**Project Report: Semantic Spotter**

**Executive Summary**

This report documents the Semantic Spotter project — a document-aware question-and-answer system designed to extract facts and answer queries from insurance PDF documents. It describes objectives, stepwise design, implementation details, verification approach, challenges and corrective actions, and lessons learned. The report is presented as a clear, step-by-step account suitable for technical handover and operational planning.

* **Objectives**
* Primary objective
* Build a dependable workflow to ingest insurance policy PDFs and answer user questions with accurate, source-referenced responses.
* Secondary objectives
* Provide page-level source references for each answer.
* Verify response quality via automated assessors focused on relevance, accuracy, and fidelity.
* Produce a modular design that can be reused for other document types.
* **Scope**
  + Inputs: Digital PDF insurance documents stored on an attached drive.
  + Outputs: Natural-language responses, source file name, and page labels for provenance.
  + Exclusions: Scanned-only PDFs without OCR processing; production scaling and access control (covered in recommendations).
* **High-level System Architecture**
  + The system is organized into discrete layers. Each layer has distinct responsibilities and well-defined interfaces.
  + Layers (overview)
    - Data Layer — storage of source files and metadata.
    - Ingestion Layer — PDF parsing and document object creation.
    - Segmentation Layer — break text into manageable segments with overlap.
    - Representation & Index Layer — create vector-style representations and store them in a similarity index.
    - Retrieval Layer — retrieve best-matching segments for queries.
    - Response Generation Layer — synthesize final answers conditioned on retrieved segments using an external response service.
    - Verification Layer — automated assessors to measure relevance, accuracy, and fidelity.
    - Presentation Layer — interactive user interface and batch test reporting.
* **Detailed Layer Descriptions**
  + **Data Layer**
    - Components: Remote drive mount, file catalog.
    - Metadata captured: file\_name, page\_label, segment\_id.
    - Notes: Maintain consistent naming and timestamps for traceability.
  + **Ingestion Layer**
    - Components: PDF loader that extracts text and per-page metadata.
    - Behaviour: Preserve page boundaries; attach page label metadata to extracted text.
  + **Segmentation Layer**
    - Components: Sentence-based segmenter with configurable segment\_size and overlap.
    - Purpose: Produce semantically coherent segments to improve retrieval quality and limit context length for the response service.
  + **Representation & Index Layer**
    - Components: Service that converts text segments to semantic vectors and a similarity index to store them.
    - Purpose: Enable nearest-neighbor style retrieval across a document corpus.
    - Persistence: Option to store index on disk or in a persistent vector database for scale.
  + **Retrieval Layer**
    - Components: Search interface that queries the similarity index and returns top-k segments and their metadata.
    - Behaviour: Return both text and provenance metadata to downstream components.
  + **Response Generation Layer**
    - Components: External text response service that accepts a user query plus retrieved supporting segments and returns a synthesized answer.
    - Requirements: Deterministic configuration for reproducibility (zero randomness), token/length limits, and maximum context controls.
  + **Verification Layer**
    - Components: Three automated assessors
      * Relevance Assessor: Checks topical match between question and answer.
      * Accuracy Assessor: Compares answer against reference text extracts.
      * Fidelity Assessor: Ensures claims in the answer are grounded in returned segments.
    - Inputs: Question, generated answer, retrieved segments, and selected reference passages.
    - Outputs: Scores, pass/fail decisions, and textual feedback.
  + **Presentation Layer**
    - Components: Interactive session loop for manual queries and a batch testing function that returns tabular results (pandas DataFrame).
    - Output formats: Terminal prints and HTML-formatted display when run in notebook environments.
* **Implementation Details**
* **Setup**
  + Install required libraries for PDF parsing, index management, the external response service connector, and tabular output utilities.
  + Mount the remote drive to access files.
  + Securely read service credentials from a protected source and set environment variables.
* **Ingestion**
  + Use the PDF loader to extract text and page-level metadata for each document.
  + Validate successful extraction by checking non-empty text and page count.
* **Segmentation**
  + Configure the segmenter to use segment\_size=512 and segment\_overlap=20 (tunable).
  + Run segmentation to produce segment objects with text and metadata.
* **Representations and Index**
  + Send segment text to the representation service to obtain vector-style representations.
  + Build a similarity index from representations; persist index for reuse.
* **Retrieval and Answering**
  + Create a query flow: accept user text → query index for top-k segments → assemble context payload for response service.
  + Configure response service: deterministic behavior, max length, and verbosity limits.
  + Receive synthesized answer and attach file\_name and page\_label from the top provenance segment(s).
* **Verification and Testing**
  + Generate test questions from document segments using a deterministic generator.
  + For each question:
  + Retrieve answer with provenance.
  + Run assessors and collect scores and feedback.
  + Aggregate results into a table for analysis.
* **Data Flow (step-by-step)**
  + User or test harness issues a question.
  + Retrieval Layer queries the similarity index with the question.
  + Top-k segments and metadata are returned.
  + Response Generation Layer constructs a context payload of segments + question and calls the external response service.
  + Service returns an answer string.
  + Verification Layer runs three assessors on the answer and supporting segments.
* **Testing and Verification Plan**
  + Manual tests: Interactive session to confirm correctness on common queries (company name, addresses, page numbers).
  + Batch tests: Predefined set of questions processed with testing\_pipeline to produce a DataFrame: Question, Answer, Page.
  + Automated assessors: Use the generated test questions and store per-question metrics: relevance score, accuracy score, fidelity score, pass flags, and assessor notes.
  + Acceptance criteria: predefined score thresholds (example: relevance >= 0.7, accuracy >= 0.75, fidelity pass).
* **Challenges and Mitigations**
  + Text extraction inconsistency
    - Problem: PDFs vary widely (text vs scanned).
    - Fix: Add an OCR step for scanned pages (Tesseract or commercial OCR). Validate counts and sample outputs.
  + Context-length limits for response service
    - Problem: Documents exceed service context window.
    - Fix: Use segmentation and retrieval to limit supporting text. Adjust top-k and overlap to balance recall and context size.
  + Unsupported claims in responses
    - Problem: Response may contain statements not present in source segments.
    - Fix: Run fidelity assessment; always attach provenance and present answer as "supported by" followed by source snippets.
  + Cost and rate constraints
    - Problem: External service and representation calls incur cost and rate limits.
    - Fix: Cache representations, reuse indices, batch representation requests, and implement retry/backoff.
  + Reference selection for accuracy checks
    - Problem: Choosing a correct reference snippet is nontrivial.
    - Fix: Use segment-level references and curate a small manual ground-truth set for critical queries.
* **Conclusion**
  + Semantic Spotter provides a reproducible, layered approach to document question answering with provenance and automated verification. The system is designed for modular enhancements: ingestion, retrieval, response generation, and verification can each be improved independently to meet production needs.