Salesforce Project Documentation

Legal Contract Analyzer

Name : N K LAKSHMI NARASIMHA MURTHY

Email : nlnara20lm@gmail.com

Roll No : 22B81A6684

Legal Contract Analyzer

1. Project Overview

The **Legal Contract Analyzer** is an AI-powered Salesforce-native application designed to automate legal contract review and negotiation. It streamlines contract analysis, reduces legal risks, ensures compliance, and accelerates negotiations by providing AI-assisted suggestions.

Key Features / Business Value

- Automated clause extraction and risk scoring.
- AI-driven negotiation suggestions.
- Real-time dashboards and status notifications.
- Enterprise-grade security, scalability, and audit readiness.
- Tested and validated with **81% Apex test coverage**.

2. Objectives

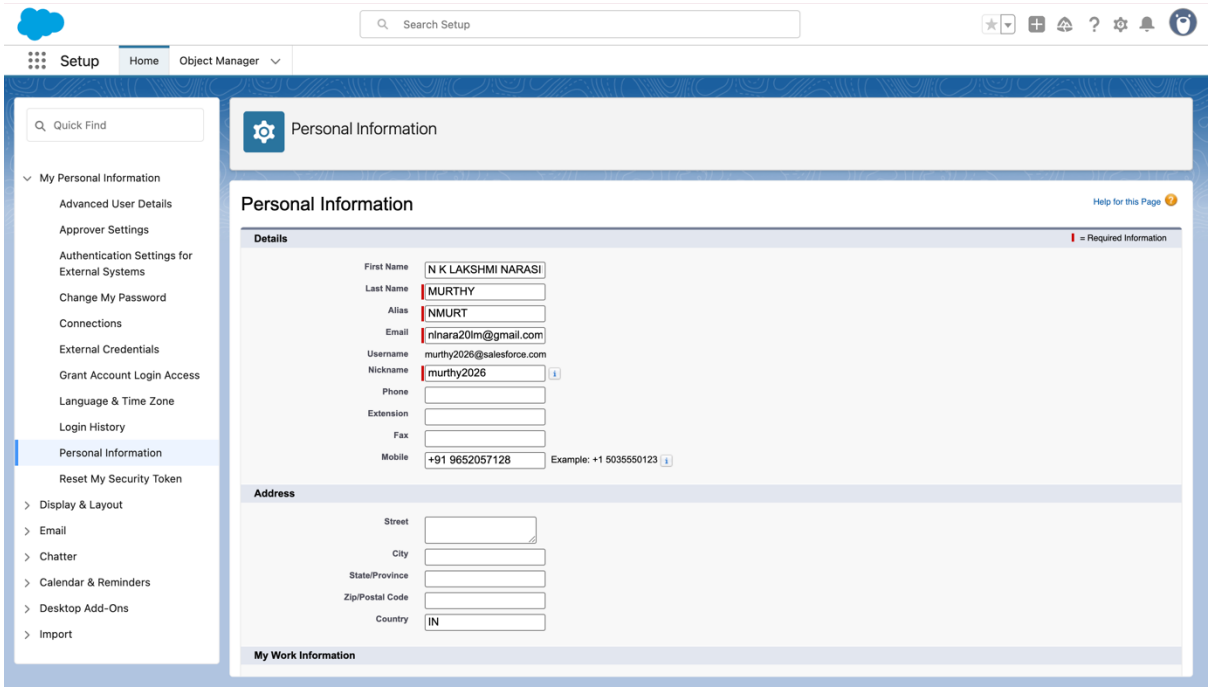
- Automate contract reviews by extracting clauses and assessing risks.
- Standardize accuracy through clause detection and risk scoring.
- Accelerate negotiations with AI-generated alternative wordings.
- Ensure compliance with GDPR/CCPA and legal policies.
- Provide visibility via dashboards, reports, and audit trails.

Phase 1 – Problem Understanding & Industry Analysis

- **Requirement Gathering**: Contracts take time, high risk of human error.
- **Stakeholders**: Legal teams, law firms, compliance officers, enterprises.
- **Business Process Mapping**: From contract upload → AI analysis → negotiation → approval.
- **Industry Use Case**: LegalTech is trending, focus on risk/compliance automation.

Phase 2 – Org Setup & Configuration

- Salesforce **Enterprise Edition Developer Org**.
- Setup included company profile, users, roles, and permission sets.
- Configured fiscal year and business hours.
- Sandbox used for development, changes deployed via **Change Sets**.



2.1 Developer Org Information

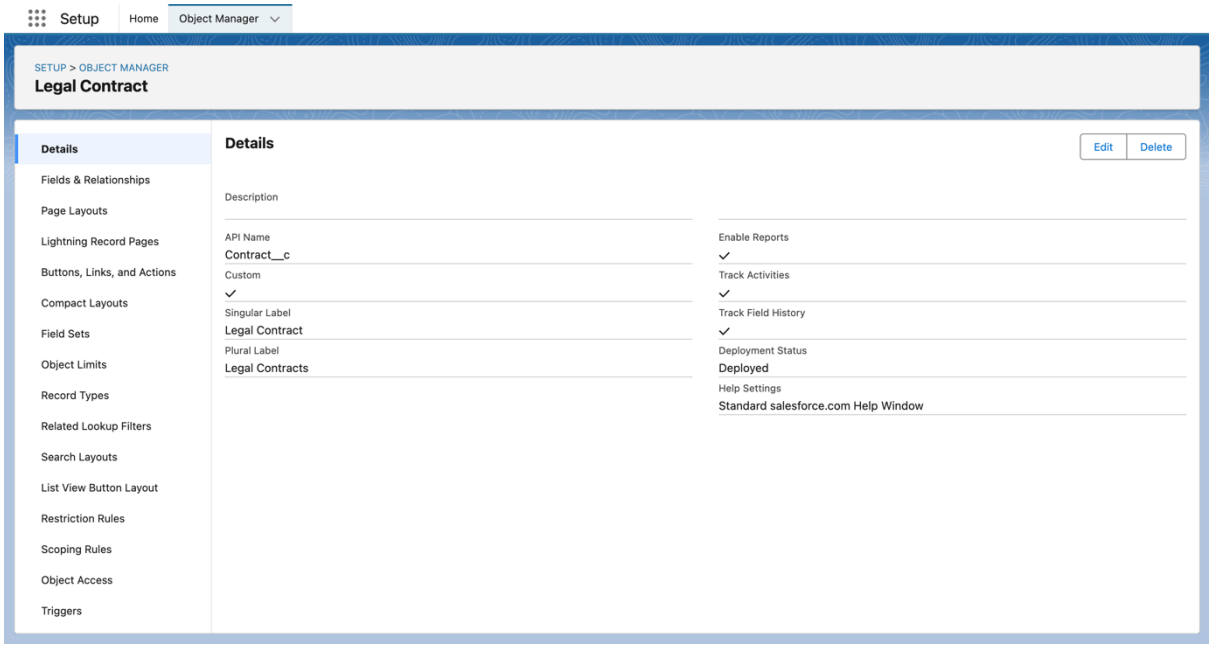
Phase 3 – Data Modeling & Relationships

Custom Objects

- **Contract__c** → Metadata, risk, status.
- **Clause__c** → Clause text, type, risk level.
- **Status_Change_Event__e** → Platform event for async processing.

Relationships

- Master-Detail: Clause__c → Contract__c.
- Lookup: Uploaded_By__c, Requested_By__c → User.



3.1 Custom Contract Object

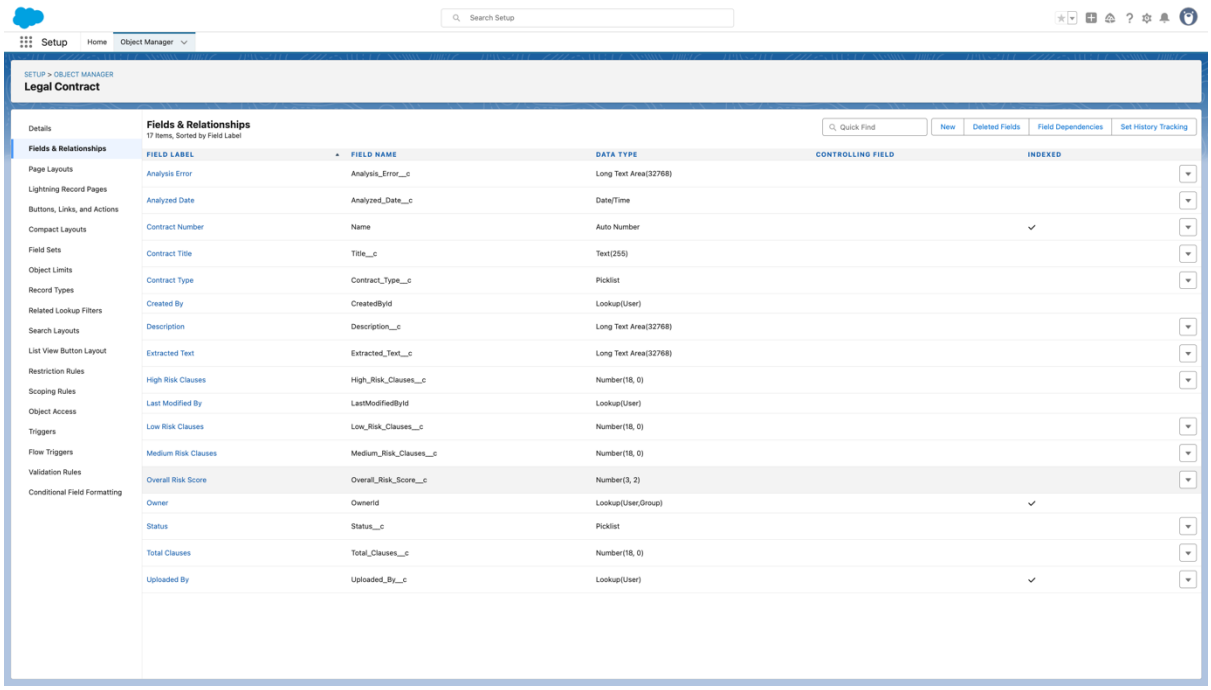
♦ Usage of Contract and Clause Objects

Contract__c (Parent Object)

- This is the **central object** in the application.
- It represents the **entire legal contract** uploaded or created by the legal team.
- Key details stored:

- Contract Title, Parties, Effective Date, Status (Draft/Approved/Rejected).
- Overall **Risk Score** (calculated from all its clauses).
- Metadata such as Uploaded By, Department, and Type (NDA, Vendor Agreement, Employment Contract, etc.).

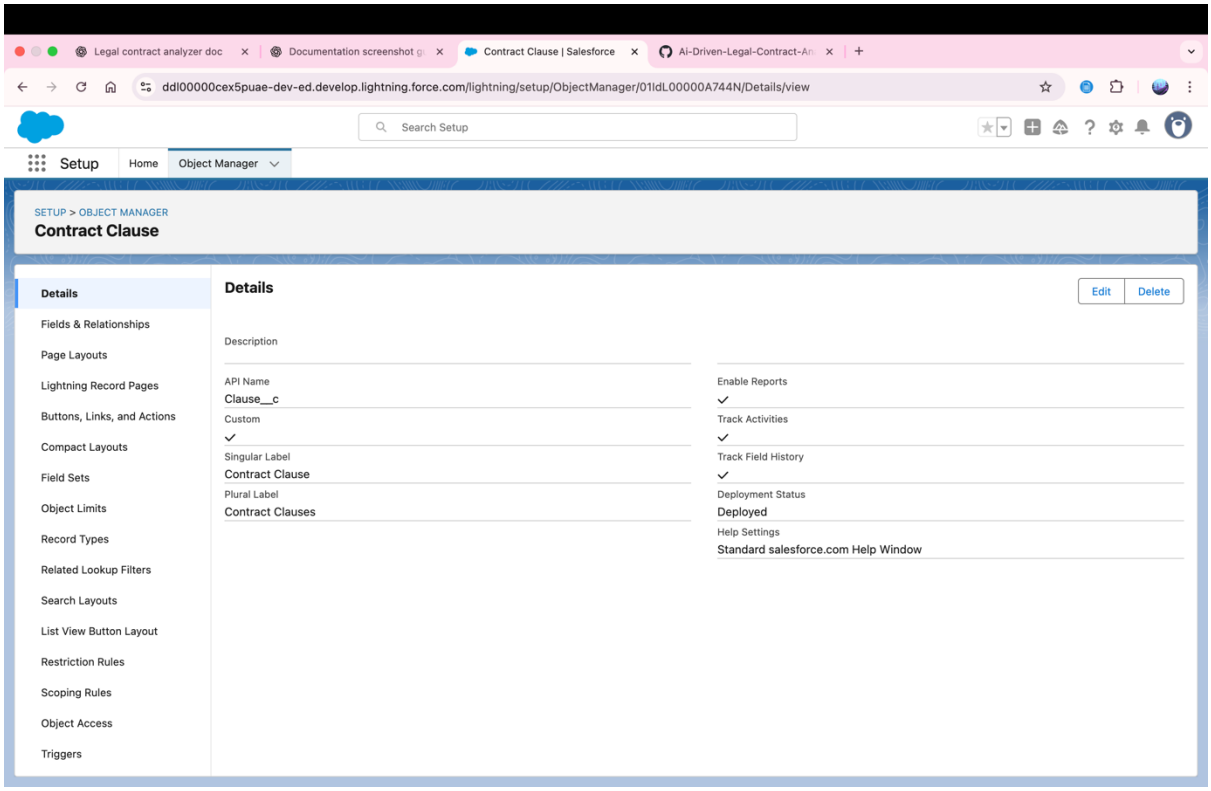
- Usage: Contracts are the **entry point** for analysis — once a contract is created/uploaded, AI services parse it into smaller parts (clauses).



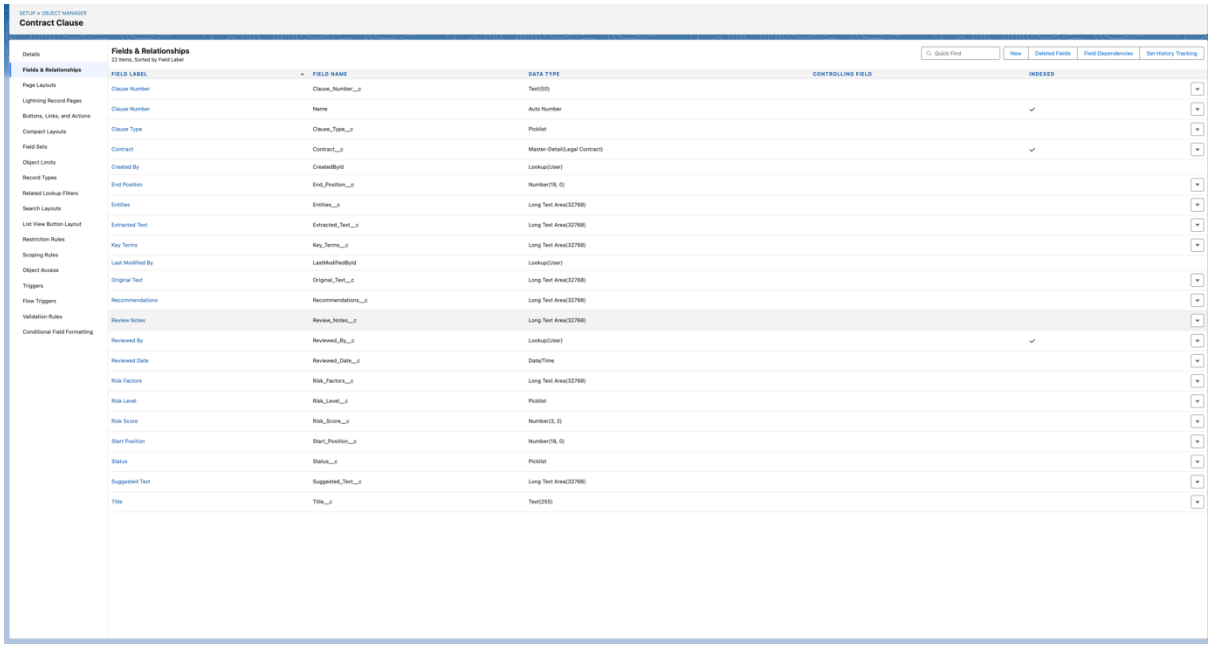
3.2 Fields and Relationships of Contract Object

Clause__c (Child Object)

- Each **clause** within a contract is stored as a separate record under **Clause__c**.
- Key details stored:
  - Clause Text, Category (Termination, Liability, Confidentiality, etc.).
  - Risk Level (Low, Medium, High).
  - AI-assigned confidence score.
- Usage:
  - Breaks down contracts into **granular pieces** for analysis.
  - Makes it easier for the AI and legal teams to assess **specific risks** without reviewing the full contract manually.
  - Provides a base for **AI Negotiation Suggestions** (e.g., “Replace termination period from 90 days → 30 days”).



3.3 Clause Custom Object



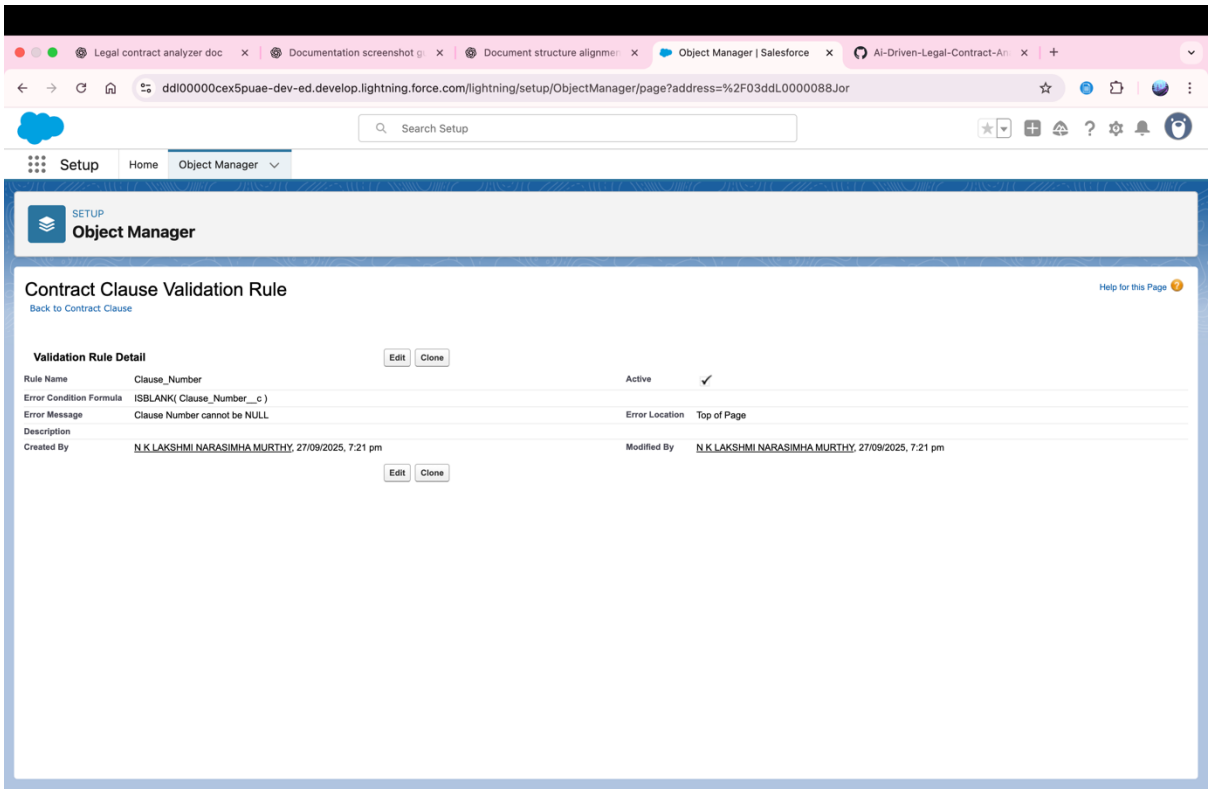
3.4 Clause Object Fields and Realtionships

🔷 Relationships Between Contract and Clause

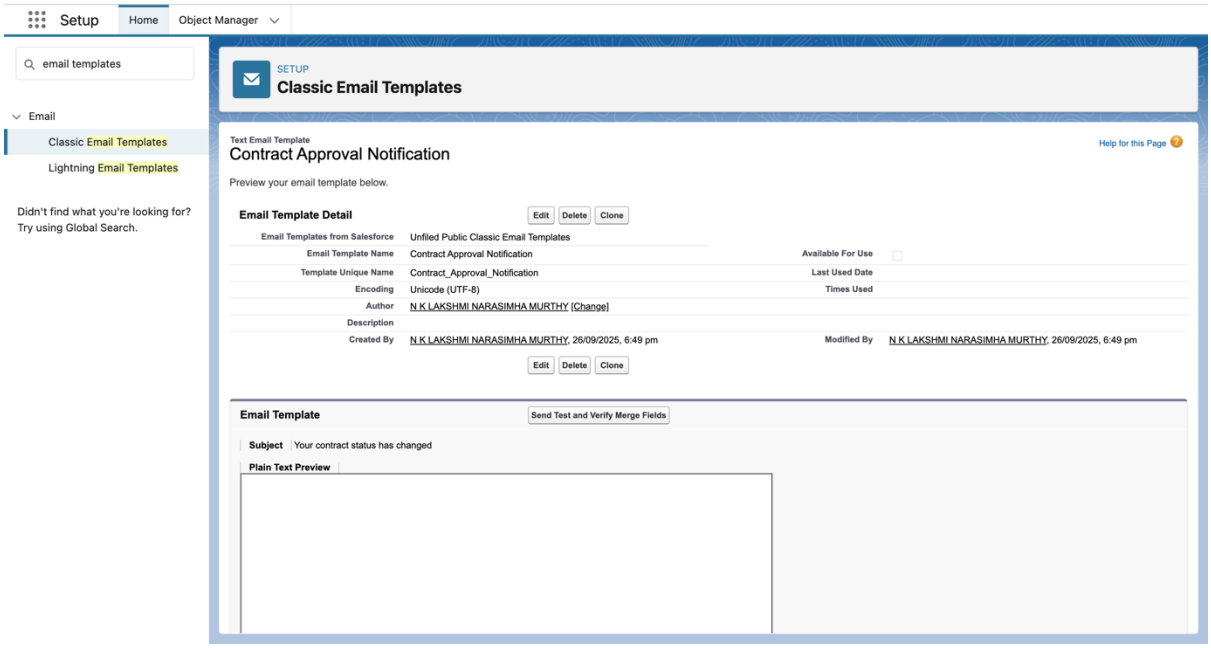
- **Master-Detail Relationship: Clause__c → Contract__c**
  - Every clause belongs to **exactly one contract**.
  - If a contract is deleted, its associated clauses are automatically deleted (data integrity).
  - Aggregated roll-up summaries are enabled: e.g., *“Count of High-Risk Clauses per Contract.”*
- **User Lookups (Uploaded_By__c, Requested_By__c)**
  - Provides accountability: who uploaded the contract and who requested the review.
  - Useful for audit trails and compliance tracking.

Phase 4 – Process Automation (Admin)

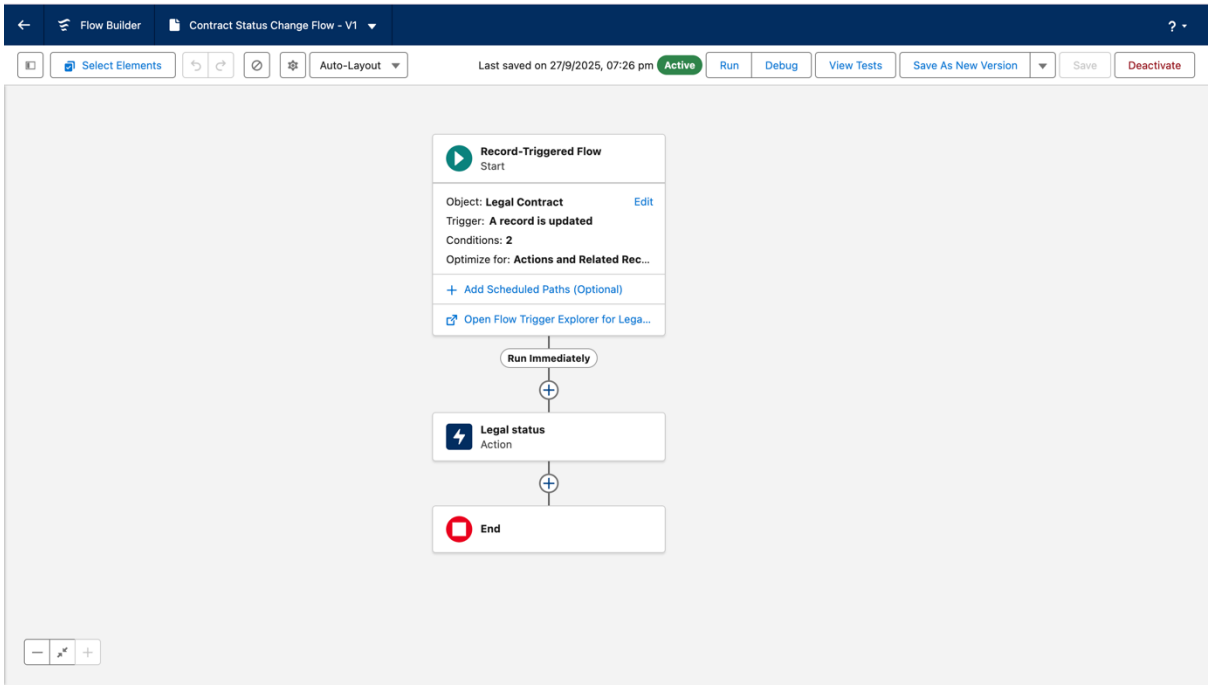
- **Validation Rules**: Required fields (Title, Risk Level, Status).
- **Auto-Numbering**: CT-{00000}, CL-{00000}, NS-{00000}.
- **Flows**: (Planned) Record-triggered flows for stakeholder notifications.
- **Approval Process**: Contracts can move through review → approve/reject with email alerts.



4.1 Validation Rule for Clause Object



4.2 Email Template for Contract Approval Notification



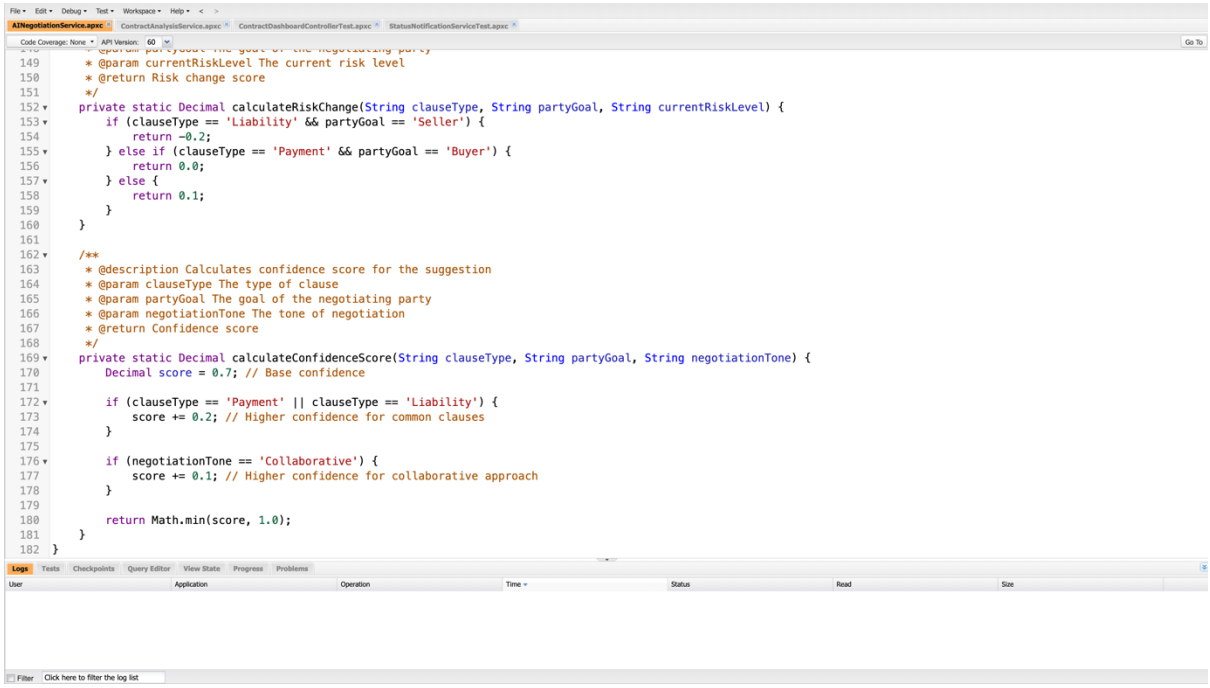
4.3 Contract Status Flow (Approved or Rejected sends Notification through Email)

Phase 5 – Apex Programming (Developer)

- **ContractAnalysisService**: Parses contract text, extracts clauses, computes risk.
- **AINegotiationService**: Generates AI negotiation text.
- **ContractDashboardController**: Supplies KPIs to LWC.
- **StatusNotificationService**: Automates approval/rejection emails.

Triggers

- ContractStatusTrigger (fires on approval/rejection).
- ClauseStatusTrigger (fires on clause status updates).



```
    * @param currentRiskLevel The current risk level
    * @return Risk change score
    */
   private static Decimal calculateRiskChange(String clauseType, String partyGoal, String currentRiskLevel) {
       if (clauseType == 'Liability' && partyGoal == 'Seller') {
           return -0.2;
       } else if (clauseType == 'Payment' && partyGoal == 'Buyer') {
           return 0.0;
       } else {
           return 0.1;
       }
   }

   /**
    * @description Calculates confidence score for the suggestion
    * @param clauseType The type of clause
    * @param partyGoal The goal of the negotiating party
    * @param negotiationTone The tone of negotiation
    * @return Confidence score
    */
   private static Decimal calculateConfidenceScore(String clauseType, String partyGoal, String negotiationTone) {
       Decimal score = 0.7; // Base confidence

       if (clauseType == 'Payment' || clauseType == 'Liability') {
           score += 0.2; // Higher confidence for common clauses
       }

       if (negotiationTone == 'Collaborative') {
           score += 0.1; // Higher confidence for collaborative approach
       }

       return Math.min(score, 1.0);
   }
}
```

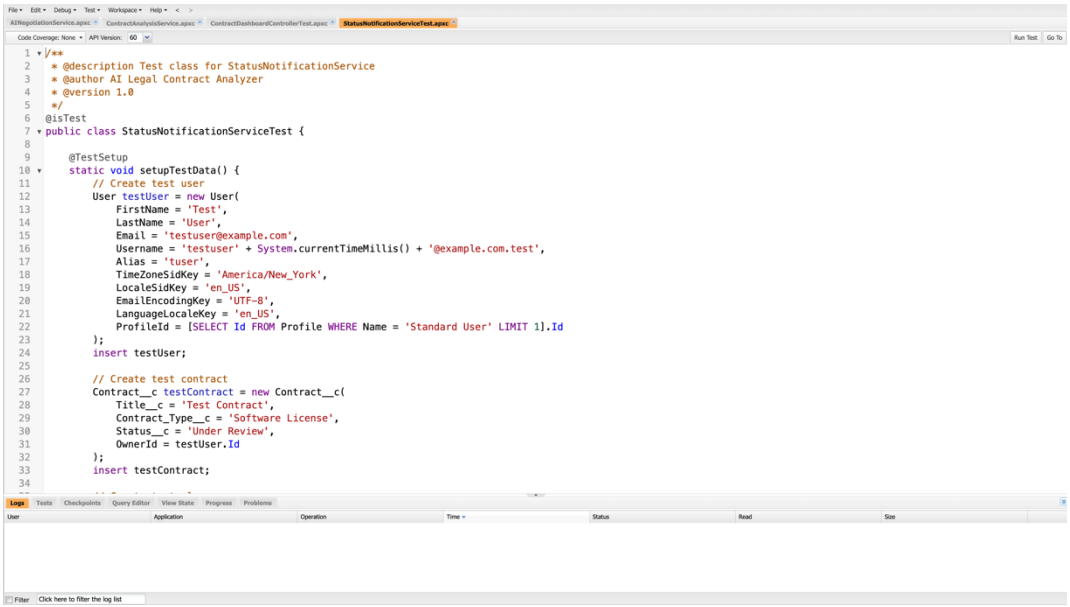
5.1 AINegotiationService

This Apex service class is designed to provide AI-powered negotiation suggestions for contract clauses within Salesforce. It analyzes individual clauses, takes into account the negotiating party's goals and preferred tone, and generates suggested revisions along with explanations, risk assessments, and confidence scores. The class also stores generated suggestions as Salesforce records, enabling contract managers and legal teams to track and apply AI-driven negotiation strategies directly in their workflows.

```
/**
 * @description Test class for ContractDashboardController
 * @author AI Legal Contract Analyzer
 * @version 1.0
 */
@isTest
public class ContractDashboardControllerTest {

    @TestSetup
    static void setupTestData() {
        // Create test user
        User testUser = new User(
            FirstName = 'Test',
            LastName = 'User',
            Email = 'testuser@example.com',
            Username = 'testuser' + System.currentTimeMillis() + '@example.com.test',
            Alias = 'tuser',
            TimeZoneSidKey = 'America/New_York',
            LocaleSidKey = 'en_US',
            EmailEncodingKey = 'UTF-8',
            LanguageLocaleKey = 'en_US',
            ProfileId = [SELECT Id FROM Profile WHERE Name = 'Standard User' LIMIT 1].Id
        );
        insert testUser;

        // Create test contracts
        List<Contract__c> contracts = new List<Contract__c>();
        for (Integer i = 0; i < 5; i++) {
            contracts.add(new Contract__c(
                Title__c = 'Test Contract ' + i,
                Contract_Type__c = 'Software License',
                Status__c = i < 2 ? 'Analyzed' : 'Under Review',
                Overall_Risk_Score__c = 0.5 + (i * 0.1),
                OwnerId = testUser.Id
```

5.2 ContractDashboardController

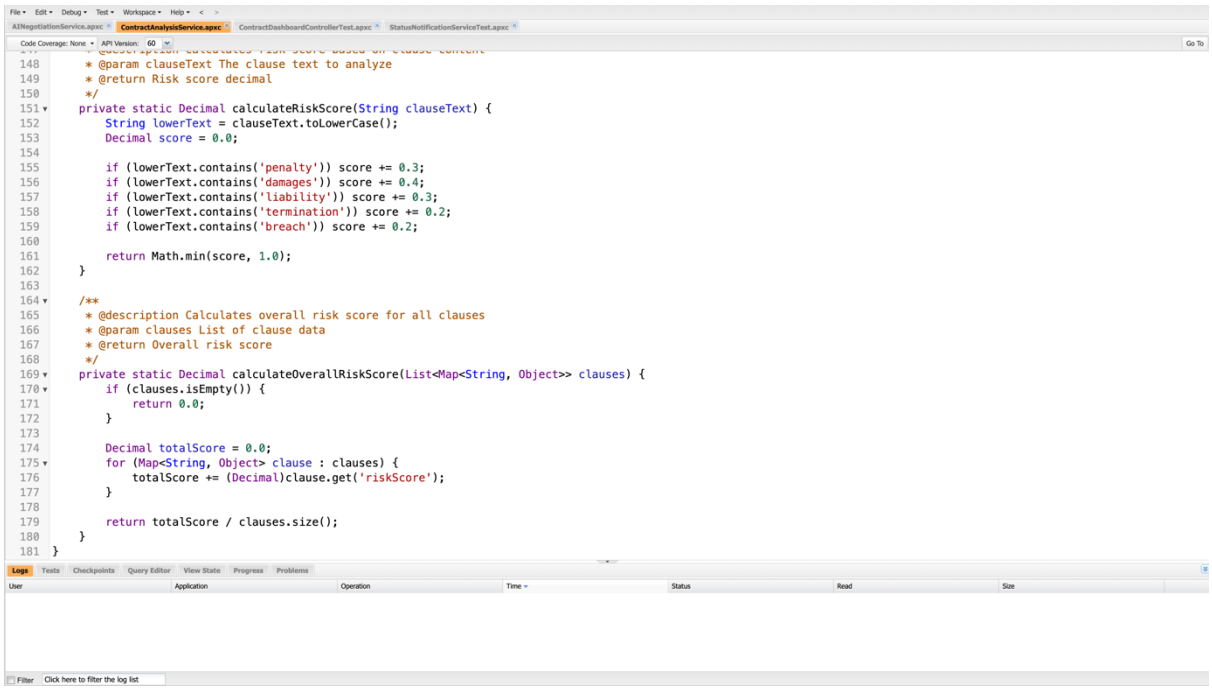
This test class validates the functionality of the ContractDashboardController in Salesforce by covering contract and clause dashboard operations. It sets up sample contract and clause data, and includes tests for retrieving contract statistics, searching and listing contracts, fetching clauses for contracts, and updating contract status and clause risk levels. The tests ensure that the controller's core dashboard features work correctly and that the data management processes are robust and reliable for legal contract analysis workflows.



```
/**
 * @description Test class for StatusNotificationService
 * @author AI Legal Contract Analyzer
 * @version 1.0
 */
@isTest
public class StatusNotificationServiceTest {

    @TestSetup
    static void setupTestData() {
        // Create test user
        User testUser = new User(
            FirstName = 'Test',
            LastName = 'User',
            Email = 'testuser@example.com',
            Username = 'testuser' + System.currentTimeMillis() + '@example.com.test',
            Alias = 'tuser',
            TimeZoneSidKey = 'America/New_York',
            LocaleSidKey = 'en_US',
            EmailEncodingKey = 'UTF-8',
            LanguageLocaleKey = 'en_US',
            ProfileId = [SELECT Id FROM Profile WHERE Name = 'Standard User' LIMIT 1].Id
        );
        insert testUser;

        // Create test contract
        Contract__c testContract = new Contract__c(
            Title__c = 'Test Contract',
            Contract_Type__c = 'Software License',
            Status__c = 'Under Review',
            OwnerId = testUser.Id
        );
        insert testContract;
```

5.3 StatusNotificationService

This test class verifies the proper functioning of the StatusNotificationService in Salesforce, which is responsible for sending status notifications related to contract and clause changes. It creates test users, contracts, and clauses, and runs tests to ensure notifications are sent correctly for both approved and rejected statuses, including edge cases with empty input lists. The class confirms that the service manages notification logic reliably in various scenarios relevant to legal contract management workflows.



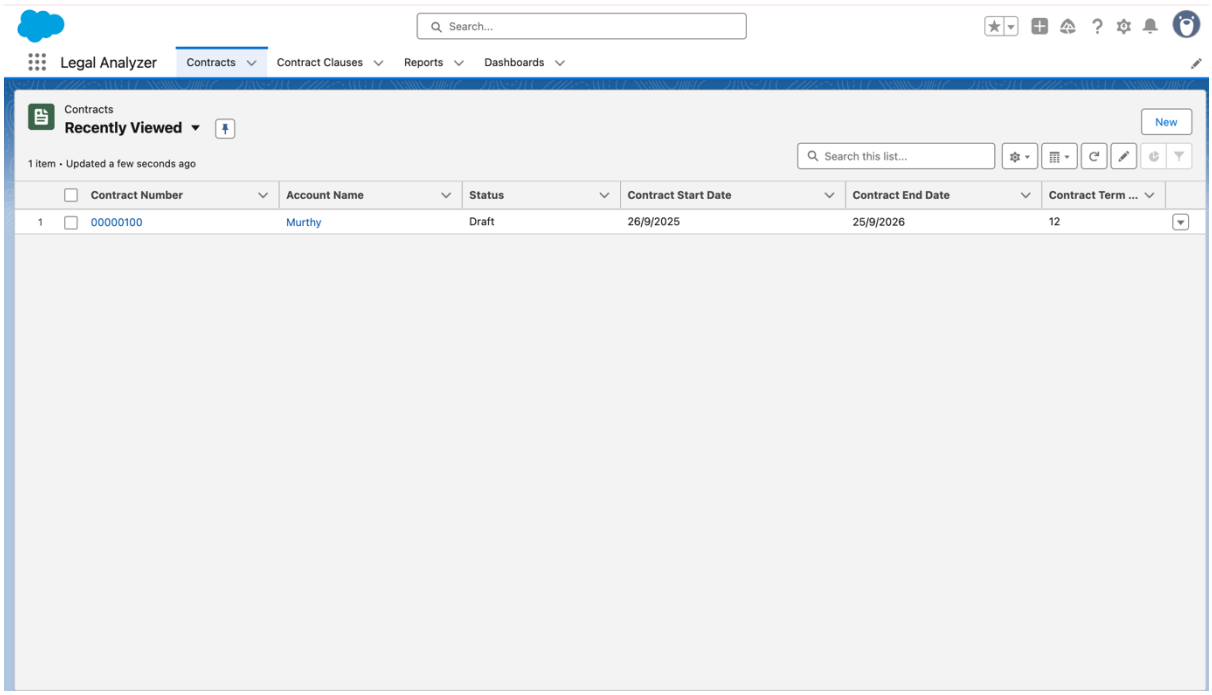
```
     * @param clauseText The clause text to analyze
     * @return Risk score decimal
     */
    private static Decimal calculateRiskScore(String clauseText) {
        String lowerText = clauseText.toLowerCase();
        Decimal score = 0.0;

        if (lowerText.contains('penalty')) score += 0.3;
        if (lowerText.contains('damages')) score += 0.4;
        if (lowerText.contains('liability')) score += 0.3;
        if (lowerText.contains('termination')) score += 0.2;
        if (lowerText.contains('breach')) score += 0.2;

        return Math.min(score, 1.0);
    }

    /**
     * @description Calculates overall risk score for all clauses
     * @param clauses List of clause data
     * @return Overall risk score
     */
    private static Decimal calculateOverallRiskScore(List<Map<String, Object>> clauses) {
        if (clauses.isEmpty()) {
            return 0.0;
        }

        Decimal totalScore = 0.0;
        for (Map<String, Object> clause : clauses) {
            totalScore += (Decimal)clause.get('riskScore');
        }

        return totalScore / clauses.size();
    }
}
```

5.4 ContractAnalysisService

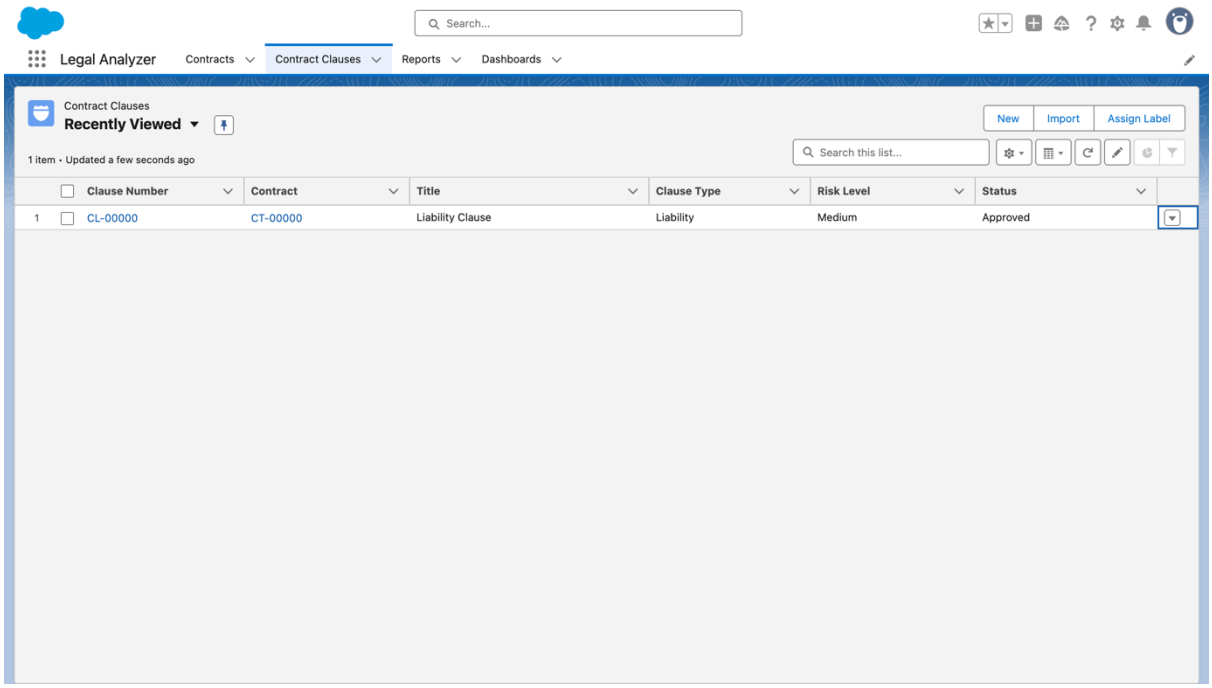
This service class provides contract analysis and clause extraction for legal workflows in Salesforce. It analyzes the extracted text from a contract, identifies key clauses using pattern matching, assesses the risk for each clause, and updates the contract record with risk metrics and clause details. Each identified clause is saved as a separate record, making contract review and risk management easier and more automated.

Phase 6 – User Interface Development

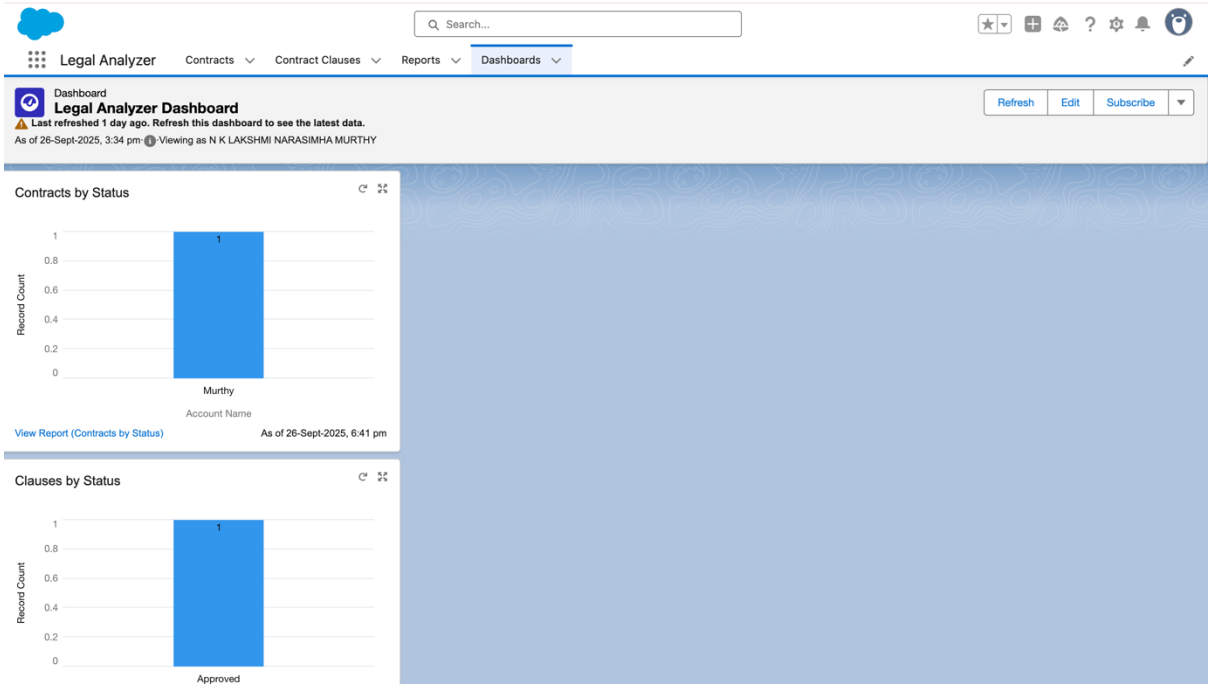
- **Lightning Web Component (LWC): contractDashboard**
  - Displays KPIs, recent contracts, risk distribution chart.
- **Tabs**: Contracts, Clauses, Negotiation Suggestions.
- **Page Layouts** for quick navigation and data visibility.



6.1 Contract User Interface



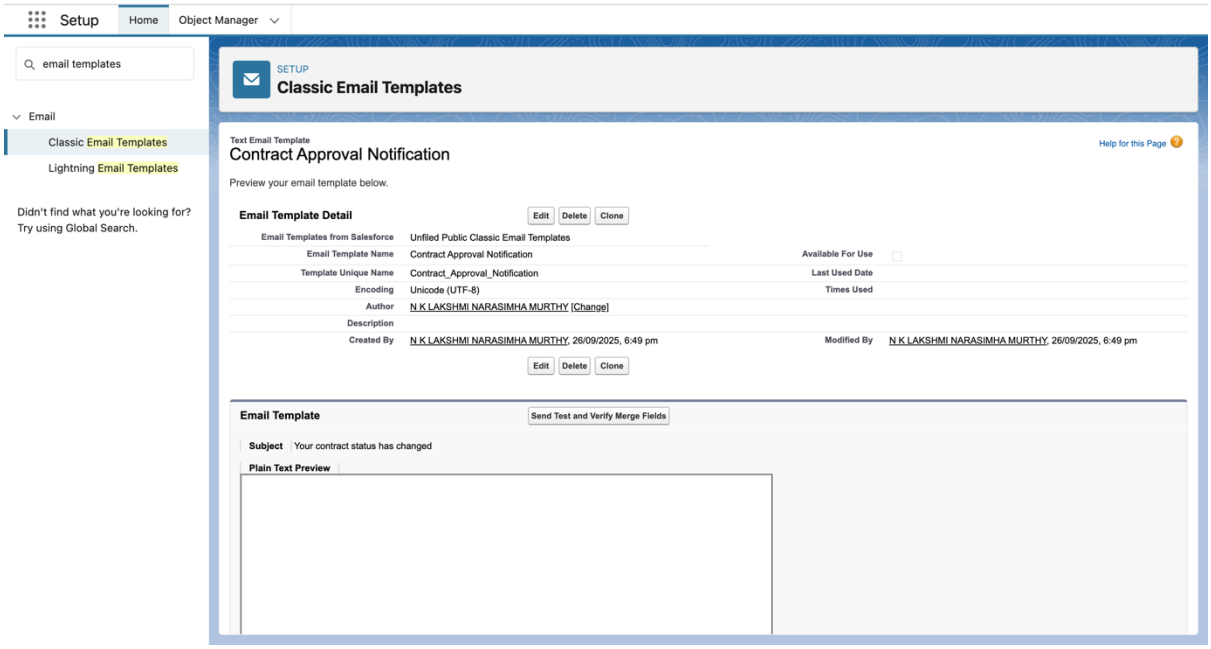
6.2 Contract Clause User Interface



6.3 Legal Analyzer Dashboard with Contract and Clause Status

Phase 7 – Integration & External Access

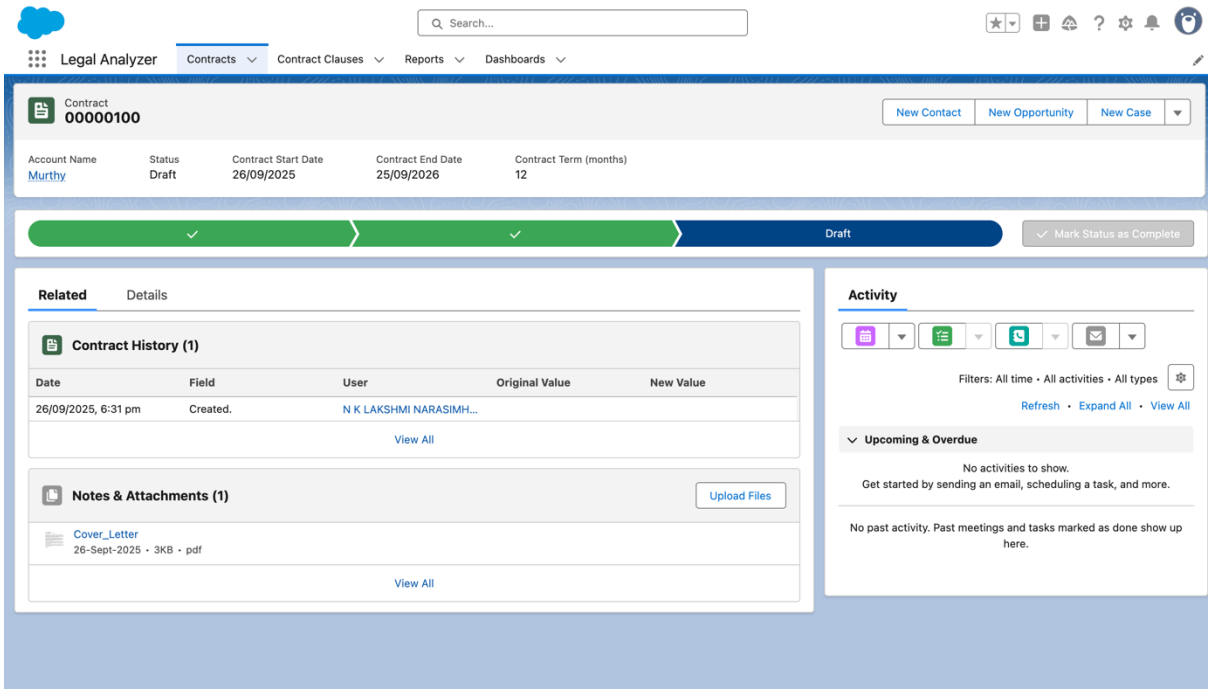
- **Platform Events**: Status_Change_Event__e for async updates.
- **Email Alerts**: Status change notifications.
- **Future Scope**: REST API integration with external AI/NLP contract risk engines.



7.1 Email template configured for approval/rejection notifications.

Phase 8 – Data Management & Deployment

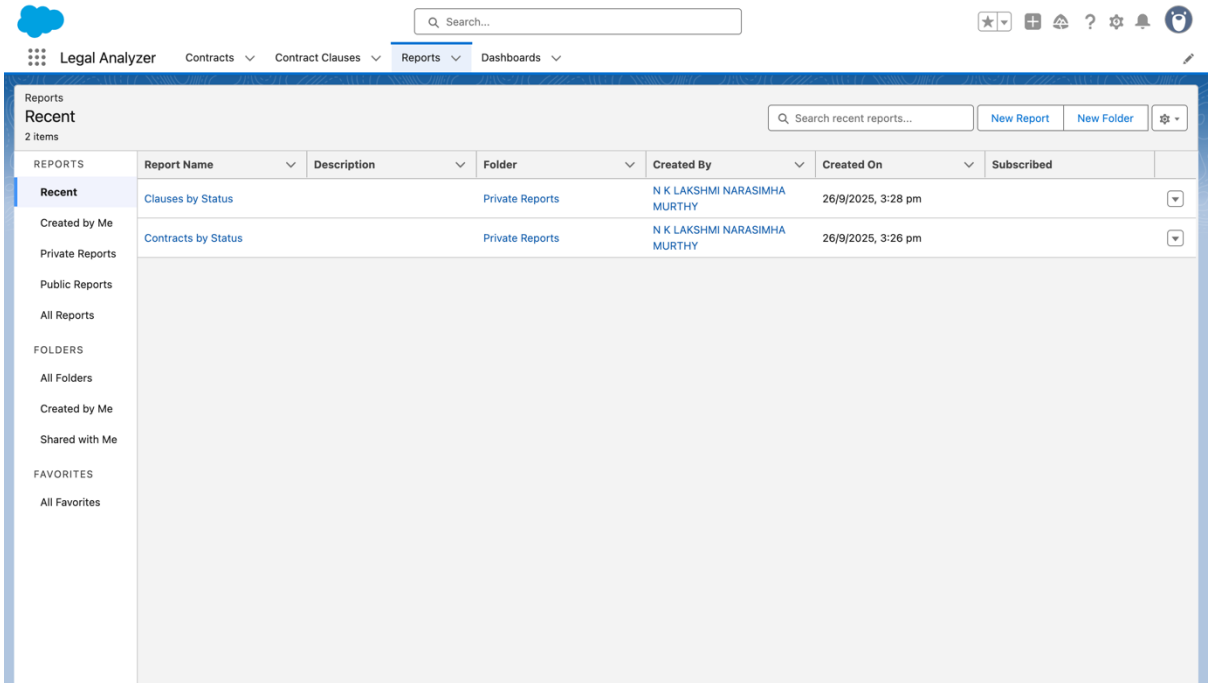
- Data import/export using **Data Loader** for test data.
- Deployment via **Change Sets**.
- Backup strategy: Salesforce Data Export service.



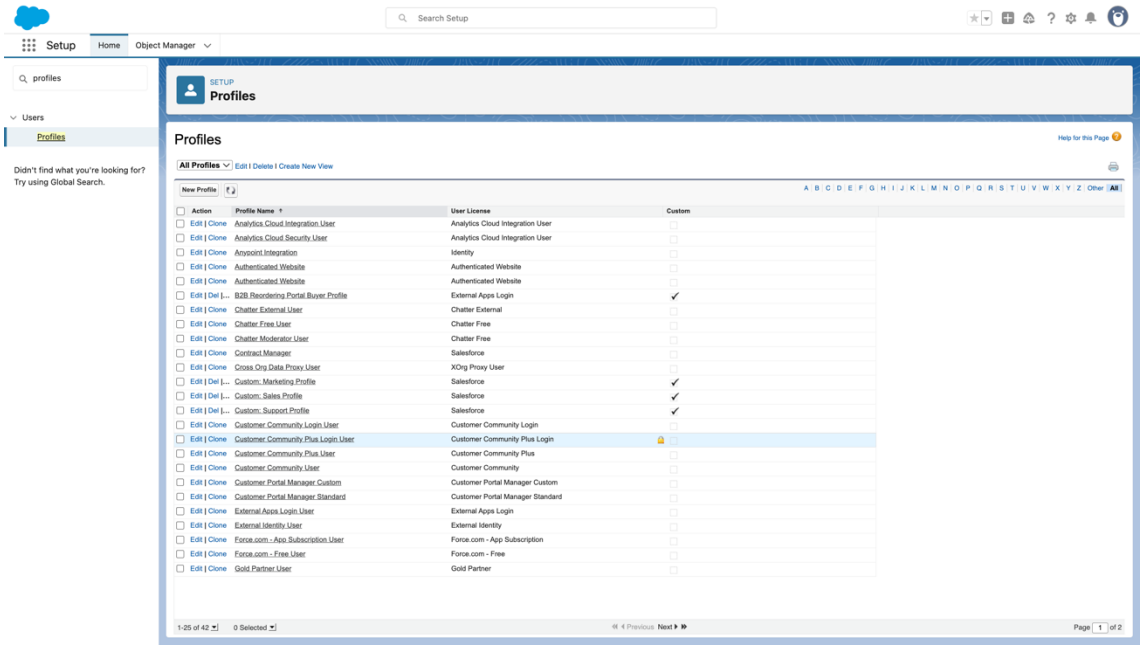
8.1 Sample test data imported using Data Loader.

Phase 9 – Reporting, Dashboards & Security Review

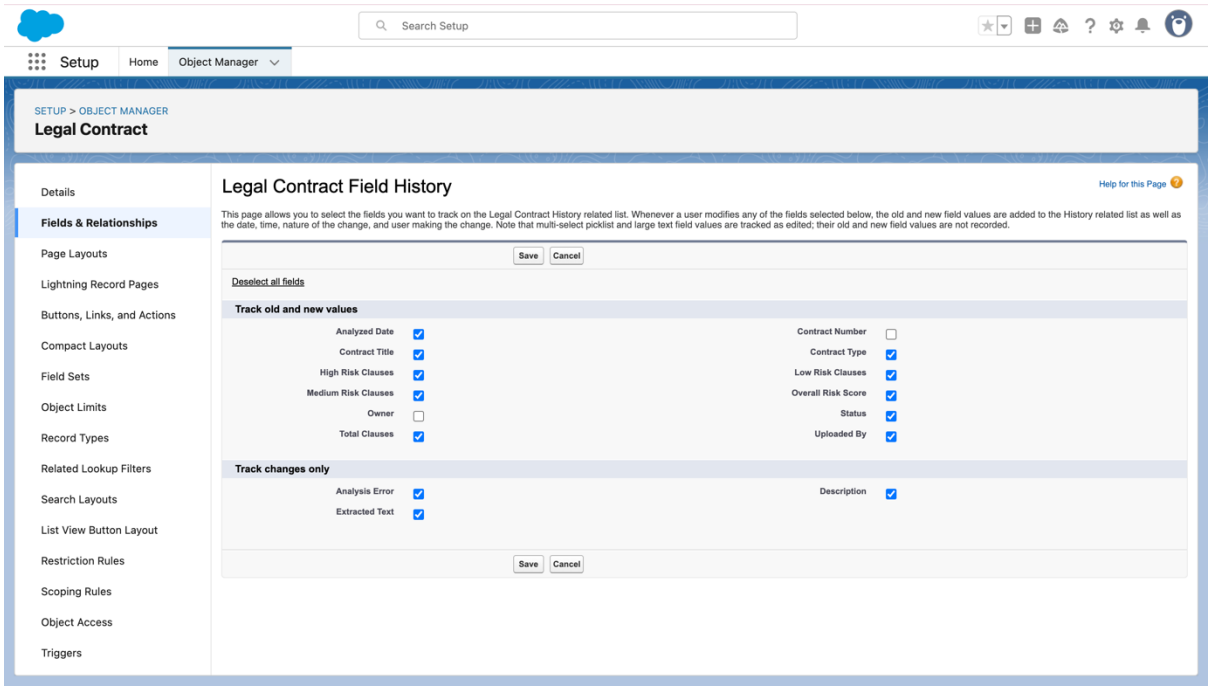
- **Reports**:
  - Contract Risk Summary (by Type/Status).
  - Clause Distribution (by Risk Level).
- **Dashboard**: “Legal Analytics” – combines KPIs, risks, and approvals.
- **Security**: Roles, profiles, permission sets, FLS, OWD.



9.1 Reports that are created



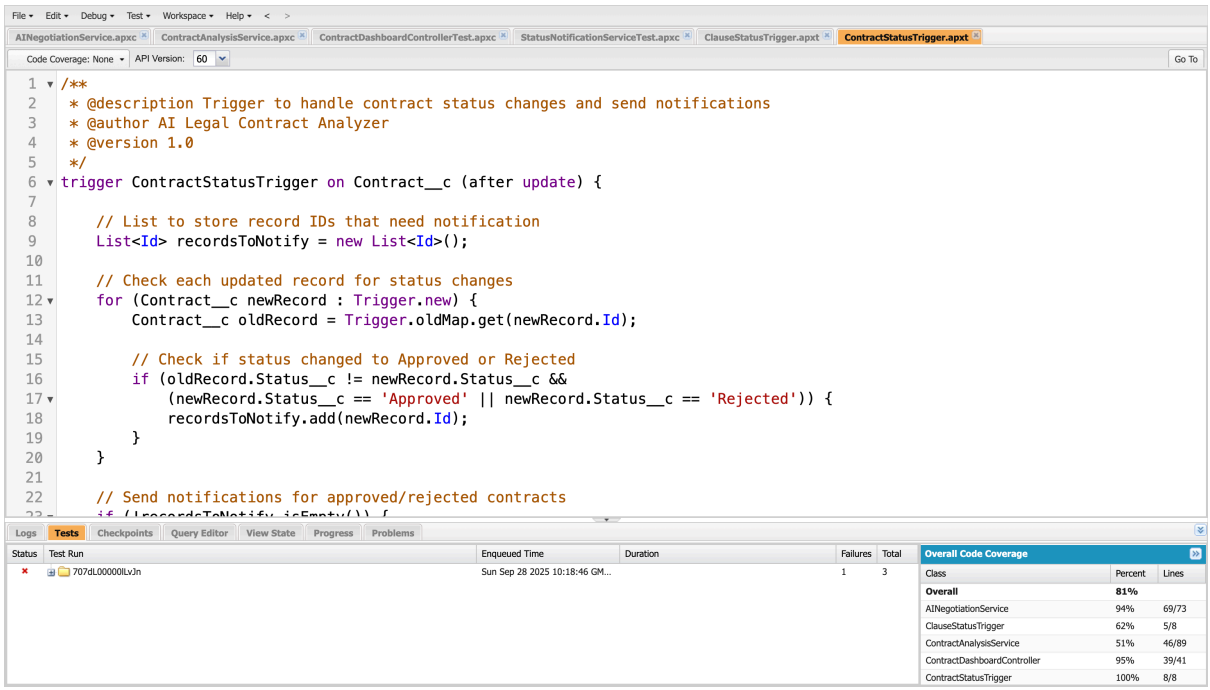
9.2 Profiles



9.3 Legal Contract Field History

Phase 10 – Quality Assurance Testing

- **Unit Testing**: 4 Apex test classes.
- **Integration Testing**: Verified event triggers and LWC wiring.
- **UI Testing**: LWC data-binding tested.
- **Coverage**: Achieved 81% (above Salesforce minimum).



10.1 Code Covergae Results

13. Conclusion

The **Legal Contract Analyzer** successfully delivers an AI-powered contract review and negotiation solution on Salesforce. It is secure, scalable, and enterprise-ready, helping legal teams save time and reduce risk.

Lessons Learned

- Deploy LWCs and services before enabling flows for stability.
- Flows built inside **Salesforce Setup** proved more reliable for CI/CD.

Future Enhancements

- Advanced flow automation for negotiation approvals.
- Extended dashboards for deeper contract analytics.
- External AI/NLP model integration for smarter risk detection.
- Batch/queueable Apex for handling bulk contracts.

14. Appendix

- Objects: force-app/main/default/objects/*
- Apex Classes: force-app/main/default/classes/*
- Triggers: force-app/main/default/triggers/*
- LWC: force-app/main/default/lwc/contractDashboard/*
- Platform Event: force-app/main/default/objects/Status_Change_Event__e/*
- Tests: force-app/main/default/classes/*Test.cls