**Here are several techniques commonly used in Generative AI to automatically create predefined prompts for an uploaded document:**

### **1. Template-based Prompt Generation**

* **Description:  
   Predefined placeholders and static templates.**
* **Example:  
   "Summarize the key points from {section\_name}."**

### **2. Metadata-based Prompt Generation**

* **Description:  
   Generate prompts dynamically using document metadata like title, author, and date.**
* **Example:  
   "Explain the implications of the report titled '{document\_title}' published on {publish\_date}."**

### **3. Extractive Prompt Generation (Keyword Extraction)**

* **Description:  
   Extract important keywords or terms using NLP (e.g., TF-IDF, RAKE).**
* **Example:  
   "Define and discuss the following terms extracted from the document: {extracted\_terms}."**

### **4. Semantic Prompting (Embedding-based)**

* **Description:  
   Use document embeddings to generate semantically relevant questions or prompts.**
* **Example:  
   "Discuss similarities between {section\_A} and {section\_B}, based on semantic analysis."**

### **5. Summarization-based Prompting**

* **Description:  
   Generate summary-based prompts using abstractive or extractive summarization models.**
* **Example:  
   "Provide a detailed explanation for: {extracted\_summary\_point}."**

### **6. Question Generation (QG)**

* **Description:  
   Automatically generate questions from the document content using Question Generation models.**
* **Example:  
   "What are the main arguments presented in section {section\_number}?"**

### **7. Named Entity-based Prompting**

* **Description:  
   Use named entity recognition (NER) to create entity-focused prompts.**
* **Example:  
   "Explain the significance of {named\_entity} as discussed in the document."**

### **8. Structured Document Prompting**

* **Description:  
   Leverage structural elements (headers, sections, tables) for prompts.**
* **Example:  
   "Describe the main points outlined under the heading '{header\_name}'."**

### **9. Prompt Refinement (LLM-based)**

* **Description:  
   Use large language models iteratively to refine initial prompts.**
* **Example:  
   Initial: "Give details about finance." → Refined: "Summarize the financial projections for fiscal year {year} in section {section}."**

### **10. Agentic Prompt Generation**

* **Description:  
   Agent-based systems dynamically generating prompts based on interactions and document context.**
* **Example:  
   "After reviewing section {section\_name}, discuss potential areas requiring further clarification."**

### **Commonly Used Tools & Libraries:**

* **OpenAI GPT Models (GPT-3.5, GPT-4, GPT-4 Turbo)**
* **Hugging Face Transformers (for summarization, question generation, and embeddings)**
* **SpaCy/NLTK (for NLP tasks like NER, Keyword extraction)**
* **LangChain/LlamaIndex (for structured prompting and embedding-based semantic retrieval)**
* **Azure Document Intelligence/Azure Cognitive Services (metadata extraction, layout analysis)**

## **Evaluation of GenAI Approaches for Generating Predefined Prompts**

### **1. OpenAI GPT-4 / GPT-3.5-turbo**

* **Description: Language models capable of generating human-like text.**
* **Use Cases: Generating example prompts, summaries, section-specific questions.**
* **Resources Needed:**
  + **OpenAI API key**
  + **Token management**
  + **Prompt engineering expertise**
* **Benefits:**
  + **High language fluency and contextual accuracy**
  + **Easy integration via API**
* **Limitations:**
  + **API cost and token limit**
  + **May generate irrelevant prompts if context is poor**
  + **Not deterministic; outputs may vary**

### **2. LangChain**

* **Description: Python framework for developing LLM-powered applications with modular components.**
* **Use Cases: Chaining document loaders, retrievers, and LLMs to generate prompts.**
* **Resources Needed:**
  + **LangChain library**
  + **Python knowledge**
  + **Chain configuration skills**
* **Benefits:**
  + **Modular and composable**
  + **Supports memory and prompt templates**
* **Limitations:**
  + **Steep learning curve**
  + **Slower execution with large chains**
  + **Debugging complexity**

### **3. LlamaIndex**

* **Description: Framework for indexing and querying document content with LLMs.**
* **Use Cases: Automatically extract relevant sections to base prompt generation.**
* **Resources Needed:**
  + **Document loaders**
  + **Index storage (in-memory/vector DB)**
  + **Python-based setup**
* **Benefits:**
  + **Efficient document structuring**
  + **Query-aware context fetching**
* **Limitations:**
  + **Additional pre-processing step**
  + **Costly if using embeddings frequently**
  + **Dependent on document cleanliness**

### **Combined Workflow Architecture**

1. **Document Upload**
2. **Text Extraction/OCR (Azure, Tesseract, etc.)**
3. **LlamaIndex (Indexing Document)**
4. **LangChain (Chaining Query + Prompt Template)**
5. **OpenAI GPT-4 (Generate Predefined Prompts)**
6. **Present Prompts in UI or Chatbot**

## **Summary Table**

| **Approach** | **Resources Needed** | **Benefits** | **Limitations** | **Implementation Effort** |
| --- | --- | --- | --- | --- |
| **GPT-4 / GPT-3.5** | **API Key, Prompt Engineering** | **Fluent output, flexible** | **Token limits, API costs, hallucinations** | **Medium** |
| **LangChain** | **Python, LangChain, chaining knowledge** | **Modular, memory support** | **Debug complexity, learning curve** | **High** |
| **LlamaIndex** | **Index setup, document loaders** | **Structured content retrieval** | **Preprocessing needed, sensitive to document quality** | **Medium** |

## **Methods for Creating Predefined Prompts**

### **1. Static Prompt Templates**

* **Description:  
   Static prompts predefined manually to guide the model explicitly.**

**Example:  
  
 plaintext  
CopyEdit  
"Compare interest rates between Document A and Document B. Highlight differences clearly."**

* **Implementation Effort:**
  + **Very low (manual writing).**
  + **Developer-friendly, no ML required.**
* **Resources Required:**
  + **None (only basic documentation/knowledge of prompts).**
* **Limitations:**
  + **Brittle to variations.**
  + **Requires manual updates if documents change structure.**

### **2. Prompt Engineering with Parameterization**

* **Description:  
   Using templates with placeholders dynamically filled by variables from the documents.**

**Example:  
  
 plaintext  
CopyEdit  
"Compare the {section} of Document A and Document B. List any differences clearly."**

* **Implementation Effort:**
  + **Low effort, minor integration required to inject variables.**
* **Resources Required:**
  + **Minimal coding (simple string templating system).**
  + **Section extraction logic from documents.**
* **Limitations:**
  + **Requires structured data extraction beforehand.**
  + **Less flexible with unforeseen variations.**

### **3. Prompt Libraries (e.g., LangChain Prompt Templates)**

* **Description:  
   Structured libraries to manage and version prompt templates.**

**Example:  
  
 python  
CopyEdit  
from langchain import PromptTemplate**

**prompt = PromptTemplate(**

**template="Analyze the {section} between Document A and B. Summarize differences.",**

**input\_variables=["section"]**

**)**

* **Implementation Effort:**
  + **Moderate (library setup, prompt template management).**
* **Resources Required:**
  + **LangChain or similar framework setup.**
  + **Minimal developer effort for maintenance.**
* **Limitations:**
  + **Still reliant on accurate section identification.**
  + **May need occasional tuning.**

### **4. Retrieval-Augmented Prompting (RAP)**

* **Description:  
   Dynamically retrieve relevant predefined prompts based on semantic matching of user intent or document content.**
* **Example:  
   Embedding user's query ("compare loan penalties") to select a relevant prompt from stored embeddings.**
* **Implementation Effort:**
  + **Moderate (setup embeddings, semantic search).**
* **Resources Required:**
  + **Vector database (e.g., Pinecone, Faiss).**
  + **Embedding model (e.g., OpenAI embeddings).**
* **Limitations:**
  + **Requires embedding and semantic search setup.**
  + **Occasional mismatches in semantic similarity.**

### **5. Dynamic Prompt Generation via LLM**

* **Description:  
   Use an LLM to dynamically generate a relevant predefined prompt based on user query or document characteristics.**

**Example Prompt to LLM:  
  
 plaintext  
CopyEdit  
"Generate a detailed prompt for comparing repayment terms in loan documents."**

* **Implementation Effort:**
  + **Moderate (integration of secondary LLM call for prompt creation).**
* **Resources Required:**
  + **Additional API calls (costs).**
  + **Small integration code snippet.**
* **Limitations:**
  + **Slightly increased latency.**
  + **May generate suboptimal prompts occasionally, requiring validation.**

### **6. Agentic Prompting**

* **Description:  
   Agents that autonomously generate and select prompts based on step-by-step reasoning and document exploration.**
* **Example Use-case:  
   An agent evaluates two documents, decides differences to highlight, and selects a suitable prompt autonomously.**
* **Implementation Effort:**
  + **High (agent setup, integration with LangChain or similar framework).**
* **Resources Required:**
  + **Agent framework (LangChain, AutoGPT, or custom).**
  + **Possibly multiple LLM calls (higher cost).**
* **Limitations:**
  + **Complexity in debugging and maintaining.**
  + **Potentially slower responses due to multi-step reasoning.**

### **7. Fine-tuned Prompt Generation Model**

* **Description:  
   Fine-tune a smaller LLM specifically to generate prompts for loan document comparisons.**
* **Example:  
   Given input "interest rate comparison," the fine-tuned model outputs a detailed predefined prompt.**
* **Implementation Effort:**
  + **High (training data collection, fine-tuning process).**
* **Resources Required:**
  + **GPU compute resources for fine-tuning.**
  + **Training dataset (example prompts).**
* **Limitations:**
  + **Requires ML expertise and infrastructure.**
  + **Cost and complexity in maintaining/updating fine-tuned models.**

### **8. Prompt Optimization with Genetic Algorithms or RLHF**

* **Description:  
   Use optimization techniques (e.g., Genetic Algorithms, Reinforcement Learning with Human Feedback - RLHF) to iteratively improve predefined prompts automatically.**
* **Example Scenario:  
   Generate multiple prompt variants, evaluate them based on user feedback, and iteratively select the best-performing prompt.**
* **Implementation Effort:**
  + **Very High (setup optimization loop and feedback system).**
* **Resources Required:**
  + **Iterative optimization framework.**
  + **User feedback or evaluation metric system.**
* **Limitations:**
  + **Significant initial investment.**
  + **Complex to maintain.**

## **Summary Table (PDF & PPT-friendly):**

| **Method** | **Effort** | **Resources Required** | **Key Limitations** | **Suitability for Predefined Prompts** |
| --- | --- | --- | --- | --- |
| **Static Templates** | **Low** | **None** | **Brittle, manual updates** | **High (simple, stable scenarios)** |
| **Parameterized Templates** | **Low** | **Basic coding** | **Less flexible** | **High** |
| **Prompt Libraries (LangChain)** | **Moderate** | **LangChain or similar** | **Dependent on extraction accuracy** | **Very High (maintainability)** |
| **Retrieval-Augmented Prompting** | **Moderate** | **Embedding models, Vector DB** | **Semantic mismatch possible** | **High (dynamic adaptability)** |
| **Dynamic Prompt Generation (LLM)** | **Moderate** | **Additional LLM API calls** | **Latency, suboptimal prompts** | **Moderate-high** |
| **Agentic Prompting** | **High** | **LangChain, agentic framework, multiple LLM calls** | **Complexity, latency, debugging difficulty** | **Moderate (for complex cases)** |
| **Fine-tuned Prompt Generation** | **High** | **GPU compute, training dataset** | **ML expertise, expensive setup** | **High (if specialized prompts needed)** |
| **GA/RLHF Prompt Optimization** | **Very High** | **Optimization framework, feedback mechanism** | **High complexity, maintenance** | **Moderate-low (advanced scenarios)** |

## **Recommendation for Your Loan Document Comparison Web App:**

* **Initial Development: Start with simple parameterized templates and LangChain-managed prompt libraries.**
* **Intermediate Improvement: Implement Retrieval-Augmented Prompting for dynamic adaptability.**
* **Advanced Enhancements: Consider fine-tuning a smaller model or adopting agentic prompting for sophisticated user scenarios, if justified by use-case complexity and volume.**