DALHOUSIE
UNIVERSITY

CSCI 5411

Advanced Cloud Architecting

Prof. Lu Yang

Mid – Term Project

Milestone – 01

Banner ID:	B00988337
Name:	Patel Het Ghanshyambhai
Date:	18th May, 2025

Table of Contents –

01.	Domain Selection and Justification	03.
02.	Problem Statement and Business Case	04.
03.	Detailed Functional Requirements	05.
04.	Non-Functional Requirements	07.
05.	Initial Architecture Design	09.
06.	Data Sequence Diagram	10.
07.	AWS Services and Tech Stack	11.
08.	Potential Architectural Challenges	13.
09.	Initial Cost Estimation	14.
10.	References	15.

1. <u>Domain Selection and Justification</u> –

Domain Selection:

ScholarSight operates in the Academic Research Assistance domain, specifically implementing Retrieval-Augmented Generation (RAG) for research paper understanding and information extraction. This domain represents a strategic intersection of natural language processing, information retrieval, and knowledge management applied to academic research workflows.

Justification:

Retrieval-Augmented Generation (RAG) represents a significant advancement in AI/GenAI by combining the strengths of retrieval-based systems with generative AI. This approach is particularly well-suited for academic research assistance because:

1. **Knowledge Grounding**: RAG ensures responses are grounded in specific source documents, essential for academic integrity.
2. **Citation Capability**: The retrieval component enables precise source attribution with page numbers and paragraphs.
3. **Reduced Hallucination**: By retrieving relevant context before generation, RAG significantly reduces AI hallucination.
4. **Efficiency**: RAG enables processing of large documents without requiring the entire content to fit in context windows.
5. **Adaptability**: The system can work with any research paper without requiring domain-specific training.

This domain leverages cutting-edge AI/GenAI techniques while addressing a practical need in academic and professional research environments.

2. <u>Problem Statement and Business Case</u> –

Problem Statement:

Researchers, students, and professionals face significant challenges when dealing with lengthy research papers:

1. **Time Consumption**: Reading a 100-page research paper can take 3-5 hours.
2. **Information Extraction**: Finding specific information requires extensive scanning and searching.
3. **Context Understanding**: Comprehending complex research concepts often requires additional context.
4. **Knowledge Synthesis**: Extracting key insights efficiently is difficult without reading the entire paper.
5. **Source Attribution**: Properly citing information from papers is time-consuming and error-prone.

These challenges lead to reduced research productivity, information overload, and potential missed insights when important sections are overlooked.

Business Case:

Target Users:

- Graduate students conducting literature reviews
- Researchers staying current with publications in their field
- Professionals needing to extract insights from technical papers
- Academics preparing for conferences or peer reviews
- Students learning complex technical subjects

Value Proposition:

- Reduce research paper review time by 70-80%
- Extract precise information with accurate source attribution
- Improve research efficiency and knowledge acquisition
- Enable more comprehensive literature reviews in less time
- Maintain academic integrity through proper citation

Market Need:
The volume of academic research continues to grow exponentially, with over 2 million new research papers published annually. According to surveys, researchers spend approximately 15-20 hours per week reading papers, yet still feel unable to keep up with relevant publications. A tool that streamlines this process addresses a critical pain point in academic and professional research workflows.

Differentiation:
Unlike general-purpose search engines or AI assistants, ScholarSight:
- Focuses specifically on research paper content
- Provides contextually accurate answers from the uploaded document
- Includes proper citations to specific sections of the paper
- Understands academic terminology and research methodologies
- Maintains source attribution and academic integrity

3. <u>Detailed Functional Requirements</u> –

3.1 Document Processing
- The system shall accept PDF uploads of research papers up to 100 pages in length.
- The system shall extract and process text from PDF documents while preserving page and paragraph information.
- The system shall handle various academic paper formats and layouts.
- The system shall split papers into appropriate chunks for retrieval while maintaining source traceability.
- The system shall generate and store vector embeddings for all paper chunks.
- The system shall maintain metadata about paper sections, page numbers, and paragraph identifiers.

3.2 Query Processing
- The system shall accept natural language questions about the uploaded paper.

- The system shall convert user queries into vector embeddings for similarity search.
- The system shall identify the specific aspects of the paper being questioned.
- The system shall support queries about methodologies, findings, conclusions, and references.
- The system shall handle academic terminology and research concepts in queries.

3.3 Retrieval System

- The system shall retrieve the most relevant paper sections based on query similarity.
- The system shall rank retrieved chunks by relevance to the query.
- The system shall include context from surrounding sections when necessary.
- The system shall retrieve information about figures, tables, and citations when relevant.
- The system shall identify and retrieve mathematical formulas and technical details when needed.

3.4 Answer Generation

- The system shall generate natural language answers based on retrieved paper sections.
- The system shall provide specific page and paragraph references for information sources.
- The system shall format mathematical formulas and technical content appropriately.
- The system shall explain complex research concepts in clear language.
- The system shall generate responses that accurately reflect the paper's content.

3.5 Export Functionality

- The system shall allow users to copy generated answers to clipboard.

- The system shall enable downloading responses in DOCX format.
- The system shall include proper formatting in exported documents.
- The system shall maintain citations and references in exported content.
- The system shall include metadata about the source paper in exports.

3.6 User Interface

- The system shall provide an intuitive interface for paper uploads.
- The system shall display a simple query input field for questions.
- The system shall present answers with proper formatting and citations.
- The system shall show source citations in an organized manner.
- The system shall include options for copying and downloading responses.

4. <u>Non-Functional Requirements</u> –

- *Security*

The system will implement HTTPS for all communications to ensure data integrity and confidentiality. It will enforce IAM roles with the principle of least privilege and sanitize user inputs to prevent injection attacks. Uploaded papers and generated embeddings will be encrypted to safeguard sensitive information, while temporary storage will be utilized to avoid retaining papers longer than necessary. API endpoints will be secured using proper authentication mechanisms to restrict unauthorized access.

- *Scalability*

To accommodate varying workloads, the system will support processing papers up to 100 pages and manage at least three simultaneous users effectively. It will implement queuing mechanisms to handle processing demands without compromising system performance.

Efficient memory management will be prioritized to maintain optimal response times irrespective of document size.

- *Performance*

The system is designed to process 100-page documents within two minutes while returning query responses within eight seconds. It will optimize text extraction for accuracy, implement vector similarity search for faster retrieval, and reduce latency in model inference to enhance overall responsiveness.

- *Reliability*

To maintain reliability, the system will handle diverse PDF formats and layouts, ensuring consistent processing results. It will implement graceful degradation strategies for large documents and provide clear error messages for processing issues. Robust error handling will be integrated to recover from failures without data loss, maintaining data integrity throughout operations.

- *Maintainability*

A modular architecture will be employed to facilitate easy component updates and system modifications. Comprehensive logging will be implemented to aid in troubleshooting and error tracking. All code and configurations will be maintained under version control, and thorough documentation will be provided for components and APIs to streamline future updates and maintenance activities.

- *Usability*

The user interface will be designed for simplicity and intuitiveness, ensuring a seamless user experience. The system will generate concise, well-formatted answers and provide real-time feedback during paper processing. Natural language queries will be supported without requiring specific syntax, enabling users to interact with the system effortlessly.

5. Initial Architecture Design –

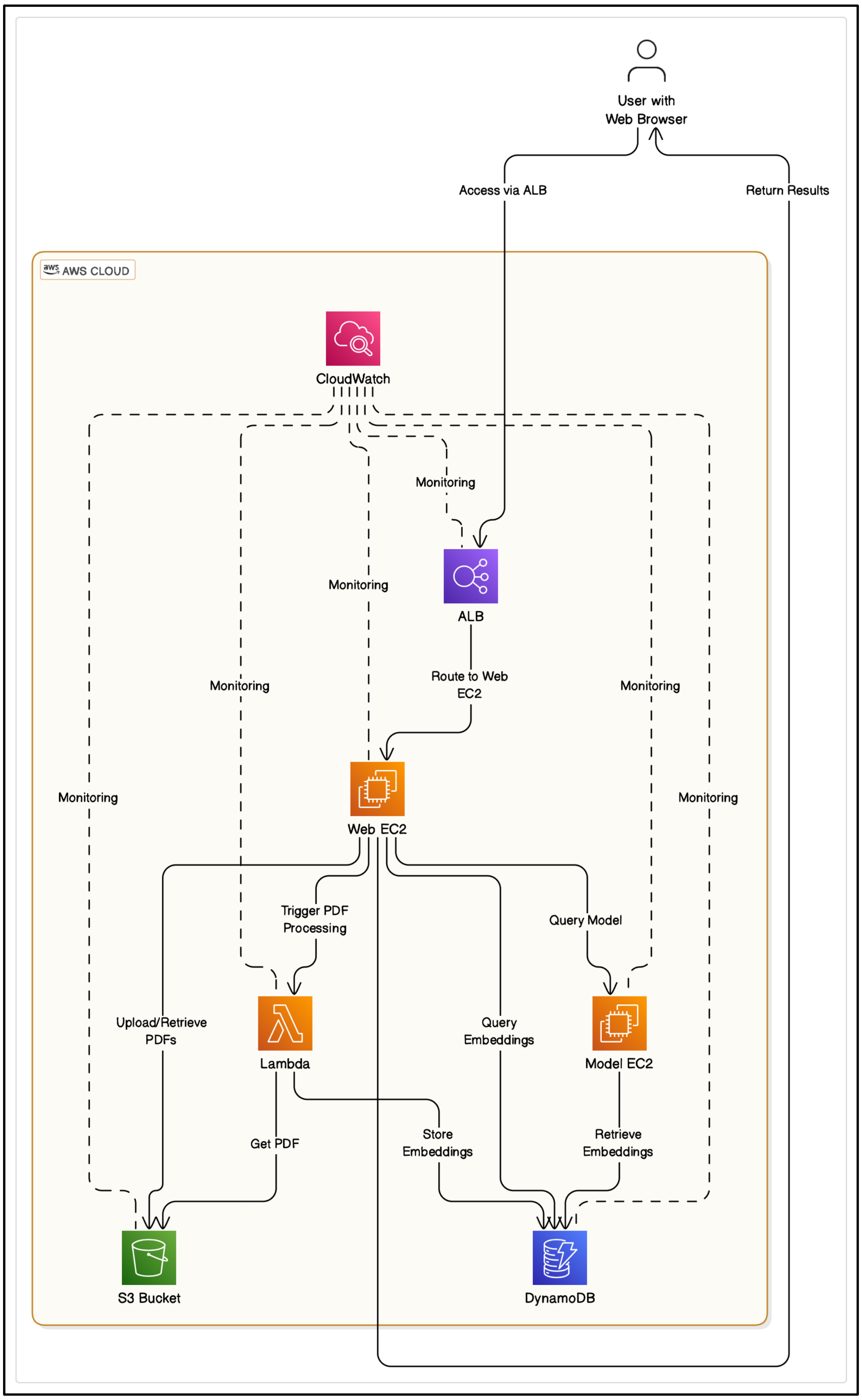

Fig. 1. Initial Architecture Diagram for ScholarSight.

6. Data Sequence Diagram – (UML Diagram)

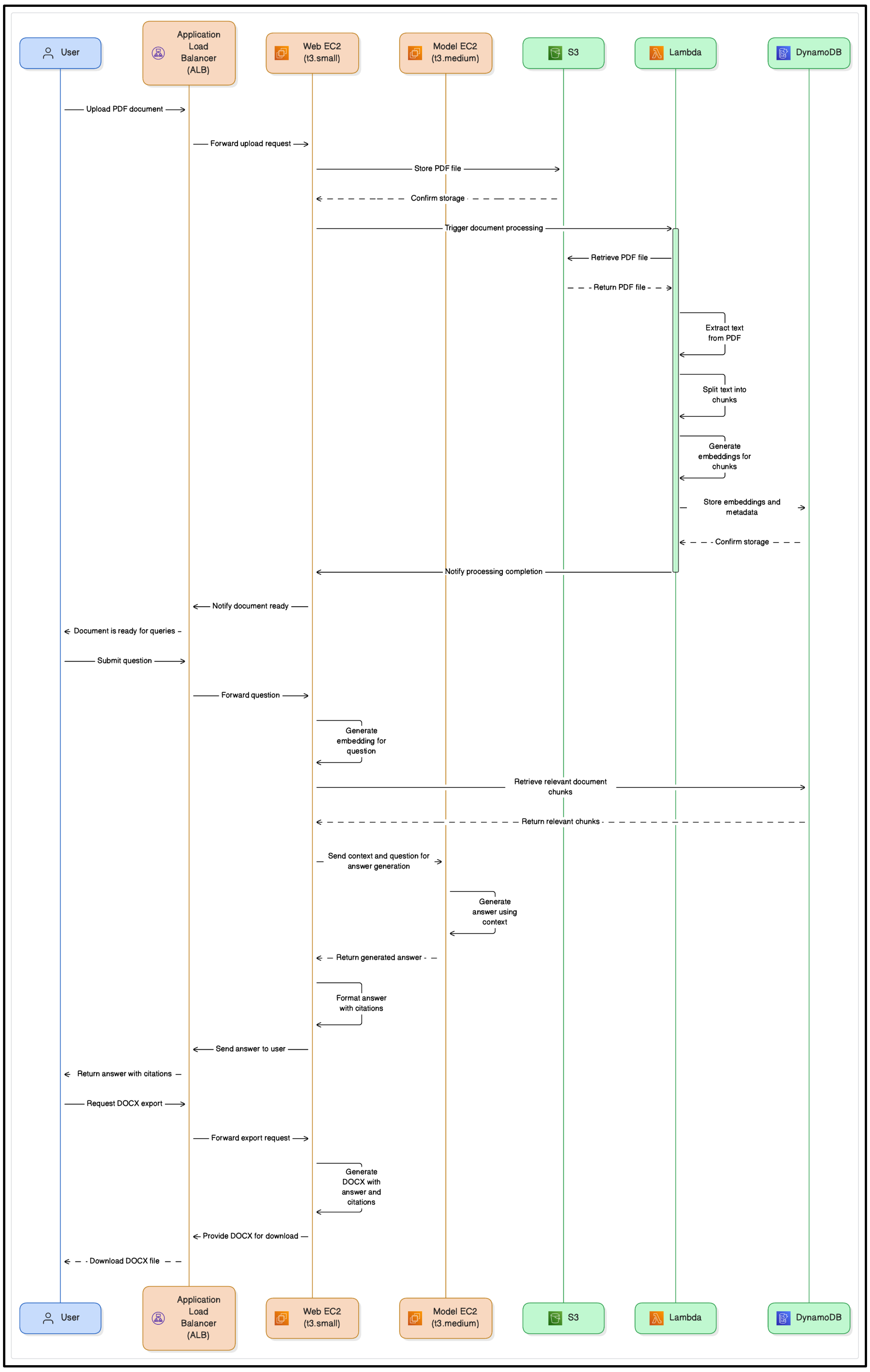
User
Application Load Balancer (ALB)
Web EC2 (t3.small)
Model EC2 (t3.medium)
S3
Lambda
DynamoDB
Upload PDF document
Forward upload request
Store PDF file
Confirm storage
Trigger document processing
Retrieve PDF file
Return PDF file
Extract text from PDF
Split text into chunks
Generate embeddings for chunks
Store embeddings and metadata
Confirm storage
Notify processing completion
Notify document ready
Document is ready for queries
Submit question
Forward question
Generate embedding for question
Retrieve relevant document chunks
Return relevant chunks
Send context and question for answer generation
Generate answer using context
Return generated answer
Format answer with citations
Send answer to user
Return answer with citations
Request DOCX export
Forward export request
Generate DOCX with answer and citations
Provide DOCX for download
Download DOCX file
User
Application Load Balancer (ALB)
Web EC2 (t3.small)
Model EC2 (t3.medium)
S3
Lambda
DynamoDB

Fig. 2. Data Sequence Diagram (UML) for ScholarSight.

7. <u>AWS Services and Tech Stack</u> –

AWS Services:

1. **Amazon EC2 (t3.small)**

- Purpose: Hosts the web application and API layer.
- Configuration: t3.small instance with auto-recovery.
- Justification: Provides sufficient compute resources for the application while remaining cost-effective.

2. **Amazon S3**

- Purpose: Temporary storage for uploaded documents.
- Configuration: Standard storage with lifecycle policies.
- Justification: Secure, scalable storage with built-in encryption and lifecycle management.

3. **AWS Lambda**

- Purpose: Asynchronous document processing.
- Configuration: Python runtime with up to 1GB memory.
- Justification: Serverless architecture for cost-effective, scalable document processing.

4. **Amazon DynamoDB**

- Purpose: Storage for document metadata and embeddings.
- Configuration: On-demand capacity mode.
- Justification: Fast, consistent performance for vector retrieval with minimal management.

5. **Amazon EC2 (t3.medium)**

- Purpose: Hosts the language model for answer generation
- Configuration: t3.medium instance with optimized model deployment
- Justification: Provides necessary compute and memory for running the language model while being more cost-effective than SageMaker for this use case.

6. **Amazon CloudWatch**
- Purpose: Monitoring and logging.
- Configuration: Basic monitoring with custom metrics.
- Justification: Comprehensive visibility into application performance and issues.

7. **AWS IAM**
- Purpose: Security and access control.
- Configuration: Custom roles with least privilege.
- Justification: Essential for securing interactions between services.

Tech Stack:

1. **Programming Languages**
- Backend: Python 3.9+ for API, processing, and ML components
- Frontend: JavaScript/React for web interface

2. **Frameworks & Libraries**
- Web Framework: Flask or FastAPI for API endpoints
- Frontend: React for user interface
- PDF Processing: PyPDF2 or pdfminer.six for text extraction
- Vector Search: FAISS for efficient similarity search
- Embedding Models: Sentence-Transformers (all-MiniLM-L6-v2)
- Language Models: T5-small or FLAN-T5-small
- Document Export: python-docx for DOCX generation
- Model Serving: TorchServe or Flask for model API
- Model Optimization: ONNX Runtime for efficient inference

3. **Development & Deployment**
- Containerization: Docker for consistent deployment
- Version Control: GitHub for code management

8. Potential Architectural Challenges –

- PDF Processing Service Scalability

The architecture must handle variable-sized PDFs with complex layouts while maintaining performance. We'll implement asynchronous Lambda processing with configurable memory allocation and timeout settings, coupled with SQS queuing to manage processing backpressure during high load periods.

- Vector Database Implementation on DynamoDB

Storing and efficiently querying high-dimensional vector embeddings in DynamoDB presents architectural challenges. Our solution includes binary serialization of vectors, composite key design for efficient retrieval, and implementing approximate nearest neighbor search algorithms directly in the application layer.

- Model Server Reliability

Ensuring reliable operation of the model-hosting EC2 instance presents architectural challenges. We'll implement health checks, automatic recovery configurations, model loading optimizations to reduce startup time, and efficient resource utilization to prevent out-of-memory errors during inference.

- Stateless Processing with Context Preservation

Maintaining document context across stateless Lambda functions requires architectural consideration. We'll implement a metadata-rich chunking strategy with parent-child relationships between chunks, store comprehensive metadata in DynamoDB, and design a context assembly system in the API layer.

- Secure Temporary Storage Design

Designing secure yet temporary document storage presents architectural security challenges. Our approach includes S3 bucket policies with strict access controls, server-side encryption, pre-signed URLs for secure access, and automated lifecycle policies to ensure document deletion after processing.

- Cross-Service Communication Latency

The distributed nature of the architecture introduces potential latency in cross-service communication. We'll implement efficient API gateway routing, optimize Lambda cold start times, utilize connection pooling for DynamoDB, and implement strategic caching at the API layer to reduce repeated queries.

- Monitoring and Logging Architecture

Designing comprehensive monitoring across distributed services requires architectural planning. We'll implement centralized logging with CloudWatch Logs, create custom metrics for RAG-specific performance indicators, design appropriate alerting thresholds, and implement distributed tracing for cross-service requests.

9. **Initial Cost Estimation –**

Cost Estimation for AWS Leaner's Lab (30 days):

Services	Configuration	Estimated Cost (30 days)
EC2 (Web)	t3.small (100 hrs)	$1.80
EC2 (Model)	t3.medium (100 hrs)	$3.60
S3	0.5GB storage, minimal	$0.02
Lambda	100 invocations, 1GB	$0.20
DynamoDB	0.5GB storage	$0.13
Data Transfer	Minimal	$0.10
CloudWatch	Basic monitoring	$0.00
Total		~ $5.85

Table. 1. Estimated cost according to initial architecture.

10. References –

[1] "How to Build a RAG Chatbot from Scratch with Minimal AI Hallucinations," Coralogix AI Blog. [Online]. Available: https://coralogix.com/ai-blog/how-to-build-a-rag-chatbot-from-scratch-with-minimal-ai-hallucinations. Accessed: May 11, 2025.

[2] "Vector Embedding," IBM Think. [Online]. Available: https://www.ibm.com/think/topics/vector-embedding. Accessed: May 18, 2025.

[3] "Which is lower cost: SageMaker or EC2?," Stack Overflow. [Online]. Available: https://stackoverflow.com/questions/52198660/which-is-lower-cost-sagemaker-or-ec2. Accessed: May 18, 2025.