book_to_slide_BY_sections_V4

July 4, 2025

1 Set up Paths

[1]:
```
# Cell 1: Setup and Configuration
import os
import re
import logging
import warnings
from docx import Document
import pdfplumber
import ollama
from tenacity import retry, stop_after_attempt, wait_exponential, RetryError
import json

# Setup Logger for this cell
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
logger = logging.getLogger(__name__)

# --- 1. CORE SETTINGS ---
# Set this to True for EPUB, False for PDF. This controls the entire notebook's␣
 ↪flow.
PROCESS_EPUB = True # for EPUB
# PROCESS_EPUB = False # for PDF

# --- 2. INPUT FILE NAMES ---
# The name of the Unit Outline file (e.g., DOCX, PDF)
UNIT_OUTLINE_FILENAME = "ICT312 Digital Forensic_Final.docx" # epub
# UNIT_OUTLINE_FILENAME = "ICT311 Applied Cryptography.docx" # pdf

EXTRACT_UO = False

# The names of the book files
EPUB_BOOK_FILENAME = "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide␣
 ↪to Computer Forensics and Investigations_ Processing Digital␣
 ↪Evidence-Cengage Learning (2018).epub"
```

```
PDF_BOOK_FILENAME = "(Chapman & Hall_CRC Cryptography and Network Security␣
 ↪Series) Jonathan Katz, Yehuda Lindell - Introduction to Modern␣
 ↪Cryptography-CRC Press (2020).pdf"

# --- 3. DIRECTORY STRUCTURE ---
# Define the base path to your project to avoid hardcoding long paths everywhere
PROJECT_BASE_DIR = "/home/sebas_dev_linux/projects/course_generator"

# Define subdirectories relative to the base path
DATA_DIR = os.path.join(PROJECT_BASE_DIR, "data")
PARSE_DATA_DIR = os.path.join(PROJECT_BASE_DIR, "Parse_data")

# Construct full paths for clarity
INPUT_UO_DIR = os.path.join(DATA_DIR, "UO")
INPUT_BOOKS_DIR = os.path.join(DATA_DIR, "books")
OUTPUT_PARSED_UO_DIR = os.path.join(PARSE_DATA_DIR, "Parse_UO")
OUTPUT_PARSED_TOC_DIR = os.path.join(PARSE_DATA_DIR, "Parse_TOC_books")
OUTPUT_DB_DIR = os.path.join(DATA_DIR, "DataBase_Chroma")

# --- 4. LLM & EMBEDDING CONFIGURATION ---
LLM_PROVIDER = "ollama"  # Can be "ollama", "openai", "gemini"
OLLAMA_HOST = "http://localhost:11434"
OLLAMA_MODEL = "qwen3:8b" # "qwen3:8b", #"mistral:latest"
EMBEDDING_MODEL_OLLAMA = "nomic-embed-text"
CHUNK_SIZE = 800
CHUNK_OVERLAP = 100

# --- 5. DYNAMICALLY GENERATED PATHS & IDs (DO NOT EDIT THIS SECTION) ---
# This section uses the settings above to create all the necessary variables␣
 ↪for later cells.

# Extract Unit ID from the filename
def print_header(text: str, char: str = "="):
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)

def extract_uo_id_from_filename(filename: str) -> str:
    match = re.match(r'^[A-Z]+\d+', os.path.basename(filename))
    if match:
        return match.group(0)
    raise ValueError(f"Could not extract a valid Unit ID from filename:␣
 ↪'{filename}'")

try:
    UNIT_ID = extract_uo_id_from_filename(UNIT_OUTLINE_FILENAME)
```

```
except ValueError as e:
    print(f"Error: {e}")
    UNIT_ID = "UNKNOWN_ID"

# Full path to the unit outline file
FULL_PATH_UNIT_OUTLINE = os.path.join(INPUT_UO_DIR, UNIT_OUTLINE_FILENAME)

# Determine which book and output paths to use based on the PROCESS_EPUB flag
if PROCESS_EPUB:
    BOOK_PATH = os.path.join(INPUT_BOOKS_DIR, EPUB_BOOK_FILENAME)
    PRE_EXTRACTED_TOC_JSON_PATH = os.path.join(OUTPUT_PARSED_TOC_DIR,␣
 ↪f"{UNIT_ID}_epub_table_of_contents.json")
else:
    BOOK_PATH = os.path.join(INPUT_BOOKS_DIR, PDF_BOOK_FILENAME)
    PRE_EXTRACTED_TOC_JSON_PATH = os.path.join(OUTPUT_PARSED_TOC_DIR,␣
 ↪f"{UNIT_ID}_pdf_table_of_contents.json")

# Define paths for the vector database
file_type_suffix = 'epub' if PROCESS_EPUB else 'pdf'
CHROMA_PERSIST_DIR = os.path.join(OUTPUT_DB_DIR,␣
 ↪f"chroma_db_toc_guided_chunks_{file_type_suffix}")
CHROMA_COLLECTION_NAME = f"book_toc_guided_chunks_{file_type_suffix}_v2"

# Define path for the parsed unit outline
PARSED_UO_JSON_PATH = os.path.join(OUTPUT_PARSED_UO_DIR, f"{os.path.
 ↪splitext(UNIT_OUTLINE_FILENAME)[0]}_parsed.json")

# --- Sanity Check Printout ---
print("--- CONFIGURATION SUMMARY ---")
print(f"Processing Mode: {'EPUB' if PROCESS_EPUB else 'PDF'}")
print(f"Unit ID: {UNIT_ID}")
print(f"Unit Outline Path: {FULL_PATH_UNIT_OUTLINE}")
print(f"Book Path: {BOOK_PATH}")
print(f"Parsed UO Output Path: {PARSED_UO_JSON_PATH}")
print(f"Parsed ToC Output Path: {PRE_EXTRACTED_TOC_JSON_PATH}")
print(f"Vector DB Path: {CHROMA_PERSIST_DIR}")
print(f"Vector DB Collection: {CHROMA_COLLECTION_NAME}")
print("--- SETUP COMPLETE ---")
```

```
--- CONFIGURATION SUMMARY ---
Processing Mode: EPUB
Unit ID: ICT312
Unit Outline Path:
/home/sebas_dev_linux/projects/course_generator/data/UO/ICT312 Digital
Forensic_Final.docx
Book Path: /home/sebas_dev_linux/projects/course_generator/data/books/Bill
Nelson, Amelia Phillips, Christopher Steuart - Guide to Computer Forensics and
```

```
Investigations_ Processing Digital Evidence-Cengage Learning (2018).epub
Parsed UO Output Path:
/home/sebas_dev_linux/projects/course_generator/Parse_data/Parse_UO/ICT312
Digital Forensic_Final_parsed.json
Parsed ToC Output Path: /home/sebas_dev_linux/projects/course_generator/Parse_da
ta/Parse_TOC_books/ICT312_epub_table_of_contents.json
Vector DB Path: /home/sebas_dev_linux/projects/course_generator/data/DataBase_Ch
roma/chroma_db_toc_guided_chunks_epub
Vector DB Collection: book_toc_guided_chunks_epub_v2
--- SETUP COMPLETE ---
```

2 System Prompt

[2]:
```
UNIT_OUTLINE_SYSTEM_PROMPT_TEMPLATE = """
You are an expert academic assistant tasked with parsing a university unit␣
 ↪outline document and extracting key information into a structured JSON␣
 ↪format.

The input will be the raw text content of a unit outline. Your goal is to␣
 ↪identify and extract the following details and structure them precisely as␣
 ↪specified in the JSON schema below. Note: do not change any key name

**JSON Output Schema:**

```json
{{
  "unitInformation": {{
    "unitCode": "string | null",
    "unitName": "string | null",
    "creditPoints": "integer | null",
    "unitRationale": "string | null",
    "prerequisites": "string | null"
  }},
  "learningOutcomes": [
    "string"
  ],
  "assessments": [
    {{
      "taskName": "string",
      "description": "string",
      "dueWeek": "string | null",
      "weightingPercent": "integer | null",
      "learningOutcomesAssessed": "string | null"
    }}
  ],
  "weeklySchedule": [
```
```

```
    {{
      "week": "string",
      "contentTopic": "string",
      "requiredReading": "string | null"
    }}
  ],
  "requiredReadings": [
    "string"
  ],
  "recommendedReadings": [
    "string"
  ]
}}

Instructions for Extraction:
Unit Information: Locate Unit Code, Unit Name, Credit Points. Capture 'Unit␣
 ↪Overview / Rationale' as unitRationale. Identify prerequisites.
Learning Outcomes: Extract each learning outcome statement.
Assessments: Each task as an object. Capture full task name, description, Due␣
 ↪Week, Weighting % (number), and Learning Outcomes Assessed.
weeklySchedule: Each week as an object. Capture Week, contentTopic, and␣
 ↪requiredReading.
Required and Recommended Readings: List full text for each.
**Important Considerations for the LLM**:
Pay close attention to headings and table structures.
If information is missing, use null for string/integer fields, or an empty list␣
 ↪[] for array fields.
Do no change keys in the template given
Ensure the output is ONLY the JSON object, starting with {{{{ and ending with␣
 ↪}}}}. No explanations or conversational text before or after the JSON.
Now, parse the following unit outline text:
--- UNIT_OUTLINE_TEXT_START ---
{outline_text}
--- UNIT_OUTLINE_TEXT_END ---
"""
```

[3]:
```
# Place this in a new cell after your imports, or within Cell 3 before the␣
 ↪functions.
# This code is based on the schema from your screenshot on page 4.

from pydantic import BaseModel, Field, ValidationError
from typing import List, Optional
import time

# Define Pydantic models that match your JSON schema
class UnitInformation(BaseModel):
    unitCode: Optional[str] = None
```

```
    unitName: Optional[str] = None
    creditPoints: Optional[int] = None
    unitRationale: Optional[str] = None
    prerequisites: Optional[str] = None

class Assessment(BaseModel):
    taskName: str
    description: str
    dueWeek: Optional[str] = None
    weightingPercent: Optional[int] = None
    learningOutcomesAssessed: Optional[str] = None

class WeeklyScheduleItem(BaseModel):
    week: str
    contentTopic: str
    requiredReading: Optional[str] = None

class ParsedUnitOutline(BaseModel):
    unitInformation: UnitInformation
    learningOutcomes: List[str]
    assessments: List[Assessment]
    weeklySchedule: List[WeeklyScheduleItem]
    requiredReadings: List[str]
    recommendedReadings: List[str]
```

3 Extrac Unit outline details to process following steps - output raw json with UO details

[4]:
```
# Cell 3: Parse Unit Outline


# --- Helper Functions for Parsing ---
def extract_text_from_file(filepath: str) -> str:
    _, ext = os.path.splitext(filepath.lower())
    if ext == '.docx':
        doc = Document(filepath)
        full_text = [p.text for p in doc.paragraphs]
        for table in doc.tables:
            for row in table.rows:
                full_text.append(" | ".join(cell.text for cell in row.cells))
        return '\n'.join(full_text)
    elif ext == '.pdf':
        with pdfplumber.open(filepath) as pdf:
            return "\n".join(page.extract_text() for page in pdf.pages if page.
 ↪extract_text())
    else:
```

```
        raise TypeError(f"Unsupported file type: {ext}")

def parse_llm_json_output(content: str) -> dict:
    try:
        match = re.search(r'\{.*\}', content, re.DOTALL)
        if not match: return None
        return json.loads(match.group(0))
    except (json.JSONDecodeError, TypeError):
        return None

@retry(stop=stop_after_attempt(3), wait=wait_exponential(min=2, max=10))
def call_ollama_with_retry(client, prompt):
    logger.info(f"Calling Ollama model '{OLLAMA_MODEL}'...")
    response = client.chat(
        model=OLLAMA_MODEL,
        messages=[{"role": "user", "content": prompt}],
        format="json",
        options={"temperature": 0.0}
    )
    if not response or 'message' not in response or not response['message'].
 ↪get('content'):
        raise ValueError("Ollama returned an empty or invalid response.")
    return response['message']['content']

# --- Main Orchestration Function for this Cell ---
def parse_and_save_outline_robust(
    input_filepath: str,
    output_filepath: str,
    prompt_template: str,
    max_retries: int = 3
):
    logger.info(f"Starting to robustly process Unit Outline: {input_filepath}")

    if not os.path.exists(input_filepath):
        logger.error(f"Input file not found: {input_filepath}")
        return

    try:
        outline_text = extract_text_from_file(input_filepath)
        if not outline_text.strip():
            logger.error("Extracted text is empty. Aborting.")
            return
    except Exception as e:
        logger.error(f"Failed to extract text from file: {e}", exc_info=True)
        return

    client = ollama.Client(host=OLLAMA_HOST)
```

```
    current_prompt = prompt_template.format(outline_text=outline_text)

    for attempt in range(max_retries):
        logger.info(f"Attempt {attempt + 1}/{max_retries} to parse outline.")

        try:
            # Call the LLM
            llm_output_str = call_ollama_with_retry(client, current_prompt)

            # Find the JSON blob in the response
            json_blob = parse_llm_json_output(llm_output_str) # Your existing␣
↪helper
            if not json_blob:
                raise ValueError("LLM did not return a parsable JSON object.")

            # *** THE KEY VALIDATION STEP ***
            # Try to parse the dictionary into your Pydantic model.
            # This will raise a `ValidationError` if keys are wrong, types are␣
↪wrong, or fields are missing.
            parsed_data = ParsedUnitOutline.model_validate(json_blob)

            # If successful, save the validated data and exit the loop
            logger.info("Successfully validated JSON structure against Pydantic␣
↪model.")
            os.makedirs(os.path.dirname(output_filepath), exist_ok=True)
            with open(output_filepath, 'w', encoding='utf-8') as f:
                # Use .model_dump_json() for clean, validated output
                f.write(parsed_data.model_dump_json(indent=2))

            logger.info(f"Successfully parsed and saved Unit Outline to:␣
↪{output_filepath}")
            return # Exit function on success

        except ValidationError as e:
            logger.warning(f"Validation failed on attempt {attempt + 1}. Error:␣
↪{e}")
            # Formulate a new prompt with the error message for self-correction
            error_feedback = (
                f"\n\nYour previous attempt failed. You MUST correct the␣
↪following errors:\n"
                f"{e}\n\n"
                f"Please regenerate the entire JSON object, ensuring it␣
↪strictly adheres to the schema "
                f"and corrects these specific errors. Do not change any key␣
↪names."
            )
```

```
            current_prompt = current_prompt + error_feedback # Append the error␣
 ↪to the prompt

        except Exception as e:
            # Catch other errors like network issues from call_ollama_with_retry
            logger.error(f"An unexpected error occurred on attempt {attempt +␣
 ↪1}: {e}", exc_info=True)
            # You might want to wait before retrying for non-validation errors
            time.sleep(5)

    logger.error(f"Failed to get valid structured data from the LLM after␣
 ↪{max_retries} attempts.")


# --- In your execution block, call the new function ---
# parse_and_save_outline(...) becomes:

if EXTRACT_UO:
    parse_and_save_outline_robust(
        input_filepath=FULL_PATH_UNIT_OUTLINE,
        output_filepath=PARSED_UO_JSON_PATH,
        prompt_template=UNIT_OUTLINE_SYSTEM_PROMPT_TEMPLATE
    )
```

4 Extract TOC from epub or epub

[4]:
```
# Cell 4: Extract Book Table of Contents (ToC)
# This cell extracts the ToC from the specified book (EPUB or PDF)
# and saves it to the path defined in Cell 1.

from ebooklib import epub, ITEM_NAVIGATION
from bs4 import BeautifulSoup
import fitz  # PyMuPDF
import json

# --- EPUB Extraction Logic ---
def parse_navpoint(navpoint, level=0):
    # (Your existing parse_navpoint function)
    title = navpoint.navLabel.text.strip()
    # Add filtering logic here if needed
    node = {"level": level, "title": title, "children": []}
    for child_navpoint in navpoint.find_all('navPoint', recursive=False):
        child_node = parse_navpoint(child_navpoint, level + 1)
        if child_node: node["children"].append(child_node)
    return node
```

```python
def parse_li(li_element, level=0):
    # (Your existing parse_li function)
    a_tag = li_element.find('a')
    if a_tag:
        title = a_tag.get_text(strip=True)
        # Add filtering logic here if needed
        node = {"level": level, "title": title, "children": []}
        nested_ol = li_element.find('ol')
        if nested_ol:
            for sub_li in nested_ol.find_all('li', recursive=False):
                child_node = parse_li(sub_li, level + 1)
                if child_node: node["children"].append(child_node)
        return node
    return None

def extract_epub_toc(epub_path, output_json_path):
    print(f"Processing EPUB ToC for: {epub_path}")
    toc_data = []
    book = epub.read_epub(epub_path)
    for nav_item in book.get_items_of_type(ITEM_NAVIGATION):
        soup = BeautifulSoup(nav_item.get_content(), 'xml')
        if nav_item.get_name().endswith('.ncx'):
            print("INFO: Found EPUB 2 (NCX) Table of Contents.")
            navmap = soup.find('navMap')
            if navmap:
                for navpoint in navmap.find_all('navPoint', recursive=False):
                    node = parse_navpoint(navpoint, level=0)
                    if node: toc_data.append(node)
        else:
            print("INFO: Found EPUB 3 (XHTML) Table of Contents.")
            toc_nav = soup.select_one('nav[epub|type="toc"]')
            if toc_nav:
                top_ol = toc_nav.find('ol')
                if top_ol:
                    for li in top_ol.find_all('li', recursive=False):
                        node = parse_li(li, level=0)
                        if node: toc_data.append(node)
        if toc_data: break

    if toc_data:
        os.makedirs(os.path.dirname(output_json_path), exist_ok=True)
        with open(output_json_path, 'w', encoding='utf-8') as f:
            json.dump(toc_data, f, indent=2, ensure_ascii=False)
        print(f"  Successfully wrote EPUB ToC to: {output_json_path}")
    else:
        print("  WARNING: No ToC data extracted from EPUB.")
```

```python
# --- PDF Extraction Logic ---
def build_pdf_hierarchy(toc_list):
    """
    Builds a hierarchical structure from a flat ToC list from PyMuPDF.
    MODIFIED: Normalizes levels to start at 0 for consistency with EPUB.
    """
    root = []
    # The parent_stack keys are now level-based, starting from -1 for the root's parent.
    parent_stack = {-1: {"children": root}}

    for level, title, page in toc_list:
        # --- FIX: NORMALIZE LEVEL TO START AT 0 ---
        # fitz/PyMuPDF ToC levels start at 1, so we subtract 1.
        normalized_level = level - 1

        node = {
            "level": normalized_level,
            "title": title.strip(),
            "page": page,
            "children": []
        }

        # Find the correct parent in the stack. The parent's level is one less than the current node's.
        # This logic correctly places the node under its parent in the hierarchy.
        parent_node = parent_stack[normalized_level - 1]
        parent_node["children"].append(node)

        # Add the current node to the stack so it can be a parent for subsequent nodes.
        parent_stack[normalized_level] = node

    return root

def extract_pdf_toc(pdf_path, output_json_path):
    print(f"Processing PDF ToC for: {pdf_path}")
    try:
        doc = fitz.open(pdf_path)
        toc = doc.get_toc()
        if not toc:
            print("  WARNING: This PDF has no embedded bookmarks (ToC).")
            hierarchical_toc = []
        else:
            print(f"INFO: Found {len(toc)} bookmark entries.")
            hierarchical_toc = build_pdf_hierarchy(toc)
```

```
        os.makedirs(os.path.dirname(output_json_path), exist_ok=True)
        with open(output_json_path, 'w', encoding='utf-8') as f:
            json.dump(hierarchical_toc, f, indent=2, ensure_ascii=False)
        print(f"  Successfully wrote PDF ToC to: {output_json_path}")

    except Exception as e:
        print(f"An error occurred during PDF ToC extraction: {e}")

# --- Execute ToC Extraction ---
if PROCESS_EPUB:
    extract_epub_toc(BOOK_PATH, PRE_EXTRACTED_TOC_JSON_PATH)
else:
    extract_pdf_toc(BOOK_PATH, PRE_EXTRACTED_TOC_JSON_PATH)
```

```
Processing EPUB ToC for:
/home/sebas_dev_linux/projects/course_generator/data/books/Bill Nelson, Amelia
Phillips, Christopher Steuart - Guide to Computer Forensics and Investigations_
Processing Digital Evidence-Cengage Learning (2018).epub
INFO: Found EPUB 2 (NCX) Table of Contents.
  Successfully wrote EPUB ToC to: /home/sebas_dev_linux/projects/course_generato
r/Parse_data/Parse_TOC_books/ICT312_epub_table_of_contents.json
```

5 Hirachical DB base on TOC

5.1 Process Book

[162]:
```
# Cell 5: Create Hierarchical Vector Database (with Sequential ToC ID and Chunk␣
 ↪ID)
# This cell processes the book, enriches it with hierarchical and sequential␣
 ↪metadata,
# chunks it, and creates the final vector database.

import os
import json
import shutil
import logging
from typing import List, Dict, Any, Tuple
from langchain_core.documents import Document
from langchain_community.document_loaders import PyPDFLoader,␣
 ↪UnstructuredEPubLoader
from langchain_ollama.embeddings import OllamaEmbeddings
from langchain_chroma import Chroma
from langchain.text_splitter import RecursiveCharacterTextSplitter

# Setup Logger for this cell
```

```
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
logger = logging.getLogger(__name__)

# --- Helper: Clean metadata values for ChromaDB ---
def clean_metadata_for_chroma(value: Any) -> Any:
    """Sanitizes metadata values to be compatible with ChromaDB."""
    if isinstance(value, list): return ", ".join(map(str, value))
    if isinstance(value, dict): return json.dumps(value)
    if isinstance(value, (str, int, float, bool)) or value is None: return value
    return str(value)

# --- Core Function to Process Book with Pre-extracted ToC ---
def process_book_with_extracted_toc(
    book_path: str,
    extracted_toc_json_path: str,
    chunk_size: int,
    chunk_overlap: int
) -> Tuple[List[Document], List[Dict[str, Any]]]:

    logger.info(f"Processing book '{os.path.basename(book_path)}' using ToC␣
 ↪from '{os.path.basename(extracted_toc_json_path)}'.")

    # 1. Load the pre-extracted hierarchical ToC
    try:
        with open(extracted_toc_json_path, 'r', encoding='utf-8') as f:
            hierarchical_toc = json.load(f)
        if not hierarchical_toc:
            logger.error(f"Pre-extracted ToC at '{extracted_toc_json_path}' is␣
 ↪empty or invalid.")
            return [], []
        logger.info(f"Successfully loaded pre-extracted ToC with␣
 ↪{len(hierarchical_toc)} top-level entries.")
    except Exception as e:
        logger.error(f"Error loading pre-extracted ToC JSON: {e}",␣
 ↪exc_info=True)
        return [], []

    # 2. Load all text elements/pages from the book
    all_raw_book_docs: List[Document] = []
    _, file_extension = os.path.splitext(book_path.lower())

    if file_extension == ".epub":
        loader = UnstructuredEPubLoader(book_path, mode="elements",␣
 ↪strategy="fast")
        try:
```

```
            all_raw_book_docs = loader.load()
            logger.info(f"Loaded {len(all_raw_book_docs)} text elements from␣
↪EPUB.")
        except Exception as e:
            logger.error(f"Error loading EPUB content: {e}", exc_info=True)
            return [], hierarchical_toc
    elif file_extension == ".pdf":
        loader = PyPDFLoader(book_path)
        try:
            all_raw_book_docs = loader.load()
            logger.info(f"Loaded {len(all_raw_book_docs)} pages from PDF.")
        except Exception as e:
            logger.error(f"Error loading PDF content: {e}", exc_info=True)
            return [], hierarchical_toc
    else:
        logger.error(f"Unsupported book file format: {file_extension}")
        return [], hierarchical_toc

    if not all_raw_book_docs:
        logger.error("No text elements/pages loaded from the book.")
        return [], hierarchical_toc

    # 3. Create enriched LangChain Documents by matching ToC to content
    final_documents_with_metadata: List[Document] = []

    # Flatten the ToC, AND add a unique sequential ID for sorting and␣
↪validation.
    flat_toc_entries: List[Dict[str, Any]] = []

    def _add_ids_and_flatten_recursive(nodes: List[Dict[str, Any]],␣
↪current_titles_path: List[str], counter: List[int]):
        """
        Recursively traverses ToC nodes to flatten them and assign a unique,␣
↪sequential toc_id.
        """
        for node in nodes:
            toc_id = counter[0]
            counter[0] += 1
            title = node.get("title", "").strip()
            if not title: continue
            new_titles_path = current_titles_path + [title]
            entry = {
                "titles_path": new_titles_path,
                "level": node.get("level"),
                "full_title_for_matching": title,
                "toc_id": toc_id
            }
```

```
            if "page" in node: entry["page"] = node["page"]
            flat_toc_entries.append(entry)
            if node.get("children"):
                _add_ids_and_flatten_recursive(node.get("children", []),␣
↪new_titles_path, counter)

    toc_id_counter = [0]
    _add_ids_and_flatten_recursive(hierarchical_toc, [], toc_id_counter)
    logger.info(f"Flattened ToC and assigned sequential IDs to␣
↪{len(flat_toc_entries)} entries.")

    # Logic for PDF metadata assignment
    if file_extension == ".pdf" and any("page" in entry for entry in␣
↪flat_toc_entries):
        logger.info("Assigning metadata to PDF pages based on ToC page numbers..
↪.")
        flat_toc_entries.sort(key=lambda x: x.get("page", -1) if x.get("page")␣
↪is not None else -1)
        for page_doc in all_raw_book_docs:
            page_num_0_indexed = page_doc.metadata.get("page", -1)
            page_num_1_indexed = page_num_0_indexed + 1
            assigned_metadata = {"source": os.path.basename(book_path),␣
↪"page_number": page_num_1_indexed}
            best_match_toc_entry = None
            for toc_entry in flat_toc_entries:
                toc_page = toc_entry.get("page")
                if toc_page is not None and toc_page <= page_num_1_indexed:
                    if best_match_toc_entry is None or toc_page >␣
↪best_match_toc_entry.get("page", -1):
                        best_match_toc_entry = toc_entry
                elif toc_page is not None and toc_page > page_num_1_indexed:
                    break
            if best_match_toc_entry:
                for i, title_in_path in␣
↪enumerate(best_match_toc_entry["titles_path"]):
                    assigned_metadata[f"level_{i+1}_title"] = title_in_path
                assigned_metadata['toc_id'] = best_match_toc_entry.get('toc_id')
            else:
                assigned_metadata["level_1_title"] = "Uncategorized PDF Page"
            cleaned_meta = {k: clean_metadata_for_chroma(v) for k, v in␣
↪assigned_metadata.items()}
            final_documents_with_metadata.append(Document(page_content=page_doc.
↪page_content, metadata=cleaned_meta))

    # Logic for EPUB metadata assignment
    elif file_extension == ".epub":
```

```
        logger.info("Assigning metadata to EPUB elements by matching ToC titles␣
↪in text...")
        toc_titles_for_search = [entry for entry in flat_toc_entries if entry.
↪get("full_title_for_matching")]
        current_hierarchy_metadata = {}
        for element_doc in all_raw_book_docs:
            element_text = element_doc.page_content.strip() if element_doc.
↪page_content else ""
            if not element_text: continue
            for toc_entry in toc_titles_for_search:
                if element_text == toc_entry["full_title_for_matching"]:
                    current_hierarchy_metadata = {"source": os.path.
↪basename(book_path)}
                    for i, title_in_path in enumerate(toc_entry["titles_path"]):
                        current_hierarchy_metadata[f"level_{i+1}_title"] =␣
↪title_in_path
                    current_hierarchy_metadata['toc_id'] = toc_entry.
↪get('toc_id')
                    if "page" in toc_entry:␣
↪current_hierarchy_metadata["epub_toc_page"] = toc_entry["page"]
                    break
            if not current_hierarchy_metadata:
                doc_metadata_to_assign = {"source": os.path.
↪basename(book_path), "level_1_title": "EPUB Preamble", "toc_id": -1}
            else:
                doc_metadata_to_assign = current_hierarchy_metadata.copy()
            cleaned_meta = {k: clean_metadata_for_chroma(v) for k, v in␣
↪doc_metadata_to_assign.items()}
            final_documents_with_metadata.
↪append(Document(page_content=element_text, metadata=cleaned_meta))

    else: # Fallback
        final_documents_with_metadata = all_raw_book_docs

    if not final_documents_with_metadata:
        logger.error("No documents were processed or enriched with hierarchical␣
↪metadata.")
        return [], hierarchical_toc

    logger.info(f"Total documents prepared for chunking:␣
↪{len(final_documents_with_metadata)}")

    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size,
        chunk_overlap=chunk_overlap,
        length_function=len
```

```
    )
    final_chunks = text_splitter.split_documents(final_documents_with_metadata)
    logger.info(f"Split into {len(final_chunks)} final chunks, inheriting␣
 ↪hierarchical metadata.")

    # --- MODIFICATION START: Add a unique, sequential chunk_id to each chunk␣
 ↪---
    logger.info("Assigning sequential chunk_id to all final chunks...")
    for i, chunk in enumerate(final_chunks):
        chunk.metadata['chunk_id'] = i
    logger.info(f"Assigned chunk_ids from 0 to {len(final_chunks) - 1}.")
    # --- MODIFICATION END ---

    return final_chunks, hierarchical_toc

# --- Main Execution Block for this Cell ---

if not os.path.exists(PRE_EXTRACTED_TOC_JSON_PATH):
    logger.error(f"CRITICAL: Pre-extracted ToC file not found at␣
 ↪'{PRE_EXTRACTED_TOC_JSON_PATH}'.")
    logger.error("Please run the 'Extract Book Table of Contents (ToC)' cell␣
 ↪(Cell 4) first.")
else:
    final_chunks_for_db, toc_reloaded = process_book_with_extracted_toc(
        book_path=BOOK_PATH,
        extracted_toc_json_path=PRE_EXTRACTED_TOC_JSON_PATH,
        chunk_size=CHUNK_SIZE,
        chunk_overlap=CHUNK_OVERLAP
    )

    if final_chunks_for_db:
        if os.path.exists(CHROMA_PERSIST_DIR):
            logger.warning(f"Deleting existing ChromaDB directory:␣
 ↪{CHROMA_PERSIST_DIR}")
            shutil.rmtree(CHROMA_PERSIST_DIR)

        logger.info(f"Initializing embedding model '{EMBEDDING_MODEL_OLLAMA}'␣
 ↪and creating new vector database...")
        embedding_model = OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA)

        vector_db = Chroma.from_documents(
            documents=final_chunks_for_db,
            embedding=embedding_model,
            persist_directory=CHROMA_PERSIST_DIR,
            collection_name=CHROMA_COLLECTION_NAME
        )
```

```
        reloaded_db = Chroma(persist_directory=CHROMA_PERSIST_DIR,␣
↪embedding_function=embedding_model, collection_name=CHROMA_COLLECTION_NAME)
        count = reloaded_db._collection.count()

        print("-" * 50)
        logger.info(f" Vector DB created successfully at:␣
↪{CHROMA_PERSIST_DIR}")
        logger.info(f" Collection '{CHROMA_COLLECTION_NAME}' contains {count}␣
↪documents.")
        print("-" * 50)
    else:
        logger.error(" Failed to generate chunks. Vector DB not created.")
```

```
2025-07-02 16:00:52,108 - INFO - Processing book 'Bill Nelson, Amelia Phillips,
Christopher Steuart - Guide to Computer Forensics and Investigations_ Processing
Digital Evidence-Cengage Learning (2018).epub' using ToC from
'ICT312_epub_table_of_contents.json'.
2025-07-02 16:00:52,109 - INFO - Successfully loaded pre-extracted ToC with 28
top-level entries.
2025-07-02 16:00:54,400 - INFO - Note: NumExpr detected 32 cores but
"NUMEXPR_MAX_THREADS" not set, so enforcing safe limit of 16.
2025-07-02 16:00:54,400 - INFO - NumExpr defaulting to 16 threads.
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[162], line 189
    187     logger.error("Please run the 'Extract Book Table of Contents (ToC)'␣
 ↪cell (Cell 4) first.")
    188 else:
--> 189     final_chunks_for_db, toc_reloaded = process_book_with_extracted_toc
    190         book_path=BOOK_PATH,
    191         extracted_toc_json_path=PRE_EXTRACTED_TOC_JSON_PATH,
    192         chunk_size=CHUNK_SIZE,
    193         chunk_overlap=CHUNK_OVERLAP
    194     )
    196     if final_chunks_for_db:
    197         if os.path.exists(CHROMA_PERSIST_DIR):

Cell In[162], line 57, in process_book_with_extracted_toc(book_path,␣
 ↪extracted_toc_json_path, chunk_size, chunk_overlap)
     55 loader = UnstructuredEPubLoader(book_path, mode="elements",␣
 ↪strategy="fast")
     56 try:
---> 57     all_raw_book_docs = loader.load()
     58     logger.info(f"Loaded {len(all_raw_book_docs)} text elements from␣
 ↪EPUB.")
```

```
    59 except Exception as e:

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/langchain_core/
 ↪document_loaders/base.py:32, in BaseLoader.load(self)
    30 def load(self) -> list[Document]:
    31     """Load data into Document objects."""
---> 32     return list(self.lazy_load())

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/langchain_community
 ↪document_loaders/unstructured.py:107, in UnstructuredBaseLoader.lazy_load(sel
 ↪)
   105 def lazy_load(self) -> Iterator[Document]:
   106     """Load file."""
--> 107     elements = self._get_elements()
   108     self._post_process_elements(elements)
   109     if self.mode == "elements":

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/langchain_community
 ↪document_loaders/epub.py:55, in UnstructuredEPubLoader._get_elements(self)
    52 def _get_elements(self) -> List:
    53     from unstructured.partition.epub import partition_epub
---> 55     return␣
 ↪partition_epub(filename=self.file_path, **self.unstructured_kwargs)

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/unstructured/
 ↪partition/epub.py:48, in partition_epub(filename, file, metadata_filename,␣
 ↪metadata_last_modified, languages, detect_language_per_element, **kwargs)
    44 exactly_one(filename=filename, file=file)
    46 last_modified = get_last_modified_date(filename) if filename else None
---> 48 html_text = convert_file_to_html_text_using_pandoc(
    49     source_format="epub", filename=filename, file=file
    50 )
    52 return partition_html(
    53     text=html_text,
    54     metadata_filename=metadata_filename or filename,
  (…)
    60     **kwargs,
    61 )

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/unstructured/
 ↪file_utils/file_conversion.py:67, in␣
 ↪convert_file_to_html_text_using_pandoc(source_format, filename, file)
    62         return convert_file_to_text(
    63             filename=tmp_file_path, source_format=source_format,␣
 ↪target_format="html"
    64         )
    66 assert filename is not None
---> 67 return convert_file_to_text(
    68     filename=filename, source_format=source_format, target_format="html
```

```
     69 )

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/unstructured/utils.
 ↪py:216, in requires_dependencies.<locals>.decorator.<locals>.wrapper(*args,␣
 ↪**kwargs)
    213 @wraps(func)
    214 def wrapper(*args: _P.args, **kwargs: _P.kwargs):
    215     run_check()
--> 216     return func(*args, **kwargs)

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/unstructured/
 ↪file_utils/file_conversion.py:17, in convert_file_to_text(filename,␣
 ↪source_format, target_format)
     14 import pypandoc
     16 try:
---> 17     text =␣
 ↪pypandoc.convert_file(filename, target_format, format=source_format, sandbox=True)
     18 except FileNotFoundError as err:
     19     msg = (
     20         f"Error converting the file to text. Ensure you have the pandoc␣
 ↪package installed on"
     21         f" your system. Installation instructions are available at"
     22         f" https://pandoc.org/installing.html. The original exception␣
 ↪text was:\n{err}"
     23     )

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/pypandoc/__init__.py:
 ↪206, in convert_file(source_file, to, format, extra_args, encoding,␣
 ↪outputfile, filters, verify_format, sandbox, cworkdir, sort_files)
    203 if len(discovered_source_files) == 1:
    204     discovered_source_files = discovered_source_files[0]
--> 206 return␣
 ↪_convert_input(discovered_source_files, format, 'path', to, extra_args=extra_args,
    207                 outputfile=outputfile, filters=filters,
    208                 verify_format=verify_format, sandbox=sandbox,
    209                 cworkdir=cworkdir, sort_files=sort_files)

File ~/miniconda3/envs/tensor2/lib/python3.10/site-packages/pypandoc/__init__.py:
 ↪451, in _convert_input(source, format, input_type, to, extra_args, outputfile␣
 ↪filters, verify_format, sandbox, cworkdir, sort_files)
    449         pass
    450 try:
--> 451     stdout, stderr = p.communicate(source if string_input else None)
    452 except OSError:
    453     # this is happening only on Py2.6 when pandoc dies before reading all
    454     # the input. We treat that the same as when we exit with an error…
    455     raise RuntimeError('Pandoc died with exitcode "%s" during conversion.
 ↪' % (p.returncode))
```

```
File ~/miniconda3/envs/tensor2/lib/python3.10/subprocess.py:1154, in Popen.
 ↪communicate(self, input, timeout)
   1151     endtime = None
   1153 try:
-> 1154     stdout, stderr = self._communicate(input, endtime, timeout)
   1155 except KeyboardInterrupt:
   1156     # https://bugs.python.org/issue25942
   1157     # See the detailed comment in .wait().
   1158     if timeout is not None:

File ~/miniconda3/envs/tensor2/lib/python3.10/subprocess.py:2021, in Popen.
 ↪_communicate(self, input, endtime, orig_timeout)
   2014     self._check_timeout(endtime, orig_timeout,
   2015                         stdout, stderr,
   2016                         skip_check_and_raise=True)
   2017     raise RuntimeError(  # Impossible :)
   2018         '_check_timeout(…, skip_check_and_raise=True) '
   2019         'failed to raise TimeoutExpired.')
-> 2021 ready = selector.select(timeout)
   2022 self._check_timeout(endtime, orig_timeout, stdout, stderr)
   2024 # XXX Rewrite these to use non-blocking I/O on the file
   2025 # objects; they are no longer using C stdio!

File ~/miniconda3/envs/tensor2/lib/python3.10/selectors.py:416, in␣
 ↪_PollLikeSelector.select(self, timeout)
    414 ready = []
    415 try:
--> 416     fd_event_list = self._selector.poll(timeout)
    417 except InterruptedError:
    418     return ready

KeyboardInterrupt:
```

5.1.1 Full Database Health & Hierarchy Diagnostic Report

[ ]:
```
# Cell 5.1: Full Database Health & Hierarchy Diagnostic Report (V5 - with␣
 ↪Content Preview)

import os
import json
import logging
import random
from typing import List, Dict, Any

# You might need to install pandas if you haven't already
try:
```

```
    import pandas as pd
    pandas_available = True
except ImportError:
    pandas_available = False

try:
    from langchain_chroma import Chroma
    from langchain_ollama.embeddings import OllamaEmbeddings
    from langchain_core.documents import Document
    langchain_available = True
except ImportError:
    langchain_available = False

# Setup Logger
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
logger = logging.getLogger(__name__)

# --- HELPER FUNCTIONS ---
def print_header(text: str, char: str = "="):
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)

def count_total_chunks(node: Dict) -> int:
    """Recursively counts all chunks in a node and its children."""
    total = node.get('_chunks', 0)
    for child_node in node.get('_children', {}).values():
        total += count_total_chunks(child_node)
    return total

def print_hierarchy_report(node: Dict, indent_level: int = 0):
    """
    Recursively prints the reconstructed hierarchy, sorting by sequential ToC␣
 ↪ID.
    """
    sorted_children = sorted(
        node.get('_children', {}).items(),
        key=lambda item: item[1].get('_toc_id', float('inf'))
    )

    for title, child_node in sorted_children:
        prefix = "  " * indent_level + "|-- "
        total_chunks_in_branch = count_total_chunks(child_node)
        direct_chunks = child_node.get('_chunks', 0)
        toc_id = child_node.get('_toc_id', 'N/A')
```

```
        print(f"{prefix}{title} [ID: {toc_id}] (Total Chuck in branch:␣
 ↪{total_chunks_in_branch}, Direct Chunk: {direct_chunks})")
        print_hierarchy_report(child_node, indent_level + 1)

def find_testable_sections(node: Dict, path: str, testable_list: List):
    """
    Recursively find sections with a decent number of "direct" chunks to test␣
 ↪sequence on.
    """
    if node.get('_chunks', 0) > 10 and not node.get('_children'):
        testable_list.append({
            "path": path,
            "toc_id": node.get('_toc_id'),
            "chunk_count": node.get('_chunks')
        })

    for title, child_node in node.get('_children', {}).items():
        new_path = f"{path} -> {title}" if path else title
        find_testable_sections(child_node, new_path, testable_list)


# --- MODIFIED TEST FUNCTION ---
def verify_chunk_sequence_and_content(vector_store: Chroma, hierarchy_tree:␣
 ↪Dict):
    """
    Selects a random ToC section, verifies chunk sequence, and displays the␣
 ↪reassembled content.
    """
    print_header("Chunk Sequence & Content Integrity Test", char="-")
    logger.info("Verifying chunk order and reassembling content for a random␣
 ↪ToC section.")

    # 1. Find a good section to test
    testable_sections = []
    find_testable_sections(hierarchy_tree, "", testable_sections)

    if not testable_sections:
        logger.warning("Could not find a suitable section with enough chunks to␣
 ↪test. Skipping content test.")
        return

    random_section = random.choice(testable_sections)
    test_toc_id = random_section['toc_id']
    section_title = random_section['path'].split(' -> ')[-1]
```

```
    logger.info(f"Selected random section for testing: '{random_section['path']}' (toc_id: {test_toc_id})")

    # 2. Retrieve all documents (content + metadata) for that toc_id
    try:
        # Use .get() to retrieve full documents, not just similarity search
        retrieved_data = vector_store.get(
            where={"toc_id": test_toc_id},
            include=["metadatas", "documents"]
        )

        # Combine metadatas and documents into LangChain Document objects
        docs = [Document(page_content=doc, metadata=meta) for doc, meta in zip(retrieved_data['documents'], retrieved_data['metadatas'])]

        logger.info(f"Retrieved {len(docs)} document chunks for toc_id {test_toc_id}.")

        if len(docs) < 1:
            logger.warning("No chunks found in the selected section. Skipping.")
            return

        # 3. Sort the documents by chunk_id
        # Handle cases where chunk_id might be missing for robustness
        docs.sort(key=lambda d: d.metadata.get('chunk_id', -1))

        chunk_ids = [d.metadata.get('chunk_id') for d in docs]
        if None in chunk_ids:
            logger.error("TEST FAILED: Some retrieved chunks are missing a 'chunk_id'.")
            return

        # 4. Verify sequence
        is_sequential = all(chunk_ids[i] == chunk_ids[i-1] + 1 for i in range(1, len(chunk_ids)))

        # 5. Reassemble and print content
        full_content = "\n".join([d.page_content for d in docs])

        print("\n" + "-"*25 + " CONTENT PREVIEW " + "-"*25)
        print(f"Title: {section_title} [toc_id: {test_toc_id}]")
        print(f"Chunk IDs: {chunk_ids}")
        print("-" * 70)
        print(full_content)
        print("-" * 23 + " END CONTENT PREVIEW " + "-"*23 + "\n")

        if is_sequential:
```

```
            logger.info("  TEST PASSED: Chunk IDs for the section are␣
 ↪sequential and content is reassembled.")
        else:
            logger.warning("TEST PASSED (with note): Chunk IDs are not␣
 ↪perfectly sequential but are in increasing order.")
            logger.warning("This is acceptable. Sorting by chunk_id␣
 ↪successfully restored narrative order.")

    except Exception as e:
        logger.error(f"TEST FAILED: An error occurred during chunk sequence␣
 ↪verification: {e}", exc_info=True)


# --- MAIN DIAGNOSTIC FUNCTION ---
def run_full_diagnostics():
    if not langchain_available:
        logger.error("LangChain components not installed. Skipping diagnostics.
 ↪")
        return
    if not pandas_available:
        logger.warning("Pandas not installed. Some reports may not be available.
 ↪")

    print_header("Full Database Health & Hierarchy Diagnostic Report")

    # 1. Connect to the Database
    logger.info("Connecting to the vector database...")
    if not os.path.exists(CHROMA_PERSIST_DIR):
        logger.error(f"FATAL: Chroma DB directory not found at␣
 ↪{CHROMA_PERSIST_DIR}.")
        return

    vector_store = Chroma(
        persist_directory=CHROMA_PERSIST_DIR,
        embedding_function=OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA),
        collection_name=CHROMA_COLLECTION_NAME
    )
    logger.info("Successfully connected to the database.")

    # 2. Retrieve ALL Metadata
    total_docs = vector_store._collection.count()
    if total_docs == 0:
        logger.warning("Database is empty. No diagnostics to run.")
        return

    logger.info(f"Retrieving metadata for all {total_docs} chunks...")
```

```
    metadatas = vector_store.get(limit=total_docs,␣
↪include=["metadatas"])['metadatas']
    logger.info("Successfully retrieved all metadata.")

    # 3. Reconstruct the Hierarchy Tree
    logger.info("Reconstructing hierarchy from chunk metadata...")
    hierarchy_tree = {'_children': {}}
    chunks_without_id = 0

    for meta in metadatas:
        toc_id = meta.get('toc_id')
        if toc_id is None or toc_id == -1:
            chunks_without_id += 1
            node_title = meta.get('level_1_title', 'Orphaned Chunks')
            if node_title not in hierarchy_tree['_children']:
                hierarchy_tree['_children'][node_title] = {'_children': {},␣
↪'_chunks': 0, '_toc_id': float('inf')}
            hierarchy_tree['_children'][node_title]['_chunks'] += 1
            continue

        current_node = hierarchy_tree
        for level in range(1, 7):
            level_key = f'level_{level}_title'
            title = meta.get(level_key)
            if not title: break
            if title not in current_node['_children']:
                current_node['_children'][title] = {'_children': {}, '_chunks':␣
↪0, '_toc_id': float('inf')}
            current_node = current_node['_children'][title]

        current_node['_chunks'] += 1
        current_node['_toc_id'] = min(current_node['_toc_id'], toc_id)

    logger.info("Hierarchy reconstruction complete.")

    # 4. Print Hierarchy Report
    print_header("Reconstructed Hierarchy Report (Book Order)", char="-")
    print_hierarchy_report(hierarchy_tree)

    # 5. Run Chunk Sequence and Content Test
    verify_chunk_sequence_and_content(vector_store, hierarchy_tree)

    # 6. Final Summary
    print_header("Diagnostic Summary", char="-")
    print(f"Total Chunks in DB: {total_docs}")

    if chunks_without_id > 0:
```

```
        logger.warning(f"Found {chunks_without_id} chunks MISSING a valid␣
 ↪'toc_id'. Check 'Orphaned' sections.")
    else:
        logger.info("All chunks contain valid 'toc_id' metadata. Sequential␣
 ↪integrity is maintained.")

    print_header("Diagnostic Complete")

# --- Execute Diagnostics ---
if 'CHROMA_PERSIST_DIR' in locals() and langchain_available:
    run_full_diagnostics()
else:
    logger.error("Skipping diagnostics: Global variables not defined or␣
 ↪LangChain not available.")
```

```
2025-07-01 22:02:40,404 - INFO - Connecting to the vector database…
2025-07-01 22:02:40,421 - INFO - Successfully connected to the database.
2025-07-01 22:02:40,458 - INFO - Retrieving metadata for all 11774 chunks…


================================================================================
               Full Database Health & Hierarchy Diagnostic Report
================================================================================

2025-07-01 22:02:41,229 - INFO - Successfully retrieved all metadata.
2025-07-01 22:02:41,229 - INFO - Reconstructing hierarchy from chunk metadata…
2025-07-01 22:02:41,239 - INFO - Hierarchy reconstruction complete.
2025-07-01 22:02:41,241 - INFO - Verifying chunk order and reassembling content
for a random ToC section.
2025-07-01 22:02:41,241 - INFO - Selected random section for testing: 'Chapter
4. Processing Crime and Incident Scenes -> Collecting Evidence in Private-Sector
Incident Scenes' (toc_id: 147)
2025-07-01 22:02:41,249 - INFO - Retrieved 24 document chunks for toc_id 147.
2025-07-01 22:02:41,249 - INFO -   TEST PASSED: Chunk IDs for the section are
sequential and content is reassembled.
2025-07-01 22:02:41,249 - WARNING - Found 21 chunks MISSING a valid 'toc_id'.
Check 'Orphaned' sections.


--------------------------------------------------------------------------------
                  Reconstructed Hierarchy Report (Book Order)
--------------------------------------------------------------------------------
|-- Preface [ID: 3] (Total Chuck in branch: 10, Direct Chunk: 10)
|-- Introduction [ID: 4] (Total Chuck in branch: 73, Direct Chunk: 73)
|-- About the Authors [ID: 5] (Total Chuck in branch: 5, Direct Chunk: 5)
|-- Acknowledgments [ID: 6] (Total Chuck in branch: 20, Direct Chunk: 20)
|-- Chapter 1. Understanding the Digital Forensics Profession and Investigations
[ID: 7] (Total Chuck in branch: 4566, Direct Chunk: 23)
  |-- An Overview of Digital Forensics [ID: 9] (Total Chuck in branch: 60,
```

Direct Chunk: 18)
    |-- Digital Forensics and Other Related Disciplines [ID: 10] (Total Chuck in branch: 18, Direct Chunk: 18)
    |-- A Brief History of Digital Forensics [ID: 11] (Total Chuck in branch: 13, Direct Chunk: 13)
    |-- Understanding Case Law [ID: 12] (Total Chuck in branch: 3, Direct Chunk: 3)
    |-- Developing Digital Forensics Resources [ID: 13] (Total Chuck in branch: 8, Direct Chunk: 8)
  |-- Preparing for Digital Investigations [ID: 14] (Total Chuck in branch: 84, Direct Chunk: 5)
    |-- Understanding Law Enforcement Agency Investigations [ID: 15] (Total Chuck in branch: 10, Direct Chunk: 10)
    |-- Following Legal Processes [ID: 16] (Total Chuck in branch: 13, Direct Chunk: 13)
    |-- Understanding Private-Sector Investigations [ID: 17] (Total Chuck in branch: 56, Direct Chunk: 3)
      |-- Establishing Company Policies [ID: 18] (Total Chuck in branch: 5, Direct Chunk: 5)
      |-- Displaying Warning Banners [ID: 19] (Total Chuck in branch: 19, Direct Chunk: 19)
      |-- Designating an Authorized Requester [ID: 20] (Total Chuck in branch: 9, Direct Chunk: 9)
      |-- Conducting Security Investigations [ID: 21] (Total Chuck in branch: 15, Direct Chunk: 15)
      |-- Distinguishing Personal and Company Property [ID: 22] (Total Chuck in branch: 5, Direct Chunk: 5)
  |-- Maintaining Professional Conduct [ID: 23] (Total Chuck in branch: 10, Direct Chunk: 10)
  |-- Preparing a Digital Forensics Investigation [ID: 24] (Total Chuck in branch: 97, Direct Chunk: 4)
    |-- An Overview of a Computer Crime [ID: 25] (Total Chuck in branch: 12, Direct Chunk: 12)
    |-- An Overview of a Company Policy Violation [ID: 26] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Taking a Systematic Approach [ID: 27] (Total Chuck in branch: 77, Direct Chunk: 16)
      |-- Assessing the Case [ID: 28] (Total Chuck in branch: 11, Direct Chunk: 11)
      |-- Planning Your Investigation [ID: 29] (Total Chuck in branch: 41, Direct Chunk: 41)
      |-- Securing Your Evidence [ID: 30] (Total Chuck in branch: 9, Direct Chunk: 9)
  |-- Procedures for Private-Sector High-Tech Investigations [ID: 31] (Total Chuck in branch: 124, Direct Chunk: 2)
    |-- Employee Termination Cases [ID: 32] (Total Chuck in branch: 2, Direct Chunk: 2)
    |-- Internet Abuse Investigations [ID: 33] (Total Chuck in branch: 19,

Direct Chunk: 19)
    |-- E-mail Abuse Investigations [ID: 34] (Total Chuck in branch: 16, Direct Chunk: 16)
    |-- Attorney-Client Privilege Investigations [ID: 35] (Total Chuck in branch: 33, Direct Chunk: 33)
    |-- Industrial Espionage Investigations [ID: 36] (Total Chuck in branch: 52, Direct Chunk: 41)
      |-- Interviews and Interrogations in High-Tech Investigations [ID: 37] (Total Chuck in branch: 11, Direct Chunk: 11)
  |-- Understanding Data Recovery Workstations and Software [ID: 38] (Total Chuck in branch: 37, Direct Chunk: 18)
    |-- Setting Up Your Workstation for Digital Forensics [ID: 39] (Total Chuck in branch: 19, Direct Chunk: 19)
  |-- Conducting an Investigation [ID: 40] (Total Chuck in branch: 109, Direct Chunk: 8)
    |-- Gathering the Evidence [ID: 41] (Total Chuck in branch: 14, Direct Chunk: 14)
    |-- Understanding Bit-stream Copies [ID: 42] (Total Chuck in branch: 8, Direct Chunk: 6)
      |-- Acquiring an Image of Evidence Media [ID: 43] (Total Chuck in branch: 2, Direct Chunk: 2)
    |-- Analyzing Your Digital Evidence [ID: 44] (Total Chuck in branch: 48, Direct Chunk: 44)
      |-- Some Additional Features of Autopsy [ID: 45] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Completing the Case [ID: 46] (Total Chuck in branch: 22, Direct Chunk: 12)
      |-- Autopsy's Report Generator [ID: 47] (Total Chuck in branch: 10, Direct Chunk: 10)
    |-- Critiquing the Case [ID: 48] (Total Chuck in branch: 9, Direct Chunk: 9)
  |-- Chapter Review [ID: inf] (Total Chuck in branch: 4022, Direct Chunk: 0)
    |-- Chapter Summary [ID: 50] (Total Chuck in branch: 211, Direct Chunk: 211)
    |-- Key Terms [ID: 51] (Total Chuck in branch: 309, Direct Chunk: 309)
    |-- Review Questions [ID: 52] (Total Chuck in branch: 1785, Direct Chunk: 1785)
    |-- Hands-On Projects [ID: 53] (Total Chuck in branch: 1527, Direct Chunk: 1527)
    |-- Case Projects [ID: 54] (Total Chuck in branch: 190, Direct Chunk: 190)
|-- Chapter 2. The Investigator's Office and Laboratory [ID: 55] (Total Chuck in branch: 331, Direct Chunk: 22)
  |-- Understanding Forensics Lab Accreditation Requirements [ID: 57] (Total Chuck in branch: 86, Direct Chunk: 7)
    |-- Identifying Duties of the Lab Manager and Staff [ID: 58] (Total Chuck in branch: 7, Direct Chunk: 7)
    |-- Lab Budget Planning [ID: 59] (Total Chuck in branch: 18, Direct Chunk: 18)
    |-- Acquiring Certification and Training [ID: 60] (Total Chuck in branch: 54, Direct Chunk: 4)

|-- International Association of Computer Investigative Specialists [ID: 61] (Total Chuck in branch: 13, Direct Chunk: 13)
|-- ISC2 Certified Cyber Forensics Professional [ID: 62] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- High Tech Crime Network [ID: 63] (Total Chuck in branch: 19, Direct Chunk: 19)
|-- EnCase Certified Examiner Certification [ID: 64] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- AccessData Certified Examiner [ID: 65] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Other Training and Certifications [ID: 66] (Total Chuck in branch: 12, Direct Chunk: 12)
|-- Determining the Physical Requirements for a Digital Forensics Lab [ID: 67] (Total Chuck in branch: 68, Direct Chunk: 3)
|-- Identifying Lab Security Needs [ID: 68] (Total Chuck in branch: 9, Direct Chunk: 9)
|-- Conducting High-Risk Investigations [ID: 69] (Total Chuck in branch: 7, Direct Chunk: 7)
|-- Using Evidence Containers [ID: 70] (Total Chuck in branch: 24, Direct Chunk: 24)
|-- Overseeing Facility Maintenance [ID: 71] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Considering Physical Security Needs [ID: 72] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Auditing a Digital Forensics Lab [ID: 73] (Total Chuck in branch: 8, Direct Chunk: 8)
|-- Determining Floor Plans for Digital Forensics Labs [ID: 74] (Total Chuck in branch: 7, Direct Chunk: 7)
|-- Selecting a Basic Forensic Workstation [ID: 75] (Total Chuck in branch: 51, Direct Chunk: 2)
|-- Selecting Workstations for a Lab [ID: 76] (Total Chuck in branch: 12, Direct Chunk: 12)
|-- Selecting Workstations for Private-Sector Labs [ID: 77] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Stocking Hardware Peripherals [ID: 78] (Total Chuck in branch: 14, Direct Chunk: 14)
|-- Maintaining Operating Systems and Software Inventories [ID: 79] (Total Chuck in branch: 9, Direct Chunk: 9)
|-- Using a Disaster Recovery Plan [ID: 80] (Total Chuck in branch: 7, Direct Chunk: 7)
|-- Planning for Equipment Upgrades [ID: 81] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Building a Business Case for Developing a Forensics Lab [ID: 82] (Total Chuck in branch: 104, Direct Chunk: 11)
|-- Preparing a Business Case for a Digital Forensics Lab [ID: 83] (Total Chuck in branch: 93, Direct Chunk: 2)
|-- Justification [ID: 84] (Total Chuck in branch: 8, Direct Chunk: 8)
|-- Budget Development [ID: 85] (Total Chuck in branch: 2, Direct Chunk:

2)
      |-- Facility Cost [ID: 86] (Total Chuck in branch: 15, Direct Chunk: 15)
      |-- Hardware Requirements [ID: 87] (Total Chuck in branch: 21, Direct
Chunk: 21)
      |-- Software Requirements [ID: 88] (Total Chuck in branch: 23, Direct
Chunk: 23)
      |-- Miscellaneous Budget Needs [ID: 89] (Total Chuck in branch: 4, Direct
Chunk: 4)
      |-- Approval and Acquisition [ID: 90] (Total Chuck in branch: 4, Direct
Chunk: 4)
      |-- Implementation [ID: 91] (Total Chuck in branch: 2, Direct Chunk: 2)
      |-- Acceptance Testing [ID: 92] (Total Chuck in branch: 6, Direct Chunk:
6)
      |-- Correction for Acceptance [ID: 93] (Total Chuck in branch: 2, Direct
Chunk: 2)
      |-- Production [ID: 94] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Chapter 3. Data Acquisition [ID: 101] (Total Chuck in branch: 390, Direct
Chunk: 28)
  |-- Understanding Storage Formats for Digital Evidence [ID: 103] (Total Chuck
in branch: 31, Direct Chunk: 5)
    |-- Raw Format [ID: 104] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Proprietary Formats [ID: 105] (Total Chuck in branch: 11, Direct Chunk:
11)
    |-- Advanced Forensic Format [ID: 106] (Total Chuck in branch: 11, Direct
Chunk: 11)
  |-- Determining the Best Acquisition Method [ID: 107] (Total Chuck in branch:
20, Direct Chunk: 20)
  |-- Contingency Planning for Image Acquisitions [ID: 108] (Total Chuck in
branch: 10, Direct Chunk: 10)
  |-- Using Acquisition Tools [ID: 109] (Total Chuck in branch: 173, Direct
Chunk: 5)
    |-- Mini-WinFE Boot CDs and USB Drives [ID: 110] (Total Chuck in branch: 9,
Direct Chunk: 9)
    |-- Acquiring Data with a Linux Boot CD [ID: 111] (Total Chuck in branch:
113, Direct Chunk: 5)
      |-- Using Linux Live CD Distributions [ID: 112] (Total Chuck in branch:
17, Direct Chunk: 17)
      |-- Preparing a Target Drive for Acquisition in Linux [ID: 113] (Total
Chuck in branch: 45, Direct Chunk: 45)
      |-- Acquiring Data with dd in Linux [ID: 114] (Total Chuck in branch: 32,
Direct Chunk: 32)
      |-- Acquiring Data with dcfldd in Linux [ID: 115] (Total Chuck in branch:
14, Direct Chunk: 14)
    |-- Capturing an Image with AccessData FTK Imager Lite [ID: 116] (Total
Chuck in branch: 46, Direct Chunk: 46)
  |-- Validating Data Acquisitions [ID: 117] (Total Chuck in branch: 32, Direct
Chunk: 5)
    |-- Linux Validation Methods [ID: 118] (Total Chuck in branch: 21, Direct

Chunk: 3)
      |-- Validating dd-Acquired Data [ID: 119] (Total Chuck in branch: 12,
Direct Chunk: 12)
      |-- Validating dcfldd-Acquired Data [ID: 120] (Total Chuck in branch: 6,
Direct Chunk: 6)
    |-- Windows Validation Methods [ID: 121] (Total Chuck in branch: 6, Direct
Chunk: 6)
  |-- Performing RAID Data Acquisitions [ID: 122] (Total Chuck in branch: 30,
Direct Chunk: 2)
    |-- Understanding RAID [ID: 123] (Total Chuck in branch: 15, Direct Chunk:
15)
    |-- Acquiring RAID Disks [ID: 124] (Total Chuck in branch: 13, Direct Chunk:
13)
  |-- Using Remote Network Acquisition Tools [ID: 125] (Total Chuck in branch:
39, Direct Chunk: 5)
    |-- Remote Acquisition with ProDiscover [ID: 126] (Total Chuck in branch:
20, Direct Chunk: 20)
    |-- Remote Acquisition with EnCase Enterprise [ID: 127] (Total Chuck in
branch: 7, Direct Chunk: 7)
    |-- Remote Acquisition with R-Tools R-Studio [ID: 128] (Total Chuck in
branch: 2, Direct Chunk: 2)
    |-- Remote Acquisition with WetStone US-LATT PRO [ID: 129] (Total Chuck in
branch: 2, Direct Chunk: 2)
    |-- Remote Acquisition with F-Response [ID: 130] (Total Chuck in branch: 3,
Direct Chunk: 3)
  |-- Using Other Forensics Acquisition Tools [ID: 131] (Total Chuck in branch:
27, Direct Chunk: 2)
    |-- PassMark Software ImageUSB [ID: 132] (Total Chuck in branch: 2, Direct
Chunk: 2)
    |-- ASR Data SMART [ID: 133] (Total Chuck in branch: 7, Direct Chunk: 7)
    |-- Runtime Software [ID: 134] (Total Chuck in branch: 12, Direct Chunk: 12)
    |-- ILookIX IXImager [ID: 135] (Total Chuck in branch: 2, Direct Chunk: 2)
    |-- SourceForge [ID: 136] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Chapter 4. Processing Crime and Incident Scenes [ID: 143] (Total Chuck in
branch: 413, Direct Chunk: 29)
  |-- Identifying Digital Evidence [ID: 145] (Total Chuck in branch: 76, Direct
Chunk: 13)
    |-- Understanding Rules of Evidence [ID: 146] (Total Chuck in branch: 63,
Direct Chunk: 63)
  |-- Collecting Evidence in Private-Sector Incident Scenes [ID: 147] (Total
Chuck in branch: 24, Direct Chunk: 24)
  |-- Processing Law Enforcement Crime Scenes [ID: 148] (Total Chuck in branch:
24, Direct Chunk: 6)
    |-- Understanding Concepts and Terms Used in Warrants [ID: 149] (Total Chuck
in branch: 18, Direct Chunk: 18)
  |-- Preparing for a Search [ID: 150] (Total Chuck in branch: 40, Direct Chunk:
2)
    |-- Identifying the Nature of the Case [ID: 151] (Total Chuck in branch: 3,

Direct Chunk: 3)
    |-- Identifying the Type of OS or Digital Device [ID: 152] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Determining Whether You Can Seize Computers and Digital Devices [ID: 153] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Getting a Detailed Description of the Location [ID: 154] (Total Chuck in branch: 7, Direct Chunk: 7)
    |-- Determining Who Is in Charge [ID: 155] (Total Chuck in branch: 2, Direct Chunk: 2)
    |-- Using Additional Technical Expertise [ID: 156] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Determining the Tools You Need [ID: 157] (Total Chuck in branch: 11, Direct Chunk: 11)
    |-- Preparing the Investigation Team [ID: 158] (Total Chuck in branch: 3, Direct Chunk: 3)
  |-- Securing a Digital Incident or Crime Scene [ID: 159] (Total Chuck in branch: 9, Direct Chunk: 9)
  |-- Seizing Digital Evidence at the Scene [ID: 160] (Total Chuck in branch: 72, Direct Chunk: 4)
    |-- Preparing to Acquire Digital Evidence [ID: 161] (Total Chuck in branch: 8, Direct Chunk: 8)
    |-- Processing Incident or Crime Scenes [ID: 162] (Total Chuck in branch: 34, Direct Chunk: 34)
    |-- Processing Data Centers with RAID Systems [ID: 163] (Total Chuck in branch: 2, Direct Chunk: 2)
    |-- Using a Technical Advisor [ID: 164] (Total Chuck in branch: 9, Direct Chunk: 9)
    |-- Documenting Evidence in the Lab [ID: 165] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Processing and Handling Digital Evidence [ID: 166] (Total Chuck in branch: 11, Direct Chunk: 11)
  |-- Storing Digital Evidence [ID: 167] (Total Chuck in branch: 18, Direct Chunk: 7)
    |-- Evidence Retention and Media Storage Needs [ID: 168] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Documenting Evidence [ID: 169] (Total Chuck in branch: 7, Direct Chunk: 7)
  |-- Obtaining a Digital Hash [ID: 170] (Total Chuck in branch: 42, Direct Chunk: 42)
  |-- Reviewing a Case [ID: 171] (Total Chuck in branch: 79, Direct Chunk: 8)
    |-- Sample Civil Investigation [ID: 172] (Total Chuck in branch: 23, Direct Chunk: 23)
    |-- An Example of a Criminal Investigation [ID: 173] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Reviewing Background Information for a Case [ID: 174] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Planning the Investigation [ID: 175] (Total Chuck in branch: 7, Direct Chunk: 7)

```
    |-- Conducting the Investigation: Acquiring Evidence with OSForensics [ID:
176] (Total Chuck in branch: 33, Direct Chunk: 33)
|-- Chapter 5. Working with Windows and CLI Systems [ID: 183] (Total Chuck in
branch: 471, Direct Chunk: 22)
  |-- Understanding File Systems [ID: 185] (Total Chuck in branch: 33, Direct
Chunk: 3)
    |-- Understanding the Boot Sequence [ID: 186] (Total Chuck in branch: 8,
Direct Chunk: 8)
    |-- Understanding Disk Drives [ID: 187] (Total Chuck in branch: 14, Direct
Chunk: 14)
    |-- Solid-State Storage Devices [ID: 188] (Total Chuck in branch: 8, Direct
Chunk: 8)
  |-- Exploring Microsoft File Structures [ID: 189] (Total Chuck in branch: 71,
Direct Chunk: 5)
    |-- Disk Partitions [ID: 190] (Total Chuck in branch: 39, Direct Chunk: 39)
    |-- Examining FAT Disks [ID: 191] (Total Chuck in branch: 27, Direct Chunk:
24)
      |-- Deleting FAT Files [ID: 192] (Total Chuck in branch: 3, Direct Chunk:
3)
  |-- Examining NTFS Disks [ID: 193] (Total Chuck in branch: 168, Direct Chunk:
14)
    |-- NTFS System Files [ID: 194] (Total Chuck in branch: 6, Direct Chunk: 6)
    |-- MFT and File Attributes [ID: 195] (Total Chuck in branch: 20, Direct
Chunk: 20)
    |-- MFT Structures for File Data [ID: 196] (Total Chuck in branch: 69,
Direct Chunk: 3)
      |-- MFT Header Fields [ID: 197] (Total Chuck in branch: 7, Direct Chunk:
7)
      |-- Attribute 0x10: Standard Information [ID: 198] (Total Chuck in branch:
8, Direct Chunk: 8)
      |-- Attribute 0x30: File Name [ID: 199] (Total Chuck in branch: 18, Direct
Chunk: 18)
      |-- Attribute 0x40: Object_ID [ID: 200] (Total Chuck in branch: 10, Direct
Chunk: 10)
      |-- Attribute 0x80: Data for a Resident File [ID: 201] (Total Chuck in
branch: 8, Direct Chunk: 8)
      |-- Attribute 0x80: Data for a Nonresident File [ID: 202] (Total Chuck in
branch: 6, Direct Chunk: 6)
      |-- Interpreting a Data Run [ID: 203] (Total Chuck in branch: 9, Direct
Chunk: 9)
    |-- NTFS Alternate Data Streams [ID: 204] (Total Chuck in branch: 14, Direct
Chunk: 14)
    |-- NTFS Compressed Files [ID: 205] (Total Chuck in branch: 3, Direct Chunk:
3)
    |-- NTFS Encrypting File System [ID: 206] (Total Chuck in branch: 5, Direct
Chunk: 5)
    |-- EFS Recovery Key Agent [ID: 207] (Total Chuck in branch: 8, Direct
Chunk: 8)
```

|-- Deleting NTFS Files [ID: 208] (Total Chuck in branch: 20, Direct Chunk: 20)
|-- Resilient File System [ID: 209] (Total Chuck in branch: 9, Direct Chunk: 9)
|-- Understanding Whole Disk Encryption [ID: 210] (Total Chuck in branch: 26, Direct Chunk: 11)
|-- Examining Microsoft BitLocker [ID: 211] (Total Chuck in branch: 9, Direct Chunk: 9)
|-- Examining Third-Party Disk Encryption Tools [ID: 212] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Understanding the Windows Registry [ID: 213] (Total Chuck in branch: 56, Direct Chunk: 9)
|-- Exploring the Organization of the Windows Registry [ID: 214] (Total Chuck in branch: 18, Direct Chunk: 18)
|-- Examining the Windows Registry [ID: 215] (Total Chuck in branch: 29, Direct Chunk: 29)
|-- Understanding Microsoft Startup Tasks [ID: 216] (Total Chuck in branch: 47, Direct Chunk: 3)
|-- Startup in Windows 7, Windows 8, and Windows 10 [ID: 217] (Total Chuck in branch: 5, Direct Chunk: 5)
|-- Startup in Windows NT and Later [ID: 218] (Total Chuck in branch: 39, Direct Chunk: 10)
|-- Startup Files for Windows Vista [ID: 219] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Startup Files for Windows XP [ID: 220] (Total Chuck in branch: 17, Direct Chunk: 17)
|-- Windows XP System Files [ID: 221] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Contamination Concerns with Windows XP [ID: 222] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Understanding Virtual Machines [ID: 223] (Total Chuck in branch: 48, Direct Chunk: 10)
|-- Creating a Virtual Machine [ID: 224] (Total Chuck in branch: 38, Direct Chunk: 38)
|-- Chapter 6. Current Digital Forensics Tools [ID: 231] (Total Chuck in branch: 315, Direct Chunk: 22)
|-- Evaluating Digital Forensics Tool Needs [ID: 233] (Total Chuck in branch: 184, Direct Chunk: 10)
|-- Types of Digital Forensics Tools [ID: 234] (Total Chuck in branch: 11, Direct Chunk: 4)
|-- Hardware Forensics Tools [ID: 235] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Software Forensics Tools [ID: 236] (Total Chuck in branch: 5, Direct Chunk: 5)
|-- Tasks Performed by Digital Forensics Tools [ID: 237] (Total Chuck in branch: 153, Direct Chunk: 3)
|-- Acquisition [ID: 238] (Total Chuck in branch: 22, Direct Chunk: 22)
|-- Validation and Verification [ID: 239] (Total Chuck in branch: 14,

Direct Chunk: 14)
      |-- Extraction [ID: 240] (Total Chuck in branch: 25, Direct Chunk: 25)
      |-- Reconstruction [ID: 241] (Total Chuck in branch: 66, Direct Chunk: 66)
      |-- Reporting [ID: 242] (Total Chuck in branch: 23, Direct Chunk: 23)
    |-- Tool Comparisons [ID: 243] (Total Chuck in branch: 6, Direct Chunk: 6)
    |-- Other Considerations for Tools [ID: 244] (Total Chuck in branch: 4,
Direct Chunk: 4)
  |-- Digital Forensics Software Tools [ID: 245] (Total Chuck in branch: 41,
Direct Chunk: 4)
    |-- Command-Line Forensics Tools [ID: 246] (Total Chuck in branch: 14,
Direct Chunk: 14)
    |-- Linux Forensics Tools [ID: 247] (Total Chuck in branch: 19, Direct
Chunk: 4)
      |-- Smart [ID: 248] (Total Chuck in branch: 4, Direct Chunk: 4)
      |-- Helix 3 [ID: 249] (Total Chuck in branch: 3, Direct Chunk: 3)
      |-- Kali Linux [ID: 250] (Total Chuck in branch: 2, Direct Chunk: 2)
      |-- Autopsy and Sleuth Kit [ID: 251] (Total Chuck in branch: 4, Direct
Chunk: 4)
      |-- Forcepoint Threat Protection [ID: 252] (Total Chuck in branch: 2,
Direct Chunk: 2)
    |-- Other GUI Forensics Tools [ID: 253] (Total Chuck in branch: 4, Direct
Chunk: 4)
  |-- Digital Forensics Hardware Tools [ID: 254] (Total Chuck in branch: 38,
Direct Chunk: 3)
    |-- Forensic Workstations [ID: 255] (Total Chuck in branch: 13, Direct
Chunk: 7)
      |-- Building Your Own Workstation [ID: 256] (Total Chuck in branch: 6,
Direct Chunk: 6)
    |-- Using a Write-Blocker [ID: 257] (Total Chuck in branch: 17, Direct
Chunk: 17)
    |-- Recommendations for a Forensic Workstation [ID: 258] (Total Chuck in
branch: 5, Direct Chunk: 5)
  |-- Validating and Testing Forensics Software [ID: 259] (Total Chuck in
branch: 30, Direct Chunk: 2)
    |-- Using National Institute of Standards and Technology Tools [ID: 260]
(Total Chuck in branch: 13, Direct Chunk: 13)
    |-- Using Validation Protocols [ID: 261] (Total Chuck in branch: 15, Direct
Chunk: 6)
      |-- Digital Forensics Examination Protocol [ID: 262] (Total Chuck in
branch: 5, Direct Chunk: 5)
      |-- Digital Forensics Tool Upgrade Protocol [ID: 263] (Total Chuck in
branch: 4, Direct Chunk: 4)
|-- Chapter 7. Linux and Macintosh File Systems [ID: 270] (Total Chuck in
branch: 274, Direct Chunk: 17)
  |-- Examining Linux File Structures [ID: 272] (Total Chuck in branch: 131,
Direct Chunk: 77)
    |-- File Structures in Ext4 [ID: 273] (Total Chuck in branch: 54, Direct
Chunk: 8)

|-- Inodes [ID: 274] (Total Chuck in branch: 22, Direct Chunk: 22)
|-- Hard Links and Symbolic Links [ID: 275] (Total Chuck in branch: 24, Direct Chunk: 24)
|-- Understanding Macintosh File Structures [ID: 276] (Total Chuck in branch: 58, Direct Chunk: 6)
|-- An Overview of Mac File Structures [ID: 277] (Total Chuck in branch: 23, Direct Chunk: 23)
|-- Forensics Procedures in Mac [ID: 278] (Total Chuck in branch: 29, Direct Chunk: 18)
|-- Acquisition Methods in macOS [ID: 279] (Total Chuck in branch: 11, Direct Chunk: 11)
|-- Using Linux Forensics Tools [ID: 280] (Total Chuck in branch: 68, Direct Chunk: 5)
|-- Installing Sleuth Kit and Autopsy [ID: 281] (Total Chuck in branch: 21, Direct Chunk: 21)
|-- Examining a Case with Sleuth Kit and Autopsy [ID: 282] (Total Chuck in branch: 42, Direct Chunk: 42)
|-- Chapter 8. Recovering Graphics Files [ID: 289] (Total Chuck in branch: 240, Direct Chunk: 19)
|-- Recognizing a Graphics File [ID: 291] (Total Chuck in branch: 54, Direct Chunk: 4)
|-- Understanding Bitmap and Raster Images [ID: 292] (Total Chuck in branch: 13, Direct Chunk: 13)
|-- Understanding Vector Graphics [ID: 293] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Understanding Metafile Graphics [ID: 294] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Understanding Graphics File Formats [ID: 295] (Total Chuck in branch: 14, Direct Chunk: 14)
|-- Understanding Digital Photograph File Formats [ID: 296] (Total Chuck in branch: 19, Direct Chunk: 2)
|-- Examining the Raw File Format [ID: 297] (Total Chuck in branch: 5, Direct Chunk: 5)
|-- Examining the Exchangeable Image File Format [ID: 298] (Total Chuck in branch: 12, Direct Chunk: 12)
|-- Understanding Data Compression [ID: 299] (Total Chuck in branch: 101, Direct Chunk: 2)
|-- Lossless and Lossy Compression [ID: 300] (Total Chuck in branch: 8, Direct Chunk: 8)
|-- Locating and Recovering Graphics Files [ID: 301] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Identifying Graphics File Fragments [ID: 302] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Repairing Damaged Headers [ID: 303] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Searching for and Carving Data from Unallocated Space [ID: 304] (Total Chuck in branch: 39, Direct Chunk: 9)
|-- Planning Your Examination [ID: 305] (Total Chuck in branch: 4, Direct

Chunk: 4)
      |-- Searching for and Recovering Digital Photograph Evidence [ID: 306]
(Total Chuck in branch: 26, Direct Chunk: 26)
    |-- Rebuilding File Headers [ID: 307] (Total Chuck in branch: 22, Direct
Chunk: 22)
    |-- Reconstructing File Fragments [ID: 308] (Total Chuck in branch: 15,
Direct Chunk: 15)
  |-- Identifying Unknown File Formats [ID: 309] (Total Chuck in branch: 47,
Direct Chunk: 14)
    |-- Analyzing Graphics File Headers [ID: 310] (Total Chuck in branch: 5,
Direct Chunk: 5)
    |-- Tools for Viewing Images [ID: 311] (Total Chuck in branch: 5, Direct
Chunk: 5)
    |-- Understanding Steganography in Graphics Files [ID: 312] (Total Chuck in
branch: 16, Direct Chunk: 16)
    |-- Using Steganalysis Tools [ID: 313] (Total Chuck in branch: 7, Direct
Chunk: 7)
  |-- Understanding Copyright Issues with Graphics [ID: 314] (Total Chuck in
branch: 19, Direct Chunk: 19)
|-- Chapter 9. Digital Forensics Analysis and Validation [ID: 321] (Total Chuck
in branch: 248, Direct Chunk: 16)
  |-- Determining What Data to Collect and Analyze [ID: 323] (Total Chuck in
branch: 99, Direct Chunk: 6)
    |-- Approaching Digital Forensics Cases [ID: 324] (Total Chuck in branch:
35, Direct Chunk: 30)
      |-- Refining and Modifying the Investigation Plan [ID: 325] (Total Chuck
in branch: 5, Direct Chunk: 5)
    |-- Using Autopsy to Validate Data [ID: 326] (Total Chuck in branch: 23,
Direct Chunk: 8)
      |-- Installing NSRL Hashes in Autopsy [ID: 327] (Total Chuck in branch:
15, Direct Chunk: 15)
    |-- Collecting Hash Values in Autopsy [ID: 328] (Total Chuck in branch: 35,
Direct Chunk: 35)
  |-- Validating Forensic Data [ID: 329] (Total Chuck in branch: 51, Direct
Chunk: 3)
    |-- Validating with Hexadecimal Editors [ID: 330] (Total Chuck in branch:
31, Direct Chunk: 28)
      |-- Using Hash Values to Discriminate Data [ID: 331] (Total Chuck in
branch: 3, Direct Chunk: 3)
    |-- Validating with Digital Forensics Tools [ID: 332] (Total Chuck in
branch: 17, Direct Chunk: 17)
  |-- Addressing Data-Hiding Techniques [ID: 333] (Total Chuck in branch: 82,
Direct Chunk: 2)
    |-- Hiding Files by Using the OS [ID: 334] (Total Chuck in branch: 3, Direct
Chunk: 3)
    |-- Hiding Partitions [ID: 335] (Total Chuck in branch: 8, Direct Chunk: 8)
    |-- Marking Bad Clusters [ID: 336] (Total Chuck in branch: 6, Direct Chunk:
6)

|-- Bit-Shifting [ID: 337] (Total Chuck in branch: 34, Direct Chunk: 34)
|-- Understanding Steganalysis Methods [ID: 338] (Total Chuck in branch: 10, Direct Chunk: 10)
|-- Examining Encrypted Files [ID: 339] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Recovering Passwords [ID: 340] (Total Chuck in branch: 15, Direct Chunk: 15)
|-- Chapter 10. Virtual Machine Forensics, Live Acquisitions, and Network Forensics [ID: 347] (Total Chuck in branch: 287, Direct Chunk: 17)
|-- An Overview of Virtual Machine Forensics [ID: 349] (Total Chuck in branch: 176, Direct Chunk: 7)
|-- Type 2 Hypervisors [ID: 350] (Total Chuck in branch: 32, Direct Chunk: 6)
|-- Parallels Desktop [ID: 351] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- KVM [ID: 352] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Microsoft Hyper-V [ID: 353] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- VMware Workstation and Workstation Player [ID: 354] (Total Chuck in branch: 14, Direct Chunk: 14)
|-- VirtualBox [ID: 355] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Conducting an Investigation with Type 2 Hypervisors [ID: 356] (Total Chuck in branch: 103, Direct Chunk: 70)
|-- Other VM Examination Methods [ID: 357] (Total Chuck in branch: 13, Direct Chunk: 13)
|-- Using VMs as Forensics Tools [ID: 358] (Total Chuck in branch: 20, Direct Chunk: 20)
|-- Working with Type 1 Hypervisors [ID: 359] (Total Chuck in branch: 34, Direct Chunk: 34)
|-- Performing Live Acquisitions [ID: 360] (Total Chuck in branch: 18, Direct Chunk: 15)
|-- Performing a Live Acquisition in Windows [ID: 361] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Network Forensics Overview [ID: 362] (Total Chuck in branch: 76, Direct Chunk: 4)
|-- The Need for Established Procedures [ID: 363] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Securing a Network [ID: 364] (Total Chuck in branch: 13, Direct Chunk: 13)
|-- Developing Procedures for Network Forensics [ID: 365] (Total Chuck in branch: 41, Direct Chunk: 8)
|-- Reviewing Network Logs [ID: 366] (Total Chuck in branch: 11, Direct Chunk: 11)
|-- Using Network Tools [ID: 367] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Using Packet Analyzers [ID: 368] (Total Chuck in branch: 18, Direct Chunk: 18)
|-- Investigating Virtual Networks [ID: 369] (Total Chuck in branch: 7,

Direct Chunk: 7)
    |-- Examining the Honeynet Project [ID: 370] (Total Chuck in branch: 7, Direct Chunk: 7)
|-- Chapter 11. E-mail and Social Media Investigations [ID: 377] (Total Chuck in branch: 302, Direct Chunk: 20)
  |-- Exploring the Role of E-mail in Investigations [ID: 379] (Total Chuck in branch: 11, Direct Chunk: 11)
  |-- Exploring the Roles of the Client and Server in E-mail [ID: 380] (Total Chuck in branch: 10, Direct Chunk: 10)
  |-- Investigating E-mail Crimes and Violations [ID: 381] (Total Chuck in branch: 101, Direct Chunk: 4)
    |-- Understanding Forensic Linguistics [ID: 382] (Total Chuck in branch: 6, Direct Chunk: 6)
    |-- Examining E-mail Messages [ID: 383] (Total Chuck in branch: 28, Direct Chunk: 8)
      |-- Copying an E-mail Message [ID: 384] (Total Chuck in branch: 20, Direct Chunk: 20)
    |-- Viewing E-mail Headers [ID: 385] (Total Chuck in branch: 33, Direct Chunk: 33)
    |-- Examining E-mail Headers [ID: 386] (Total Chuck in branch: 14, Direct Chunk: 14)
    |-- Examining Additional E-mail Files [ID: 387] (Total Chuck in branch: 6, Direct Chunk: 6)
    |-- Tracing an E-mail Message [ID: 388] (Total Chuck in branch: 7, Direct Chunk: 7)
    |-- Using Network E-mail Logs [ID: 389] (Total Chuck in branch: 3, Direct Chunk: 3)
  |-- Understanding E-mail Servers [ID: 390] (Total Chuck in branch: 33, Direct Chunk: 13)
    |-- Examining UNIX E-mail Server Logs [ID: 391] (Total Chuck in branch: 12, Direct Chunk: 12)
    |-- Examining Microsoft E-mail Server Logs [ID: 392] (Total Chuck in branch: 8, Direct Chunk: 8)
  |-- Using Specialized E-mail Forensics Tools [ID: 393] (Total Chuck in branch: 91, Direct Chunk: 22)
    |-- Using Magnet AXIOM to Recover E-mail [ID: 394] (Total Chuck in branch: 14, Direct Chunk: 14)
    |-- Using a Hex Editor to Carve E-mail Messages [ID: 395] (Total Chuck in branch: 41, Direct Chunk: 41)
    |-- Recovering Outlook Files [ID: 396] (Total Chuck in branch: 7, Direct Chunk: 7)
    |-- E-mail Case Studies [ID: 397] (Total Chuck in branch: 7, Direct Chunk: 7)
  |-- Applying Digital Forensics Methods to Social Media Communications [ID: 398] (Total Chuck in branch: 36, Direct Chunk: 23)
    |-- Forensics Tools for Social Media Investigations [ID: 399] (Total Chuck in branch: 13, Direct Chunk: 13)
|-- Chapter 12. Mobile Device Forensics and the Internet of Anything [ID: 406]

(Total Chuck in branch: 169, Direct Chunk: 8)
  |-- Understanding Mobile Device Forensics [ID: 408] (Total Chuck in branch: 65, Direct Chunk: 19)
    |-- Mobile Phone Basics [ID: 409] (Total Chuck in branch: 24, Direct Chunk: 24)
    |-- Inside Mobile Devices [ID: 410] (Total Chuck in branch: 22, Direct Chunk: 11)
      |-- SIM Cards [ID: 411] (Total Chuck in branch: 11, Direct Chunk: 11)
  |-- Understanding Acquisition Procedures for Mobile Devices [ID: 412] (Total Chuck in branch: 75, Direct Chunk: 30)
    |-- Mobile Forensics Equipment [ID: 413] (Total Chuck in branch: 30, Direct Chunk: 6)
      |-- SIM Card Readers [ID: 414] (Total Chuck in branch: 7, Direct Chunk: 7)
      |-- Mobile Phone Forensics Tools and Methods [ID: 415] (Total Chuck in branch: 17, Direct Chunk: 17)
    |-- Using Mobile Forensics Tools [ID: 416] (Total Chuck in branch: 15, Direct Chunk: 15)
  |-- Understanding Forensics in the Internet of Anything [ID: 417] (Total Chuck in branch: 21, Direct Chunk: 21)
|-- Chapter 13. Cloud Forensics [ID: 424] (Total Chuck in branch: 290, Direct Chunk: 20)
  |-- An Overview of Cloud Computing [ID: 426] (Total Chuck in branch: 42, Direct Chunk: 2)
    |-- History of the Cloud [ID: 427] (Total Chuck in branch: 4, Direct Chunk: 4)
    |-- Cloud Service Levels and Deployment Methods [ID: 428] (Total Chuck in branch: 13, Direct Chunk: 13)
    |-- Cloud Vendors [ID: 429] (Total Chuck in branch: 15, Direct Chunk: 15)
    |-- Basic Concepts of Cloud Forensics [ID: 430] (Total Chuck in branch: 8, Direct Chunk: 8)
  |-- Legal Challenges in Cloud Forensics [ID: 431] (Total Chuck in branch: 56, Direct Chunk: 2)
    |-- Service Level Agreements [ID: 432] (Total Chuck in branch: 25, Direct Chunk: 17)
      |-- Policies, Standards, and Guidelines for CSPs [ID: 433] (Total Chuck in branch: 5, Direct Chunk: 5)
      |-- CSP Processes and Procedures [ID: 434] (Total Chuck in branch: 3, Direct Chunk: 3)
    |-- Jurisdiction Issues [ID: 435] (Total Chuck in branch: 6, Direct Chunk: 6)
    |-- Accessing Evidence in the Cloud [ID: 436] (Total Chuck in branch: 23, Direct Chunk: 3)
      |-- Search Warrants [ID: 437] (Total Chuck in branch: 10, Direct Chunk: 10)
      |-- Subpoenas and Court Orders [ID: 438] (Total Chuck in branch: 10, Direct Chunk: 10)
  |-- Technical Challenges in Cloud Forensics [ID: 439] (Total Chuck in branch: 33, Direct Chunk: 5)

|-- Considering Writing Style [ID: 476] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Including Signposts [ID: 477] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Designing the Layout and Presentation of Reports [ID: 478] (Total Chuck in branch: 85, Direct Chunk: 44)
|-- Providing Supporting Material [ID: 479] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Formatting Consistently [ID: 480] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Explaining Examination and Data Collection Methods [ID: 481] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Including Calculations [ID: 482] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Providing for Uncertainty and Error Analysis [ID: 483] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Explaining Results and Conclusions [ID: 484] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Providing References [ID: 485] (Total Chuck in branch: 20, Direct Chunk: 20)
|-- Including Appendixes [ID: 486] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Generating Report Findings with Forensics Software Tools [ID: 487] (Total Chuck in branch: 78, Direct Chunk: 2)
|-- Using Autopsy to Generate Reports [ID: 488] (Total Chuck in branch: 76, Direct Chunk: 76)
|-- Chapter 15. Expert Testimony in Digital Investigations [ID: 495] (Total Chuck in branch: 329, Direct Chunk: 14)
|-- Preparing for Testimony [ID: 497] (Total Chuck in branch: 62, Direct Chunk: 15)
|-- Documenting and Preparing Evidence [ID: 498] (Total Chuck in branch: 12, Direct Chunk: 12)
|-- Reviewing Your Role as a Consulting Expert or an Expert Witness [ID: 499] (Total Chuck in branch: 9, Direct Chunk: 9)
|-- Creating and Maintaining Your CV [ID: 500] (Total Chuck in branch: 8, Direct Chunk: 8)
|-- Preparing Technical Definitions [ID: 501] (Total Chuck in branch: 12, Direct Chunk: 12)
|-- Preparing to Deal with the News Media [ID: 502] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Testifying in Court [ID: 503] (Total Chuck in branch: 155, Direct Chunk: 2)
|-- Understanding the Trial Process [ID: 504] (Total Chuck in branch: 10, Direct Chunk: 10)
|-- Providing Qualifications for Your Testimony [ID: 505] (Total Chuck in branch: 59, Direct Chunk: 59)
|-- General Guidelines on Testifying [ID: 506] (Total Chuck in branch: 47, Direct Chunk: 27)

|-- Using Graphics During Testimony [ID: 507] (Total Chuck in branch: 10, Direct Chunk: 10)
|-- Avoiding Testimony Problems [ID: 508] (Total Chuck in branch: 7, Direct Chunk: 7)
|-- Understanding Prosecutorial Misconduct [ID: 509] (Total Chuck in branch: 3, Direct Chunk: 3)
|-- Testifying During Direct Examination [ID: 510] (Total Chuck in branch: 12, Direct Chunk: 12)
|-- Testifying During Cross-Examination [ID: 511] (Total Chuck in branch: 25, Direct Chunk: 25)
|-- Preparing for a Deposition or Hearing [ID: 512] (Total Chuck in branch: 33, Direct Chunk: 6)
|-- Guidelines for Testifying at Depositions [ID: 513] (Total Chuck in branch: 19, Direct Chunk: 10)
|-- Recognizing Deposition Problems [ID: 514] (Total Chuck in branch: 9, Direct Chunk: 9)
|-- Guidelines for Testifying at Hearings [ID: 515] (Total Chuck in branch: 8, Direct Chunk: 8)
|-- Preparing Forensics Evidence for Testimony [ID: 516] (Total Chuck in branch: 65, Direct Chunk: 34)
|-- Preparing a Defense of Your Evidence-Collection Methods [ID: 517] (Total Chuck in branch: 31, Direct Chunk: 31)
|-- Chapter 16. Ethics for the Expert Witness [ID: 524] (Total Chuck in branch: 310, Direct Chunk: 13)
|-- Applying Ethics and Codes to Expert Witnesses [ID: 526] (Total Chuck in branch: 58, Direct Chunk: 11)
|-- Forensics Examiners' Roles in Testifying [ID: 527] (Total Chuck in branch: 5, Direct Chunk: 5)
|-- Considerations in Disqualification [ID: 528] (Total Chuck in branch: 22, Direct Chunk: 22)
|-- Traps for Unwary Experts [ID: 529] (Total Chuck in branch: 16, Direct Chunk: 16)
|-- Determining Admissibility of Evidence [ID: 530] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- Organizations with Codes of Ethics [ID: 531] (Total Chuck in branch: 26, Direct Chunk: 2)
|-- International Society of Forensic Computer Examiners [ID: 532] (Total Chuck in branch: 11, Direct Chunk: 11)
|-- International High Technology Crime Investigation Association [ID: 533] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- American Bar Association [ID: 535] (Total Chuck in branch: 5, Direct Chunk: 5)
|-- American Psychological Association [ID: 536] (Total Chuck in branch: 2, Direct Chunk: 2)
|-- Ethical Difficulties in Expert Testimony [ID: 537] (Total Chuck in branch: 19, Direct Chunk: 6)
|-- Ethical Responsibilities Owed to You [ID: 538] (Total Chuck in branch: 9, Direct Chunk: 9)

|-- Standard Forensics Tools and Tools You Create [ID: 539] (Total Chuck in branch: 4, Direct Chunk: 4)
|-- An Ethics Exercise [ID: 540] (Total Chuck in branch: 194, Direct Chunk: 4)
|-- Performing a Cursory Exam of a Forensic Image [ID: 541] (Total Chuck in branch: 18, Direct Chunk: 18)
|-- Performing a Detailed Exam of a Forensic Image [ID: 542] (Total Chuck in branch: 33, Direct Chunk: 33)
|-- Performing the Exam [ID: 543] (Total Chuck in branch: 76, Direct Chunk: 2)
|-- Preparing for an Examination [ID: 544] (Total Chuck in branch: 74, Direct Chunk: 74)
|-- Interpreting Attribute 0x80 Data Runs [ID: 545] (Total Chuck in branch: 44, Direct Chunk: 2)
|-- Finding Attribute 0x80 an MFT Record [ID: 546] (Total Chuck in branch: 21, Direct Chunk: 21)
|-- Configuring Data Interpreter Options in WinHex [ID: 547] (Total Chuck in branch: 6, Direct Chunk: 6)
|-- Calculating Data Runs [ID: 548] (Total Chuck in branch: 15, Direct Chunk: 15)
|-- Carving Data Run Clusters Manually [ID: 549] (Total Chuck in branch: 19, Direct Chunk: 19)
|-- Lab Manual for Guide to Computer Forensics and Investigations [ID: 556] (Total Chuck in branch: 2165, Direct Chunk: 1)
|-- Chapter 12. Mobile Device Forensics [ID: 792] (Total Chuck in branch: 13, Direct Chunk: 10)
|-- Lab 12.1. Examining Cell Phone Storage Devices [ID: 794] (Total Chuck in branch: 1, Direct Chunk: 1)
|-- Lab 12.2. Using FTK Imager Lite to View Text Messages, Phone Numbers, and Photos [ID: 799] (Total Chuck in branch: 1, Direct Chunk: 1)
|-- Lab 12.3. Using Autopsy to Search Cloud Backups of Mobile Devices [ID: 804] (Total Chuck in branch: 1, Direct Chunk: 1)
|-- Chapter 1. Understanding the Digital Forensics Profession and Investigations [ID: inf] (Total Chuck in branch: 1673, Direct Chunk: 0)
|-- Lab 1.1. Installing Autopsy for Windows [ID: 560] (Total Chuck in branch: 1670, Direct Chunk: 1)
|-- Objectives [ID: 561] (Total Chuck in branch: 702, Direct Chunk: 345)
|-- Materials Required [ID: 562] (Total Chuck in branch: 357, Direct Chunk: 357)
|-- Activity [ID: 563] (Total Chuck in branch: 967, Direct Chunk: 967)
|-- Lab 1.2. Downloading FTK Imager Lite [ID: 565] (Total Chuck in branch: 1, Direct Chunk: 1)
|-- Lab 1.3. Downloading WinHex [ID: 570] (Total Chuck in branch: 1, Direct Chunk: 1)
|-- Lab 1.4. Using Autopsy for Windows [ID: 575] (Total Chuck in branch: 1, Direct Chunk: 1)
|-- Chapter 2. The Investigator's Office and Laboratory [ID: inf] (Total Chuck in branch: 5, Direct Chunk: 0)
|-- Lab 2.1. Wiping a USB Drive Securely [ID: 582] (Total Chuck in branch:

1, Direct Chunk: 1)
    |-- Lab 2.2. Using Directory Snoop to Image a USB Drive [ID: 587] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 2.3. Converting a Raw Image to an .E01 Image [ID: 592] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 2.4. Imaging Evidence with FTK Imager Lite [ID: 597] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 2.5. Viewing Images in FTK Imager Lite [ID: 602] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 3. Data Acquisition [ID: inf] (Total Chuck in branch: 70, Direct Chunk: 0)
    |-- Lab 3.1. Creating a DEFT Zero Forensic Boot CD and USB Drive [ID: 609] (Total Chuck in branch: 67, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 66, Direct Chunk: 0)
        |-- Creating a DEFT Zero Boot CD [ID: 613] (Total Chuck in branch: 14, Direct Chunk: 14)
        |-- Creating a Bootable USB DEFT Zero Drive [ID: 614] (Total Chuck in branch: 14, Direct Chunk: 14)
        |-- Learning DEFT Zero Features [ID: 615] (Total Chuck in branch: 38, Direct Chunk: 38)
    |-- Lab 3.2. Examining a FAT Image [ID: 617] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 3.3. Examining an NTFS Image [ID: 622] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 3.4. Examining an HFS+ Image [ID: 627] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 4. Processing Crime and Incident Scenes [ID: inf] (Total Chuck in branch: 36, Direct Chunk: 0)
    |-- Lab 4.1. Creating a Mini-WinFE Boot CD [ID: 634] (Total Chuck in branch: 33, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 32, Direct Chunk: 0)
        |-- Setting Up Mini-WinFE [ID: 638] (Total Chuck in branch: 8, Direct Chunk: 8)
        |-- Creating a Mini-WinFE ISO Image [ID: 639] (Total Chuck in branch: 24, Direct Chunk: 24)
    |-- Lab 4.2. Using Mini-WinFE to Boot and Image a Windows Computer [ID: 641] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 4.3. Testing the Mini-WinFE Write-Protection Feature [ID: 646] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 4.4. Creating an Image with Guymager [ID: 651] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 5. Working with Windows and CLI Systems [ID: inf] (Total Chuck in branch: 4, Direct Chunk: 0)
    |-- Lab 5.1. Using DART to Export Windows Registry Files [ID: 658] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 5.2. Examining the SAM Hive [ID: 663] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 5.3. Examining the SYSTEM Hive [ID: 668] (Total Chuck in branch: 1,

Direct Chunk: 1)
    |-- Lab 5.4. Examining the ntuser.dat Registry File [ID: 673] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 6. Current Digital Forensics Tools [ID: inf] (Total Chuck in branch: 79, Direct Chunk: 0)
    |-- Lab 6.1. Using Autopsy 4.7.0 to Search an Image File [ID: 680] (Total Chuck in branch: 41, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 40, Direct Chunk: 0)
        |-- Installing Autopsy 4.7.0 [ID: 684] (Total Chuck in branch: 12, Direct Chunk: 12)
        |-- Searching E-mail in Autopsy 4.7.0 [ID: 685] (Total Chuck in branch: 28, Direct Chunk: 28)
    |-- Lab 6.2. Using OSForensics to Search an Image of a Hard Drive [ID: 687] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 6.3. Examining a Corrupt Image File with FTK Imager Lite, Autopsy, and WinHex [ID: 692] (Total Chuck in branch: 37, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 36, Direct Chunk: 0)
        |-- Testing an Image File in Autopsy 4.3.0 [ID: 696] (Total Chuck in branch: 16, Direct Chunk: 16)
        |-- Examining Image Files in WinHex [ID: 697] (Total Chuck in branch: 20, Direct Chunk: 20)
  |-- Chapter 7. Linux and Macintosh File Systems [ID: inf] (Total Chuck in branch: 3, Direct Chunk: 0)
    |-- Lab 7.1. Using Autopsy to Process a Mac OS X Image [ID: 701] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 7.2. Using Autopsy to Process a Mac OS 9 Image [ID: 706] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 7.3. Using Autopsy to Process a Linux Image [ID: 711] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 8. Recovering Graphics Files [ID: inf] (Total Chuck in branch: 3, Direct Chunk: 0)
    |-- Lab 8.1. Using Autopsy to Analyze Multimedia Files [ID: 718] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 8.2. Using OSForensics to Analyze Multimedia Files [ID: 723] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 8.3. Using WinHex to Analyze Multimedia Files [ID: 728] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 9. Digital Forensics Analysis and Validation [ID: inf] (Total Chuck in branch: 9, Direct Chunk: 0)
    |-- Lab 9.1. Using Autopsy to Search for Keywords in an Image [ID: 735] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 9.2. Validating File Hash Values with FTK Imager Lite [ID: 740] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 9.3. Validating File Hash Values with WinHex [ID: 745] (Total Chuck in branch: 7, Direct Chunk: 1)
      |-- Objectives [ID: inf] (Total Chuck in branch: 6, Direct Chunk: 0)
        |-- Materials Required: [ID: 747] (Total Chuck in branch: 6, Direct Chunk: 6)

|-- Chapter 10. Virtual Machine Forensics, Live Acquisitions, and Network Forensics [ID: inf] (Total Chuck in branch: 143, Direct Chunk: 0)
    |-- Lab 10.1. Analyzing a Forensic Image Hosting a Virtual Machine [ID: 752] (Total Chuck in branch: 41, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 40, Direct Chunk: 0)
        |-- Installing MD5 Hashes in Autopsy [ID: 756] (Total Chuck in branch: 8, Direct Chunk: 8)
        |-- Analyzing a Windows Image Containing a Virtual Machine [ID: 757] (Total Chuck in branch: 32, Direct Chunk: 32)
    |-- Lab 10.2. Conducting a Live Acquisition [ID: 759] (Total Chuck in branch: 45, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 44, Direct Chunk: 0)
        |-- Installing Tools for Live Acquisitions [ID: 763] (Total Chuck in branch: 16, Direct Chunk: 16)
        |-- Exploring Tools for Live Acquisitions [ID: 764] (Total Chuck in branch: 14, Direct Chunk: 14)
        |-- Capturing Data in a Live Acquisition [ID: 765] (Total Chuck in branch: 14, Direct Chunk: 14)
    |-- Lab 10.3. Using Kali Linux for Network Forensics [ID: 767] (Total Chuck in branch: 57, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 56, Direct Chunk: 0)
        |-- Installing Kali Linux [ID: 771] (Total Chuck in branch: 26, Direct Chunk: 26)
        |-- Mounting Drives in Kali Linux [ID: 772] (Total Chuck in branch: 12, Direct Chunk: 12)
        |-- Identifying Open Ports and Making a Screen Capture [ID: 773] (Total Chuck in branch: 18, Direct Chunk: 18)
  |-- Chapter 11. E-mail and Social Media Investigations [ID: inf] (Total Chuck in branch: 3, Direct Chunk: 0)
    |-- Lab 11.1. Using OSForensics to Search for E-mails and Mailboxes [ID: 777] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 11.2. Using Autopsy to Search for E-mails and Mailboxes [ID: 782] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 11.3. Finding Google Searches and Multiple E-mail Accounts [ID: 787] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 13. Cloud Forensics [ID: inf] (Total Chuck in branch: 3, Direct Chunk: 0)
    |-- Lab 13.1. Examining Dropbox Cloud Storage [ID: 811] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 13.2. Examining Google Drive Cloud Storage [ID: 816] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 13.3. Examining OneDrive Cloud Storage [ID: 821] (Total Chuck in branch: 1, Direct Chunk: 1)
  |-- Chapter 14. Report Writing for High-Tech Investigations [ID: inf] (Total Chuck in branch: 3, Direct Chunk: 0)
    |-- Lab 14.1. Investigating Corporate Espionage [ID: 828] (Total Chuck in branch: 1, Direct Chunk: 1)
    |-- Lab 14.2. Adding Evidence to a Case [ID: 833] (Total Chuck in branch: 1,

```
Direct Chunk: 1)
    |-- Lab 14.3. Preparing a Report [ID: 838] (Total Chuck in branch: 1, Direct
Chunk: 1)
  |-- Chapter 15. Expert Testimony in Digital Investigations [ID: inf] (Total
Chuck in branch: 44, Direct Chunk: 0)
    |-- Lab 15.1. Conducting a Preliminary Investigation [ID: 845] (Total Chuck
in branch: 1, Direct Chunk: 1)
    |-- Lab 15.2. Investigating an Arsonist [ID: 850] (Total Chuck in branch: 1,
Direct Chunk: 1)
    |-- Lab 15.3. Recovering a Password from Password-Protected Files [ID: 855]
(Total Chuck in branch: 42, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 41, Direct Chunk: 0)
        |-- Verifying the Existence of a Warning Banner [ID: 859] (Total Chuck
in branch: 12, Direct Chunk: 12)
        |-- Recovering a Password from Password-Protected Files [ID: 860] (Total
Chuck in branch: 29, Direct Chunk: 29)
  |-- Chapter 16. Ethics for the Expert Witness [ID: inf] (Total Chuck in
branch: 73, Direct Chunk: 0)
    |-- Lab 16.1. Rebuilding an MFT Record from a Corrupt Image [ID: 864] (Total
Chuck in branch: 73, Direct Chunk: 1)
      |-- Activity [ID: inf] (Total Chuck in branch: 72, Direct Chunk: 0)
        |-- Creating a Duplicate Forensic Image [ID: 868] (Total Chuck in
branch: 14, Direct Chunk: 14)
        |-- Determining the Offset Byte Address of the Corrupt MFT Record [ID:
869] (Total Chuck in branch: 12, Direct Chunk: 12)
        |-- Copying the Corrected MFT Record [ID: 870] (Total Chuck in branch:
20, Direct Chunk: 20)
        |-- Extracting Additional Evidence [ID: 871] (Total Chuck in branch: 26,
Direct Chunk: 26)
|-- Appendix A. Certification Test References [ID: 873] (Total Chuck in branch:
56, Direct Chunk: 56)
|-- Appendix B. Digital Forensics References [ID: 874] (Total Chuck in branch:
109, Direct Chunk: 109)
|-- Appendix C. Digital Forensics Lab Considerations [ID: 875] (Total Chuck in
branch: 58, Direct Chunk: 58)
|-- Appendix D. Legacy File System and Forensics Tools [ID: 876] (Total Chuck in
branch: 59, Direct Chunk: 59)
|-- EPUB Preamble [ID: inf] (Total Chuck in branch: 21, Direct Chunk: 21)

--------------------------------------------------------------------------------
                    Chunk Sequence & Content Integrity Test
--------------------------------------------------------------------------------

------------------------- CONTENT PREVIEW -------------------------
Title: Collecting Evidence in Private-Sector Incident Scenes [toc_id: 147]
Chunk IDs: [1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947,
1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960]
-------------------------------------------------------------------
```

Collecting Evidence in Private-Sector Incident Scenes

Private-sector organizations include small to medium businesses, large corporations, and non-government organizations (NGOs), which might get funding from the government or other agencies. In the United States, NGOs and similar agencies must comply with state public disclosure and federal Freedom of Information Act (FOIA) laws and make certain documents available as public records. State public disclosure laws define state public records as open and available for inspection. For example, divorces recorded in a public office, such as a courthouse, become matters of public record unless a judge orders the documents sealed. Anyone can request a copy of a public divorce decree. Figure 4-3 shows an excerpt of a public disclosure law for the state of Idaho.

State public disclosure laws apply to state records, but the FOIA allows citizens to request copies of public documents created by federal agencies. The FOIA was originally enacted in the 1960s, and several subsequent amendments have broadened its laws. Some Web sites now provide copies of publicly accessible records for a fee.

ISPs and other communication companies make up a special category of private-sector businesses. ISPs can investigate computer abuse committed by their employees but not by customers. They must preserve customer privacy, especially when dealing with e-mail. However, federal regulations related to the Homeland Security Act and the PATRIOT Act of 2001 have redefined how ISPs and large organizations operate and maintain their records. ISPs and other communication companies can be called on to investigate customers' activities that are deemed to create an emergency situation. An emergency situation under the PATRIOT Act is defined as the immediate risk of death or personal injury, such as finding a bomb threat in an e-mail.

As recent events have shown, the government monitors e-mails for the occurrence of keywords. Incidents such as the Edward Snowden case have made public the amount of electronic surveillance done by the U.S. government and the governments of other countries. Some provisions of these federal regulations have been revised over the past few years, so you should stay abreast of their implications. For example, in March 2017, the U.S. Congress voted to allow ISPs to sell customers' browsing histories without their explicit permission. (See www.congress.gov/bill/115th-congress/senate-joint-resolution/34 for more details.)

Investigating and controlling computer incident scenes in private-sector environments is much easier than in crime scenes. In the private sector, the incident scene is often a workplace, such as a contained office or manufacturing area, where a policy violation is being investigated. Everything from the computers used to violate a company policy to the surrounding facility is under a controlled authority-that is, company management. Typically, businesses have inventory databases of computer hardware and software. Having access to these databases and knowing what applications are on suspected computers help identify the forensics tools needed to analyze a policy violation and the best way to conduct the analysis. For example, companies might have a preferred Web browser, such as Microsoft

conduct the analysis. For example, companies might have a preferred Web browser, such as Microsoft Internet Explorer, Microsoft Edge, Mozilla Firefox, or Google

Chrome. Knowing which browser a suspect used helps you develop standard examination procedures to identify data downloaded to the suspect's workstation.
To investigate employees suspected of improper use of company digital assets, a company policy statement about misuse of digital assets allows private-sector investigators to conduct covert surveillance with little or no cause and access company computer systems and digital devices without a warrant, which is an advantage. Law enforcement investigators can't do the same, however, without sufficient reason for a warrant.
However, if a company doesn't display a warning banner or publish a policy stating that it reserves the right to inspect digital assets at will, employees have an expectation of privacy (as explained in Chapter 1). When an employee is being investigated, this expected privacy prevents the employer from legally conducting an intrusive investigation. A well-defined company policy, therefore, should state that an employer has the right to examine, inspect, or access any company-owned digital assets. If a company issues a policy statement to all employees, the employer can investigate digital assets at will without any privacy right restrictions; this practice might violate the privacy laws of countries in the EU, for example. As a standard practice, companies should use both warning banners
countries in the EU, for example. As a standard practice, companies should use both warning banners and policy statements. For example, if an incident is escalated to a criminal complaint, prosecutors prefer showing juries warning banners instead of policy manuals. A warning banner leaves a much stronger impression on a jury.
In addition to making sure a company has a policy statement or a warning banner, private-sector investigators should know under what circumstances they can examine an employee's computer. With a policy statement, an employer can freely initiate any inquiry necessary to protect the company or organization. However, organizations must also have a well-defined process describing when an investigation can be initiated. At a minimum, most company policies require that employers have a "reasonable suspicion" that a law or policy is being violated. For example, if a policy states that employees can't use company computers for outside business and a supervisor notices a change in work behavior that could indicate an employee is violating this rule, generally it's enough to warrant an that could indicate an employee is violating this rule, generally it's enough to warrant an investigation. However, some countries require notifying employees that they're being investigated if they're suspected of criminal behavior at work.
If a private-sector investigator finds that an employee is committing or has committed a crime, the employer can file a criminal complaint with the police. Some businesses, such as banks, have a regulatory requirement to report crimes. In the United States, the employer must turn over all evidence to the police for prosecution. If this evidence had been collected by a law enforcement officer, it would require a warrant, which would be difficult to get without sufficient probable cause. In "Processing Law Enforcement Crime Scenes" later in this chapter, you learn more about probable cause and how it applies to a criminal investigation.
Employers are usually interested in enforcing company policy, not seeking out

and prosecuting employees, so typically they approve digital investigations only to identify employees who are misusing company assets. Private-sector investigators are, therefore, concerned mainly with protecting company assets, such as intellectual property. Finding evidence of a criminal act during an investigation escalates the investigation from an internal civil matter to an external criminal complaint. In some situations, such as the discovery of child pornography, the company or its agents must notify law enforcement immediately. If you discover evidence of a crime during a company policy investigation, first determine whether the incident meets the elements of criminal law. You might have to consult with your organization's attorney to determine whether the situation is a potential crime. Next, inform management of the incident; they might have other concerns, such as protecting confidential business data that could be included with the criminal evidence (called "commingled data"). In this case, coordinate with management and the organization's attorney to determine the best way to protect commingled data. After you submit evidence containing sensitive information to the police, it becomes public record. Public record laws do include exceptions for protecting sensitive company information; ultimately, however, a

laws do include exceptions for protecting sensitive company information; ultimately, however, a judge decides what to protect.

After you discover illegal activity and document and report the crime, stop your investigation to make sure you don't violate Fourth Amendment restrictions on obtaining evidence. If the information you supply is specific enough to meet the criteria for a search warrant, the police are responsible for obtaining a warrant that requests any new evidence. If you follow police instructions to gather additional evidence without a search warrant after you have reported the crime, you run the risk of becoming an agent of law enforcement. Instead, consult with your organization's attorney on how to respond to a police request for information. The police and prosecutor should issue a subpoena for any additional new evidence, which minimizes your exposure to potential civil liability. In addition,

additional new evidence, which minimizes your exposure to potential civil liability. In addition, you should keep all documentation of evidence collected to investigate an internal company policy violation. Later in this section, you learn more about using affidavits in an internal investigation.

One example of a company policy violation involves employees observing another employee accessing pornographic Web sites. If your organization's policy requires you to determine whether any evidence supports this accusation, you could start by extracting log file data from the proxy server (used to connect a company LAN to the Internet) and conducting a forensic examination of the subject's computer. Suppose that during your examination, you find adult and child pornography. Further examination of the subject's hard disk reveals that the employee has been collecting child pornography in separate folders on his workstation's hard drive. In the United States, possessing child pornography is a crime under federal and state criminal statutes. These situations aren't uncommon and make life

a crime under federal and state criminal statutes. These situations aren't uncommon and make life difficult for investigators who don't want to be guilty

of possession of this contraband on their forensic workstations.
You survey the remaining content of the subject's drive and find that he's a lead engineer for the team developing your company's latest high-tech bicycle. He placed the child pornography images in a subfolder where the bicycle plans are stored. By doing so, he has commingled contraband with the company's confidential design plans for the bicycle. Your discovery poses two problems in dealing with this contraband evidence. First, you must report the crime to the police; all U.S. states and most countries have legal and moral codes when evidence of sexual exploitation of children is found. Second, you must also protect sensitive company information. Letting the high-tech bicycle plans become part of the criminal evidence might make it public record, and the design work will then be available
of the criminal evidence might make it public record, and the design work will then be available to competitors. Your first step is to ask your organization's attorney how to deal with the commingled contraband data and sensitive design plans.
Your next step is to work with the attorney to write an affidavit confirming your findings. The attorney should indicate in the affidavit that the evidence is commingled with company secrets, and releasing the information will be detrimental to the company's financial health. When the affidavit is completed, you sign it before a notary, and then deliver the affidavit and the recovered evidence with log files to the police, where you make a criminal complaint. At the same time, the attorney goes to court and requests that all evidence recovered from the hard disk that's not related to the complaint and is a company trade secret be protected from public viewing. You and the attorney have reported the crime and taken steps to protect the sensitive data.
Now suppose the detective assigned to the case calls you. In the evidence you've turned over to the police, the detective notices that the suspect is collecting most of his contraband from e-mail attachments. The prosecutor needs you to collect more evidence to determine whether the suspect is transmitting contraband pictures to other potential suspects. The detective realizes that collecting more evidence might make you an agent of law enforcement and violate the employee's Fourth Amendment rights, so she writes an affidavit for a search warrant, ensuring that any subsequent instructions to you are legal. Before collecting any additional information, you wait until you or your organization's attorney gets a subpoena, search warrant, or other court order.
---------- END CONTENT PREVIEW ----------

Diagnostic Summary

Total Chunks in DB: 11774

==========

Diagnostic Complete

==========

5.2 Test Data Base for content development

Require Description

[ ]:
```
# Cell 6: Verify Vector Database (Final Version with Rich Diagnostic Output)

import os
import json
import re
import random
import logging
from typing import List, Dict, Any, Tuple, Optional

# Third-party imports
try:
    from langchain_chroma import Chroma
    from langchain_ollama.embeddings import OllamaEmbeddings
    from langchain_core.documents import Document
    langchain_available = True
except ImportError:
    langchain_available = False

# Setup Logger for this cell
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# --- HELPER FUNCTIONS ---


def print_results(query_text: str, results: list, where_filter: Optional[Dict]= None):
    """
    Richly prints query results, showing the query, filter, and retrieved documents.
    """
    print("\n" + "-"*10 + " DIAGNOSTIC: RETRIEVAL RESULTS " + "-"*10)
    print(f"QUERY: '{query_text}'")
    if where_filter:
        print(f"FILTER: {json.dumps(where_filter, indent=2)}")

    if not results:
        print("--> No documents were retrieved for this query and filter.")
        print("-" * 55)
        return

    print(f"--> Found {len(results)} results. Displaying top {min(len(results), 3)}:")
```

```
    for i, doc in enumerate(results[:3]):
        print(f"\n[ RESULT {i+1} ]")
        content_preview = doc.page_content.replace('\n', ' ').strip()
        print(f"  Content : '{content_preview[:200]}...'")
        print(f"  Metadata: {json.dumps(doc.metadata, indent=2)}")
    print("-" * 55)


# --- HELPER FUNCTIONS FOR FINDING DATA (UNCHANGED) ---
def find_deep_entry(nodes: List[Dict], current_path: List[str] = []) ->␣
 ↪Optional[Tuple[Dict, List[str]]]:
    shuffled_nodes = random.sample(nodes, len(nodes))
    for node in shuffled_nodes:
        if node.get('level', 0) >= 2 and node.get('children'): return node,␣
 ↪current_path + [node['title']]
        if node.get('children'):
            path = current_path + [node['title']]
            deep_entry = find_deep_entry(node['children'], path)
            if deep_entry: return deep_entry
    return None

def find_chapter_title_by_number(toc_data: List[Dict], chap_num: int) ->␣
 ↪Optional[List[str]]:
    def search_nodes(nodes, num, current_path):
        for node in nodes:
            path = current_path + [node['title']]
            if re.match(rf"(Chapter\s)?{num}[.:\s]", node.get('title', ''), re.
 ↪IGNORECASE): return path
            if node.get('children'):
                found_path = search_nodes(node['children'], num, path)
                if found_path: return found_path
        return None
    return search_nodes(toc_data, chap_num, [])


# --- ENHANCED TEST CASES with DIAGNOSTIC OUTPUT ---

def basic_retrieval_test(db, outline):
    print_header("Test 1: Basic Retrieval", char="-")
    try:
        logger.info("Goal: Confirm the database is live and contains␣
 ↪thematically relevant content.")
        logger.info("Strategy: Perform a simple similarity search using the␣
 ↪course's 'unitName'.")
        query_text = outline.get("unitInformation", {}).get("unitName",␣
 ↪"introduction")
```

```
        logger.info(f"Action: Searching for query: '{query_text}'...")
        results = db.similarity_search(query_text, k=1)

        print_results(query_text, results) # <--- SHOW THE EVIDENCE

        logger.info("Verification: Check if at least one document was returned.
↪")
        assert len(results) > 0, "Basic retrieval query returned no results."

        logger.info("  Result: TEST 1 PASSED. The database is online and␣
↪responsive.")
        return True
    except Exception as e:
        logger.error(f"  Result: TEST 1 FAILED. Reason: {e}")
        return False

def deep_hierarchy_test(db, toc):
    print_header("Test 2: Deep Hierarchy Retrieval", char="-")
    try:
        logger.info("Goal: Verify that the multi-level hierarchical metadata␣
↪was ingested correctly.")
        logger.info("Strategy: Find a random, deeply nested sub-section and use␣
↪a precise filter to retrieve it.")
        deep_entry_result = find_deep_entry(toc)
        assert deep_entry_result, "Could not find a suitable deep entry (level␣
↪>= 2) to test."
        node, path = deep_entry_result
        query = node['title']

        logger.info(f"  - Selected random deep section: {' -> '.join(path)}")
        conditions = [{f"level_{i+1}_title": {"$eq": title}} for i, title in␣
↪enumerate(path)]
        w_filter = {"$and": conditions}

        logger.info("Action: Performing a similarity search with a highly␣
↪specific '$and' filter.")
        results = db.similarity_search(query, k=1, filter=w_filter)

        print_results(query, results, w_filter) # <--- SHOW THE EVIDENCE

        logger.info("Verification: Check if the precisely filtered query␣
↪returned any documents.")
        assert len(results) > 0, "Deeply filtered query returned no results."
```

```
        logger.info("  Result: TEST 2 PASSED. Hierarchical metadata is␣
↪structured correctly.")
        return True
    except Exception as e:
        logger.error(f"  Result: TEST 2 FAILED. Reason: {e}")
        return False

def advanced_alignment_test(db, outline, toc):
    print_header("Test 3: Advanced Unit Outline Alignment", char="-")
    try:
        logger.info("Goal: Ensure a weekly topic from the syllabus can be␣
↪mapped to the correct textbook chapter(s).")
        logger.info("Strategy: Pick a random week, find its chapter, and query␣
↪for the topic filtered by that chapter.")
        week_to_test = random.choice(outline['weeklySchedule'])
        logger.info(f"  - Selected random week: Week {week_to_test['week']} -␣
↪'{week_to_test['contentTopic']}'")

        reading = week_to_test.get('requiredReading', '')
        chap_nums_str = re.findall(r'\d+', reading)
        assert chap_nums_str, f"Could not find chapter numbers in required␣
↪reading: '{reading}'"
        logger.info(f"  - Extracted required chapter number(s):␣
↪{chap_nums_str}")

        chapter_paths = [find_chapter_title_by_number(toc, int(n)) for n in␣
↪chap_nums_str]
        chapter_paths = [path for path in chapter_paths if path is not None]
        assert chapter_paths, f"Could not map chapter numbers {chap_nums_str}␣
↪to a valid ToC path."

        level_1_titles = list(set([path[0] for path in chapter_paths]))
        logger.info(f"  - Mapped to top-level ToC entries: {level_1_titles}")

        or_filter = [{"level_1_title": {"$eq": title}} for title in␣
↪level_1_titles]
        w_filter = {"$or": or_filter} if len(or_filter) > 1 else or_filter[0]
        query = week_to_test['contentTopic']

        logger.info("Action: Searching for the weekly topic, filtered by the␣
↪mapped chapter(s).")
        results = db.similarity_search(query, k=5, filter=w_filter)

        print_results(query, results, w_filter) # <--- SHOW THE EVIDENCE
```

```
        logger.info("Verification: Check if at least one returned document is␣
↪from the correct chapter.")
        assert len(results) > 0, "Alignment query returned no results for the␣
↪correct section/chapter."

        logger.info("  Result: TEST 3 PASSED. The syllabus can be reliably␣
↪aligned with the textbook content.")
        return True
    except Exception as e:
        logger.error(f"  Result: TEST 3 FAILED. Reason: {e}")
        return False

def content_sequence_test(db, outline):
    print_header("Test 4: Content Sequence Verification", char="-")
    try:
        logger.info("Goal: Confirm that chunks for a topic can be re-ordered to␣
↪form a coherent narrative.")
        logger.info("Strategy: Retrieve several chunks for a random topic and␣
↪verify their 'chunk_id' is sequential.")
        topic_query = random.choice(outline['weeklySchedule'])['contentTopic']

        logger.info(f"Action: Performing similarity search for topic:␣
↪'{topic_query}' to get a set of chunks.")
        results = db.similarity_search(topic_query, k=10)

        print_results(topic_query, results) # <--- SHOW THE EVIDENCE

        docs_with_id = [doc for doc in results if 'chunk_id' in doc.metadata]
        assert len(docs_with_id) > 3, "Fewer than 4 retrieved chunks have a␣
↪'chunk_id' to test."

        chunk_ids = [doc.metadata['chunk_id'] for doc in docs_with_id]
        sorted_ids = sorted(chunk_ids)

        logger.info(f"  - Retrieved and sorted chunk IDs: {sorted_ids}")
        logger.info("Verification: Check if the sorted list of chunk_ids is␣
↪strictly increasing.")
        is_ordered = all(sorted_ids[i] >= sorted_ids[i-1] for i in range(1,␣
↪len(sorted_ids)))
        assert is_ordered, "The retrieved chunks' chunk_ids are not in␣
↪ascending order when sorted."

        logger.info("  Result: TEST 4 PASSED. Narrative order can be␣
↪reconstructed using 'chunk_id'.")
        return True
    except Exception as e:
```

```
        logger.error(f"  Result: TEST 4 FAILED. Reason: {e}")
        return False

# --- MAIN VERIFICATION EXECUTION ---
def run_verification():
    print_header("Database Verification Process")

    if not langchain_available:
        logger.error("LangChain libraries not found. Aborting tests.")
        return

    required_files = {
        "Chroma DB": CHROMA_PERSIST_DIR,
        "ToC JSON": PRE_EXTRACTED_TOC_JSON_PATH,
        "Parsed Outline": PARSED_UO_JSON_PATH
    }
    for name, path in required_files.items():
        if not os.path.exists(path):
            logger.error(f"Required '{name}' not found at '{path}'. Please run␣
 ↪previous cells.")
            return

    with open(PRE_EXTRACTED_TOC_JSON_PATH, 'r', encoding='utf-8') as f:
        toc_data = json.load(f)
    with open(PARSED_UO_JSON_PATH, 'r', encoding='utf-8') as f:
        unit_outline_data = json.load(f)

    logger.info("Connecting to DB and initializing components...")
    embeddings = OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA)
    vector_store = Chroma(
        persist_directory=CHROMA_PERSIST_DIR,
        embedding_function=embeddings,
        collection_name=CHROMA_COLLECTION_NAME
    )

    results_summary = [
        basic_retrieval_test(vector_store, unit_outline_data),
        deep_hierarchy_test(vector_store, toc_data),
        advanced_alignment_test(vector_store, unit_outline_data, toc_data),
        content_sequence_test(vector_store, unit_outline_data)
    ]

    passed_count = sum(filter(None, results_summary))
    failed_count = len(results_summary) - passed_count

    print_header("Verification Summary")
    print(f"Total Tests Run: {len(results_summary)}")
```

```
    print(f"  Passed: {passed_count}")
    print(f"  Failed: {failed_count}")
    print_header("Verification Complete", char="=")

# --- Execute Verification ---
# Assumes global variables from Cell 1 are available in the notebook's scope
run_verification()
```

```
2025-07-01 21:02:48,736 - INFO - Connecting to DB and initializing components…
2025-07-01 21:02:48,746 - INFO - Goal: Confirm the database is live and contains
thematically relevant content.
2025-07-01 21:02:48,746 - INFO - Strategy: Perform a simple similarity search
using the course's 'unitName'.
2025-07-01 21:02:48,747 - INFO - Action: Searching for query: 'Digital
Forensic'…
2025-07-01 21:02:48,814 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:48,818 - INFO - Verification: Check if at least one document
was returned.
2025-07-01 21:02:48,818 - INFO -   Result: TEST 1 PASSED. The database is online
and responsive.
2025-07-01 21:02:48,819 - INFO - Goal: Verify that the multi-level hierarchical
metadata was ingested correctly.
2025-07-01 21:02:48,819 - INFO - Strategy: Find a random, deeply nested sub-
section and use a precise filter to retrieve it.
2025-07-01 21:02:48,820 - INFO -   - Selected random deep section: Chapter 6.
Current Digital Forensics Tools -> Digital Forensics Hardware Tools -> Forensic
Workstations
2025-07-01 21:02:48,820 - INFO - Action: Performing a similarity search with a
highly specific '$and' filter.


================================================================================
                        Database Verification Process
================================================================================

--------------------------------------------------------------------------------
                            Test 1: Basic Retrieval
--------------------------------------------------------------------------------

---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
QUERY: 'Digital Forensic'
--> Found 1 results. Displaying top 1:

[ RESULT 1 ]
  Content : 'An Overview of Digital Forensics…'
  Metadata: {
  "chunk_id": 156,
```

```
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "level_1_title": "Chapter 1. Understanding the Digital Forensics Profession
and Investigations",
  "toc_id": 9,
  "level_2_title": "An Overview of Digital Forensics"
}
```

Test 2: Deep Hierarchy Retrieval

```
2025-07-01 21:02:48,953 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:48,965 - INFO - Verification: Check if the precisely filtered
query returned any documents.
2025-07-01 21:02:48,965 - INFO -   Result: TEST 2 PASSED. Hierarchical metadata
is structured correctly.
2025-07-01 21:02:48,966 - INFO - Goal: Ensure a weekly topic from the syllabus
can be mapped to the correct textbook chapter(s).
2025-07-01 21:02:48,966 - INFO - Strategy: Pick a random week, find its chapter,
and query for the topic filtered by that chapter.
2025-07-01 21:02:48,967 - INFO -   - Selected random week: Week Week 9 - 'Email
and Social Media.'
2025-07-01 21:02:48,967 - INFO -   - Extracted required chapter number(s):
['2019', '978', '1', '337', '56894', '4', '11']
2025-07-01 21:02:48,970 - INFO -   - Mapped to top-level ToC entries: ['Chapter
11. E-mail and Social Media Investigations', 'Chapter 4. Processing Crime and
Incident Scenes', 'Chapter 1. Understanding the Digital Forensics Profession and
Investigations']
2025-07-01 21:02:48,970 - INFO - Action: Searching for the weekly topic,
filtered by the mapped chapter(s).
2025-07-01 21:02:49,080 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:49,099 - INFO - Verification: Check if at least one returned
document is from the correct chapter.
2025-07-01 21:02:49,100 - INFO -   Result: TEST 3 PASSED. The syllabus can be
reliably aligned with the textbook content.
2025-07-01 21:02:49,101 - INFO - Goal: Confirm that chunks for a topic can be
re-ordered to form a coherent narrative.
2025-07-01 21:02:49,101 - INFO - Strategy: Retrieve several chunks for a random
topic and verify their 'chunk_id' is sequential.
2025-07-01 21:02:49,102 - INFO - Action: Performing similarity search for topic:
'Current Computer Forensics Tools.' to get a set of chunks.


---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
```

```
QUERY: 'Forensic Workstations'
FILTER: {
  "$and": [
    {
      "level_1_title": {
        "$eq": "Chapter 6. Current Digital Forensics Tools"
      }
    },
    {
      "level_2_title": {
        "$eq": "Digital Forensics Hardware Tools"
      }
    },
    {
      "level_3_title": {
        "$eq": "Forensic Workstations"
      }
    }
  ]
}
--> Found 1 results. Displaying top 1:

[ RESULT 1 ]
  Content : 'Forensic Workstations…'
  Metadata: {
  "level_3_title": "Forensic Workstations",
  "level_2_title": "Digital Forensics Hardware Tools",
  "toc_id": 255,
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "chunk_id": 3311
}
--------------------------------------------------------

--------------------------------------------------------------------------------
                    Test 3: Advanced Unit Outline Alignment
--------------------------------------------------------------------------------

---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
QUERY: 'Email and Social Media.'
FILTER: {
  "$or": [
    {
      "level_1_title": {
        "$eq": "Chapter 11. E-mail and Social Media Investigations"
      }
```

```
    },
    {
      "level_1_title": {
        "$eq": "Chapter 4. Processing Crime and Incident Scenes"
      }
    },
    {
      "level_1_title": {
        "$eq": "Chapter 1. Understanding the Digital Forensics Profession and
Investigations"
      }
    }
  ]
}
--> Found 5 results. Displaying top 3:

[ RESULT 1 ]
  Content : 'Chapter 11. E-mail and Social Media Investigations…'
  Metadata: {
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "chunk_id": 5378,
  "toc_id": 377,
  "level_1_title": "Chapter 11. E-mail and Social Media Investigations"
}

[ RESULT 2 ]
  Content : 'Chapter 11. E-mail and Social Media Investigations…'
  Metadata: {
  "chunk_id": 10484,
  "level_1_title": "Chapter 11. E-mail and Social Media Investigations",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "toc_id": 377
}

[ RESULT 3 ]
  Content : 'Social media can contain a lot of information, including the
following:…'
  Metadata: {
  "level_2_title": "Applying Digital Forensics Methods to Social Media
Communications",
  "chunk_id": 5636,
  "toc_id": 398,
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
```

```
Learning (2018).epub",
  "level_1_title": "Chapter 11. E-mail and Social Media Investigations"
}
--------------------------------------------------

--------------------------------------------------------------------------------
                    Test 4: Content Sequence Verification
--------------------------------------------------------------------------------

2025-07-01 21:02:49,217 - INFO - HTTP Request: POST
http://127.0.0.1:11434/api/embed "HTTP/1.1 200 OK"
2025-07-01 21:02:49,220 - INFO -   - Retrieved and sorted chunk IDs: [49, 3138,
3141, 3160, 3164, 3166, 3267, 3271, 3308, 9541]
2025-07-01 21:02:49,221 - INFO - Verification: Check if the sorted list of
chunk_ids is strictly increasing.
2025-07-01 21:02:49,221 - INFO -  Result: TEST 4 PASSED. Narrative order can be
reconstructed using 'chunk_id'.


---------- DIAGNOSTIC: RETRIEVAL RESULTS ----------
QUERY: 'Current Computer Forensics Tools.'
--> Found 10 results. Displaying top 3:

[ RESULT 1 ]
  Content : 'Chapter 6. Current Digital Forensics Tools…'
  Metadata: {
  "toc_id": 231,
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "chunk_id": 3138
}

[ RESULT 2 ]
  Content : 'Chapter 6. Current Digital Forensics Tools…'
  Metadata: {
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "chunk_id": 9541,
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "toc_id": 231
}

[ RESULT 3 ]
  Content : 'Software Forensics Tools…'
  Metadata: {
  "level_3_title": "Types of Digital Forensics Tools",
```

```
  "level_4_title": "Software Forensics Tools",
  "level_1_title": "Chapter 6. Current Digital Forensics Tools",
  "source": "Bill Nelson, Amelia Phillips, Christopher Steuart - Guide to
Computer Forensics and Investigations_ Processing Digital Evidence-Cengage
Learning (2018).epub",
  "level_2_title": "Evaluating Digital Forensics Tool Needs",
  "toc_id": 236,
  "chunk_id": 3166
}
----------------------------------------------------

================================================================================
                              Verification Summary
================================================================================
Total Tests Run: 4
 Passed: 4
 Failed: 0

================================================================================
                             Verification Complete
================================================================================
```

6 Content Generation

6.1 Planning Agent

[111]:
```
# # Cell 7: The Data-Driven Planning Agent (Final Hierarchical Version)

# import os
# import json
# import re
# import math
# import logging
# from typing import List, Dict, Any, Optional

# # Setup Logger and LangChain components
# logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
# logger = logging.getLogger(__name__)
# try:
#     from langchain_chroma import Chroma
#     from langchain_ollama.embeddings import OllamaEmbeddings
#     langchain_available = True
# except ImportError:
#     langchain_available = False

# def print_header(text: str, char: str = "="):
```

```
#     """Prints a centered header to the console."""
#     print("\n" + char * 80)
#     print(text.center(80))
#     print(char * 80)

# class PlanningAgent:
#     """
#     An agent that creates a hierarchical content plan, adaptively partitions␣
 ↪content
#     into distinct lecture decks, and allocates presentation time.
#     """
#     def __init__(self, master_config: Dict, vector_store: Optional[Any] =␣
 ↪None):
#         self.config = master_config['processed_settings']
#         self.unit_outline = master_config['unit_outline']
#         self.book_toc = master_config['book_toc']
#         self.flat_toc_with_ids = self._create_flat_toc_with_ids()
#         self.vector_store = vector_store
#         logger.info("Data-Driven PlanningAgent initialized successfully.")

#     def _create_flat_toc_with_ids(self) -> List[Dict]:
#         """Creates a flattened list of the ToC for easy metadata lookup."""
#         flat_list = []
#         def flatten_recursive(nodes, counter):
#             for node in nodes:
#                 node_id = counter[0]; counter[0] += 1
#                 flat_list.append({'toc_id': node_id, 'title': node.
 ↪get('title', ''), 'node': node})
#                 if node.get('children'):
#                     flatten_recursive(node.get('children'), counter)
#         flatten_recursive(self.book_toc, [0])
#         return flat_list

#     def _identify_relevant_chapters(self, weekly_schedule_item: Dict) ->␣
 ↪List[int]:
#         """Extracts chapter numbers precisely from the 'requiredReading'␣
 ↪string."""
#         reading_str = weekly_schedule_item.get('requiredReading', '')
#         match = re.search(r'Chapter(s)?', reading_str, re.IGNORECASE)
#         if not match: return []
#         search_area = reading_str[match.start():]
#         chap_nums_str = re.findall(r'\d+', search_area)
#         if chap_nums_str:
#             return sorted(list(set(int(n) for n in chap_nums_str)))
#         return []

#     def _find_chapter_node(self, chapter_number: int) -> Optional[Dict]:
```

```
#         """Finds the ToC node for a specific chapter number."""
#         for item in self.flat_toc_with_ids:
#             if re.match(rf"Chapter\s{chapter_number}(?:\D|$)", item['title']):
#                 return item['node']
#         return None

#     def _build_topic_plan_tree(self, toc_node: Dict) -> Dict:
#         """
#         Recursively builds a hierarchical plan tree from any ToC node,
#         annotating it with direct and total branch chunk counts.
#         """
#         node_metadata = next((item for item in self.flat_toc_with_ids if␣
 ↪item['node'] is toc_node), None)
#         if not node_metadata: return {}

#         retrieved_docs = self.vector_store.get(where={'toc_id':␣
 ↪node_metadata['toc_id']})
#         direct_chunk_count = len(retrieved_docs.get('ids', []))

#         plan_node = {
#             "title": node_metadata['title'],
#             "toc_id": node_metadata['toc_id'],
#             "chunk_count": direct_chunk_count,
#             "total_chunks_in_branch": 0,
#             "slides_allocated": 0,
#             "children": []
#         }

#         child_branch_total = 0
#         for child_node in toc_node.get('children', []):
#             if any(ex in child_node.get('title', '').lower() for ex in␣
 ↪["review", "introduction", "summary", "key terms"]):
#                 continue
#             child_plan_node = self._build_topic_plan_tree(child_node)
#             if child_plan_node:
#                 plan_node['children'].append(child_plan_node)
#                 child_branch_total += child_plan_node.
 ↪get('total_chunks_in_branch', 0)

#         plan_node['total_chunks_in_branch'] = direct_chunk_count +␣
 ↪child_branch_total
#         return plan_node

#     def _allocate_slides_to_tree(self, plan_tree: Dict, content_slides_budget:
 ↪ int):
```

```
#         """Performs a two-pass safety-net allocation on a hierarchical plan␣
 ↪tree."""
#         leaf_nodes = []
#         def find_leaves(node):
#             if not node.get('children'):
#                 leaf_nodes.append(node)
#             for child in node.get('children', []):
#                 find_leaves(child)
#         find_leaves(plan_tree)

#         if not leaf_nodes or content_slides_budget <= 0: return plan_tree

#         # Pass 1: Safety Net
#         slides_per_topic = 1 if content_slides_budget >= len(leaf_nodes) else␣
 ↪0
#         for node in leaf_nodes:
#             node['slides_allocated'] = slides_per_topic

#         remaining_budget = content_slides_budget - (len(leaf_nodes) *␣
 ↪slides_per_topic)

#         # Pass 2: Proportional Distribution
#         if remaining_budget > 0:
#             total_leaf_chunks = sum(node['chunk_count'] for node in␣
 ↪leaf_nodes)
#             if total_leaf_chunks > 0:
#                 # Distribute remaining slides based on chunk weight
#                 for node in leaf_nodes:
#                     node['slides_allocated'] += round((node['chunk_count'] /␣
 ↪total_leaf_chunks) * remaining_budget)

#         # Pass 3: Sum totals upwards
#         def sum_slides_upwards(node):
#             if not node.get('children'):
#                 return node['slides_allocated']
#             node['slides_allocated'] = sum(sum_slides_upwards(child) for␣
 ↪child in node['children'])
#             return node['slides_allocated']
#         sum_slides_upwards(plan_tree)
#         return plan_tree

#     def create_content_plan_for_week(self, week_number: int) ->␣
 ↪Optional[Dict]:
#         """Orchestrates the adaptive planning and partitioning process."""
#         print_header(f"Planning Week {week_number}", char="*")
```

```
#         weekly_schedule_item = self.
 ↪unit_outline['weeklySchedule'][week_number - 1]
#         chapter_numbers = self.
 ↪_identify_relevant_chapters(weekly_schedule_item)
#         if not chapter_numbers: return None

#         num_decks = self.config['week_session_setup'].
 ↪get('sessions_per_week', 1)

#         # 1. Build a full plan tree for each chapter to get its weight.
#         chapter_plan_trees = [self._build_topic_plan_tree(self.
 ↪_find_chapter_node(cn)) for cn in chapter_numbers if self.
 ↪_find_chapter_node(cn)]
#         total_weekly_chunks = sum(tree.get('total_chunks_in_branch', 0) for␣
 ↪tree in chapter_plan_trees)

#         # 2. NEW: Adaptive Partitioning Strategy
#         partitionable_units = []
#         num_chapters = len(chapter_plan_trees)

#         if num_chapters >= num_decks:
#             logger.info(f"Partitioning strategy: Distributing {num_chapters}␣
 ↪whole chapters across {num_decks} decks.")
#             partitionable_units = chapter_plan_trees
#         else:
#             logger.info(f"Partitioning strategy: Splitting sub-topics from␣
 ↪{num_chapters} chapter(s) across {num_decks} decks.")
#             for chapter_tree in chapter_plan_trees:
#                 partitionable_units.extend(chapter_tree.get('children', []))

#         # 3. Partition the chosen units into decks using a bin-packing␣
 ↪algorithm
#         decks = [[] for _ in range(num_decks)]
#         deck_weights = [0] * num_decks
#         sorted_units = sorted(partitionable_units, key=lambda x: x.
 ↪get('total_chunks_in_branch', 0), reverse=True)

#         for unit in sorted_units:
#             lightest_deck_index = deck_weights.index(min(deck_weights))
#             decks[lightest_deck_index].append(unit)
#             deck_weights[lightest_deck_index] += unit.
 ↪get('total_chunks_in_branch', 0)

#         # 4. Plan each deck
#         content_slides_per_week = self.config['slide_count_strategy'].
 ↪get('target', 25)
```

```
#         final_deck_plans = []
#         for i, deck_content_trees in enumerate(decks):
#             deck_number = i + 1
#             deck_chunk_weight = sum(tree.get('total_chunks_in_branch', 0) for␣
 ↪tree in deck_content_trees)
#             deck_slide_budget = round((deck_chunk_weight /␣
 ↪total_weekly_chunks) * content_slides_per_week) if total_weekly_chunks > 0␣
 ↪else 0

#             logger.info(f"--- Planning Deck {deck_number}/{num_decks} |␣
 ↪Topics: {[t['title'] for t in deck_content_trees]} | Weight:␣
 ↪{deck_chunk_weight} chunks | Slide Budget: {deck_slide_budget} ---")

#             # The allocation function is recursive and works on any tree or␣
 ↪sub-tree
#             planned_content = [self._allocate_slides_to_tree(tree,␣
 ↪round(deck_slide_budget * (tree.get('total_chunks_in_branch', 0) /␣
 ↪deck_chunk_weight))) if deck_chunk_weight > 0 else tree for tree in␣
 ↪deck_content_trees]

#             final_deck_plans.append({
#                 "deck_number": deck_number,
#                 "deck_title": f"{self.config.get('unit_name', 'Course')} -␣
 ↪Week {week_number}, Lecture {deck_number}",
#                 "session_content": planned_content
#             })

#         return {
#             "week": week_number,
#             "overall_topic": weekly_schedule_item.get('contentTopic'),
#             "deck_plans": final_deck_plans
#         }
```

[112]:
```
# Cell 7: The Data-Driven Planning Agent (Final Hierarchical Version)

import os
import json
import re
import math
import logging
from typing import List, Dict, Any, Optional

# Setup Logger and LangChain components
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')
logger = logging.getLogger(__name__)
```

```
try:
    from langchain_chroma import Chroma
    from langchain_ollama.embeddings import OllamaEmbeddings
    langchain_available = True
except ImportError:
    langchain_available = False

def print_header(text: str, char: str = "="):
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)

class PlanningAgent:
    """
    An agent that creates a hierarchical content plan, adaptively partitions␣
 ↪content
    into distinct lecture decks, and allocates presentation time.
    """
    def __init__(self, master_config: Dict, vector_store: Optional[Any] = None):
        self.config = master_config['processed_settings']
        self.unit_outline = master_config['unit_outline']
        self.book_toc = master_config['book_toc']
        self.flat_toc_with_ids = self._create_flat_toc_with_ids()
        self.vector_store = vector_store
        logger.info("Data-Driven PlanningAgent initialized successfully.")

    def _create_flat_toc_with_ids(self) -> List[Dict]:
        """Creates a flattened list of the ToC for easy metadata lookup."""
        flat_list = []
        def flatten_recursive(nodes, counter):
            for node in nodes:
                node_id = counter[0]; counter[0] += 1
                flat_list.append({'toc_id': node_id, 'title': node.get('title',␣
 ↪''), 'node': node})
                if node.get('children'):
                    flatten_recursive(node.get('children'), counter)
        flatten_recursive(self.book_toc, [0])
        return flat_list

    def _identify_relevant_chapters(self, weekly_schedule_item: Dict) ->␣
 ↪List[int]:
        """Extracts chapter numbers precisely from the 'requiredReading' string.
 ↪"""
        reading_str = weekly_schedule_item.get('requiredReading', '')
        match = re.search(r'Chapter(s)?', reading_str, re.IGNORECASE)
        if not match: return []
```

```
        search_area = reading_str[match.start():]
        chap_nums_str = re.findall(r'\d+', search_area)
        if chap_nums_str:
            return sorted(list(set(int(n) for n in chap_nums_str)))
        return []

    def _find_chapter_node(self, chapter_number: int) -> Optional[Dict]:
        """Finds the ToC node for a specific chapter number."""
        for item in self.flat_toc_with_ids:
            if re.match(rf"Chapter\s{chapter_number}(?:\D|$)", item['title']):
                return item['node']
        return None

    def _build_topic_plan_tree(self, toc_node: Dict) -> Dict:
        """
        Recursively builds a hierarchical plan tree from any ToC node,
        annotating it with direct and total branch chunk counts.
        """
        node_metadata = next((item for item in self.flat_toc_with_ids if␣
↪item['node'] is toc_node), None)
        if not node_metadata: return {}

        retrieved_docs = self.vector_store.get(where={'toc_id':␣
↪node_metadata['toc_id']})
        direct_chunk_count = len(retrieved_docs.get('ids', []))

        plan_node = {
            "title": node_metadata['title'],
            "toc_id": node_metadata['toc_id'],
            "chunk_count": direct_chunk_count,
            "total_chunks_in_branch": 0,
            "slides_allocated": 0,
            "children": []
        }

        child_branch_total = 0
        for child_node in toc_node.get('children', []):
            if any(ex in child_node.get('title', '').lower() for ex in␣
↪["review", "introduction", "summary", "key terms"]):
                continue
            child_plan_node = self._build_topic_plan_tree(child_node)
            if child_plan_node:
                plan_node['children'].append(child_plan_node)
                child_branch_total += child_plan_node.
↪get('total_chunks_in_branch', 0)
```

```
        plan_node['total_chunks_in_branch'] = direct_chunk_count +␣
↪child_branch_total
        return plan_node

    def _allocate_slides_to_tree(self, plan_tree: Dict, content_slides_budget:␣
↪int):
        """
        Performs a two-pass safety-net allocation for content slides and,
        based on configuration, adds interactive slides (with a reference␣
↪toc_id)
        at specified depths.
        """
        leaf_nodes = []

        def find_leaves(node):
            if not node.get('children'):
                leaf_nodes.append(node)
            for child in node.get('children', []):
                find_leaves(child)

        find_leaves(plan_tree)

        if not leaf_nodes or content_slides_budget <= 0:
            return plan_tree

        # --- Content Slide Allocation ---
        slides_per_topic = 1 if content_slides_budget >= len(leaf_nodes) else 0
        for node in leaf_nodes:
            node['slides_allocated'] = slides_per_topic

        remaining_budget = content_slides_budget - (len(leaf_nodes) *␣
↪slides_per_topic)

        if remaining_budget > 0:
            total_leaf_chunks = sum(node.get('chunk_count', 0) for node in␣
↪leaf_nodes)
            if total_leaf_chunks > 0:
                for node in leaf_nodes:
                    node['slides_allocated'] += round((node.get('chunk_count',␣
↪0) / total_leaf_chunks) * remaining_budget)

        # --- Interactive Slide Tagging with Depth Control ---
        def add_interactive_nodes(node, depth, interactive_deep):
            max_depth = 2 if interactive_deep else 1

            if depth > 0 and depth <= max_depth and node.get('children'):
```

```
                # --- THIS IS THE KEY MODIFICATION ---
                # Add the toc_id from the parent node to the interactive␣
↪activity.
                node['interactive_activity'] = {
                    "title": f"{node.get('title')} (Interactive Activity)",
                    "toc_id": node.get('toc_id'), # Inherit toc_id for context␣
↪retrieval
                    "slides_allocated": 1
                }

            for child in node.get('children', []):
                add_interactive_nodes(child, depth + 1, interactive_deep)

        if self.config.get('interactive', False):
            interactive_deep = self.config.get('interactive_deep', False)
            logger.info(f"Interactive mode ON. Deep interaction:␣
↪{interactive_deep}. Adding placeholders...")
            add_interactive_nodes(plan_tree, 0, interactive_deep)

        # --- Sum totals up the tree ---
        def sum_slides_upwards(node):
            total_slides = node.get('interactive_activity', {}).
↪get('slides_allocated', 0)
            if not node.get('children'):
                total_slides += node.get('slides_allocated', 0)
            else:
                total_slides += sum(sum_slides_upwards(child) for child in␣
↪node['children'])
            node['slides_allocated'] = total_slides
            return total_slides

        sum_slides_upwards(plan_tree)
        return plan_tree

    def create_content_plan_for_week(self, week_number: int) -> Optional[Dict]:
        """Orchestrates the adaptive planning and partitioning process."""
        print_header(f"Planning Week {week_number}", char="*")

        weekly_schedule_item = self.unit_outline['weeklySchedule'][week_number␣
↪- 1]
        chapter_numbers = self._identify_relevant_chapters(weekly_schedule_item)
        if not chapter_numbers: return None

        num_decks = self.config['week_session_setup'].get('sessions_per_week',␣
↪1)
```

```
        # 1. Build a full plan tree for each chapter to get its weight.
        chapter_plan_trees = [self._build_topic_plan_tree(self.
↪_find_chapter_node(cn)) for cn in chapter_numbers if self.
↪_find_chapter_node(cn)]
        total_weekly_chunks = sum(tree.get('total_chunks_in_branch', 0) for␣
↪tree in chapter_plan_trees)

        # 2. NEW: Adaptive Partitioning Strategy
        partitionable_units = []
        num_chapters = len(chapter_plan_trees)

        if num_chapters >= num_decks:
            logger.info(f"Partitioning strategy: Distributing {num_chapters}␣
↪whole chapters across {num_decks} decks.")
            partitionable_units = chapter_plan_trees
        else:
            logger.info(f"Partitioning strategy: Splitting sub-topics from␣
↪{num_chapters} chapter(s) across {num_decks} decks.")
            for chapter_tree in chapter_plan_trees:
                partitionable_units.extend(chapter_tree.get('children', []))

        # 3. Partition the chosen units into decks using a bin-packing algorithm
        decks = [[] for _ in range(num_decks)]
        deck_weights = [0] * num_decks
        sorted_units = sorted(partitionable_units, key=lambda x: x.
↪get('total_chunks_in_branch', 0), reverse=True)

        for unit in sorted_units:
            lightest_deck_index = deck_weights.index(min(deck_weights))
            decks[lightest_deck_index].append(unit)
            deck_weights[lightest_deck_index] += unit.
↪get('total_chunks_in_branch', 0)

        # 4. Plan each deck
        content_slides_per_week = self.config['slide_count_strategy'].
↪get('target', 25)
        final_deck_plans = []
        for i, deck_content_trees in enumerate(decks):
            deck_number = i + 1
            deck_chunk_weight = sum(tree.get('total_chunks_in_branch', 0) for␣
↪tree in deck_content_trees)
            deck_slide_budget = round((deck_chunk_weight / total_weekly_chunks)␣
↪* content_slides_per_week) if total_weekly_chunks > 0 else 0
```

```
            logger.info(f"--- Planning Deck {deck_number}/{num_decks} | Topics:␣
↪{[t['title'] for t in deck_content_trees]} | Weight: {deck_chunk_weight}␣
↪chunks | Slide Budget: {deck_slide_budget} ---")

            # The allocation function is recursive and works on any tree or␣
↪sub-tree
            planned_content = [self._allocate_slides_to_tree(tree,␣
↪round(deck_slide_budget * (tree.get('total_chunks_in_branch', 0) /␣
↪deck_chunk_weight))) if deck_chunk_weight > 0 else tree for tree in␣
↪deck_content_trees]

            final_deck_plans.append({
                "deck_number": deck_number,
                "deck_title": f"{self.config.get('unit_name', 'Course')} - Week␣
↪{week_number}, Lecture {deck_number}",
                "session_content": planned_content
            })

        return {
            "week": week_number,
            "overall_topic": weekly_schedule_item.get('contentTopic'),
            "deck_plans": final_deck_plans
        }
```

6.2 Content Generator Class (no yet addressed focus planning)

6.3 Orquestrator (Addressing paint points)

Description:

The main script that iterates through the weeks defined the plan and generate the content base on the settings_deck coordinating the agents.

Parameters and concideration - 1 hour in the setting session_time_duration_in_hour - is 18-20 slides at the time so it is require to calculate this according to the given value but this also means per session so sessions_per_week is a multiplicator factor that
- if apply_topic_interactive is available will add an extra slide and add extra 5 min time but to determine this is required to plan all the content first and then calculate then provide a extra time

settings_deck.json

{ “course_id”: ““,”unit_name”: ““,”interactive”: true, “interactive_deep”: false, “slide_count_strategy”: { “method”: “per_week”, “interactive_slides_per_week”: 0 – > sum all interactive counts “interactive_slides_per_session”: 0, – > Total # of slides produced if “interactive” is true other wise remains 0 “target_total_slides”: 0, –> Total Content Slides per week that cover the total - will be the target in the cell 7
“slides_content_per_session”: 0, –> Total # (target_total_slides/sessions_per_week) “total_slides_deck_week”: 0, –> target_total_slides + interactive_slides_per_week + (framework (4 + Time for Title, Agenda, Summary, End) * sessions_per_week) “Tota_slides_session”: 0 –> content_slides_per_session + interactive_slides_per_session + framework (4 + Time for

Title, Agenda, Summary, End) }, “week_session_setup”: { “sessions_per_week”: 1, “distribution_strategy”: “even”, “interactive_time_in_hour”: 0, –> find the value in ahours of the total # (“interactive_slides” * “TIME_PER_INTERACTIVE_SLIDE_MINS”)/60
“total_session_time_in_hours”: 0 –> this is going to be egual or similar to session_time_duration_in_hour if “interactive” is false obvisuly base on the global varaibles it will be the calculation of “interactive_time_in_hour” “session_time_duration_in_hour”: 2, — > this is the time that the costumer need for delivery this is a constrain is not modified never is used for reference },

“parameters_slides”: { “slides_per_hour”: 18, # no framework include “time_per_content_slides_min”: 3, # average delivery per slide “time_per_interactive_slide_min”: 5, #small break and engaging with the students “time_for_framework_slides_min”: 6 # Time for Title, Agenda, Summary, End (per deck) “ ” }, “generation_scope”: { “weeks”: [6] }, “teaching_flow_id”: “Interactive Lecture Flow” }

teaching_flows.json

{ “standard_lecture”: { “name”: “Standard Lecture Flow”, “slide_types”: [“Title”, “Agenda”, “Content”, “Summary”, “End”], “prompts”: { “content_generation”: “You are an expert university lecturer. Your audience is undergraduate students. Based on the following context, create a slide that provides a detailed explanation of the topic ‘{sub_topic}’. The content should be structured with bullet points for key details. Your output MUST be a single JSON object with a ‘title’ (string) and ‘content’ (list of strings) key.”, “summary_generation”: “You are an expert university lecturer creating a summary slide. Based on the following list of topics covered in this session, generate a concise summary of the key takeaways. The topics are: {topic_list}. Your output MUST be a single JSON object with a ‘title’ (string) and ‘content’ (list of strings) key.” }, “slide_schemas”: { “Content”: {“title”: “string”, “content”: “list[string]”}, “Summary”: {“title”: “string”, “content”: “list[string]”} } }, “apply_topic_interactive”: { “name”: “Interactive Lecture Flow”, “slide_types”: [“Title”, “Agenda”, “Content”, “Application”, “Summary”, “End”], “prompts”: { “content_generation”: “You are an expert university lecturer in Digital Forensics. Your audience is undergraduate students. Based on the provided context, create a slide explaining the concept of ‘{sub_topic}’. The content should be clear, concise, and structured with bullet points for easy understanding. Your output MUST be a single JSON object with a ‘title’ (string) and ‘content’ (list of strings) key.”, “application_generation”: “You are an engaging university lecturer creating an interactive slide. Based on the concept of ‘{sub_topic}’, create a multiple-choice question with exactly 4 options (A, B, C, D) to test understanding. The slide title must be ‘Let’s Apply This:’. Clearly indicate the correct answer within the content. Your output MUST be a single JSON object with a ‘title’ (string) and ‘content’ (list of strings) key.”, “summary_generation”: “You are an expert university lecturer creating a summary slide. Based on the following list of concepts and applications covered in this session, generate a concise summary of the key takeaways. The topics are: {topic_list}. Your output MUST be a single JSON object with a ‘title’ (string) and ‘content’ (list of strings) key.” }, “slide_schemas”: { “Content”: {“title”: “string”, “content”: “list[string]”}, “Application”: {“title”: “string”, “content”: “list[string]”}, “Summary”: {“title”: “string”, “content”: “list[string]”} } } }

[113]:
```
# Cell 8: Configuration and Scoping for Content Generation (Corrected)

import os
import json
```

```
import logging

# Setup Logger for this cell
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# --- 1. DEFINE FILE PATHS AND GLOBAL TEST SETTINGS ---
# Assumes these variables are loaded from a previous setup cell (like Cell 1)
# PROJECT_BASE_DIR, PARSED_UO_JSON_PATH, PRE_EXTRACTED_TOC_JSON_PATH must be defined.

# New configuration file paths
CONFIG_DIR = os.path.join(PROJECT_BASE_DIR, "configs")
SETTINGS_DECK_PATH = os.path.join(CONFIG_DIR, "settings_deck.json")
TEACHING_FLOWS_PATH = os.path.join(CONFIG_DIR, "teaching_flows.json")

# New output path for the processed settings
PROCESSED_SETTINGS_PATH = os.path.join(CONFIG_DIR, "processed_settings.json")

# --- Global Test Overrides (for easy testing) ---
TEST_OVERRIDE_WEEKS = None
TEST_OVERRIDE_FLOW_ID = None
TEST_OVERRIDE_SESSIONS_PER_WEEK = None
TEST_OVERRIDE_DISTRIBUTION_STRATEGY = None

def print_header(text: str, char: str = "="):
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)

def process_and_load_configurations():
    """
    PHASE 1: Loads configurations, calculates a PRELIMINARY time-based slide budget,
    and saves the result as 'processed_settings.json' for the Planning Agent.
    """
    print_header("Phase 1: Configuration and Scoping Process", char="-")

    # --- Load all input files ---
    logger.info("Loading all necessary configuration and data files...")
    try:
        os.makedirs(CONFIG_DIR, exist_ok=True)
        with open(PARSED_UO_JSON_PATH, 'r', encoding='utf-8') as f: unit_outline = json.load(f)
```

```
        with open(PRE_EXTRACTED_TOC_JSON_PATH, 'r', encoding='utf-8') as f:␣
↪book_toc = json.load(f)
        with open(SETTINGS_DECK_PATH, 'r', encoding='utf-8') as f:␣
↪settings_deck = json.load(f)
        with open(TEACHING_FLOWS_PATH, 'r', encoding='utf-8') as f:␣
↪teaching_flows = json.load(f)
        logger.info("All files loaded successfully.")
    except FileNotFoundError as e:
        logger.error(f"FATAL: A required configuration file was not found: {e}")
        return None

    # --- Pre-process and Refine Settings ---
    logger.info("Pre-processing settings_deck for definitive plan...")
    processed_settings = json.loads(json.dumps(settings_deck))

    unit_info = unit_outline.get("unitInformation", {})
    processed_settings['course_id'] = unit_info.get("unitCode",␣
↪"UNKNOWN_COURSE")
    processed_settings['unit_name'] = unit_info.get("unitName", "Unknown Unit␣
↪Name")

    # --- Apply test overrides IF they are not None ---
    logger.info("Applying overrides if specified...")
    # This block now correctly sets the teaching_flow_id based on the␣
↪interactive flag.
    if TEST_OVERRIDE_FLOW_ID is not None:
        processed_settings['teaching_flow_id'] = TEST_OVERRIDE_FLOW_ID
        logger.info(f"OVERRIDE: teaching_flow_id set to␣
↪'{TEST_OVERRIDE_FLOW_ID}'")
    else:
        # If no override, use the 'interactive' boolean from the file as the␣
↪source of truth.
        is_interactive = processed_settings.get('interactive', False)
        if is_interactive:
            processed_settings['teaching_flow_id'] = 'apply_topic_interactive'
        else:
            processed_settings['teaching_flow_id'] = 'standard_lecture'
        logger.info(f"Loaded from settings: 'interactive' is {is_interactive}.␣
↪Set teaching_flow_id to '{processed_settings['teaching_flow_id']}'.")

    # The 'interactive' flag is now always consistent with the teaching_flow_id.
    processed_settings['interactive'] = "interactive" in␣
↪processed_settings['teaching_flow_id'].lower()

    if TEST_OVERRIDE_SESSIONS_PER_WEEK is not None:
```

```
        processed_settings['week_session_setup']['sessions_per_week'] =␣
↪TEST_OVERRIDE_SESSIONS_PER_WEEK
        logger.info(f"OVERRIDE: sessions_per_week set to␣
↪{TEST_OVERRIDE_SESSIONS_PER_WEEK}")

    if TEST_OVERRIDE_DISTRIBUTION_STRATEGY is not None:
        processed_settings['week_session_setup']['distribution_strategy'] =␣
↪TEST_OVERRIDE_DISTRIBUTION_STRATEGY
        logger.info(f"OVERRIDE: distribution_strategy set to␣
↪'{TEST_OVERRIDE_DISTRIBUTION_STRATEGY}'")

    if TEST_OVERRIDE_WEEKS is not None:
        processed_settings['generation_scope']['weeks'] = TEST_OVERRIDE_WEEKS
        logger.info(f"OVERRIDE: generation_scope weeks set to␣
↪{TEST_OVERRIDE_WEEKS}")

    # --- DYNAMIC SLIDE BUDGET CALCULATION (Phase 1) ---
    logger.info("Calculating preliminary slide budget based on session time...")

    params = processed_settings.get('parameters_slides', {})
    SLIDES_PER_HOUR = params.get('slides_per_hour', 18)

    duration_hours = processed_settings['week_session_setup'].
↪get('session_time_duration_in_hour', 1.0)
    sessions_per_week = processed_settings['week_session_setup'].
↪get('sessions_per_week', 1)

    slides_content_per_session = int(duration_hours * SLIDES_PER_HOUR)
    target_total_slides = slides_content_per_session * sessions_per_week

    processed_settings['slide_count_strategy']['target_total_slides'] =␣
↪target_total_slides
    processed_settings['slide_count_strategy']['slides_content_per_session'] =␣
↪slides_content_per_session
    logger.info(f"Preliminary weekly content slide target calculated:␣
↪{target_total_slides} slides.")

    # --- Resolve Generation Scope if not overridden ---
    if TEST_OVERRIDE_WEEKS is None and processed_settings.
↪get('generation_scope', {}).get('weeks') == "all":
        num_weeks = len(unit_outline.get('weeklySchedule', []))
        processed_settings['generation_scope']['weeks'] = list(range(1,␣
↪num_weeks + 1))

    # --- Save the processed settings to disk ---
```

```
    logger.info(f"Saving preliminary processed configuration to:␣
 ↪{PROCESSED_SETTINGS_PATH}")
    with open(PROCESSED_SETTINGS_PATH, 'w', encoding='utf-8') as f:
        json.dump(processed_settings, f, indent=2)
    logger.info("File saved successfully.")

    # --- Assemble master config for optional preview ---
    master_config = {
        "processed_settings": processed_settings,
        "unit_outline": unit_outline,
        "book_toc": book_toc,
        "teaching_flows": teaching_flows
    }

    print_header("Phase 1 Configuration Complete", char="-")
    logger.info("Master configuration object is ready for the Planning Agent.")
    return master_config

# --- EXECUTE THE CONFIGURATION PROCESS ---
master_config = process_and_load_configurations()

# Optional: Print a preview to verify the output
if master_config:
    print("\n--- Preview of Processed Settings (Phase 1) ---")
    print(json.dumps(master_config['processed_settings'], indent=2,␣
 ↪sort_keys=True))
    if master_config.get('processed_settings', {}).get('generation_scope', {}).
 ↪get('weeks'):
        print(f"\nNumber of weeks to generate:␣
 ↪{len(master_config['processed_settings']['generation_scope']['weeks'])}")
    print("-----------------------------------------------")
```

```
2025-07-03 01:12:04,045 - INFO - Loading all necessary configuration and data
files…
2025-07-03 01:12:04,048 - INFO - All files loaded successfully.
2025-07-03 01:12:04,049 - INFO - Pre-processing settings_deck for definitive
plan…
2025-07-03 01:12:04,049 - INFO - Applying overrides if specified…
2025-07-03 01:12:04,049 - INFO - Loaded from settings: 'interactive' is True.
Set teaching_flow_id to 'apply_topic_interactive'.
2025-07-03 01:12:04,050 - INFO - Calculating preliminary slide budget based on
session time…
2025-07-03 01:12:04,050 - INFO - Preliminary weekly content slide target
calculated: 27 slides.
2025-07-03 01:12:04,051 - INFO - Saving preliminary processed configuration to:
/home/sebas_dev_linux/projects/course_generator/configs/processed_settings.json
2025-07-03 01:12:04,052 - INFO - File saved successfully.
```

```
2025-07-03 01:12:04,052 - INFO - Master configuration object is ready for the
Planning Agent.

--------------------------------------------------------------------------------
                    Phase 1: Configuration and Scoping Process
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
                          Phase 1 Configuration Complete
--------------------------------------------------------------------------------

--- Preview of Processed Settings (Phase 1) ---
{
  "course_id": "ICT312",
  "generation_scope": {
    "weeks": [
      7
    ]
  },
  "interactive": true,
  "interactive_deep": false,
  "parameters_slides": {
    "slides_per_hour": 18,
    "time_for_framework_slides_min": 6,
    "time_per_content_slides_min": 3,
    "time_per_interactive_slide_min": 5
  },
  "slide_count_strategy": {
    "interactive_slides_per_session": 0,
    "interactive_slides_per_week": 0,
    "method": "per_week",
    "slides_content_per_session": 27,
    "target_total_slides": 27,
    "total_slides_deck_week": 0,
    "total_slides_session": 0
  },
  "teaching_flow_id": "apply_topic_interactive",
  "unit_name": "Digital Forensic",
  "week_session_setup": {
    "distribution_strategy": "even",
    "interactive_time_in_hour": 0,
    "session_time_duration_in_hour": 1.5,
    "sessions_per_week": 1,
    "total_session_time_in_hours": 0
  }
}
```

```
Number of weeks to generate: 1
--------------------------------------------
```

[114]:
```python
# In Cell 9,

logger.info("--- Initializing Data-Driven Planning Agent Test ---")

if langchain_available:
    logger.info("Connecting to ChromaDB for the Planning Agent...")
    try:
        # 1. Connect to DB and Load all configurations
        vector_store = Chroma(
            persist_directory=CHROMA_PERSIST_DIR,
            embedding_function=OllamaEmbeddings(model=EMBEDDING_MODEL_OLLAMA),
            collection_name=CHROMA_COLLECTION_NAME
        )
        logger.info("Database connection successful.")

        logger.info("Loading configuration files for Planning Agent...")
        with open(os.path.join(CONFIG_DIR, "processed_settings.json"), 'r') as␣
 ↪f:
            processed_settings = json.load(f)
        with open(PRE_EXTRACTED_TOC_JSON_PATH, 'r') as f:
            book_toc = json.load(f)
        with open(PARSED_UO_JSON_PATH, 'r') as f:
            unit_outline = json.load(f)
        logger.info("Configuration files loaded.")

        master_config_from_file = {
            "processed_settings": processed_settings,
            "unit_outline": unit_outline,
            "book_toc": book_toc
        }

        # 2. Initialize the Planning Agent
        planning_agent = PlanningAgent(master_config_from_file,␣
 ↪vector_store=vector_store)

        # 3. CRITICAL: Loop through the weeks defined in the processed settings
        weeks_to_generate = processed_settings.get('generation_scope', {}).
 ↪get('weeks', [])
        logger.info(f"Found {len(weeks_to_generate)} week(s) to plan:␣
 ↪{weeks_to_generate}")

        for week_to_test in weeks_to_generate:
            logger.info(f"--> Generating draft plan for Week {week_to_test}")
```

```
            content_plan = planning_agent.
 ↪create_content_plan_for_week(week_to_test)

            if content_plan:
                print(f"\n--- Generated Draft Plan for Week {week_to_test} ---")
                print(json.dumps(content_plan, indent=2))

                # Save the generated plan to a file
                PLAN_OUTPUT_DIR = os.path.join(PROJECT_BASE_DIR,␣
 ↪"generated_plans")
                os.makedirs(PLAN_OUTPUT_DIR, exist_ok=True)
                plan_filename = f"{processed_settings.get('course_id',␣
 ↪'COURSE')}_Week{week_to_test}_plan_draft.json"
                plan_filepath = os.path.join(PLAN_OUTPUT_DIR, plan_filename)
                with open(plan_filepath, 'w') as f:
                    json.dump(content_plan, f, indent=2)
                logger.info(f"\nSuccessfully saved DRAFT content plan for Week␣
 ↪{week_to_test} to: {plan_filepath}")
            else:
                logger.error(f"Failed to generate content plan for Week␣
 ↪{week_to_test}.")

    except Exception as e:
        logger.error(f"An error occurred during the planning process: {e}",␣
 ↪exc_info=True)

else:
    logger.error("LangChain/Chroma libraries not found. Cannot run the Planning␣
 ↪Agent.")
```

```
2025-07-03 01:12:04,060 - INFO - --- Initializing Data-Driven Planning Agent
Test ---
2025-07-03 01:12:04,060 - INFO - Connecting to ChromaDB for the Planning
Agent…
2025-07-03 01:12:04,072 - INFO - Database connection successful.
2025-07-03 01:12:04,072 - INFO - Loading configuration files for Planning
Agent…
2025-07-03 01:12:04,073 - INFO - Configuration files loaded.
2025-07-03 01:12:04,074 - INFO - Data-Driven PlanningAgent initialized
successfully.
2025-07-03 01:12:04,075 - INFO - Found 1 week(s) to plan: [7]
2025-07-03 01:12:04,076 - INFO - --> Generating draft plan for Week 7
2025-07-03 01:12:04,128 - INFO - Partitioning strategy: Distributing 2 whole
chapters across 1 decks.
2025-07-03 01:12:04,129 - INFO - --- Planning Deck 1/1 | Topics: ['Chapter 7.
Linux and Macintosh File Systems', 'Chapter 8. Recovering Graphics Files'] |
Weight: 514 chunks | Slide Budget: 25 ---
```

```
2025-07-03 01:12:04,130 - INFO - Interactive mode ON. Deep interaction: False.
Adding placeholders…
2025-07-03 01:12:04,130 - INFO - Interactive mode ON. Deep interaction: False.
Adding placeholders…
2025-07-03 01:12:04,131 - INFO -
Successfully saved DRAFT content plan for Week 7 to: /home/sebas_dev_linux/proje
cts/course_generator/generated_plans/ICT312_Week7_plan_draft.json


********************************************************************************
                                Planning Week 7
********************************************************************************

--- Generated Draft Plan for Week 7 ---
{
  "week": 7,
  "overall_topic": "Linux Boot Processes and File Systems. Recovering Graphics
Files.",
  "deck_plans": [
    {
      "deck_number": 1,
      "deck_title": "Digital Forensic - Week 7, Lecture 1",
      "session_content": [
        {
          "title": "Chapter 7. Linux and Macintosh File Systems",
          "toc_id": 270,
          "chunk_count": 17,
          "total_chunks_in_branch": 274,
          "slides_allocated": 16,
          "children": [
            {
              "title": "Examining Linux File Structures",
              "toc_id": 272,
              "chunk_count": 77,
              "total_chunks_in_branch": 131,
              "slides_allocated": 5,
              "children": [
                {
                  "title": "File Structures in Ext4",
                  "toc_id": 273,
                  "chunk_count": 8,
                  "total_chunks_in_branch": 54,
                  "slides_allocated": 4,
                  "children": [
                    {
                      "title": "Inodes",
                      "toc_id": 274,
                      "chunk_count": 22,
```

```
                    "total_chunks_in_branch": 22,
                    "slides_allocated": 2,
                    "children": []
                  },
                  {
                    "title": "Hard Links and Symbolic Links",
                    "toc_id": 275,
                    "chunk_count": 24,
                    "total_chunks_in_branch": 24,
                    "slides_allocated": 2,
                    "children": []
                  }
                ]
              }
            ],
            "interactive_activity": {
              "title": "Examining Linux File Structures (Interactive
Activity)",
              "toc_id": 272,
              "slides_allocated": 1
            }
          },
          {
            "title": "Understanding Macintosh File Structures",
            "toc_id": 276,
            "chunk_count": 6,
            "total_chunks_in_branch": 58,
            "slides_allocated": 5,
            "children": [
              {
                "title": "An Overview of Mac File Structures",
                "toc_id": 277,
                "chunk_count": 23,
                "total_chunks_in_branch": 23,
                "slides_allocated": 2,
                "children": []
              },
              {
                "title": "Forensics Procedures in Mac",
                "toc_id": 278,
                "chunk_count": 18,
                "total_chunks_in_branch": 29,
                "slides_allocated": 2,
                "children": [
                  {
                    "title": "Acquisition Methods in macOS",
                    "toc_id": 279,
                    "chunk_count": 11,
```

```
                    "total_chunks_in_branch": 11,
                    "slides_allocated": 2,
                    "children": []
                  }
                ]
              }
            ],
            "interactive_activity": {
              "title": "Understanding Macintosh File Structures (Interactive
Activity)",
              "toc_id": 276,
              "slides_allocated": 1
            }
          },
          {
            "title": "Using Linux Forensics Tools",
            "toc_id": 280,
            "chunk_count": 5,
            "total_chunks_in_branch": 68,
            "slides_allocated": 6,
            "children": [
              {
                "title": "Installing Sleuth Kit and Autopsy",
                "toc_id": 281,
                "chunk_count": 21,
                "total_chunks_in_branch": 21,
                "slides_allocated": 2,
                "children": []
              },
              {
                "title": "Examining a Case with Sleuth Kit and Autopsy",
                "toc_id": 282,
                "chunk_count": 42,
                "total_chunks_in_branch": 42,
                "slides_allocated": 3,
                "children": []
              }
            ],
            "interactive_activity": {
              "title": "Using Linux Forensics Tools (Interactive Activity)",
              "toc_id": 280,
              "slides_allocated": 1
            }
          }
        ]
      },
      {
        "title": "Chapter 8. Recovering Graphics Files",
```

```
"toc_id": 289,
"chunk_count": 19,
"total_chunks_in_branch": 240,
"slides_allocated": 13,
"children": [
  {
    "title": "Recognizing a Graphics File",
    "toc_id": 291,
    "chunk_count": 4,
    "total_chunks_in_branch": 54,
    "slides_allocated": 4,
    "children": [
      {
        "title": "Understanding Bitmap and Raster Images",
        "toc_id": 292,
        "chunk_count": 13,
        "total_chunks_in_branch": 13,
        "slides_allocated": 1,
        "children": []
      },
      {
        "title": "Understanding Vector Graphics",
        "toc_id": 293,
        "chunk_count": 2,
        "total_chunks_in_branch": 2,
        "slides_allocated": 0,
        "children": []
      },
      {
        "title": "Understanding Metafile Graphics",
        "toc_id": 294,
        "chunk_count": 2,
        "total_chunks_in_branch": 2,
        "slides_allocated": 0,
        "children": []
      },
      {
        "title": "Understanding Graphics File Formats",
        "toc_id": 295,
        "chunk_count": 14,
        "total_chunks_in_branch": 14,
        "slides_allocated": 1,
        "children": []
      },
      {
        "title": "Understanding Digital Photograph File Formats",
        "toc_id": 296,
        "chunk_count": 2,
```

```
      "total_chunks_in_branch": 19,
      "slides_allocated": 1,
      "children": [
        {
          "title": "Examining the Raw File Format",
          "toc_id": 297,
          "chunk_count": 5,
          "total_chunks_in_branch": 5,
          "slides_allocated": 0,
          "children": []
        },
        {
          "title": "Examining the Exchangeable Image File Format",
          "toc_id": 298,
          "chunk_count": 12,
          "total_chunks_in_branch": 12,
          "slides_allocated": 1,
          "children": []
        }
      ]
    }
  ],
  "interactive_activity": {
    "title": "Recognizing a Graphics File (Interactive Activity)",
    "toc_id": 291,
    "slides_allocated": 1
  }
},
{
  "title": "Understanding Data Compression",
  "toc_id": 299,
  "chunk_count": 2,
  "total_chunks_in_branch": 101,
  "slides_allocated": 6,
  "children": [
    {
      "title": "Lossless and Lossy Compression",
      "toc_id": 300,
      "chunk_count": 8,
      "total_chunks_in_branch": 8,
      "slides_allocated": 1,
      "children": []
    },
    {
      "title": "Locating and Recovering Graphics Files",
      "toc_id": 301,
      "chunk_count": 6,
      "total_chunks_in_branch": 6,
```

```
                  "slides_allocated": 0,
                  "children": []
                },
                {
                  "title": "Identifying Graphics File Fragments",
                  "toc_id": 302,
                  "chunk_count": 3,
                  "total_chunks_in_branch": 3,
                  "slides_allocated": 0,
                  "children": []
                },
                {
                  "title": "Repairing Damaged Headers",
                  "toc_id": 303,
                  "chunk_count": 6,
                  "total_chunks_in_branch": 6,
                  "slides_allocated": 0,
                  "children": []
                },
                {
                  "title": "Searching for and Carving Data from Unallocated
Space",
                  "toc_id": 304,
                  "chunk_count": 9,
                  "total_chunks_in_branch": 39,
                  "slides_allocated": 2,
                  "children": [
                    {
                      "title": "Planning Your Examination",
                      "toc_id": 305,
                      "chunk_count": 4,
                      "total_chunks_in_branch": 4,
                      "slides_allocated": 0,
                      "children": []
                    },
                    {
                      "title": "Searching for and Recovering Digital Photograph
Evidence",
                      "toc_id": 306,
                      "chunk_count": 26,
                      "total_chunks_in_branch": 26,
                      "slides_allocated": 2,
                      "children": []
                    }
                  ]
                },
                {
                  "title": "Rebuilding File Headers",
```

```
                  "toc_id": 307,
                  "chunk_count": 22,
                  "total_chunks_in_branch": 22,
                  "slides_allocated": 1,
                  "children": []
                },
                {
                  "title": "Reconstructing File Fragments",
                  "toc_id": 308,
                  "chunk_count": 15,
                  "total_chunks_in_branch": 15,
                  "slides_allocated": 1,
                  "children": []
                }
              ],
              "interactive_activity": {
                "title": "Understanding Data Compression (Interactive
Activity)",
                "toc_id": 299,
                "slides_allocated": 1
              }
            },
            {
              "title": "Identifying Unknown File Formats",
              "toc_id": 309,
              "chunk_count": 14,
              "total_chunks_in_branch": 47,
              "slides_allocated": 2,
              "children": [
                {
                  "title": "Analyzing Graphics File Headers",
                  "toc_id": 310,
                  "chunk_count": 5,
                  "total_chunks_in_branch": 5,
                  "slides_allocated": 0,
                  "children": []
                },
                {
                  "title": "Tools for Viewing Images",
                  "toc_id": 311,
                  "chunk_count": 5,
                  "total_chunks_in_branch": 5,
                  "slides_allocated": 0,
                  "children": []
                },
                {
                  "title": "Understanding Steganography in Graphics Files",
                  "toc_id": 312,
```

```
                    "chunk_count": 16,
                    "total_chunks_in_branch": 16,
                    "slides_allocated": 1,
                    "children": []
                  },
                  {
                    "title": "Using Steganalysis Tools",
                    "toc_id": 313,
                    "chunk_count": 7,
                    "total_chunks_in_branch": 7,
                    "slides_allocated": 0,
                    "children": []
                  }
                ],
                "interactive_activity": {
                  "title": "Identifying Unknown File Formats (Interactive
Activity)",
                  "toc_id": 309,
                  "slides_allocated": 1
                }
              },
              {
                "title": "Understanding Copyright Issues with Graphics",
                "toc_id": 314,
                "chunk_count": 19,
                "total_chunks_in_branch": 19,
                "slides_allocated": 1,
                "children": []
              }
            ]
          }
        ]
      }
    ]
}
```


[115]:
```
# Cell 10: Orchestrator for Finalizing Plan and Calculating Time/Budget (Final␣
↪Corrected Schema)

import os
import json
import logging
import math

# --- Setup and Logging ---
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
↪%(message)s')
```

```
logger = logging.getLogger(__name__)

# --- Helper Functions ---
def print_header(text: str, char: str = "="):
    """Prints a centered header to the console."""
    print("\n" + char * 80)
    print(text.center(80))
    print(char * 80)

def analyze_plan_and_finalize_settings(draft_plan: Dict, initial_settings:␣
 ↪Dict) -> Dict:
    """
    Analyzes a draft plan to count slides, calculates the final time budget per␣
 ↪your
    detailed schema, and populates the settings object.
    """
    print_header("Phase 2: Analyzing Plan and Finalizing Budget", char="-")

    final_settings = json.loads(json.dumps(initial_settings))
    params = final_settings.get('parameters_slides', {})

    # Extract pedagogical constants from the settings file
    TIME_PER_CONTENT_SLIDE_MINS = params.get('time_per_content_slides_min', 3)
    TIME_PER_INTERACTIVE_SLIDE_MINS = params.
 ↪get('time_per_interactive_slide_min', 5)
    TIME_FOR_FRAMEWORK_SLIDES_MINS = params.
 ↪get('time_for_framework_slides_min', 6)
    FRAMEWORK_SLIDES_PER_DECK = 4 # Fixed number for Title, Agenda, Summary, End
    MINS_PER_HOUR = 60

    # --- 1. Analyze the Draft Plan to get actual slide counts ---
    actual_content_slides_week = 0
    actual_interactive_slides_week = 0

    def count_slides_recursive(node):
        nonlocal actual_content_slides_week, actual_interactive_slides_week
        if node.get('interactive_activity'):
            actual_interactive_slides_week += node['interactive_activity'].
 ↪get('slides_allocated', 0)

        if not node.get('children'):
            actual_content_slides_week += node.get('slides_allocated', 0)
        else:
            for child in node.get('children', []):
                count_slides_recursive(child)

    num_decks = len(draft_plan.get('deck_plans', []))
```

```
    for deck in draft_plan.get('deck_plans', []):
        for content_tree in deck.get('session_content', []):
            count_slides_recursive(content_tree)

    # --- 2. Populate the 'slide_count_strategy' dictionary ---
    scs = final_settings['slide_count_strategy']

    # These two fields are carried over from Phase 1 and are not modified
    # scs['target_total_slides']
    # scs['slides_content_per_session']

    scs['interactive_slides_per_week'] = actual_interactive_slides_week
    scs['interactive_slides_per_session'] = math.
↪ceil(actual_interactive_slides_week / num_decks) if num_decks > 0 else 0

    # Correct the typo and use the corrected calculation logic
    if 'Tota_slides_session' in scs:
        del scs['Tota_slides_session'] # Delete the typo if it exists
    scs['total_slides_session'] = scs['slides_content_per_session'] +␣
↪scs['interactive_slides_per_session'] + FRAMEWORK_SLIDES_PER_DECK
    scs['total_slides_deck_week'] = scs['target_total_slides'] +␣
↪scs['interactive_slides_per_week'] + (FRAMEWORK_SLIDES_PER_DECK * num_decks)

    # --- 3. Populate the 'week_session_setup' dictionary using PER-SESSION␣
↪logic ---
    wss = final_settings['week_session_setup']

    # Calculate per-session time components in minutes
    content_time_mins_per_session = scs['slides_content_per_session'] *␣
↪TIME_PER_CONTENT_SLIDE_MINS
    interactive_time_mins_per_session = scs['interactive_slides_per_session'] *␣
↪TIME_PER_INTERACTIVE_SLIDE_MINS

    # Update the dictionary with values in hours
    wss['interactive_time_in_hour'] = round(interactive_time_mins_per_session /␣
↪MINS_PER_HOUR, 2)

    # Calculate total time for a single session
    total_time_mins_per_session = content_time_mins_per_session +␣
↪interactive_time_mins_per_session + TIME_FOR_FRAMEWORK_SLIDES_MINS
    wss['total_session_time_in_hours'] = round(total_time_mins_per_session /␣
↪MINS_PER_HOUR, 2)

    logger.info(f"Analysis Complete: Total Content Slides:␣
↪{actual_content_slides_week}, Total Interactive Slides:␣
↪{actual_interactive_slides_week}")
```

```
    logger.info(f"PER SESSION Calculation:␣
↪Content({content_time_mins_per_session}m) +␣
↪Interactive({interactive_time_mins_per_session}m) +␣
↪Framework({TIME_FOR_FRAMEWORK_SLIDES_MINS}m) =␣
↪{total_time_mins_per_session}m")
    logger.info(f"Final Estimated Delivery Time PER SESSION:␣
↪{wss['total_session_time_in_hours']} hours")

    return final_settings

# --- Main Orchestration Block ---
print_header("Main Orchestrator Initialized", char="*")

try:
    # 1. Load the DRAFT plan and PRELIMINARY settings
    logger.info("Loading draft plan and preliminary configurations...")

    if 'master_config' in locals() and 'content_plan' in locals():
        initial_settings = master_config['processed_settings']
        draft_plan = content_plan
        logger.info("Loaded draft plan and settings from previous cell's memory.
↪")
    else:
        # Fallback to loading from files
        weeks_to_generate = initial_settings.get('generation_scope', {}).
↪get('weeks', [])
        if not weeks_to_generate: raise ValueError("No weeks to generate found␣
↪in settings.")
        week_to_load = weeks_to_generate[0]
        logger.info(f"Loading from files for Week {week_to_load}...")
        with open(PROCESSED_SETTINGS_PATH, 'r') as f: initial_settings = json.
↪load(f)
        plan_filename = f"{initial_settings.get('course_id',␣
↪'COURSE')}_Week{week_to_load}_plan_draft.json"
        plan_filepath = os.path.join(PROJECT_BASE_DIR, "generated_plans",␣
↪plan_filename)
        with open(plan_filepath, 'r') as f: draft_plan = json.load(f)

    # 2. PHASE 2: Analyze the plan and finalize the settings
    finalized_settings = analyze_plan_and_finalize_settings(draft_plan,␣
↪initial_settings)

    # 3. Save the FINAL, enriched settings to disk
    final_settings_path = os.path.join(CONFIG_DIR, "final_processed_settings.
↪json")
    logger.info(f"Saving finalized settings to {final_settings_path}")
```

```
    with open(final_settings_path, 'w', encoding='utf-8') as f:
        json.dump(finalized_settings, f, indent=2)
    logger.info("Finalized settings saved. Ready for Content Generation stage.")

    print("\n--- Finalized Processed Settings ---")
    print(json.dumps(finalized_settings, indent=2))

except Exception as e:
    logger.error(f"An unexpected error occurred: {e}", exc_info=True)
```

```
2025-07-03 01:12:04,144 - INFO - Loading draft plan and preliminary
configurations…
2025-07-03 01:12:04,144 - INFO - Loaded draft plan and settings from previous
cell's memory.
2025-07-03 01:12:04,145 - INFO - Analysis Complete: Total Content Slides: 23,
Total Interactive Slides: 6
2025-07-03 01:12:04,145 - INFO - PER SESSION Calculation: Content(81m) +
Interactive(30m) + Framework(6m) = 117m
2025-07-03 01:12:04,146 - INFO - Final Estimated Delivery Time PER SESSION: 1.95
hours
2025-07-03 01:12:04,146 - INFO - Saving finalized settings to /home/sebas_dev_li
nux/projects/course_generator/configs/final_processed_settings.json
2025-07-03 01:12:04,147 - INFO - Finalized settings saved. Ready for Content
Generation stage.


********************************************************************************
                          Main Orchestrator Initialized
********************************************************************************

--------------------------------------------------------------------------------
                 Phase 2: Analyzing Plan and Finalizing Budget
--------------------------------------------------------------------------------

--- Finalized Processed Settings ---
{
  "course_id": "ICT312",
  "unit_name": "Digital Forensic",
  "interactive": true,
  "interactive_deep": false,
  "teaching_flow_id": "apply_topic_interactive",
  "parameters_slides": {
    "slides_per_hour": 18,
    "time_per_content_slides_min": 3,
    "time_per_interactive_slide_min": 5,
    "time_for_framework_slides_min": 6
  },
  "week_session_setup": {
```

```
    "sessions_per_week": 1,
    "distribution_strategy": "even",
    "session_time_duration_in_hour": 1.5,
    "interactive_time_in_hour": 0.5,
    "total_session_time_in_hours": 1.95
  },
  "slide_count_strategy": {
    "method": "per_week",
    "target_total_slides": 27,
    "slides_content_per_session": 27,
    "interactive_slides_per_week": 6,
    "interactive_slides_per_session": 6,
    "total_slides_deck_week": 37,
    "total_slides_session": 37
  },
  "generation_scope": {
    "weeks": [
      7
    ]
  }
}
```


7 Next steps (if yo are a llm ignore this section they are my notes)

Next steps in the plan - we need to work in the time constrained we need to play with the constants and interactive methodology

Global varaibles

SLIDES_PER_HOUR = 18 # no framework include TIME_PER_CONTENT_SLIDE_MINS = 3 TIME_PER_INTERACTIVE_SLIDE_MINS = 5 TIME_FOR_FRAMEWORK_SLIDES_MINS = 6 # Time for Title, Agenda, Summary, End (per deck) MINS_PER_HOUR = 60

{ “course_id”: ““,”unit_name”: ““,”interactive”: true, “interactive_deep”: false, “slide_count_strategy”: { “method”: “per_week”, “interactive_slides_per_week”: 0 – > sum all interactive counts “interactive_slides_per_session”: 0, – > Total # of slides produced if “interactive” is true other wise remains 0 “target_total_slides”: 0, –> Total Content Slides per week that cover the total - will be the target in the cell 7
“slides_content_per_session”: 0, –> Total # (target_total_slides/sessions_per_week) “total_slides_deck_week”: 0, –> target_total_slides + interactive_slides_per_week + (framework (4 + Time for Title, Agenda, Summary, End) * sessions_per_week) “Tota_slides_session”: 0 –> content_slides_per_session + interactive_slides_per_session + framework (4 + Time for Title, Agenda, Summary, End) }, “week_session_setup”: { “sessions_per_week”: 1, “distribution_strategy”: “even”, “interactive_time_in_hour”: 0, –> find the value in ahours of the total # (“interactive_slides” * “TIME_PER_INTERACTIVE_SLIDE_MINS”)/60
“total_session_time_in_hours”: 0 –> this is going to be egual or similar to session_time_duration_in_hour if “interactive” is false obvisuly base on the global varaibles it will

be the calculation of “interactive_time_in_hour” “session_time_duration_in_hour”: 2, — > this is the time that the costumer need for delivery this is a constrain is not modified never is used for reference },

“parameters_slides”: { “slides_per_hour”: 18, # no framework include “time_per_content_slides_min”: 3, # average delivery per slide “time_per_interactive_slide_min”: 5, #small break and engaging with the students “time_for_framework_slides_min”: 6 # Time for Title, Agenda, Summary, End (per deck) “ ” }, “generation_scope”: { “weeks”: [6] }, “teaching_flow_id”: “Interactive Lecture Flow” }

“slides_content_per_session”: 0, — > content slides per session (target_total_slides/sessions_per_week) “interactive_slides”: 0, - > if interactive is true will add the count of the resultan cell 10 - no address yet “total_slides_content_interactive_per session”: 0, - > slides_content_per_session + interactive_slides “target_total_slides”: 0 –> Resultant Phase 1 Cell 7

- Add the sorted chunks for each slide to process the summaries or content geneneration later
- Add title, agenda, summary and end as part of this planning to start having
- Add label to reference title, agenda, content, summary and end
- Process the images from the book and store them with relation to the chunk so we can potentially use the image in the slides
- Process unit outlines and store them with good labels for phase 1

Next steps

Chunnk relation wwith the weights of the number of the slides per subtopic, haave in mind that 1 hour of delivery is like 20-25 slides

to ensure to move to the case to handle i wourl like to ensure the concepts are clear when we discussde about sessions and week, sessions in this context is number of classes that we have for week, if we say week , 3 sessions in one week or sessions_per_week = 3 is 3 classes per week that require 3 different set of

https://youtu.be/6xcCwlDx6f8?si=7QxFyzuNVppHBQ-c