Comparison of Contract Management and Legal AI Platforms:

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **PandaDoc** | **Harvey AI** | **DocDraft** |
| Document Automation | Yes | Yes | Yes |
| eSignature Integration | Yes | No | No |
| Legal Research Capabilities | No | Yes | No |
| Predictive Analytics | No | Yes | No |
| Compliance Checks | Limited | Advanced | Advanced |
| CRM Integration | Yes | No | No |
| Collaboration Tools | Yes | Limited | Yes |
| Customizable AI Models | No | Yes | No |
| Clause Library | Limited | No | Yes |
| Attorney Consultation | No | No | Yes |

Imperative vs. Agentic Approach Comparison:

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Imperative Approach** (Traditional) | **Declarative Approach** (Agentic AI) |
| **Definition** | Step-by-step instructions to achieve a goal. | Specifies the goal and constraints, allowing AI agents to figure out the best steps. |
| **Control** | Explicitly defines every action and decision logic. | AI autonomously determines actions based on high-level goals. |
| **Flexibility** | Rigid and requires manual updates for changes. | Adaptive, dynamically adjusting based on new data and conditions. |
| **Example** | Writing explicit loops and conditionals to handle a process. | Using AI agents that decide when and how to execute tasks. |
| **Scalability** | Limited by human-defined logic; harder to extend. | Easily scales as AI agents can generalize solutions across tasks. |
| **Error Handling** | Requires explicit error handling in the code. | AI can self-correct and learn from past interactions. |
| **Human Involvement** | Heavy involvement in coding and debugging logic. | Minimal intervention; humans define objectives, and AI executes. |
| **Efficiency** | Time-consuming, as every scenario must be manually handled. | AI optimizes execution and improves efficiency over time. |
| **Use Case Example** | A chatbot that follows predefined decision trees. | An autonomous AI assistant that adapts to user queries and context dynamically. |

Comparison of LangChain and LangGraph:

|  |  |  |
| --- | --- | --- |
| **Feature** | **LangChain** | **LangGraph** |
| Workflow Structure | Sequential chains | Dynamic graphs |
| Complexity Handling | Simple/moderate workflows | Moderate/complex workflows |
| Iterative Processing | Limited | Fully supported (loops) |
| Conditional Logic | Basic conditions | Advanced conditions |
| Multi-Agent Coordination | Basic support | Robust collaboration |
| State Management | Basic retention | Advanced tracking |
| Use Case Suitability | Straightforward tasks | Complex adaptive tasks |
| Flexibility | Rigid structure | Highly flexible |
| Debugging and Monitoring | Good support | Enhanced visibility |
| Real-time Decision Making | Limited | Extensive support |
| Ideal Applications | Simple Q&A, summarization | Complex multi-agent apps |

*Actors and Their Roles:*

|  |  |
| --- | --- |
| **Actor** | **Role** |
| Sales User | Responsible for creating and managing contract documents and negotiations with clients. |
| Legal User | Oversees contract compliance, manages legal clauses, and reviews contracts. |
| Administrator | Manages platform users, roles, permissions, and oversees system configurations. |

Textual Description of Use Case: Create Contract

|  |  |
| --- | --- |
| **UC 1: Create Contract** | |
| Actor | Sales User |
| Goal | Create a new contract |
| Preconditions | User must be authenticated |
| Postconditions | Successful creation of a contract draft |
| Nominal Scenario | 1. User accesses the Contract Repository page. 2. User clicks on "Generate a New Contract." 3. User selects one of the following input methods: voice prompt, image upload, or text prompt. 4. System extracts key contract information. 5. If a new client or mismatched client information is detected, the system prompts the user for confirmation. 6. User confirms key information. 7. System automatically generates the contract step-by-step. |
| Alternative Scenario | Missing required information – Display an alternative message indicating the specific missing information. |

Textual Description of Use Case: Create Clause Request

|  |  |
| --- | --- |
| **UC 2: Create Clause Request** | |
| Actor | Sales User |
| Goal | Create a clause request to add, modify, or delete clauses |
| Preconditions | User must be authenticated |
| Postconditions | Clause request submitted and pending Legal User action |
| Nominal Scenario | 1. User accesses the Contract Editing page. 2. User initiates a clause request from the left panel in the clause area or by right-clicking on an article, clause, or sub-clause. 3. User enters the description of the request. 4. User optionally refines the request description using an LLM-powered "Refine" button. 5. User submits the clause request. 6. System notifies the Legal User of the submitted clause request. |
| Error Scenario | Invalid or incomplete request – System displays an error message indicating required corrections. |

Textual Description of Use Case: Review Contract

|  |  |
| --- | --- |
| **UC 3: Review Contract** | |
| Actor | Legal User or Sales User |
| Goal | Review a contract for accuracy, consistency, and legal security |
| Preconditions | User must be authenticated |
| Postconditions | Contract reviewed with provided suggestions and quality score |
| Nominal Scenario | 1. User accesses the Editor View (Contract Editing page). 2. User selects "Review" in the right panel. 3. User chooses between "Accuracy and Consistency" and "Legal Security" review options. 4. User initiates the review process. 5. System completes the review and provides a quality score (\%). 6. System presents suggestions generated by the LLM for the user to accept or discard. |
| Error Scenario | Review fails or incomplete – System displays a message indicating the review error and required next steps. |