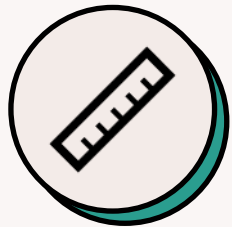
Objectives



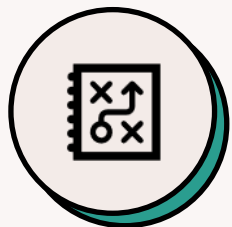
Automate contract text extraction and preprocessing.



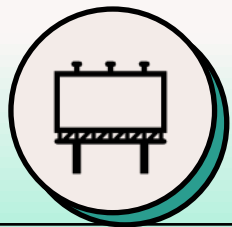
Identify and classify legal clauses using NLP and AI models.



Recognize complex legal terms and generate plain-English explanations.



Simplify contract language while preserving legal accuracy.



Provide users with an intuitive interface for contract analysis.

ClauseEase - Modules



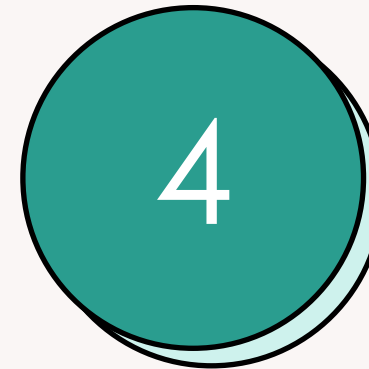
**Document Ingestion
& Text Preprocessing**



**Legal Clause
Detection**



**Legal Term
Recognition**



**Language
Simplification**



**Context Linking &
Feedback**

Requirement Analysis

Hardware:

- **OS:** Compatible with Windows, macOS, or Linux.
- **RAM:** Minimum **4GB** RAM required.
- **CPU:** Processor speed of Core i5 or equivalent/higher recommended (GPU recommended for higher evaluation models)

Basic Requirements:

- **Python 3.10** or higher
- **Web Browser:** Google Chrome / Edge / Firefox (latest version)
- **IDE / Code Editor:** VS Code / PyCharm / IntelliJ

Software Requirement Analysis

- **Flask**
- HTML5, CSS3, JavaScript
- **Jinja2** templating (Flask integrated)
- **Docker** Desktop (for deployment)

Integrations:

Model Name	Purpose
prajjwal1/bert-mini	Legal clause classification
google/flan-t5-small	Legal text simplification / paraphrasing
en_core_web_sm (spaCy)	Entity extraction & linguistic analysis

- **Google Firebase (Cloud Firestore)** → NoSQL database

Py Libraries/ Dependencies	Purpose / Usage
fitz (PyMuPDF)	PDF text extraction
python-docx	DOCX text extraction
re	Regular expressions for cleaning text
nltk	Sentence tokenization (sent_tokenize)
spacy	Named Entity Recognition (NER)
torch (PyTorch)	Running Legal-BERT / Google t5 model
transformers	Pretrained AI models (Legal-BERT, t5-small)
scikit-learn	Basic ML utilities (label encoding, evaluation)
numpy	Numerical operations
firebase-admin	Firestore connection in Flask
json / os	File handling, data serialization

Data Flow Diagrams (DFDs)

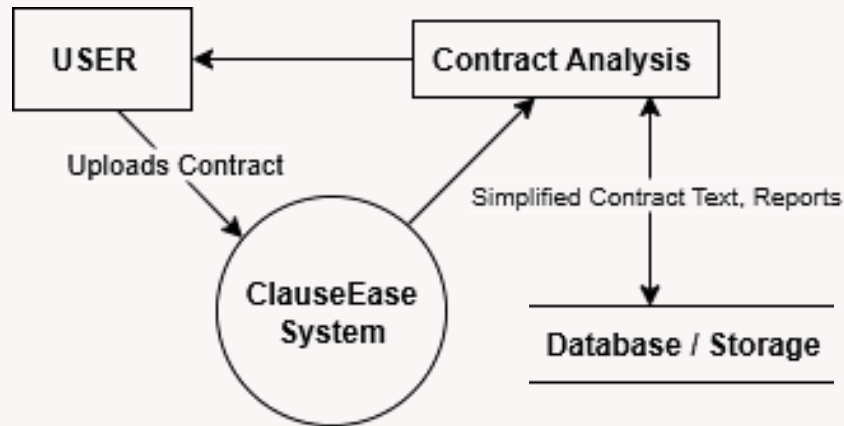


Fig.1: 0-level DFD

Note:

The output can be shown into **Basic**, **Intermediate** and **Advanced** paraphrases, according to user Query

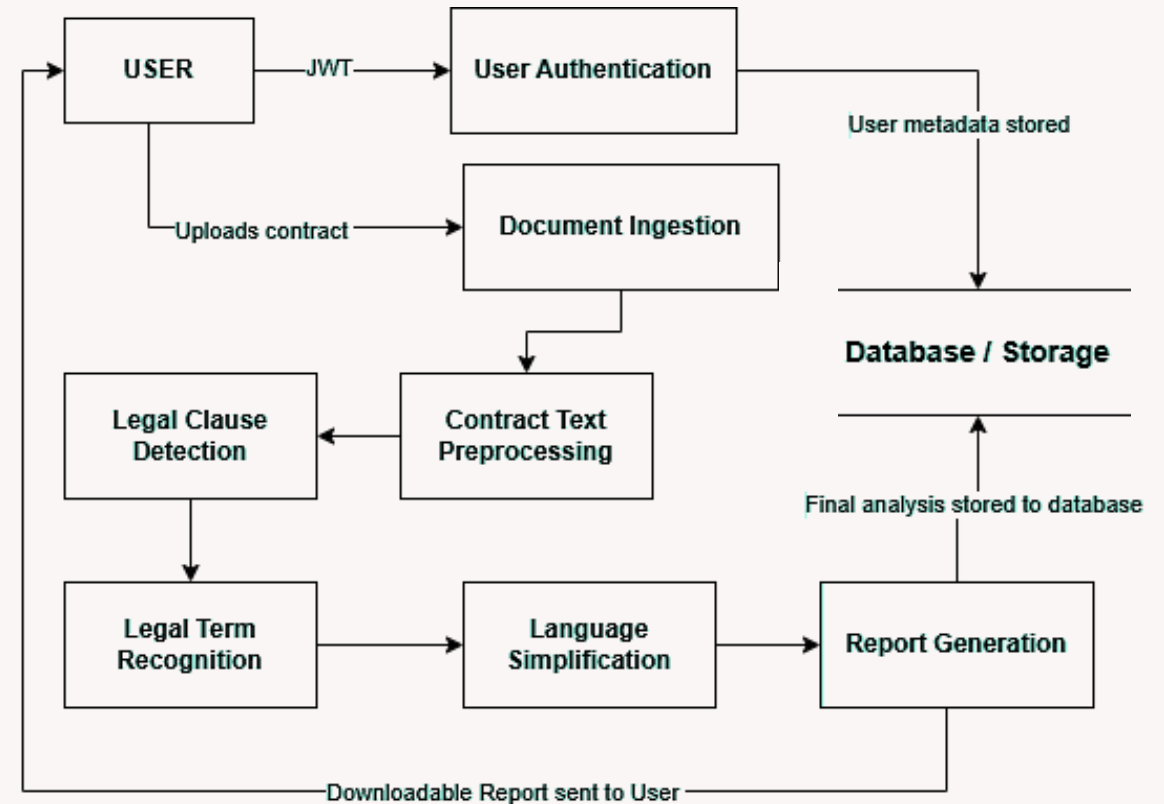
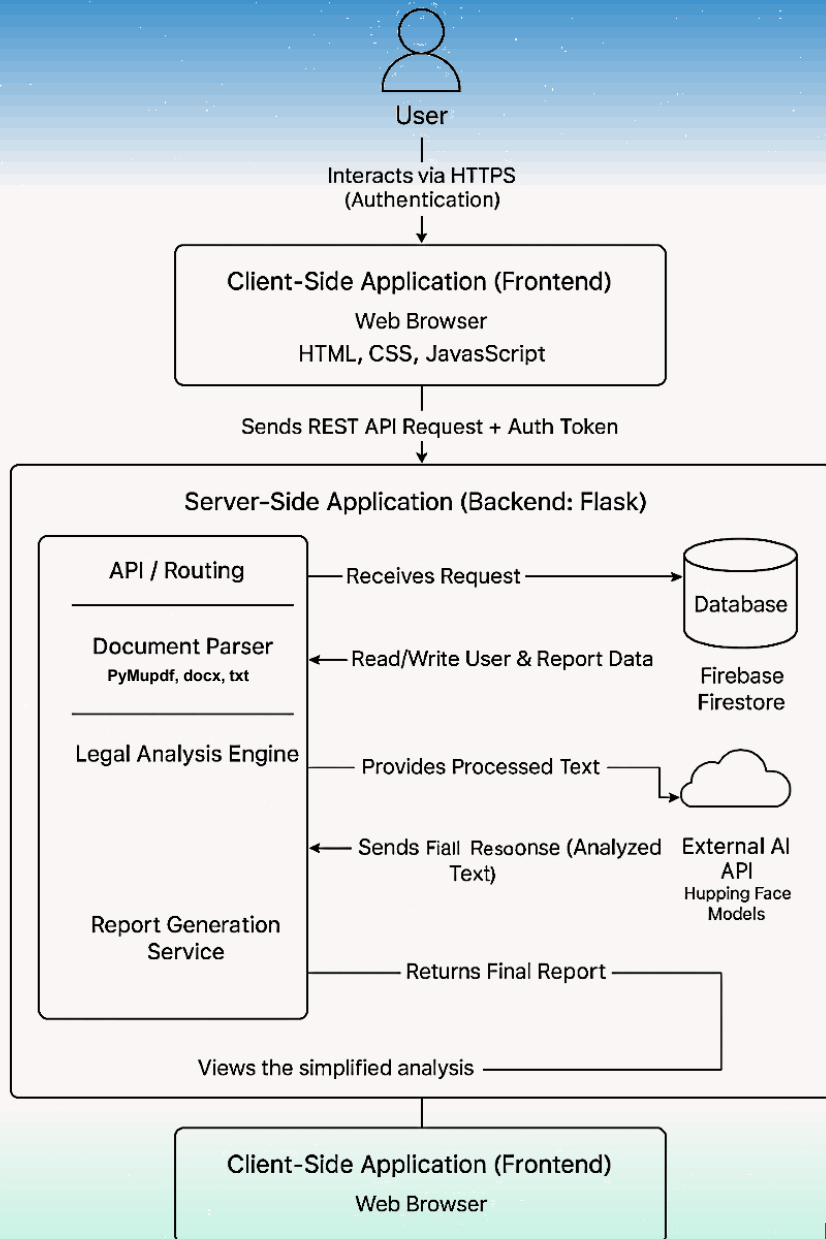


Fig.2: Level-1 DFD

System Architecture

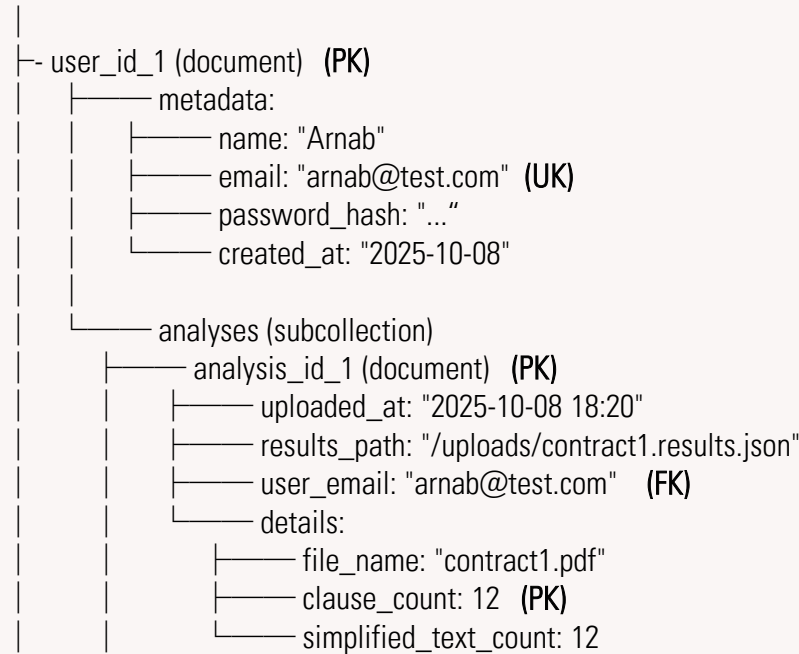


- ❑ ClauseEase uses a **Flask backend** integrated with AI models for clause detection and simplification.
- The **frontend** provides an interactive user interface, while **Firestore** securely stores user data and processed clauses.
- Requests flow from the **Frontend → Backend → AI HuggingFace models → Results** returned and saved for real-time access.

Fig. 3: System Architecture Diagram

Database Table Description

users (collection)



Index:

PK: Primary Key
UK: Unique Key
FK: Foreign Key

Collection / Subcollection	Description	Key Fields / Documents
users	Main collection storing all registered user accounts and their profile metadata.	Document ID: unique user identifier (UID) Fields: name, email, password_hash, created_at
users → {user_id} → analyses	Subcollection under each user storing all uploaded contract analyses. Each document corresponds to one analyzed contract.	Document ID: unique analysis ID Fields: results_path (path to JSON result), uploaded_at, user_email, details (summary info like file name, clause count, etc.)
analyses → file_details	(If nested further) Holds specific details about each uploaded contract file, extracted clauses, and results metadata.	Fields: file_name, file_size, file_type, results_path, simplified_text_count, clause_count
legal_terms	A global collection storing predefined legal terms and definitions for recognition and glossary support.	term, definition, category

Testing & Implementation

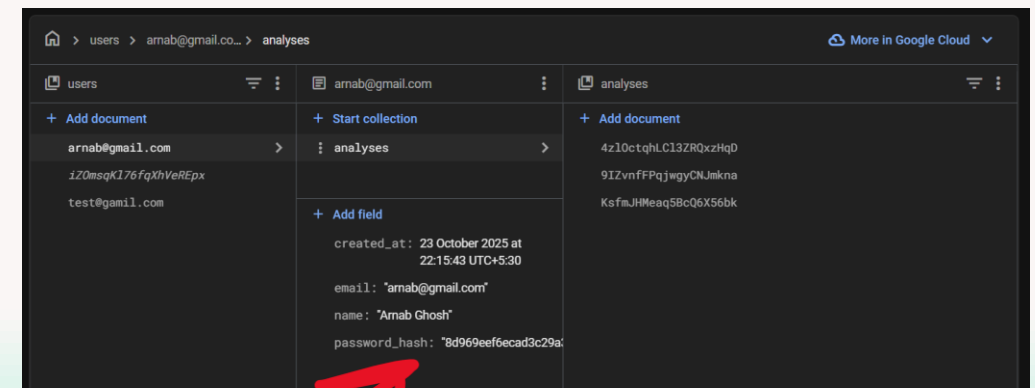
Testing Techniques

- ❑ **Unit Testing:** Tested each individual module:
 - *Text Extraction, Clause Detection, Simplification, and Glossary Mapping* modules.
- ❑ **Integration Testing:** Verified seamless communication between:
 - Flask backend, HuggingFace models, and Firestore database.
- ❑ **API Testing (Postman):**
 - All Flask routes (e.g. /api/analyze) tested using **Postman** for correct request/response handling. {json output}
- ❑ **User Interface Testing:**
 - Ensured responsive layout, smooth tab switching, and accurate result rendering in HTML frontend.
- ❑ **Performance Testing:**
 - Verified handling of large contracts (**PDF/DOCX**) and AI inference time optimization.
- ❑ **Security Testing:**
 - Password hashing and authentication verified for data safety.



Implementation

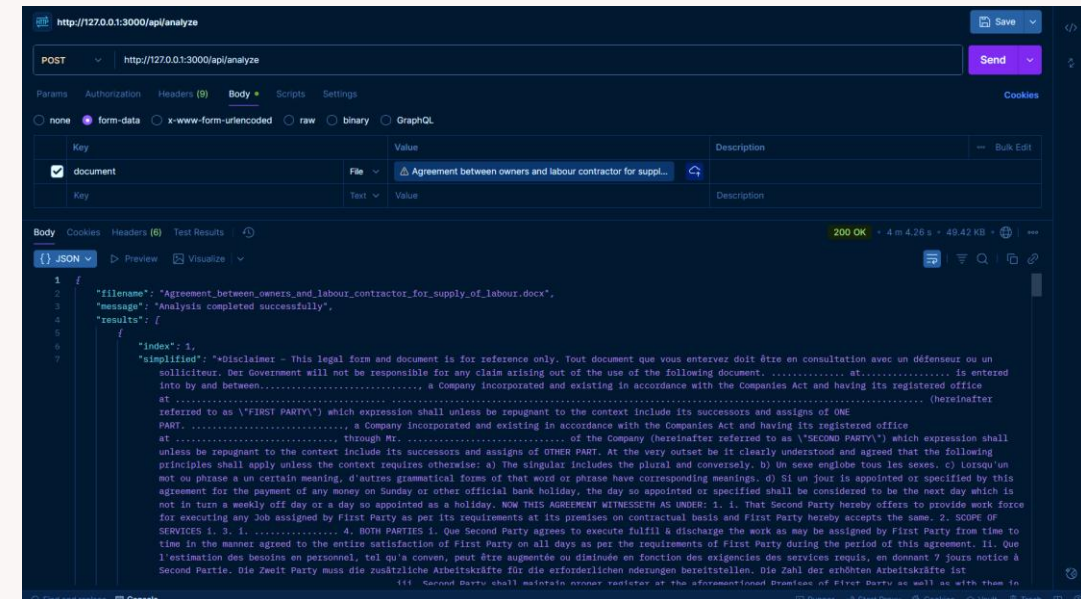
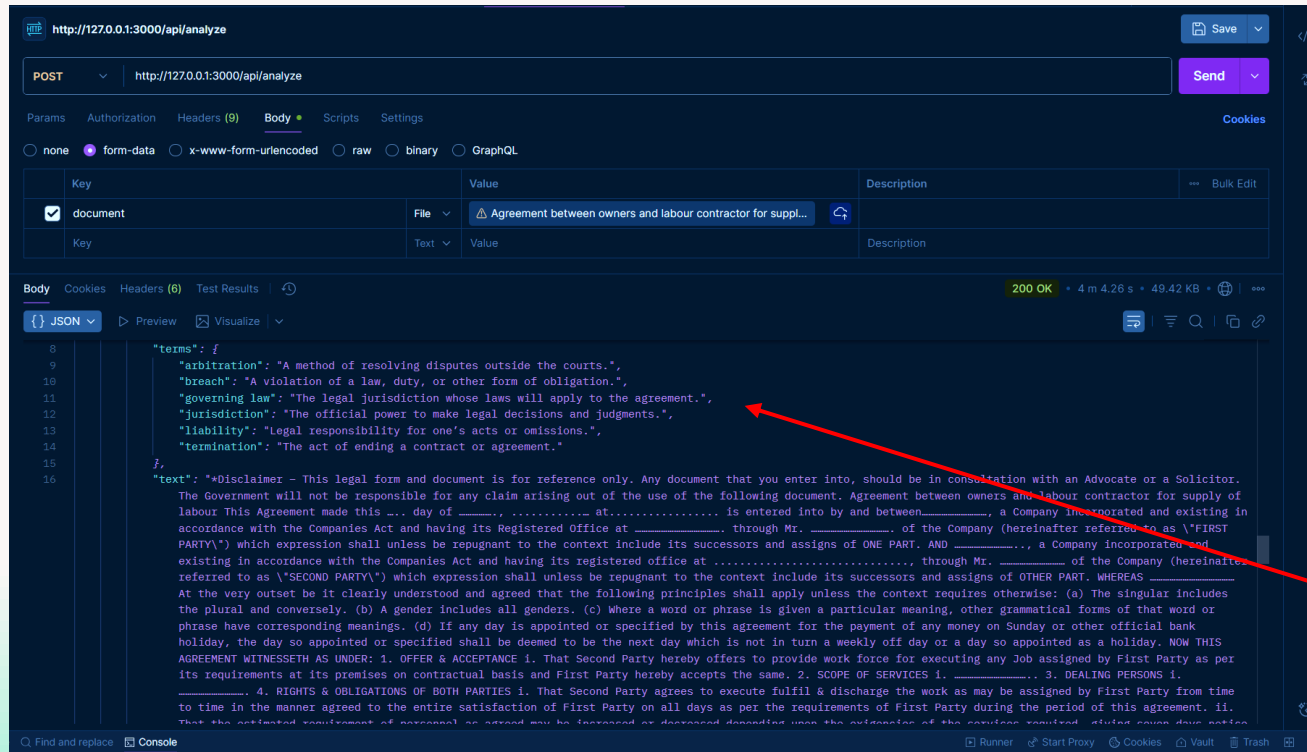
- ❑ **Frontend:** Built using HTML/CSS/JS with a clean UI following *layout.html* theme.
- ❑ **Backend:** Implemented using **Flask** for routing and model orchestration.
- ❑ **AI Processing:** BERT/NLP HF models handle clause detection and simplification logic.
- ❑ **Database:** User and analysis data securely stored in **Firestore (NoSQL)** collections.
- ❑ **Deployment:** Docker Desktop



Application Programming Interface (API) Testing

❑ Testing Tools Used:

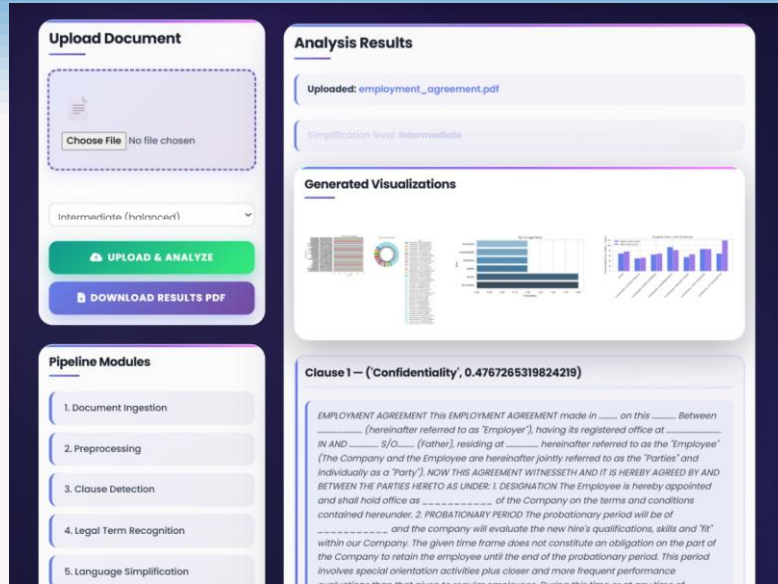
- ❖ **Postman:** API validation
- ❖ **Pytest:** Backend module testing
- ❖ **Firebase Console:** Firestore data integrity checks



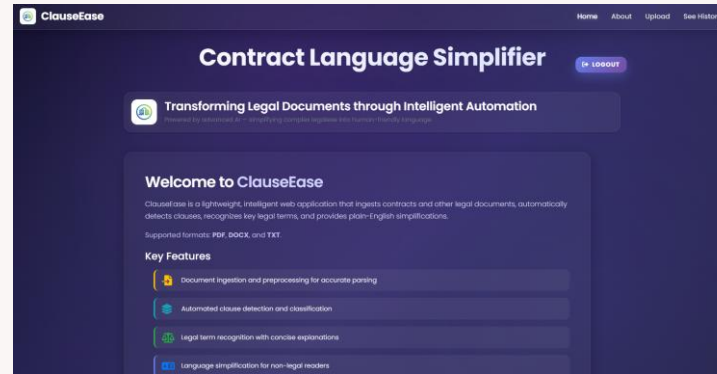
Simplified text output

Legal Terms recognized from the document

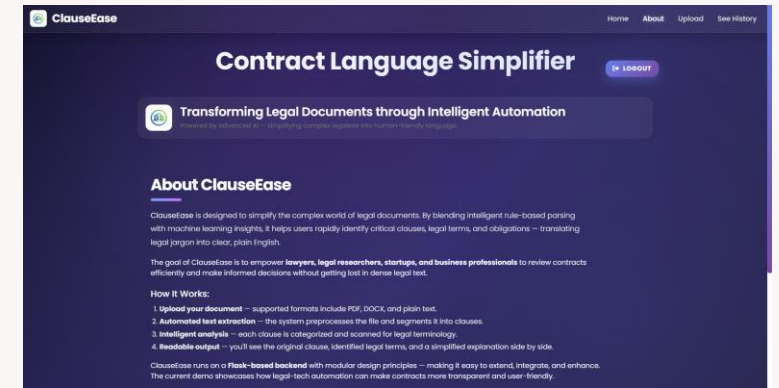
Snapshots



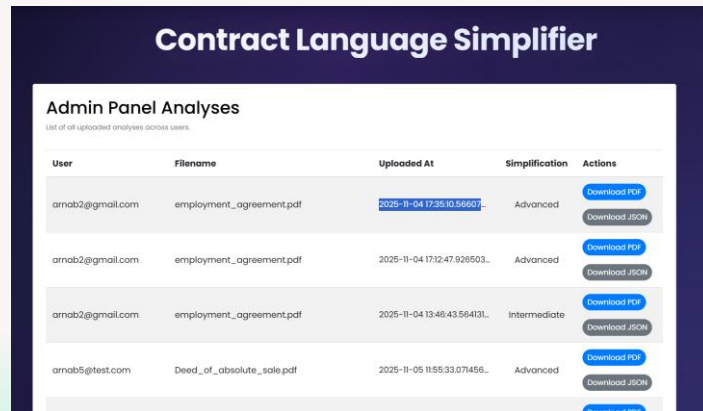
Img1: Analysis Report



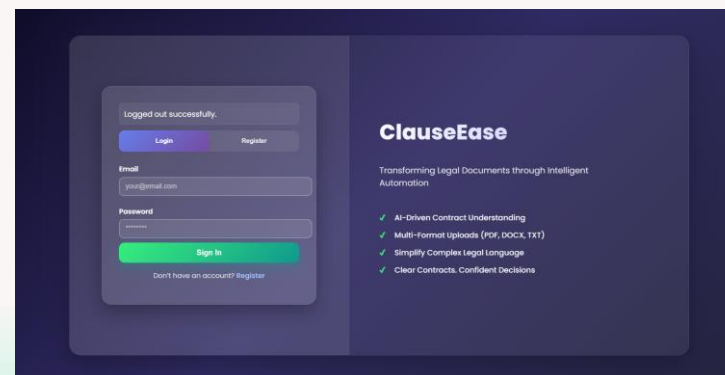
Img2: Home page



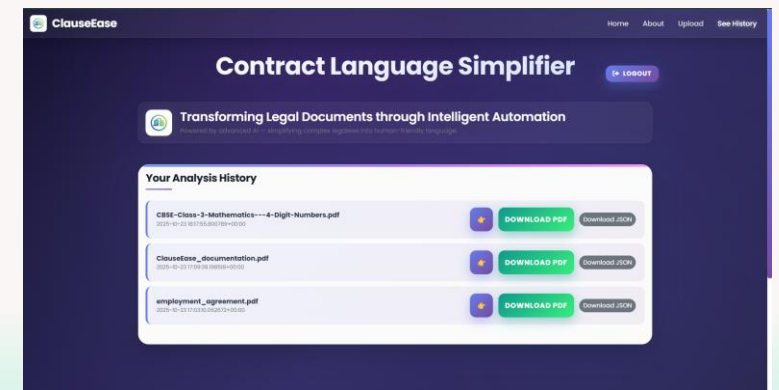
Img4: About page



Img6: Admin panel



Img3: Login/ Signup Page



Img5: See history page

Conclusion

Summary

ClauseEase successfully simplifies complex legal contracts using AI and NLP. It reduces the effort required to interpret legal documents and improves accessibility for users without legal expertise.

Future Enhancements

- Integration of multilingual support for global use.
- Real-time clause comparison across multiple contracts.
- Improved summarization using advanced LLMs.
- Legal chatbot assistant for query-based explanations.
- Integration with document management systems for enterprise use.

References

- ❑ Hugging Face Transformers Library.
- ❑ NLTK & SpaCy documentation.
- ❑ Legal Text Simplification Research Papers.
- ❑ Flask & Bootstrap Official Documentation.
- ❑ Infosys Springboard Learning Resources.

Thank you

Project: ClauseEase | Contract Language Simplifier
Infosys SpringBoard Virtual Internship 6.0

ClauseEase

Transforming Legal Documents through Intelligent Automation

- ✓ AI-Driven Contract Understanding
- ✓ Multi-Format Uploads (PDF, DOCX, TXT)
- ✓ Simplify Complex Legal Language
- ✓ Clear Contracts. Confident Decisions