

Problem Statement & Objectives



- ***Project Objectives***
- *Convert **any** contract (scanned or digital) documents into a structured CAD automatically*
- *Ensure all extracted information is **traceable**, correct, and easy to read*
- *Generate outputs in **JSON, DOCX, and PDF***
- *Provide additional tools like **compliance checking** and **conflict detection***
- *Reduce manual work while increasing speed and consistency*
- *Repo Link:*
https://github.com/Deepakc766/UGP2_CAD_Generation

Overall System Architecture



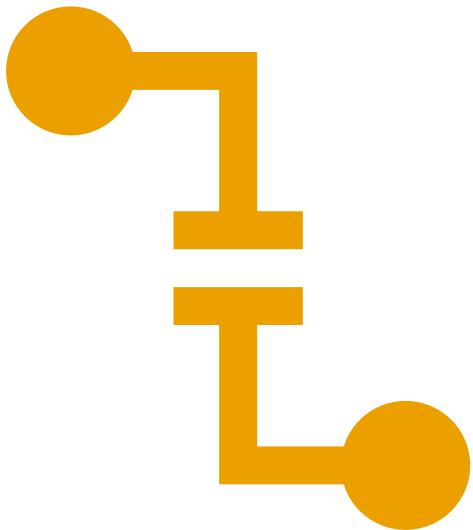
- **How the System Works (High-Level Flow)**
- **PDF Extraction**
 - If text is available → extract directly using pdfplumber
 - If scanned → use OCR (Tesseract)
 - Maintain page markers for traceability
- **LLM Processing**
 - Approaches vary (RAG, Sliding JSON, One-Shot), but all aim to understand contract text
- **CAD Generation**
 - Output structured CAD in JSON
 - Convert JSON into **DOCX and PDF summary**
- **Additional Modules**
 - **Compliance check:** validates rules and requirements
 - **Conflict detection:** identifies contradictions

Approach-1: RAG (Retrieval-Augmented Generation)



- **How it Works**
 - Contract text is broken into small chunks of window size 1024 or 2048 and overlapping of 256/128 token.
 - Chunks are converted into embeddings for similarity searching.
 - Stored in Fiaass(vector database) for efficient searching
 - When asked a question, the system retrieves the most relevant chunks and return on console and hosted using local port and streamlit
 - LLM (FLAN-T5) generates answers and CAD sections and better summarisation
- **Pros**
 - Good for interactive Q&A
 - Works well for smaller documents but not for large contract document like 150+ tokens because LLM(fplant-t5) has maximum token limit of 1024.
- **Cons**
 - CAD sections generated separately → inconsistent final output
 - May miss cross-section linkages (global context missing and confusion in similarity)

Approach-2: Sliding Windows



- **How it Works**
- Entire contract is split into **overlapping token windows again into window size of 1024 and overlap of 128 tokens.**
- For each window, LLM outputs a **strict JSON block**
- All JSON blocks are merged into a final CAD
- Page references are added by searching for quotes in extracted pages
- **Advantages**
- More stable structure than RAG
- Works better for large documents but if LLM has large token size limit like GPT-4,5 or claude.
- **Limitations**
- Still not fully “global”—final consistency depends on merging and leads to bad results sometimes.
- Hard to manage if the contract is extremely long above 100 pages.

Approach-3: One-Shot ChatGPT API (Final Method)

The screenshot shows a dark-themed web application interface for generating CAD (Contract Automation Data) from contracts. On the left, a sidebar titled "Model & Controls" includes a dropdown for "Select OpenAI model" (set to "gpt-4.1"), a temperature slider (set to 0.00), and dropdowns for "Max output tokens (CAD JSON)" (set to 8000) and "Summary PDF: no. of salient features rows" (set to 6). At the bottom of the sidebar is a file upload area with a "Drag and drop file here" placeholder and a "Browse files" button.

The main content area has four sections:

- Chat with the Contract (One-Shot):** A blue button labeled "Upload & Extract a PDF to enable chat."
- CAD Generator:** A blue button labeled "Upload & Extract a PDF to enable CAD generation."
- Compliance Check (One-Shot per rule):** A blue button labeled "Upload a contract first."
- Conflict Detection:** A blue button labeled "Upload a contract first."

- **How the Final Approach Works**
- The **entire contract** (after pdfplumber + OCR) is passed into a **large-context ChatGPT model** — no chunking, no retrieval, no merging.
- A **strict JSON CAD schema** (\approx 45+ fields across salient features, payment, notices, risks, disputes, etc.) ensures the model outputs a **complete and consistent CAD**.
- The model is forced to produce **valid JSON** using `response_format="json_object"` and rules such as: *unknown → null/false/[] if not found*
- Because the whole contract is read at once, the model understands:
 - cross-clause dependencies
 - temporal relationships
 - party roles, dates, and payment logic
- The output is a **single CAD JSON object**—no sliding windows or merging like Approach-2.
- **Post-Processing Pipeline**
- **JSON Validation & Repair** – fixes minor formatting issues.
- **DOCX Generation** – section headings + tables in Arial 11pt for professional readability.
- **PDF Summary** – clean 1-page report created using reportlab(although we can change the number of pages from left bar).

Why Approach-3 Is the Best

- **1. Highest Accuracy**
- No missing clauses because the **whole contract** is seen at once.
- Eliminates window mismatch, overlap drifting, and retrieval errors from earlier approaches.
- Ensures dates, parties, payment terms, and DLP are all **globally consistent**.
- **2. Most Consistent Output**
- CAD fields stay synchronized across the JSON schema.
- Employer/Contractor information does not conflict.
- All payment sections (advance, retention, final settlement) align perfectly.
- **3. Simplest & Most Reliable Pipeline**
- Compared to earlier methods:
- No embeddings
- No Chroma DB
- No token windows or merging
- No post-LLM recomposition

⚠ Conflict Detection

Include LLM conflict miner and merge with regex (recommended) ?

Run Conflict Detection Export Conflicts JSON Export Conflicts CSV

LLM conflict miner not found in this file. Only regex conflicts will be shown.

3 potential conflicts detected.

1. Category: Financial / Payment ↗

Type: retention_mismatch
Severity: High
Confidence: 0.65
Message: Different retention percentages found: [4, 5, 7]%.
Suggested quick fix: Clarify a single retention % and update annex.

Easy Readability: CAD Generation

- What Makes the CAD Easy to Read?**
- One-shot JSON ensures each section of the CAD is well-structured
- DOCX contains neat tables for each major part
- PDF summary presents all key information on one page
- Eliminates unnecessary text and extracts only meaningful details
- Sample Example on Right side of Notice Clause.

4. Important clauses pertaining to project progress - EOT, Escalation, Variation, Suspension, etc.

Sl. No.	Topic	Clause No.	Summary
1	Extension of Time (EOT)	3.5, 7.4, 26.3	EOT may be granted for force majeure, delays in drawings/approvals, or suspension >30 days; claims must be notified within 14 days
2	Escalation	Annexure B	Price escalation applicable beyond 180 days from commencement; formula based on indices for cement, steel, fuel, labour, misc.
3	Variation	7.1-7.3	Employer may instruct variations; Contractor to submit quotation within 7 days; Engineer's determination binding until arbitration
4	Suspension	26.1-26.4	Employer may instruct suspension; EOT if >30 days; payment for idle resources by mutual agreement
5	Liquidated Damages (LD)	19.2, Annexure B	Delay in mobilization beyond 15 days from site possession: LD 0.05%/day; cap 10% of Contract Price

Accuracy: Conflict Detection

- **Implemented Conflict Checks**
- Your system automatically detects contract inconsistencies such as:
- Commencement date vs site possession date
- Payment term mismatch
- Retention percentage mismatch
- Arbitration vs court contradiction
- Security/performance guarantee inconsistencies
- Other 10 checks are there in code .
- **Why This Matters :**
- Conflicts cause disputes during project execution
- Automated detection helps contract reviewers catch issues early

⚠️ Conflict Detection

Include LLM conflict miner and merge with regex (recommended) ?

Run Conflict Detection **Export Conflicts JSON** **Export Conflicts CSV**

LLM conflict miner not found in this file. Only regex conflicts will be shown.

3 potential conflicts detected.

1. Category: Financial / Payment ↗

Type: retention_mismatch
Severity: High
Confidence: 0.65
Message: Different retention percentages found: [4, 5, 7]%.
Suggested quick fix: Clarify a single retention % and update annex.

Completeness: Compliance Check

- **What It Does**
- Validates whether the contract contains specific rules or required sections
- Each rule produces a small JSON output with:
 - Is the rule present?
 - Summary of text
 - A direct quote
 - Page or clause number
 - Confidence score
- **Why It Helps**
- Ensures contract meets legal/technical standards
- Helps users identify missing or incomplete clauses
- We can also download compliance report as well.

 **Compliance Check (One-Shot per rule)**

Enter compliance rules (one per line)

Does the contract specify payment terms?
Is there a termination clause?
What is the governing law?

Run Compliance Check **Export Compliance CSV**

Compliance Report

Show raw compliance debug

Rule: Does the contract specify payment terms?
Present: True
Summary: Yes, payment terms are specified, including advance, interim, and final payment timelines, but with some internal conflicts.
Quote: Mobilization Advance: 10% of Contract Value against Bank Guarantee. Interim Payments: 90% of the value of work done, payable within 30 days from certification by the Engineer. Annexure C—Payment Schedule: Interim payments stated as within 45 days of certification.
Sources: page 2, clause 4.2, clause 4.3, page 11, Annexure C
Confidence: 0.95

Results & Comparison

- Overall Comparison of Approaches
- Approach-1 (RAG)
- Works but limited because of separate chunks
- Good for basic Q&A, weak for full CAD
- Approach-2 (Sliding JSON Windows)
- More robust than RAG
- Better structure but merging is complex
- Approach-3 (One-Shot ChatGPT)
- Best performance overall
- Highest accuracy, consistency, readability
- Required for production-level CAD generation

CAD Generator

Generate CAD (JSON + DOCX + PDF)

CAD generated.

Download CAD JSON

Download CAD DOCX

Download CAD PDF

Limitations & Future Work

- **Current Limitations**

- **Privacy Concern for Legal Documents:**

Sending confidential contracts to cloud-based models may violate company or legal policies.

Solution: Use an **Enterprise API key** or private deployment with strict data-retention controls.

- **Token Size Limitations:**

Very long contracts can exceed the model's maximum context window, causing incomplete processing.

Solution: Use models with **larger context sizes** or apply adaptive chunking only when needed.

- **OCR Errors in Scanned PDFs:**

Low-quality scans lead to incorrect text extraction, affecting accuracy of CAD, compliance, and conflicts.

- **Conflict Detection Coverage:**

Not all possible legal inconsistencies are detected; current system implements only selected major rules.

- **Formatting Variations:**

Contracts differ in structure (tables, annexures, multi-column layouts), causing extraction inconsistencies.

Solution :

API(enterprise) Key might work ,They won't train model on private data and large token size model like gpt-5