Cloud-Based Research Paper Summarization Tool using Amazon Bedrock and OpenSearch

Sindhu Jyoti Dutta, Sheel Patel, Yashavika Singh
Department of Computer Science
New York University, New York, USA
{sd6201, sjp9507, ys6668}@nyu.edu

***Abstract*—The overwhelming pace of academic publishing has created significant barriers to efficient knowledge consumption, particularly in fast-growing disciplines such as computer science, healthcare, and data science. Manual reading and synthesis of large volumes of papers are time-consuming and prone to oversight. This project presents a cloud-native, serverless application that leverages foundation models via Amazon Bedrock to automate research paper summarization and semantic search. Users can upload research papers, receive concise AI-generated summaries, extract key concepts, and perform similarity-based retrieval using vector embeddings stored in OpenSearch. The backend pipeline is built entirely using AWS services, including Lambda, S3, DynamoDB, and Bedrock, ensuring scalability and cost-efficiency. The frontend offers an intuitive interface for file upload, summary review, keyword-based search, and trending research exploration. By integrating LLM-based summarization with vector search and cloud orchestration, our tool significantly streamlines the academic research process for students, faculty, and professionals alike.**

Keywords— research summarization, Amazon Bedrock, vector search, OpenSearch, large language models, serverless architecture, cloud computing, academic productivity.

I. Problem Statement

With the exponential growth of academic publications across disciplines, researchers are facing an unprecedented information overload. Manually reading, understanding, and comparing papers is time-consuming and inefficient. According to estimates, more than 2.5 million new research articles are published annually across journals and conference proceedings. This has made it increasingly difficult for researchers, graduate students, and industry professionals to stay current with the latest developments.

Traditional approaches to literature review involve manual scanning of titles and abstracts, followed by full-paper readings for promising candidates. This process is not only inefficient but often fails to scale in fast-paced domains like computer science, machine learning, and bioinformatics.

II. Motivation

The need for tools that can automate parts of the literature review process is more critical than ever. Even advanced search platforms like Semantic Scholar and Google Scholar offer limited support for direct summarization or intelligent contextual search beyond keyword matching.

Recent advances in Large Language Models (LLMs) have introduced new possibilities in natural language understanding, enabling automated summarization, question answering, and document clustering with human-comparable accuracy. These capabilities can be harnessed to assist researchers by transforming unstructured papers into structured summaries and semantically indexed vectors, drastically reducing the time needed to consume and compare relevant literature.

Cloud platforms such as Amazon Web Services (AWS) offer robust infrastructure for building scalable, event-driven systems. Services like Amazon Bedrock now provide access to state-of-the-art foundational models including Mistral, Claude, and Titan without requiring users to host or fine-tune models themselves. When combined with serverless compute tools like AWS Lambda, OpenSearch, and DynamoDB, this creates a powerful platform for deploying lightweight, cost-effective machine learning pipelines.

III. Existing Solutions

Several tools and platforms have attempted to address the growing complexity of academic research and paper discovery. While each offers certain advantages, they often fall short in terms of summarization capabilities, extensibility, or integration with modern cloud-native AI services. Below are a few prominent examples:

- **Semantic Scholar:** A free AI-powered academic search engine developed by the Allen Institute for AI. It indexes millions of papers and extracts citations and influence scores but does not offer automated summarization. It relies heavily on metadata and citation networks.
- **Research Rabbit:** A platform designed for visual literature discovery. It allows users to follow citation trails and build collections but does not support AI-generated summaries or keyword-based vector search. Its primary strength lies in network visualizations.
- **Connected Papers:** This tool helps researchers explore prior and derivative work through graph-based clustering. While useful for topic exploration, it lacks deep document-level insights or semantic search features.
- **Scholarcy:** An AI-driven tool that attempts to summarize research papers and generate flashcards. However, it is limited in terms of API access and cannot be integrated seamlessly into academic workflows or scaled using cloud-native infrastructure.
- **Google Scholar and Microsoft Academic Graph:** These search engines provide keyword-based queries and citation tracking but do not incorporate summarization

or semantic embeddings. They are widely used but offer minimal support for automated research synthesis.

While these solutions provide value in discovery and network exploration, they generally do not support end-to-end summarization, vector-based semantic search, or modular backend integration. Our proposed system addresses these limitations by offering a serverless, event-driven pipeline that leverages LLMs through Amazon Bedrock and integrates vector search with OpenSearch.

IV. System Architecture

The architecture of our research paper summarization tool is modular, event-driven, and fully serverless. It leverages a range of AWS-managed services including Amazon Bedrock, Lambda, S3, SQS, DynamoDB, OpenSearch, Polly, SES, and API Gateway. The design emphasizes scalability, fault tolerance, modularity, and extensibility — ensuring a seamless user experience from upload to search, summarization, playback, and Q&A.

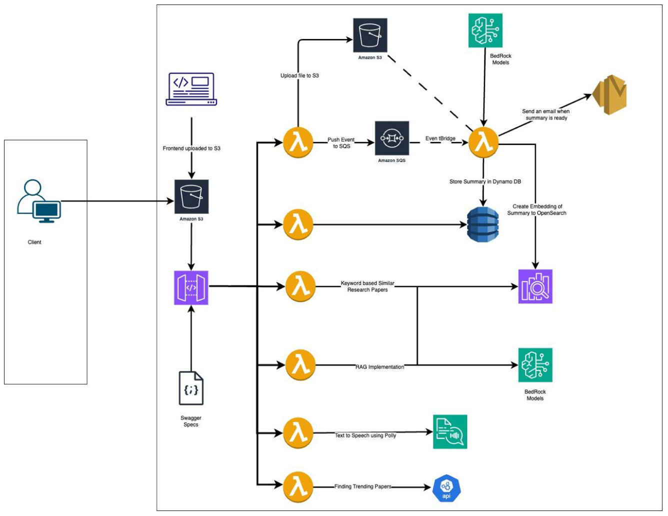


Fig. 1. Updated system architecture showing Lambda-driven microservices, Bedrock-powered LLM and embedding flows, and auxiliary modules like RAG and TTS.

A. Overview of System Components

- **Frontend Interface (Amazon S3):** A web application built with HTML, CSS, and JavaScript, hosted on Amazon S3. It allows users to upload research papers, view summaries, play audio, search similar papers, and interact with trending literature. The frontend is fully responsive and communicates asynchronously with backend APIs using Fetch.
- **API Gateway and Swagger/OpenAPI Specs:** All backend services are exposed via RESTful APIs configured through Amazon API Gateway. These endpoints are described using Swagger/OpenAPI specifications, which provide built-in validation and automatic frontend integration.
- **Upload Handler Lambda + Amazon SQS:** When a paper is uploaded, it is received by a Lambda function that stores it in S3 and publishes a message to an SQS queue. This decoupling allows asynchronous, non-blocking processing of files and protects the system from load spikes.
- **Summarization Pipeline (Lambda + Bedrock + SES):** The summarizer Lambda is triggered by the SQS message. It retrieves the paper, splits the content, and sequentially invokes the Mistral model via Amazon Bedrock. Once a summary is created, the system stores it in DynamoDB and optionally sends a notification email using Amazon SES.
- **Metadata Storage (DynamoDB):** All summaries, keywords, and associated metadata (e.g., filename, user email) are persisted in DynamoDB. It supports low-latency access and is well-suited to semi-structured data with variable schema fields.
- **Text Embedding and Indexing (Lambda + Bedrock + OpenSearch):** A dedicated Lambda creates semantic embeddings of summaries and keywords via Bedrock and indexes them in OpenSearch using its k-NN plugin. This enables fast, meaningful retrieval.
- **Semantic Search Handler (Lambda + OpenSearch):** When users enter queries, they are embedded and matched against the vector index in OpenSearch. The top-k results are returned based on cosine similarity, enhancing the search beyond keyword matching.
- **RAG-based Q&A Module (Lambda + Bedrock):** This Lambda handles natural language questions about a given paper. It retrieves the summary from storage, builds a context prompt, and queries Bedrock to generate responses tailored to the uploaded content.
- **Text-to-Speech Module (Lambda + Polly):** A TTS Lambda function integrates with Amazon Polly to convert summaries to audio. It allows users to listen to summaries directly from the UI, promoting accessibility and multitasking.
- **Trending Paper Ingestion (CloudWatch + Lambda):** This Lambda is scheduled to run via CloudWatch Events and pulls trending papers from APIs like arXiv or Semantic Scholar. The papers are summarized, embedded, and indexed for search.

B. Design Choices and Rationale

- **Fully Serverless Architecture:** AWS Lambda, S3, DynamoDB, and OpenSearch reduce operational burden and scale automatically with usage. This model is cost-effective and fits academic or startup deployment needs.
- **Event-Driven Processing:** S3 PUT events, SQS messaging, and CloudWatch schedules enable reactive, loosely coupled workflows. Each task can fail, retry, and scale independently, improving system reliability.
- **Use of Amazon Bedrock:** Bedrock provides access to proprietary LLMs without infrastructure management.

Mistral was chosen for its strong summarization performance and prompt responsiveness.

- **Vector Retrieval with OpenSearch:** OpenSearch k-NN plugin enables scalable, native vector indexing. It eliminates the need for external vector DBs like Pinecone, and integrates smoothly with IAM/VPC security.
- **Integrated Notifications and Accessibility:** The use of SES and Polly adds practical value: users can receive email summaries and listen to content audibly, which enhances reach and inclusivity.
- **Security and Cost Awareness:** IAM roles are scoped per function. No sensitive user data is stored. Services are pay-per-use, and additional cost controls can be added via budgets, quotas, and caching.

C. Implementation Details

1) Backend Lambda Functions: Each function is deployed using AWS SAM and written in Python 3.11. The major Lambda functions are:

- **Upload Lambda:** Handles file validation and stores papers in S3. Publishes SQS messages to trigger summarization.
- **Summarization Lambda:** Processes files, interacts with Bedrock, and stores results. Sends email via SES.
- **Embedding Lambda:** Generates vector embeddings from summaries and keywords and stores them in OpenSearch.
- **Search Lambda:** Accepts user queries, embeds them, and returns top-k similar results from OpenSearch.
- **Q&A Lambda:** Retrieves paper summary and uses Bedrock to answer user-submitted questions.
- **TTS Lambda:** Uses Amazon Polly to return an audio stream for a given summary.
- **Trending Fetcher Lambda:** Fetches recent papers and processes them through the same pipeline.

2) Frontend and Integration: The frontend is hosted on S3 and built as a single-page app. It includes:

- A drag-and-drop uploader and progress tracker.
- Sections for real-time summaries, trending content, and similar paper discovery.
- A search bar and Q&A interface for each uploaded paper.
- A "Play" button to trigger TTS playback of summaries.

API calls are handled via the Fetch API. CORS and IAM policies ensure secure interactions.

3) System Workflow Summary:

1) User uploads a paper via the frontend.
2) Upload Lambda stores it in S3 and queues an event in SQS.
3) Summarization Lambda consumes the message, generates the summary, and stores it in DynamoDB.
4) Embedding Lambda creates vector representations and indexes them in OpenSearch.
5) Search or Q&A Lambda is triggered based on user query.
6) Polly Lambda handles TTS playback requests.
7) CloudWatch regularly triggers ingestion of new external research.

V. Results

The implementation of the cloud-based research paper summarization tool has been validated through extensive functional testing, user feedback, and demonstration of end-to-end features. This section outlines the primary system capabilities as illustrated through the following screenshots.

A. Upload and Summarization Interface

The homepage provides an intuitive interface for uploading research papers. As shown in Figure 2, users can drag and drop PDF or text files, enter their email address (optional), and receive an AI-generated summary after successful upload and processing.

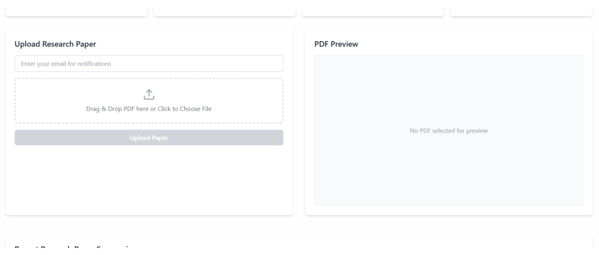
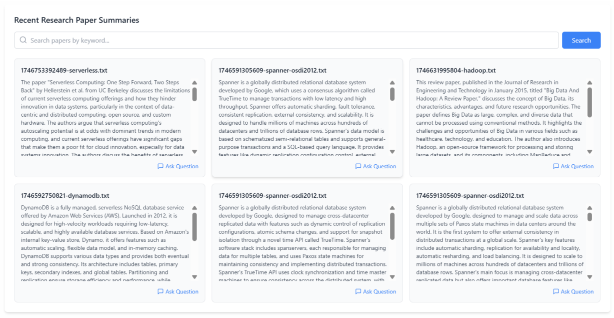

Upload Research Paper
Enter your email for notifications
Drag & Drop PDF here or Click to Choose File
Upload Paper
PDF Preview
No PDF selected for preview

Fig. 2. Homepage interface: User uploads a research paper and receives a summary.

B. Summary Display and Keyword Preview

Once the summary is generated using Amazon Bedrock, the system displays the result under "Recent Research Paper Summaries" (Figure 3). The summary is presented in a card layout along with the original filename, and includes an option for keyword-specific Q&A.



1746753392489-serverless.txt

The paper "Serverless Computing: One Step Forward, Two Steps Back" by Hellerstein et al. from UC Berkeley discusses the limitations of current serverless computing offerings and how they hinder innovation in data systems, particularly in the context of data-centric and distributed computing, open source, and custom hardware. The authors argue that serverless computing's autoscaling potential is at odds with dominant trends in modern computing, and current serverless offerings have significant gaps that make them a poor fit for cloud innovation, especially for data

1746591305609-spanner-osdi2012.txt

Spanner is a globally distributed relational database system developed by Google, which uses a consensus algorithm called TrueTime to manage transactions with low latency and high throughput. Spanner offers automatic sharding, fault tolerance, consistent replication, external consistency, and scalability. It is designed to handle millions of machines across hundreds of datacenters and trillions of database rows. Spanner's data model is based on schematized semi-relational tables and supports general-purpose transactions and a SQL-based query language. It provides

1746631995804-hadoop.txt

This review paper, published in the Journal of Research in Engineering and Technology in January 2015, titled "Big Data And Hadoop: A Review Paper," discusses the concept of Big Data, its characteristics, advantages, and future research opportunities. The paper defines Big Data as large, complex, and diverse data that cannot be processed using conventional methods. It highlights the challenges and opportunities of Big Data in various fields such as healthcare, technology, and education. The author also introduces Hadoop, an open-source framework for processing and storing

1746592750821-dynamodb.txt

DynamoDB is a fully managed, serverless NoSQL database service offered by Amazon Web Services (AWS). Launched in 2012, it is designed for high-velocity workloads requiring low-latency, scalable, and highly available database services. Based on Amazon's internal key-value store, Dynamo, it offers features such as automatic scaling, flexible data model, and in-memory caching. DynamoDB supports various data types and provides both eventual and strong consistency. Its architecture includes tables, primary keys, secondary indexes, and global tables. Partitioning and

1746591305609-spanner-osdi2012.txt

Spanner is a globally distributed relational database system developed by Google, designed to manage cross-datacenter replicated data with features such as dynamic control of replication configurations, atomic schema changes, and support for snapshot isolation through a novel time API called TrueTime. Spanner's software stack includes spanservers, each responsible for managing data for multiple tables, and uses Paxos state machines for maintaining consistency and implementing distributed transactions. Spanner's TrueTime API uses clock synchronization and time master

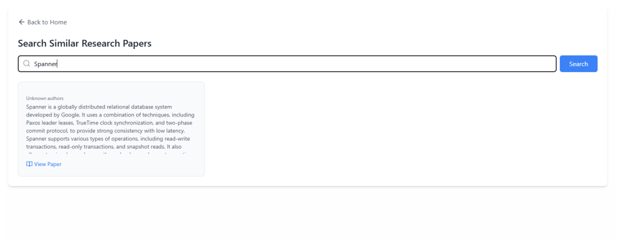
1746591305609-spanner-osdi2012.txt

Spanner is a globally distributed relational database system developed by Google, designed to manage and scale data across multiple sets of Paxos state machines in data centers around the world. It is the first system to offer external consistency in distributed transactions at a global scale. Spanner's key features include automatic sharding, replication for availability and locality, automatic resharding, and load balancing. It is designed to scale to millions of machines across hundreds of datacenters and trillions of database rows. Spanner's main focus is managing cross-datacenter

Fig. 3. Summary results: AI-generated summary cards from uploaded papers.

C. Semantic Search and Retrieval

As shown in Figure 4, users can enter a keyword or phrase (e.g., "spanner") to retrieve related papers based on vector similarity search. The embedded query is matched against the OpenSearch index and top results are returned. This enables semantic exploration beyond exact keyword matching.



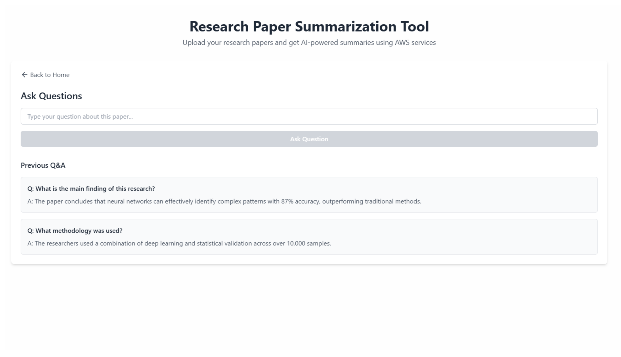
Unknown authors

Spanner is a globally distributed relational database system developed by Google. It uses a combination of techniques, including Paxos leader leases, TrueTime clock synchronization, and two-phase commit protocol, to provide strong consistency with low latency. Spanner supports various types of operations, including read-write transactions, read-only transactions, and snapshot reads. It also

Fig. 4. Semantic Search: Query-based retrieval using OpenSearch vector similarity.

D. Q&A Interaction

The Q&A module allows users to ask specific questions about a summarized paper, such as "What was the main finding?" or "What methodology was used?" (Figure 5). These responses are generated from extracted keywords and context, demonstrating the system's ability to enable deeper interaction with academic content.


Research Paper Summarization Tool

Upload your research papers and get AI-powered summaries using AWS services

Previous Q&A

Q: What is the main finding of this research?

A: The paper concludes that neural networks can effectively identify complex patterns with 87% accuracy, outperforming traditional methods.

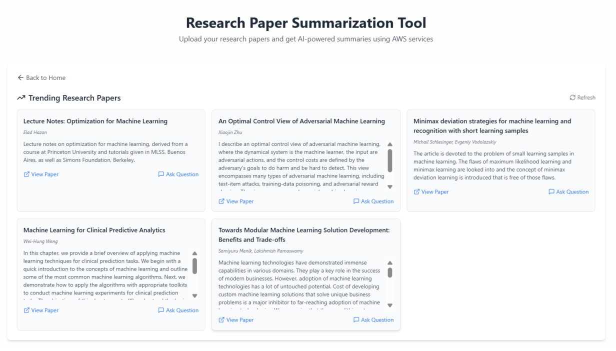
Q: What methodology was used?

A: The researchers used a combination of deep learning and statistical validation across over 10,000 samples.

Fig. 5. Q&A Interface: Users ask questions related to a paper's summary.

E. Trending Papers Feed

Figure 6 displays the trending papers section, automatically populated via a scheduled Lambda function that fetches recent articles from public APIs like arXiv and Semantic Scholar. These are summarized and embedded in the same pipeline as user uploads, enabling discovery of new research.


Research Paper Summarization Tool

Upload your research papers and get AI-powered summaries using AWS services

Lecture Notes: Optimization for Machine Learning

Elad Hazan

Lecture notes on optimization for machine learning, derived from a course at Princeton University and tutorials given in MLSS, Buenos Aires, as well as Simons Foundation, Berkeley.

An Optimal Control View of Adversarial Machine Learning

Xiaojin Zhu

I describe an optimal control view of adversarial machine learning, where the dynamical system is the machine learner, the input are adversarial actions, and the control costs are defined by the adversary's goals to do harm and be hard to detect. This view encompasses many types of adversarial machine learning, including test-item attacks, training-data poisoning, and adversarial reward

Minimax deviation strategies for machine learning and recognition with short learning samples

Michail Schlesinger, Evgeniy Vodolazskiy

The article is devoted to the problem of small learning samples in machine learning. The flaws of maximum likelihood learning and minimax learning are looked into and the concept of minimax deviation learning is introduced that is free of those flaws.

Machine Learning for Clinical Predictive Analytics

Wei-Hung Weng

In this chapter, we provide a brief overview of applying machine learning techniques for clinical prediction tasks. We begin with a quick introduction to the concepts of machine learning and outline some of the most common machine learning algorithms. Next, we demonstrate how to apply the algorithms with appropriate toolkits to conduct machine learning experiments for clinical prediction

Towards Modular Machine Learning Solution Development: Benefits and Trade-offs

Samiyuru Menik, Lakshmish Ramaswamy

Machine learning technologies have demonstrated immense capabilities in various domains. They play a key role in the success of modern businesses. However, adoption of machine learning technologies has a lot of untouched potential. Cost of developing custom machine learning solutions that solve unique business problems is a major inhibitor to far-reaching adoption of machine

Fig. 6. Trending Feed: Summarized recent papers fetched from external APIs.

F. Summary of Results

The system demonstrates the following core capabilities:

- Seamless file upload and summarization using Bedrock (Mistral model).
- Structured metadata generation including summaries and keyword extraction.
- Semantic search using OpenSearch vector indexing.
- Interactive Q&A with context-aware responses.
- Trending paper ingestion from external academic sources.

These results validate the system's practical utility for students and researchers, showing that AI-powered summarization and retrieval can significantly accelerate literature review and exploration.

VI. Limitations

While the system performs well, it currently relies on proprietary APIs (e.g., Amazon Bedrock), which incur costs. Summarization quality depends on the LLM's performance and may occasionally produce inaccurate or oversimplified results. The system does not yet support user authentication or bulk paper ingestion.

VII. Conclusion and Future Work

In this project, we have presented a scalable, serverless cloud-based system that leverages large language models (LLMs) for the automated summarization and semantic retrieval of research papers. By integrating Amazon Bedrock with AWS Lambda, DynamoDB, OpenSearch, and S3, we developed a robust end-to-end pipeline capable of ingesting papers, generating high-quality summaries, extracting keywords, and enabling vector-based similarity search — all through a user-friendly web interface.

Our solution addresses the growing challenge of academic information overload, particularly in fields like computer science and healthcare, where new research is published daily. The tool empowers users to quickly digest the essence of complex papers without manually reading every section, and to discover related works using semantic embeddings rather than traditional keyword search. The use of Bedrock's foundation models — particularly Mistral — provided high-quality summarization results, while OpenSearch's k-NN plugin enabled real-time vector search without leaving the AWS ecosystem.

Beyond the core functionality, our system includes thoughtful features such as a trending paper module powered by CloudWatch Events, a Q&A interaction module powered by a Retrieval-Augmented Generation (RAG)-like approach, and keyword extraction for enhanced indexing. All services were orchestrated using event-driven architectures, which ensured low latency, modularity, and cost-efficiency.

We also introduced a text-to-speech (TTS) feature using Amazon Polly, allowing users to listen to the generated summaries through a simple playback interface. This improves accessibility for users with visual impairments and enhances user experience for mobile or multitasking environments.

There are multiple exciting directions to extend this project:

- **Multi-format Input Support:** Future versions will include native PDF and DOCX ingestion with structure-preserving parsing using libraries like PyMuPDF or AWS Textract.
- **User Authentication and History:** We plan to add authentication via Amazon Cognito, allowing users to track uploaded papers, revisit past summaries, and manage search history.
- **Personalized Summaries:** Integrating prompt-tuning or retrieval-augmented generation (RAG) pipelines to generate summaries tailored to users' background (e.g., undergrad vs PhD).
- **Feedback Loop and Active Learning:** Incorporating thumbs-up/down or "Was this helpful?" feedback on summaries to fine-tune prompt templates and select optimal models over time.
- **Cross-Paper Clustering and Recommendation:** Using the embedding space to build research paper clusters and enable content-based recommendations beyond keyword matches.
- **Mobile-Friendly Experience:** Improving UI/UX responsiveness and implementing push notifications for summary readiness or trending updates.
- **Cost Optimization:** Exploring spot Lambda containers, model caching, and usage quotas for institutional deployment.

In conclusion, this project demonstrates the feasibility and effectiveness of integrating generative AI with cloud-native infrastructure to support academic research workflows. With continuous improvements in foundational models, serverless tools, and vector search capabilities, systems like this can significantly transform how researchers access, consume, and interact with scholarly content.

Demo and Video Presentation

A working demo of the tool is available at: http://research-paper-summarisation.s3-website-us-east-1.amazonaws.com
The video showcasing our project is available at: https://youtu.be/eXXfsw-C9l4

Acknowledgments

This project was completed as part of NYU's Cloud Computing curriculum.