Enhanced Job Workflow Capture & Synthesis Framework (EJWCS): Technical Specification and Implementation Roadmap

I. Strategic Overview and The EJWCS Framework Mandate

I.A. Defining Job Workflow Synthesis in the Age of AI

The traditional reliance on static job descriptions and subjective subject matter expert (SME) knowledge is no longer sufficient to support advanced operational analysis, training curriculum development, or automation initiatives. The current technological landscape necessitates a transition to dynamic, machine-readable workflow models that provide precision, consistency, and scalability for analyzing complex professional tasks.

The core challenge addressed by the Enhanced Job Workflow Capture & Synthesis Framework (EJWCS) is the transformation of unstructured human expertise—captured through interviews or documentation—into rigorously structured data artifacts. These artifacts must be suitable for integration into modern AI pipelines and enterprise resource planning systems. Successful extraction and standardization of workflow data are critical steps toward building advanced AI systems capable of generating summaries, extracting complex data structures, and supporting scalable workforce planning.

I.B. Architectural Philosophy: Structured Extraction and Standardization

The EJWCS framework is designed around a fundamental "schema-first" architectural philosophy. This approach mandates that the Large Language Model (LLM) processing core operates not as a creative generator, but as a reliable data formatter. To achieve this, the system relies heavily on explicitly defining and enforcing the desired output structure using Pydantic models. This mechanism is essential for guiding the LLM to consistently produce valid, deeply nested JSON output, overcoming the frequent "JSON Problem" where models invent fields, omit data, or introduce type mismatches when processing complex schemas.

Furthermore, the extracted workflow data must adhere to global occupational taxonomies to ensure maximum interoperability and utility across different enterprise functions, including talent management and training. The framework mandates standardization by mapping extracted skills, knowledge, and roles to globally recognized frameworks, specifically the European Skills, Competences, and Occupations (ESCO) classification and the U.S. Department of Labor's Occupational Information Network (O*NET) Content Model. This dual standardization elevates the resulting data from mere internal documentation to a universally recognized asset.

I.C. The Four Phases of EJWCS: Capture, Extract, Synthesize, Validate

The EJWCS operates as a closed-loop system defined by four integrated phases: Capture, Extract, Synthesize, and Validate. This pipeline ensures a comprehensive transition from raw, qualitative input to a quantitative, visualizable output.
The framework's effectiveness hinges on the seamless integration of specialized components at each layer, as detailed below.
Table I: Enhanced Job Workflow Capture & Synthesis Framework (EJWCS) Component Breakdown

Layer	**Component/Agent**	**Core Function**	**Required Inputs**	**Relevant Output**
Capture	Input Ingestion Module	Pre-processes raw data (transcription, diarization, text cleaning).	SME interviews, existing SOPs, job descriptions.	Clean, time-stamped, speaker-attributed textual transcript.
Extraction	WORKFLOW_EXTRACTOR (LLM)	Generates structured JSON output based on the defined Pydantic schema, identifying tasks, dependencies, and conditions.	Pre-processed textual input, full JSON Schema definition (Prompt).	Raw, validated JSON workflow object.
Standardization	Taxonomy Mapper	Maps extracted skills, knowledge, and roles to specific identifiers.	Raw JSON output, ESCO/O*NET database access.	Standardized JSON with ESCO/O*NET IDs.
Synthesis	Visualization Generator	Converts structured workflow logic (dependencies, conditions) into display format (Mermaid syntax).	Mapped JSON output, Visualization Logic Library.	Mermaid Diagram Text/Code.
Validation	Audit and Review Module	Ensures extracted semantics and taxonomy tags match expert intent.	Synthesized Mermaid output, Standardized JSON.	Verified, audit-ready workflow blueprint.

II. EJWCS Component Architecture and Methodology

II.A. Input Layer Design: Source Ingestion and Pre-processing

The initial phase of the EJWCS pipeline focuses on securing high-fidelity input data, recognizing that the quality of the final output is directly proportional to the richness of the source material.

SME Interview Methodology (DACUM Foundation)

The ingestion process is structurally anchored in the DACUM (Developing A Curriculum) methodology. DACUM provides a validated framework for documenting occupational duties and tasks. This process is critical because it moves beyond generalized job duties to capture specific performance steps, crucial decision points, essential underlying knowledge, and required industry standards, all gathered directly from a subject matter expert. This ensures that the textual input provided to the LLM agent is rich enough to satisfy the requirements of the standardized schema, particularly in defining conditional logic and knowledge requirements.

Technical Pipeline for Audio/Video Capture

For the capture of SME interviews, the choice of transcription service is dictated by the need for advanced features necessary for workflow integrity. The Azure AI Speech batch transcription service is recommended over standard OpenAI or Azure OpenAI Whisper API deployments for this critical use case.
While the Whisper model via Azure OpenAI is suitable for fast processing of individual files up to 25MB, the specialized requirements of detailed workflow analysis demand features only available through the Azure AI Speech service. These features include:

1. **Large File Handling:** Support for transcribing files up to 1GB, essential for handling long SME interview sessions.
2. **Diarization:** The ability to distinguish between different speakers participating in the conversation.
3. **Word-level Timestamps:** Providing temporal markers for the transcribed text.

Justification for High-Fidelity Transcription Investment

The requirement for diarization fundamentally drives the architectural decision for the Input Ingestion Module. Workflow analysis demands a clear understanding of task ownership; that is, *who* performs *what* task. If a multi-participant SME interview (e.g., an accountant and a senior auditor discussing the month-end close) results in a single, undifferentiated text block, the LLM agent cannot reliably attribute a specific task description to the correct occupational role (the role_owner field).
The Azure AI Speech solution, which includes diarization, enables reliable extraction of the role_owner field, linking the specific speech segment to the identified speaker. This capability ensures that the extracted workflow segments are accurately attributed to the corresponding role (e.g., Staff Accountant vs. Senior Manager).
Although the Azure AI Speech service is known to be significantly more expensive—with transcription costs cited at approximately $1.00 USD per hour, compared to $0.36 USD per hour for the standard OpenAI Whisper API —this difference in cost (a roughly 3x premium) is structurally justified. The inability to attribute tasks reliably due to a lack of diarization would render the subsequent LLM extraction and standardization steps functionally unreliable for role definition and compliance tracking. Therefore, the higher expense for superior data fidelity is a necessary architectural investment.

II.B. The Central Processing Unit: The WORKFLOW_EXTRACTOR

Agent

The WORKFLOW_EXTRACTOR LLM agent serves as the central processing unit, mapping the clean, time-stamped textual input onto the rigorously defined, deeply nested JSON structure. The LLM chosen for this task must prioritize adherence to structure and possess strong contextual processing capabilities, given the length of SME transcripts. The operational core of the agent is the mandatory use of a defined Pydantic model. By inputting the full schema definition directly into the prompt, the model becomes both the validator and the generator, forcing the LLM to output precise JSON. This technique resolves the common difficulty developers face in ensuring LLMs consistently produce valid JSON, especially when dealing with complex structures involving nested objects, dependencies, and conditional logic.

II.C. The Output Layer: Visualization and Data Repository Structure

Visualization Mandate

The synthesized workflow output must be immediately actionable and comprehensible by both technical and non-technical stakeholders. This requirement is met by mandating the use of Mermaid syntax. Mermaid is a modern, web-friendly text-based tool that allows users to generate clear, structured diagrams using plain text (Diagrams-as-Code). Mermaid integrates seamlessly with many collaboration and visual workspace tools, making it an ideal choice for developer teams needing to visualize complex processes and collaborate asynchronously.

Semantic Consistency and Data Repository

The centralized data repository is structured to store three interconnected artifacts: the standardized JSON object, the corresponding Mermaid syntax, and the linked occupational taxonomy identifiers.
Crucially, the architecture dictates that both the structured JSON and the resultant Mermaid text must be stored. This requirement is founded on the principle of semantic preservation. Workflow modeling, particularly based on BPMN (Business Process Model and Notation) semantics, involves strict rules—for instance, an Exclusive (XOR) gateway means only one path may be taken, regardless of how the conditional logic is written. The JSON structure, derived directly from the Pydantic schema, captures this semantic intent (the core business logic). If only the Mermaid code were stored, any future change in visualization tools or syntaxes could potentially lose the core, standardized logic defined by the framework. Storing the JSON ensures the core business process logic remains preserved, auditable, and analytically accessible for future applications, independent of the visualization layer.

III. The Workflow Extraction Engine: Agent Specification and Prompt Design

III.A. Principles of Structured Prompt Engineering (Schema-First Approach)

The successful function of the WORKFLOW_EXTRACTOR relies entirely on a meticulously engineered prompt that imposes strict structural boundaries on the LLM output. The overarching goal of the prompt is to instruct the LLM to analyze the high-fidelity input text (e.g., "The accountant reconciles the accounts , and if a variance exists, they must inform the manager") and accurately map these elements onto the highly specific fields of the JobWorkflowSchema. The structured prompt must include three mandatory components to maximize compliance and minimize error: 1) a clear System Instruction defining the agent's role and constraints; 2) the raw input context; and 3) the full Pydantic schema definition, provided as a code block for definitive structural guidance.

III.B. Defining the Pydantic Base Model for Job Workflow

The Pydantic base model, referred to as JobWorkflowSchema, is the definitive blueprint for all extracted data. It is rigorously defined to capture sequential flow, complexity (conditional routing), dependency mapping, and required competency fields, ensuring the resulting data object is suitable for downstream analytics and curriculum design.

Table II: Proposed Core Schema Fields for EJWCS Workflow Extraction

Field Name (JSON Key)	**Data Type**	**Description**	**Standardization Reference**	**Architectural Implication**
workflow_name	String	High-level process name (e.g., "Monthly Financial Close").	Internal Index	Essential for data grouping and retrieval.
tasks	List	Array of individual, chronologically ordered task steps.	BPMN Sequence Flow	Defines the execution path.
TaskObject (Nested):				
task_id	String (UUID)	Unique identifier for the specific task instance.	Internal Index	Essential key for defining task relationships (dependencies, precedes_tasks).
task_description	String	Detailed, action-oriented description of the step.	DACUM/SME Input	Direct input from expert transcription.
role_owner	String	The responsible occupation/role.	O*NET/ESCO Occupations	Requires reliable diarization in Input Layer.
precedes_tasks	List	List of immediate successor tasks.	BPMN Sequence Flow	Dictates the visualization flow.
dependencies	List	List of preceding tasks that must be completed.	BPMN Sequence Flow	Defines execution constraints and fan-in logic.
conditional_logic	Object	Defines branching pathways based	BPMN Gateways	Crucial for modeling

Field Name (JSON Key)	Data Type	Description	Standardization Reference	Architectural Implication
		on an outcome (e.g., decision gateway).		complexity and decision points (e.g., audit findings).
required_knowledge	List[Object]	Associated conceptual knowledge required (e.g., GAAP, IFRS).	ESCO Knowledge Pillar	Essential link for training system integration.
required_skill_tags	List	Associated skills and competences needed.	ESCO Skill/Competence Pillar	Core standardization linkage for competency management.

III.C. Detailed Prompt for the WORKFLOW_EXTRACTOR Agent (Annotated Structure)

The full extraction prompt is verbose and serves as the operating instruction set for the LLM.

1. **System Instruction (Role & Constraint):** This block explicitly sets the agent's identity ("EJWCS Workflow Extractor agent") and enforces the strict constraint: the agent's *sole purpose* is to produce a single, valid JSON object conforming strictly to the defined Pydantic schema. It instructs the LLM that if information for a field is missing, it must return null but *never* omit the key itself.
2. **Input Context and Variables:** The entirety of the clean, pre-processed transcript derived from the SME interview (including time stamps and speaker attribution from the diarization process) is inserted here.
3. **Schema Enforcement Block:** The full, detailed Pydantic class definition (JobWorkflowSchema and the nested TaskObject), including docstrings and validation constraints (such as min_items=1 for mandatory lists) , is printed verbatim into the prompt. This provides the LLM with the indisputable structural blueprint it must adhere to.
4. **Semantic Mapping Rules:** This critical section provides the LLM with specific instructions on translating natural language workflow concepts into the structured fields:
   - *Dependencies:* The prompt defines how to identify prerequisite tasks, instructing the LLM that phrases like "before starting X, Y must be complete" must result in Y being placed into the dependencies list of X.
   - *Conditional Routing:* The prompt directs the agent to identify explicit decision points and map phrases such as "If the account balance matches, the process stops; otherwise, we must escalate the variance investigation" to a structured conditional_logic object, defining the condition statement and the resulting branched task IDs. This ensures the complexity of process management, such as the conditional logic found in accounting reconciliation , is accurately captured.

IV. Standardized Workflow Modeling and Schema

Integration

The Standardization layer is responsible for taking the raw JSON output and linking it to established occupational taxonomies, providing context and measurable data for workforce development and talent analysis.

IV.A. Comparative Analysis of Occupational Taxonomies (ESCO vs. O*NET)

Effective standardization requires leveraging the optimal strengths of available taxonomies.
ESCO (European Skills, Competences, and Occupations): Developed by the European Commission, ESCO organizes occupations, skills, and knowledge into distinct pillars. Its primary strength lies in the highly granular distinction it maintains between **Skills/Competences** and **Knowledge** concepts. This tripartite structure is highly beneficial for detailed curriculum design, allowing for precise identification of the foundational knowledge (e.g., GAAP principles) required for a specific task versus the competence (e.g., financial analysis) applied during the execution of that task.
O*NET (Occupational Information Network): This US Department of Labor framework provides a hierarchical content model describing the characteristics of occupations through six domains. O*NET collects and codifies nearly 277 descriptors, making it a robust source for broad occupational classification and standardized job descriptors primarily utilized across North America.

Dual Mapping Requirement

Limiting the framework to a single taxonomy would limit its global application and functional utility. Therefore, the EJWCS framework mandates **dual taxonomy mapping**. The Taxonomy Mapper component must perform distinct lookups to optimize the utility of both systems.
ESCO is designated as the primary taxonomy for mapping granular details: the required_knowledge field is mapped to the ESCO Knowledge Pillar, and the required_skill_tags field is mapped to the ESCO Skill/Competence Pillar. This utilization is driven by ESCO's explicit separation of these two entities, which is crucial for detailed training analysis. O*NET serves as the secondary mapping standard, utilized primarily for the higher-level occupational classification required for the role_owner field, ensuring maximum interoperability in North American talent markets. This dual-strategy ensures the system provides both highly detailed training inputs and broadly recognized occupational labels.

IV.B. Defining Workflow Entities: Task, Sub-Task, Decision Node, and Skill Tag

Task and Sub-Task Definition

For the structured workflow to be analytically useful, tasks extracted must be defined at an atomic level. Workflow analysis requires a detailed review of each step and any involved substeps. The LLM extraction process is calibrated to identify the smallest measurable units of work that contribute to the completion of the overall duty (e.g., a Staff Accountant's duty of

"Financial Reporting" contains the atomic task "Conduct account reconciliation").

Knowledge and Skill Linkage

Every extracted task must have its corresponding conceptual anchors defined. The required_knowledge list is directly mapped to the ESCO Knowledge Pillar, thereby creating a verifiable and traceable linkage between an action and the academic or conceptual understanding needed to execute it. Similarly, required_skill_tags are mapped to the ESCO Skill/Competence Pillar. These linkages are foundational for automating the generation of highly accurate, standardized training materials.

Decision Node Semantics

Decision nodes, represented within the JSON's conditional_logic object, require strict semantic validation. The Standardization layer must ensure that the extracted decision logic accurately reflects BPMN best practices, prioritizing clarity over ambiguous conditional flows. For instance, determining whether a decision point, such as choosing an escalation path based on an audit finding, should be modeled as an Exclusive (XOR) gateway, where only one path is taken, or an Inclusive gateway, where multiple paths might be executed in parallel. The system must translate the natural language intent captured in the transcript into the correct BPMN semantic type, guaranteeing the visualization reflects the true process constraint.

V. Synthesis and Visualization: Generating Actionable BPMN Diagrams

The Synthesis layer is the final stage where the structured data is converted into a visual, navigable format using the Visualization Generator component.

V.A. Conversion Algorithm: Structured JSON to Mermaid Syntax

The Visualization Generator converts the logical components within the standardized JSON into text strings compliant with the Mermaid syntax.

Sequential Flow and Dependency Mapping

Sequential flow is the most straightforward conversion. Each task in the tasks array, where Task A precedes Task B (indicated by Task B's presence in Task A's precedes_tasks field), is translated into a simple Mermaid directional link, generally represented as Task_A --> Task_B. Dependencies defined in the JSON (dependencies field) are critical for structuring parallel sequences and merge points. The generator identifies parallel convergences, or fan-in nodes, which are then rendered with specialized styling in the Mermaid output, such as double circle nodes ((fan-in)) for clear identification.

Handling Complex Conditional Logic and Gateways

Modeling conditional logic requires the explicit introduction of decision nodes to represent BPMN gateways. The conditional_logic object dictates the introduction of a diamond-shaped

node in the diagram.
If the JSON logic indicates mutually exclusive paths (the most common requirement for compliance and financial workflows, such as reconciliation checks), the generator translates this into an Exclusive (XOR) gateway equivalent. The resulting Mermaid output uses conditional edges (-- condition -->) where the condition is explicitly labeled on the flow line, ensuring the visualization clearly communicates that one and only one path is possible at that decision point. This rigorous adherence to BPMN semantics—represented graphically by the Mermaid output—ensures the synthesized diagram is not only visually appealing but also technically accurate regarding process execution.

V.B. Visualization Tool Chain and Integration Requirements

Dependency Management

The generation of the Mermaid syntax (the text string) itself requires minimal technical overhead. However, for environments where static image export (PNG or SVG) is required for documentation or archiving, the tool chain must include external dependencies. This typically necessitates the installation and configuration of GraphViz binaries, alongside the necessary Python visualization extras (pip install agent-framework[viz]).

Integration with Collaboration Tools

A critical strategic advantage of using Mermaid is its inherent compatibility with modern collaborative visual workspaces, such as Miro. The final synthesis step ensures that the output is not merely a static image but the embeddable Mermaid text. This enables non-technical stakeholders (HR, Operations) to instantly embed the generated workflow into their collaboration environments, fostering real-time review and co-editing. By treating the Mermaid text as a primary, embeddable artifact, the architecture maximizes the practical utility of the framework for immediate enterprise application.

V.C. Output Validation and Human-in-the-Loop Review

The final stage of the EJWCS pipeline involves a rigorous validation step to ensure fidelity between the raw SME input and the synthesized workflow blueprint.

1. **Schema Compliance:** This is an automated check confirming that the raw JSON output from the WORKFLOW_EXTRACTOR adheres to all structural constraints defined in the Pydantic schema.
2. **Semantic Accuracy:** Human analysts perform a manual review to verify that the visual representation (Mermaid diagram) and the underlying JSON logic accurately reflect the SME's described intent, particularly regarding complex sequencing and decision-making. Specific attention is paid to ensuring that decision points identified as XOR gateways are truly mutually exclusive, adhering to BPMN standards.
3. **Taxonomy Mapping Quality:** The relevance and accuracy of the ESCO/O*NET tags associated with each task are audited to ensure they genuinely reflect the required knowledge and skills for that specific step.

VI. Implementation Roadmap: Accounting Proof of

Concept (PoC)

VI.A. PoC Rationale and Scope Definition

The initial Proof of Concept (PoC) for the EJWCS framework is strategically targeted at the Accounting domain. Accounting workflows provide an ideal testing environment due to their inherent structure, high dependency on regulation (e.g., GAAP compliance), and reliance on strict, measurable conditional checks (e.g., reconciliation, variance analysis). This environment will provide a rigorous test of the system's ability to extract complex conditional logic and dependencies accurately.
The PoC will focus on three core functional areas typical of a Staff Accountant role :

1. **General Ledger Account Reconciliation:** A process defined by critical decision points (conditional logic) regarding the presence or absence of discrepancies.
2. **Monthly Financial Close Support:** A sequence-intensive process requiring high dependency mapping and strict chronological adherence.
3. **Audit Assistance:** Involves highly specific document preparation tasks and compliance checks.

VI.B. Phase 1: Foundation and Data Ingestion (T 0 - T 4 Weeks)

This phase establishes the foundational environment and secure data capture pipeline. Subject Matter Experts (SMEs) who are experienced Staff Accountants or CPAs will be recruited for structured interviews utilizing the DACUM approach to ensure comprehensive capture of performance steps and essential knowledge.
The technical architecture for high-fidelity audio capture will be finalized, specifically confirming the configuration of Azure AI Speech Batch Transcription. This step is non-negotiable, as the successful execution of the entire PoC depends on the reliable capture of speaker identity via diarization, enabling the accurate assignment of the role_owner field in the final output. Furthermore, the final Pydantic model structure (Table II) will be validated against anticipated Accounting terminology and compliance requirements.

VI.C. Phase 2: Agent Calibration and Schema Testing (T 5 - T 8 Weeks)

Phase 2 focuses on tuning the core WORKFLOW_EXTRACTOR agent to achieve high schema adherence and semantic accuracy.
Initial interview transcripts from Phase 1 will be processed through the LLM agent. Iterative refinement of the detailed prompt (Section III.C) will be executed to minimize schema violations and maximize the accuracy of extracting complex dependencies and conditional nodes. A specific calibration focus will be placed on tasks involving decision-making, such as modeling the XOR logic required when a GL account reconciliation succeeds or fails. The primary metric focus during this phase is achieving a schema compliance rate of greater than 95% in the raw JSON output.

VI.D. Phase 3: End-to-End Workflow Synthesis and Visualization Validation (T 9 - T 12 Weeks)

The final phase tests the end-to-end operational readiness of the EJWCS framework.

The Taxonomy Mapper component will be executed, verifying that the extracted tasks successfully map to both the granular ESCO Skill/Knowledge concepts and the occupational classifications in O*NET. The Visualization Generator will convert these standardized JSON outputs into Mermaid syntax diagrams, rigorously auditing the generated conditional edges and decision gateways (XOR vs. Inclusive) for accurate BPMN semantic representation. This phase concludes with the delivery of a fully documented library containing 5–8 standardized Accounting workflows, complete with standardized JSON and embeddable Mermaid output files.

Table III: Accounting Proof of Concept Implementation Roadmap (Q1/Q2)

Milestone Group	Key Deliverables	Target Output	Metrics for Success
Phase 1: Foundation	Selection of initial Staff Accountant SME pool; Final EJWCS Schema (Pydantic) validated; Transcription pipeline tested.	10 hours of transcribed, diarized SME interview data.	95% compliance rate for WORKFLOW_EXTRACTOR output validation (initial test).
Phase 2: Agent Calibration	Refinement of the WORKFLOW_EXTRACTOR prompt for financial close and reconciliation tasks.	Accurate extraction of the "Account Reconciliation" sub-workflow (including decision nodes).	90% human validator agreement on derived conditional logic; <5% LLM invention of fields.
Phase 3: Synthesis & Audit	Generation of five complete, standardized workflow diagrams in Mermaid format.	Fully documented PoC library of Accounting Workflows; Integration demonstration with collaboration platform.	Zero visualization errors; successful mapping to ESCO taxonomy for >90% of extracted skills; validated BPMN semantics (XOR vs. Inclusive).

VI.E. Key Success Metrics and Risk Mitigation Strategy

The primary metric for the PoC is **Semantic Accuracy**, defined as the percentage of extracted workflow tasks where the sequence flow, dependencies, and conditional routing match human expert consensus, verified through the formalized Mermaid diagram output.

Risk Mitigation: The two critical technical risks are LLM hallucination and data fidelity. The risk of LLM deviation from the JSON schema is mitigated through the strict enforcement of the Pydantic model specification and the implementation of immediate rejection and re-prompting protocols for any invalid JSON output. The risk of poor-quality SME input leading to ambiguous extraction is mitigated by mandating the DACUM interview methodology and the use of the high-fidelity, diarized Azure AI Speech service during the Capture phase. These foundational controls ensure that the input is precise and the output is structurally sound.

VII. Conclusion and Strategic Expansion

The Enhanced Job Workflow Capture & Synthesis Framework (EJWCS) represents a critical evolution in organizational knowledge management. By leveraging a structure-enforcing LLM agent guided by Pydantic schemas, the framework transforms amorphous, qualitative expertise into quantitative, standardized, and machine-readable data structures.

The architecture's reliance on dual occupational taxonomy mapping (ESCO for granular skills and knowledge, O*NET for broad occupational classification) provides unparalleled utility for global talent mobility and precise curriculum design. Furthermore, the commitment to Diagrams-as-Code visualization through Mermaid ensures the resulting workflow blueprints are immediately scalable, auditable, and easily integrated into collaborative enterprise environments.

The successful execution of the Accounting Proof of Concept will validate the system's capacity to handle the complex, conditional logic inherent in regulated professional domains. Upon validation, the EJWCS framework is strategically positioned for rapid expansion into other regulated, knowledge-intensive fields, such as Legal Compliance, Auditing, and Healthcare Administration, where the precise capture and standardization of complex professional workflows offer substantial strategic advantage for automation and workforce planning.

Works cited

1. JSON Prompting for LLMs: A Practical Guide with Python Coding Examples - MarkTechPost, https://www.marktechpost.com/2025/08/23/json-prompting-for-llms-a-practical-guide-with-python-coding-examples/ 2. Forcing LLM JSON Outputs: How to make LLM output complex JSONs | by Diana Zagirova, https://medium.com/@d.zagirowa/forcing-llm-json-outputs-how-to-make-llm-output-complex-jsons-a8bb00e87f71 3. The ESCO Classification | European Skills, Competences, Qualifications and Occupations (ESCO), https://esco.ec.europa.eu/en/classification 4. O*NET | U.S. Department of Labor, https://www.dol.gov/agencies/eta/onet 5. DACUM Chart - WIDS, https://www.wids.org/resources/resource-downloads/dacum-chart 6. Dacum…A Tool For Documenting Industrial Involvement In Curriculum Design - ASEE PEER, https://peer.asee.org/dacum-a-tool-for-documenting-industrial-involvement-in-curriculum-design.pdf 7. The Whisper model from OpenAI - Azure AI services | Microsoft Learn, https://learn.microsoft.com/en-us/azure/ai-services/speech-service/whisper-overview 8. How to Do an Accounting Workflow Analysis and Make Improvements, https://mycpadashboard.com/how-to-do-an-accounting-workflow-analysis/ 9. Best 3 Staff Accountant Job Description Template by Invedus, https://invedus.com/blog/staff-accountant-job-description-template/ 10. Whisper API at Azure - more technically advanced, but the price?, https://community.openai.com/t/whisper-api-at-azure-more-technically-advanced-but-the-price/402937 11. Mermaid Diagrams: A Guide with Miro, https://miro.com/diagramming/what-is-mermaid/ 12. Data based XOR Gateway vs Conditional sequence flow - Camunda Forum, https://forum.camunda.io/t/data-based-xor-gateway-vs-conditional-sequence-flow/9353 13. ESCO Skills Hierarchy | INDA HR - Intelligent Data Analysis for HR, https://api.inda.ai/hr/docs/ESCO%20Skills%20Hierarchy 14. Workflow Visualization | Microsoft Learn, https://learn.microsoft.com/en-us/agent-framework/tutorials/workflows/visualization