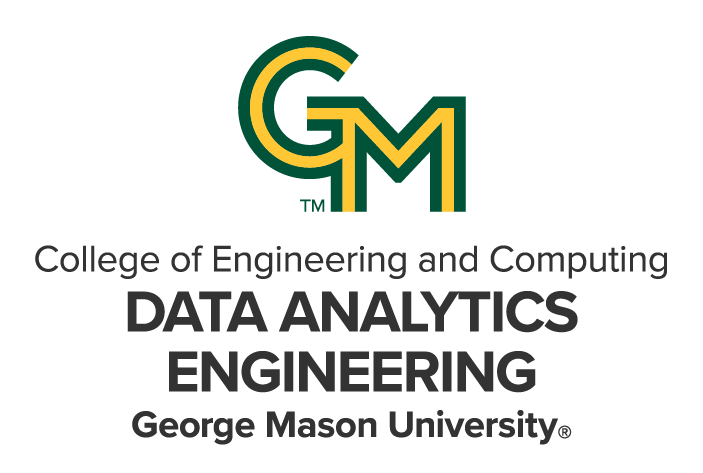
A black rectangle with a black background

Description automatically generated with low confidence

DAEN 690

Project Report

Robert Jay Dunlap

Romin Rajesh Katre

Sai Ashish Reddy Shetpally

Soumith Alloju

Nithiya Varsha Markandan Rajendran

Zijie He

Spring 2025

Proposal Helper: Intelligent Proposal Generator

**About the Cover**

This semester, the DAEN program is proud to spotlight one of our esteemed capstone partners, Daniel Erasmus—a visionary whose groundbreaking work influences leaders worldwide. As the founder and CEO of Erasmus.AI, Daniel is a renowned futurist and a pioneer in scenario planning, artificial intelligence, and strategic foresight. His innovative approach to blending AI with human-centric decision-making has profoundly shaped global conversations on technology, sustainability, and future-readiness. Through his thought leadership, Daniel continues to inspire organizations across the globe to embrace change and build resilient futures.

At Erasmus.AI, Daniel conceived and led the development of ClimateGPT—the world’s first foundational AI model family focused on climate change. Built on over a decade of collecting and processing planetary-scale datasets, this groundbreaking innovation leverages AI to uncover hidden connections in global news, from Human-Centered Extreme Weather Dashboards to maps of global innovations, risks, and breakthroughs. The Erasmus.AI platform exemplifies his commitment to using technology to inform and address some of the world’s most pressing challenges.

As co-founder of The Digital Thinking Network (DTN), Daniel has spent over 25 years leading large-scale scenario planning and transformation processes. His work has driven notable actions, such as initiating a response to food security challenges during COVID-19 that delivered 1 million meals within three months and has since provided over 60 million meals in Sub-Saharan Africa. His scenario processes have also anticipated major global events, including the Global Financial Crisis in 2006 and the Oil Price Collapse in 2012—each resulting in multi-billion-dollar benefits for his clients. In the public sector, DTN's transformative initiatives include the Rotterdam Advisory Board, which spearheaded the Rotterdam Climate Initiative in 2005 with the ambitious goal of halving CO2 emissions by 2025, and the creation of the 30-year global future scenarios Ci’Num.

An accomplished author, Daniel has written three books on innovation and the networked society, as well as numerous columns, including the Information Society column for the Financial Times Review. He has also held various prominent board positions and fellowships, including serving on the University of Stellenbosch’s Faculty of Science Advisory Board, Cambridge-based Titan Advanced Energy Solutions, and the supervisory board of the Quad9 Foundation. Through his visionary leadership, Daniel continues to shape the future across disciplines and industries.

Contents

Table of Contents

[Abstract 1](#_Toc197546872)

[Section 1: Introduction 3](#_Toc197546873)

[1.1 Report Purpose 3](#_Toc197546874)

[1.2 Report Readership 3](#_Toc197546875)

[1.3 Report Structure 3](#_Toc197546876)

[Section 2: Problem Definition 3](#_Toc197546877)

[2.1 Problem Space 3](#_Toc197546878)

[2.2 Research 4](#_Toc197546879)

[2.3 Solution Space 6](#_Toc197546880)

[2.4 Project Objectives 6](#_Toc197546881)

[2.5 Primary User Stories 7](#_Toc197546882)

[2.6 Product Vision 8](#_Toc197546883)

[2.6.1 Scenario #1 8](#_Toc197546884)

[2.6.2 Scenario #2 9](#_Toc197546885)

[Section 3: Datasets 9](#_Toc197546886)

[3.1 Overview 9](#_Toc197546887)

[3.2 Field Descriptions 9](#_Toc197546888)

[3.3 Data Context 9](#_Toc197546889)

[3.4 Data Conditioning 10](#_Toc197546890)

[3.5 Data Quality Assessment 10](#_Toc197546891)

[3.6 Other Data Sources 10](#_Toc197546892)

[3.7 Storage Medium 10](#_Toc197546893)

[3.8 Storage Security 10](#_Toc197546894)

[3.9 Storage Costs 11](#_Toc197546895)

[Section 4: Implementation 12](#_Toc197546896)

[4.1 Overview 12](#_Toc197546897)

[4.1.1 System Architecture 12](#_Toc197546898)

[4.2 Key Components of Implementation 13](#_Toc197546899)

[4.2.1 Data Ingestion and storage In S3 13](#_Toc197546900)

[4.2.2 Amazon Kendra Integration Information Retrieval 13](#_Toc197546901)

[4.2.3 AWS Bedrock Integration for AI Generated Proposal Content 13](#_Toc197546902)

[4.2.4 API Development for Chatbot Interaction 13](#_Toc197546903)

[4.3 Challenges and Solutions 14](#_Toc197546904)

[Section 5: Testing and Validation 14](#_Toc197546905)

[5.1 Testing Kendra and LLMs 14](#_Toc197546906)

[5.1.1 Testing Kendra 14](#_Toc197546907)

[5.1.2 Testing LLMs 16](#_Toc197546908)

[5.1.2.1 Claude 16](#_Toc197546909)

[5.1.2.2 Llama 18](#_Toc197546910)

[5.1.2.3 Jamba 20](#_Toc197546911)

[Section 6: Findings 22](#_Toc197546912)

[6.1.1 Model Visualizations Comparison 22](#_Toc197546913)

[6.1.2 Table Evaluation 23](#_Toc197546914)

[6.1.3 Strengths and Weaknesses Identified Through Prompt Engineering 23](#_Toc197546915)

[Section 7: Summary 24](#_Toc197546916)

[Section 8: Future Work 24](#_Toc197546917)

[Appendix A: Domain Background 27](#_Toc197546918)

[Appendix B: Glossary 29](#_Toc197546919)

[Appendix C: GitHub Repository 31](#_Toc197546920)

[Overview 31](#_Toc197546921)

[README.md Content 31](#_Toc197546922)

[GitHub Repository Link 33](#_Toc197546923)

[GitHub Repository Contents 33](#_Toc197546924)

[Appendix D: Risks 34](#_Toc197546925)

[Sprint 1 Risks 34](#_Toc197546926)

[Sprint 2 Risks 35](#_Toc197546927)

[Sprint 3 Risks 36](#_Toc197546928)

[Sprint 4 Risks 38](#_Toc197546929)

[Sprint 5 Risks 38](#_Toc197546930)

[Appendix E: Agile Development 39](#_Toc197546931)

[Scrum Framework Team Approach 39](#_Toc197546932)

[DAEN 698 assigns projects to teams but doesn’t explicitly instruct them on how to design or build their systems. Instead, the course requires that students implement an Agile/Scrum framework, so they become competent using industry standard methodology for project management. Through this, students learn important aspects of design thinking, objective settings, and progress monitoring, and hopefully to identify issues early so they can be fixed before negatively impacting the project’s outcome. 39](#_Toc197546933)

[Sprint 1 Lessons Learned 39](#_Toc197546934)

[Sprint 2 Lessons Learned 40](#_Toc197546935)

[Sprint 3 Lessons Learned 40](#_Toc197546936)

[Sprint 4 Lessons Learned 41](#_Toc197546937)

[Sprint 5 Lessons Learned 41](#_Toc197546938)

Table of Figures

[Figure 1: ChatGPT is the latest Natural Language Processing tool for Data Analytics. 5](#_Toc176093306)

[Figure 2: Data Analytics Engineering has the power to be transformative for an organization. 6](#_Toc176093307)

[Figure 3: Sprint project dates. 27](#_Toc176093308)

Table of Tables

[Table 1: Glossary Table 20](#_Toc176093309)

[Table 2: Sprint 1 Risks 23](#_Toc176093310)

[Table 3: Sprint 2 Risks 24](#_Toc176093311)

[Table 4: Sprint 3 Risks 24](#_Toc176093312)

[Table 5: Sprint 4 Risks 25](#_Toc176093313)

[Table 6: Sprint 5 Risks 26](#_Toc176093314)

**This page intentionally left blank**

Abstract

Abstract

Allwyn Corporation, an IT service provider in the competitive U.S. government contracting sector, must demonstrate relevant past performance to win new contracts. However, retrieving this information from over 15 years of project documentation is time-consuming and depends heavily on the institutional knowledge of long-term employees. This project presents the design and implementation of an AI-assisted Retrieval-Augmented Generation Large Language Model (RAG-LLM) system to streamline proposal development by automating the discovery of past performance data. The solution was built using the AWS QnABot Solution CloudFormation Stack, integrating Amazon Kendra for semantic document retrieval, Amazon Bedrock for AI-generated responses, AWS Lambda for backend orchestration, and the default Lex Web UI for user interaction. While the initial architecture relied on Allwyn’s Microsoft SharePoint document store, limitations in direct Kendra integration led to a revised design using an Amazon S3 bucket as the document repository. Three LLMs were evaluated for system integration: Anthropic Claude 3.5 Sonnet V1, AI21 Labs Jamba-Instruct, and Meta Llama 3 8B. These models were selected based on their emerging popularity, architectural variety, and commercial viability. Model performance was scored across three key criteria: (1) Did the response answer the user’s question? (2) Did it correctly identify the relevant project source? (3) Was the response proposal-ready? Testing involved 31 representative queries sourced from Allwyn's proposal writers. A prompt engineering workflow inspired by test-driven development was used to iteratively refine model outputs, yielding performance improvements exceeding 50% in certain cases. Results showed Claude achieved the highest average score across the three evaluation criteria, while Claude and Jamba tied for the most correct answers overall. This report outlines the system architecture, evaluation methodology, and test results, and demonstrates how RAG-LLM systems can accelerate, standardize, and scale knowledge extraction for proposal development.

**This page intentionally left blank**

Report

# Introduction

## Report Purpose

This report introduces an enhanced solution for Allwyn Corporation to improve their proposal development method, which demands lengthy manual hours to gather historic data and case studies. To solve this problem, we proposed an automated Generative AI Retrieval-Augmented Generation (RAG) chatbot system that retrieves data automatically, allowing users to quickly access past proposals and performance statistics. In this study, the implementation of a scalable, serverless chatbot that makes use of Amazon Web Services (AWS) to accomplish these goals is described in detail, as are the efforts that were made to achieve them. The main purpose of this study is to reduce the extensive manual effort made to retrieve past performance data and proposals.

## Report Readership

This report is meant for the management team at Allwyn Corporation, more especially the people in charge of the project who will be making decisions on technological solutions, developing the business, and generating proposals. Team members in charge of IT and development inside the organization who may be responsible for implementing the RAG solution driven by AI will also find this report useful. Critical insights into the problems with the present proposal’s creation process and how to adopt the newly created solution to improve accuracy and efficiency are provided in the report.

We hope that by reading this report, you will have a better grasp of the project's technical details, such as how we used RAG and AWS services to generate proposals. Evaluation of the solution's effect on proposal quality, turnaround time, and resource allocation will be done by the management team using the report. Also, the technical documentation will be used by the development and IT teams to make sure the system is scalable, easy to install, and maintained in the long run.

## Report Structure

This report is structured to first provide an overview of the current proposal development process at Allwyn Corporation, highlighting the challenges of manual data retrieval in the Introduction. The Problem Definition section delves deeper into the inefficiencies of gathering historical data and case studies. The Solution Overview introduces an automated Generative AI RAG chatbot system integrated with AWS Kendra, Bedrock, Lex and Lambda to streamline data retrieval. The System Design and Architecture section outlines the technical components and structure of the proposed solution, followed by the Implementation and Workflow section, which explains the step-by-step process of using the system. In the Results and Performance Evaluation section, the effectiveness of the solution is assessed by comparing manual and automated processes. The Discussion reflects on the system’s impact, challenges faced, and potential solutions. Finally, the Conclusion summarizes the key findings, while the Future Work section suggests directions for further improvements and enhancements.

# Problem Definition

## Problem Space

Allwyn Corporation, with over 15 plus years of experience in the industry, has accumulated a vast repository of past performance data, case studies, and historical project records. These documents, stored within a SharePoint repository, serve as critical references for technical proposal writers. However, retrieving relevant information from this extensive collection presents significant challenges. Proposal writers often struggle to efficiently locate precise details about past projects, requiring them to sift through numerous documents manually or seek guidance from the CEO, whom possesses institutional knowledge of the company’s historical work. This dependency is neither scalable nor efficient, leading to bottlenecks in the proposal development process.

The inefficiencies in information retrieval stem from several key issues. First, SharePoint - while a centralized document storage system, lacks an intuitive and intelligent search mechanism tailored for proposal writing. Traditional keyword-based search methods often yield overwhelming volumes of results, many of which are not contextually relevant. Writers must then manually sort through documents to identify the appropriate content, inefficiently consuming their valuable time. Additionally, inconsistencies in document organization and naming conventions further complicate the search process, increasing the likelihood of the incomplete retrieval of needed information.

Another major challenge is the reliance on human knowledge for information retrieval. The CEO and a few key individuals possess deep insights into Allwyn Corporation’s past projects, but it is impractical for proposal writers to consult them every time they need specific details. This creates an operational bottleneck, where the ability to draft accurate and compelling proposals is constrained by the availability of a few individuals. As the company continues to grow and generate an increasing number of proposals, this issue becomes more pronounced, impacting efficiency and responsiveness to new business opportunities.

The problem space extends beyond simple document retrieval; it encompasses the broader challenge of knowledge management within a large organization. The lack of an efficient system for extracting relevant information means that valuable insights remain underutilized. Without a better way to leverage historical performance data, proposal writers will continue to struggle constructing well-supported narratives that effectively demonstrate the company’s successful prior experience and capabilities. This can lead to reduced quality of their proposals, reducing their chance of winning bids.

To address these challenges, this project will implemented an AWS-based QnA Bot, a retrieval-augmented chatbot designed to intelligently fetch relevant information from Allwyn Corporation’s SharePoint repository. Unlike conventional search methods, this chatbot will leverage AI-powered natural language processing (NLP) to understand and interpret user queries, delivering precise and contextually relevant results. By integrating with AWS services such as Amazon Lex, AWS Lambda, and Amazon OpenSearch, AWS Kendra, etc., the chatbot will provide a scalable, secure, and efficient solution for retrieving critical proposal-related data.

## Research

To develop the AI-assisted proposal retrieval chatbot, our team conducted extensive research to understand both the technical feasibility and implementation. Our research focused on:

* Retrieval-Augmented Generation (RAG) and its role in AI-powered information retrieval.
* AWS services, including Amazon Lex, AWS Bedrock, Amazon Kendra, OpenSearch, and DynamoDB, to determine the best approach for integrating these tools.
* Existing proposal automation challenges and best practices from industry reports and academic papers.

This section summarizes our literature review, technical exploration, and collaborative research efforts.

1. Research Themes & Findings
   1. Understanding Retrieval-Augmented Generation (RAG) in AI-Powered Information Retrieval

To ensure our chatbot effectively assists proposal writers, we explored RAG methodology. Unlike standard generative AI models that rely solely on pre-trained knowledge, RAG improves accuracy by fetching real-time information from external sources.

Key Findings from RAG Research:

* RAG incorporates a retrieval-based search capability built on advanced vector and graph database. These systems are able to find relevant passages within documents in response to natural language questions.
* It incorporates a generative AI model to that extracts relevant sections from the responses.
* It improves accuracy, relevance, and transparency in AI-generated content.
* All existing large language models (such as OpenAI’s GPT, Anthropic Claude, and Amazon Bedrock models) work effectively with vector or graph database outputs and are an excellent choice for enterprise search applications.
  1. Exploring AWS Services for Chatbot Development

To align with our sponsor’s requirements, we explored AWS tools for building and deploying our AI chatbot.

AWS Services Considered & Their Roles:

|  |  |
| --- | --- |
| AWS Service | Possible Usage |
| Amazon Lex | Processes user queries using NLP |
| Amazon Lambda | Handles backend logic and integrates various AWS services |
| Amazon Kendra | Indexes unstructured text and retrieves the most relevant portions in response to a natural language question |
| Amazon S3 | The store of the historical proposal documents for this project |
| AWS Bedrock | Runs a variety of large language models, such as Anthropic Claude or Meta Llama |
| Amazon DynamoDB | Stores chatbot conversation history |

* 1. Understanding Proposal Automation Challenges & Industry Best Practices

Our research also included analyzing existing proposal generation challenges faced by enterprises and how AI can assist.

Challenges Identified in Traditional Proposal Writing:

* Time-Consuming Process: Writers manually search past proposals and project data, leading to inefficiencies.
* Lack of Consistency: Different teams use different formats and terminology.
* Data Retrieval Issues: Finding past performance data is cumbersome without advanced search capabilities.

How Our AI Chatbot Addresses These Issues:

* Automates proposal data retrieval using RAG-based search.
* Provides a structured search interface using AWS Kendra & OpenSearch.
* Ensures data security with AWS Cognito and IAM role-based access control.

1. Research Approach & Methodology

We organized our research using three primary methods:

1. Literature Review: Academic papers on RAG, chatbot development, and enterprise AI.
2. Technical Exploration: Hands-on testing of AWS QnABot, Amazon Kendra, OpenSearch, and Bedrock.
3. Collaborations & Discussions: Meetings with our sponsor (Allwyn Corp) to align research with business requirements.
4. Conclusion  
   Through our research, we:

* Validated the use of RAG for proposal retrieval.
* Identified the best AWS services for our chatbot.
* Understood the challenges in proposal automation and how AI can help.

In the next phase, we applied our research findings to begin implementation, focusing on data indexing, chatbot development, and integration of all the AWS services in our system.

## Solution Space

Our solution optimizes the proposal generating process for Allwyn Corporation by utilizing Retrieval-Augmented generating (RAG) to automate and enhance proposal quality. The system incorporates Amazon Kendra for intelligent content retrieval from previous documents, ensuring effective utilization of past bids, case studies, and other pertinent data. Amazon Bedrock facilitates AI-driven content production, allowing for the development of structured and contextually precise ideas. AWS Lambda facilitates real-time interactions among these services, offering a serverless and scalable architecture that reduces manual labor and enhances productivity.

Amazon S3 is utilized for the storage of historical documents and proposals, whereas Amazon Cognito facilitates secure user authentication and access control. The architecture is engineered for scalability, enabling the system to accommodate fluctuating workloads and adjust dynamically as required. By integrating these AWS services, the system guarantees that Allwyn Corporation can generate high-quality, consistent proposals that utilize data-driven insights, thereby substantially decreasing manual drafting time.

Essential performance indicators, including content relevancy, time efficiency, and system performance, will be monitored to evaluate the solution's efficacy. The solution seeks to optimize the proposal generation process, elevate the quality of proposals, and offer an AI-driven tool that assists Allwyn Corporation in increasing proposal success rates while ensuring consistency and precision in all submissions.

## Project Objectives

1. What did the team assume it would learn after finishing this project?

After completion of this project, our team assumed we would have increased understanding of generative AI models, especially as applied to retrieval-augmented generation (RAG) systems, both theoretically and practically. Also, practical knowledge on integrating the generative models within AWS services to create a chatbot system that companies can utilize for efficient retrieval of past performance and statistics.

2. What did the team assume they would achieve as a solution upon finishing this project?

The team aimed to create a Retrieval-Augmented Generation chatbot system that integrates with AWS services like Lex, Kendra, and others to automate the process of information retrieval, which can get all the relevant past performance information effectively. We wanted to reduce the extensive manual effort put into retrieving past data statistics and provide a scalable, serverless architecture.

3. What did the team assume it would achieve in terms of understanding about the problem after finishing the project?

The team would have a thorough grasp of the difficulties in retrieving past performance data manually, the shortcomings of conventional retrieval techniques, and how AI-powered solutions improve document retrieval precision, consistency, and efficiency after finishing this project. The group would also comprehend the most effective methods for handling and applying AWS services and AI to this product.

4. What did the team assume it would provide in value as a product of this project work to the world, targeted group, etc.?

The project would provide Allwyn Corporation with an intelligent chatbot system that transforms the information retrieval process, reducing the manual effort and time spent while increasing accuracy and consistency. By integrating AWS services and AI, we would provide a scalable product, improving the operational efficiency and productivity of the company.

## Primary User Stories

To utilize a generative AI proposal assistant to retrieve past success stories and provide useful information for writing proposals, we identified several key user stories to guide the development and functionality of the system.

* As a proposal writer, I want to be able to quickly search for relevant case studies and previous successful proposals when drafting a new proposal, so that I can improve the quality of the proposal and save time in searching for previous cases.
* As a Business Development Manager, I want to use AI-driven proposals that tell me the comparison and strengths and weaknesses of previous successful proposals and current proposals, so that I can purposefully judge the quality and likely success of current proposals.
* As a project manager, I want to access historical project data, including client feedback and proposal performance, so that I can improve the current proposals based on experience.
* As a CEO, I want to receive AI-generated summaries of key proposal sections based on historical success cases, so that I can quickly evaluate and efficiently approve proposals.

## Product Vision

### Scenario #1

For: Proposal Managers at Allwyn Corporation

Who: Require the formulation of precise, uniform suggestions grounded in historical performance.  
The: AWS Lex-powered chatbot solution

Is a: Dialogue-based artificial intelligence system

That: Automates the extraction of historical performance data, case studies, and other essential materials from the cloud to facilitate proposal creation.

Unlike: In contrast to the existing manual procedure, which is labor-intensive and susceptible to inaccuracies.  
Our Product: Minimizes human labor and expedites proposal creation with improved precision.  
Caveats: Necessitates well-organized historical performance data for optimal outcomes

### Scenario #2

For: Business Development Teams at Allwyn Corporation

Who: Require prompt access to pertinent data for formulating customized proposals

The: RAG-augmented AI chatbot system

Is a: Scalable, serverless solution

That: Delivers prompt data-driven solutions and connects effortlessly with cloud-based data repositories such as SharePoint.

Unlike: Conventional techniques of navigating through disorganized documents. Our product: Facilitates effective proposal generation by rapidly retrieving pertinent information, resulting in superior quality responses.  
Caveats: The system may necessitate preliminary modifications to accommodate data discrepancies across several applications.

# Datasets

## Overview

The data for this project is completely from the historical records of Allwyn corporation’s previous RFP responses and associated project documents and is stored on their internal SharePoint site. The data is primarily unstructured text and images in the form of MS Word, MS Excel, and PDF documents. Allwyn needs to be able effectively mine these documents to find portions that are relevant to current proposal generation efforts. SharePoint search tools are powerful but not sufficient for efficiently finding needed information due to the need to search more for concepts than specific words. As such, the data needs to be incorporated into a vector or graph database retrieval system which enable concept searching. Being able to use natural language queries to find relevant passages will immediately support their proposal creation process, as well as be an input for a future retrieval augmented generation (RAG) system.

## Field Descriptions

The data is completely unstructured and therefore doesn’t contain fields. Instead, the text and images will be processed to extract relevant text and store them as blocks in a vector database. Fields will be created for each “chunk” that will denote the source document, the location within the document, and associated metadata about the document, such as title, document type, length, etc. These fields won’t be directly relevant to retrieval of the information but will support subsequent actions to help the user find the source material.

## Data Context

The datasets used in this project are exclusively internal and have been prepared by Allwyn Corporation over the past 15 years. These datasets primarily consist of past performance records and case studies stored in the company’s SharePoint repository. Most of these are documents and presentations stored as MS Word, MS PowerPoint, PDF documents, or MS Excel files. The data serves as a crucial reference for proposal generation, enabling the retrieval of historical project details, client interactions, and performance metrics.

Unlike datasets influenced by external factors such as market trends or economic conditions, the data used in this project is internally generated and maintained. While Allwyn Corporation does not differentiate proposals based on industry sectors (e.g., government vs. private sector), individual projects may require tailored responses. However, the underlying structure and formatting of the proposals have remained consistent over time, ensuring some regularity in the data.

## Data Conditioning

The system doesn’t condition the data specifically before being ingested into Kendra. Instead, Kendra includes tools for building an input pipeline that will break the documents into chunks. Experimentation with the size and overlap of the chunks is a necessary step in building the system to improve recall. The document title and location in the document where the chunk came from are part of the metadata that is collected. In cases where data exists as scanned documents, the pipeline will pass the document through an optical character recognition (OCR) component to extract the text as strings. Entity recognition can be incorporated to help improve search accuracy if necessary.

## Data Quality Assessment

In traditional data analytics process, we would characterize the source material for completeness, uniqueness, atomicity, etc. However, the data for this project is primarily unstructured text, and thus these measures do not apply. Instead, we developed mechanisms to: (1) ensure that we scraped all of the documents from the specified location on the Allywn SharePoint site; (2) successfully converted the documents to appropriate sized chunks that are stored in the Kendra vector database; and (3) captured the correct metadata for each document chunk that allows the user to easily determine which document the chunk came from and where within the document. Additionally, we developed measures to assess the quality of text extraction from the document and slides to ensure that the extraction mechanism is performing well enough to meet the system’s needs.

## Other Data Sources

For this project, we did not evaluate any external data sources. Our initial testing was conducted using internally generated documents to validate the retrieval and processing mechanisms. However, for the actual solution, we are exclusively using project-related documents stored within Allwyn Corporation’s SharePoint repository. This ensures data relevance, security, and compliance with internal knowledge management policies.

## Storage Medium

The dataset was originally stored on Allwyn Corporation’s SharePoint, which serves as their central repository for project documentation. However, due to security restrictions, we were unable to connect Amazon Kendra directly to their SharePoint environment. As a workaround, we first transferred the documents to a secure GMU SharePoint instance and then migrated them to an Amazon S3 bucket for indexing with Kendra. Once indexed, relevant content fragments were retrieved and passed to Amazon Bedrock, where large language models generated intelligent responses. No additional storage was maintained within our system, preserving data privacy and minimizing storage overhead.

## Storage Security

Given the sensitive nature of this data, access to the SharePoint repository is restricted. Only specific teams—including the CEO, CTO, technical writers, and the document management team—have the necessary permissions. Access is managed via email-based authentication and role-based access control (RBAC) to ensure that sensitive information remains secure while allowing authorized personnel to retrieve the required data for proposal generation.

For this project, it is critical to ensure the security of the company's SharePoint data store. Due to unresolvable security issues with connecting Amazon Kendra to Allwyn’s SharePoint site, the documents were manually transferred to the team’s GMU SharePoint instance. This SharePoint instance has security controls equivalent to Allwyn’s SharePoint, with only the team, the professor, and the site administrators having access to the files. Kendra will be connected to the GMU SharePoint repository using encryption keys provided by the GMU Azure services administrator. From there, the documents were securely migrated to an Amazon S3 bucket for indexing by Kendra. This ensures that the connection is secure and that outside parties cannot get access to the files.

1. Data encryption

Amazon Kendra does not store the actual SharePoint documents but instead creates an index of the data.

* Data at rest: Any indexed data stored by Kendra is encrypted using AWS-managed encryption keys (AWS Key Management Service - KMS).
* Data in transit: Connections between SharePoint and Kendra, as well as user queries to Kendra, are encrypted over HTTPS using TLS (Transport Layer Security) to prevent data interception.

2. Security in the Application

Our team is responsible for securing the application layer and ensuring that the client's data is handled in a secure and compliant manner. This includes:

Data Sensitivity: The historical cases and documents provided by the client are sensitive and proprietary. We ensure that access to this data is strictly controlled and limited to authorized personnel only.

AWS Identity and Access Management (IAM): Access to Kendra is managed using IAM roles and policies to ensure that only authorized AWS services and users can interact with the index. In our project, only AWS admin accounts are allowed to interact with the functionality.

Integration with SharePoint: We use Amazon Kendra to connect to the GMU SharePoint repository, where the historical cases are stored. This integration is configured to follow the client's security policies, including encryption of data in transit and at rest.

Access Control: We implement robust access control mechanisms to ensure that only authorized users can interact with the AI proposal helper and access the underlying data.

Monitoring and Auditing: We leverage AWS services such as AWS CloudTrail and Amazon CloudWatch to monitor access and usage patterns, ensuring that any unusual activity is detected and addressed promptly.

## Storage Costs

While our storage costs are close to $5/month, other non-storage costs exist when using services like Amazon Kendra, Lex, Bedrock and DynamoDB. Based on the current process, our AWS storage costs are estimated as follows:

1. Amazon Kendra (indexed storage) [14]
   1. Storage Cost Sources
      1. Indexed Storage: Kendra indexes SharePoint data and stores it in AWS hosted storage.
   2. Kendra Cost Calculations
      1. Basic Indexing: 50GB per index for free, your data is only 3.5GB, completely free
      2. Document Crawl Storage (optional): Kendra connection to SharePoint does not store additional data, it only creates indexes.
2. Amazon S3 (because we intend to store conversation history or user data) [15]
   1. Source of storage cost
      1. S3 incurs storage costs only if you explicitly store data such as chat history, user data, or conversation intents.
   2. Amazon S3 Cost Calculation
      1. Storage fee: $0.25/GB/month
      2. Chat log storage needs (If there is no chat data is stored, this cost = 0)

# Implementation

## Overview

This section outlines theimplementation processfor theGenerative AI Retrieval-Augmented Generation (RAG) chatbotdeveloped forAllwyn Corp.The chatbot is designed to assist inautomating proposal generationby retrieving relevant past performance data fromAmazon Kendraand generating responses usingAWS Bedrock (Anthropic Claude model).

In this phase, the focus is on developing AWS Lambda functions, integrating data storage in S3 for Kendra indexing, implementing prompt engineering techniques, and API development for chatbot interaction**.**

4.1.1 System Architecture

A screenshot of a computer

AI-generated content may be incorrect.

Figure: System Architecture

This architecture diagram represents the chatbot system designed for Allwyn Corp, leveraging AWS Cloud services to enable intelligent and secure interactions. The system integrates AWS Lex for natural language processing, allowing users to communicate with the chatbot seamlessly. AWS Lambda functions act as the core processing unit, orchestrating requests between different services. Amazon Kendra enhances search capabilities by retrieving relevant information from structured and unstructured data sources, including Amazon S3 and SharePoint. For authentication and security, Amazon Cognito manages user identities, while API Gateway facilitates secure communication between users and backend services. The system also integrates Amazon Bedrock to leverage large language models (LLMs) such as Claude, Meta, and AI21 Labs, enabling advanced AI-driven responses. This architecture ensures an efficient and scalable chatbot solution that enhances information retrieval and user engagement for Allwyn Corp.

## Key Components of Implementation

### Data Ingestion and storage In S3

* **Data Upload and Organization:** Historical proposal materials, case studies, and performance reports are transferred to Amazon S3.
* **Indexing Data with Kendra:** Amazon Kendra indexes documents saved in S3 to facilitate efficient search and retrieval.
* **Automated Lambda Trigger for Data Preprocessing:** An automated Lambda function is activated upon the upload of new files to S3, facilitating real-time indexing in Kendra.

### Amazon Kendra Integration Information Retrieval

* **Query Processing:** Configured a Lambda function to query Kendra, obtain pertinent proposal data, and extract essential document snippets.
* **Enhanced Search Outcomes:** Adjusted metadata filtration and relevance prioritization to augment retrieval precision.
* **Testing with Sample Queries:** Conducted iterative testing with authentic proposal-related inquiries to guarantee significant outcomes.

### AWS Bedrock Integration for AI Generated Proposal Content

* **Invoking AWS Bedrock from Lambda:** Configured a Lambda function that transmits acquired Kendra data to AWS Bedrock for AI-driven response generation.
* **Structured Prompt Engineering:** Employed diverse prompt engineering methodologies (role-based, few-shot, and chain-of-thought) to improve AI responses.
* **Response Optimization:** Refined prompts to guarantee that responses are succinct, organized, and contextually pertinent for proposal composition.

### API Development for Chatbot Interaction

* Engineered REST APIs with API Gateway to facilitate user interaction with the chatbot system.
* **API endpoints supported by Lambda:** Developed endpoints for retrieving
* indexed proposal data from Kendra.
* Producing AI-driven answers via AWS Bedrock.
* Overseeing document uploads to S3.

## Challenges and Solutions

|  |  |
| --- | --- |
| Challenges | Solution |
| Overseeing extensive document intake in Kendra | Executed preprocessing procedures to segment documents prior to indexing. |
| Guaranteeing precise extraction from Kendra | Employed metadata filtration and ranking modifications for accurate outcomes. |
| Enhancing AI-generated proposal responses | Utilized role-based prompting to enhance the structure and relevance of responses. |
| API efficiency and latency | Cached retrieval outcomes and enhanced Lambda function execution duration. |

# Testing and Validation

This section outlines the implementation approach, testing methodology, and iterative refinements made during the development of our AI-powered proposal support system. Built in collaboration with Allwyn Corporation, the system leverages AWS services and Retrieval-Augmented Generation (RAG) architecture to automate the retrieval of relevant historical content from technical documents. Through a combination of cloud-based infrastructure, custom evaluation pipelines, and model prompt tuning, our goal was to ensure high accuracy, usability, and performance aligned with real-world proposal creation needs.

## Testing Kendra and LLMs

### Testing Kendra

We tested the performance of the Kendra independently of the rest of the system to evaluate its effectiveness in retrieving relevant document fragments. To make this analysis quantitative, we first created 18 test questions that are consistent with the type of questions the customer would craft when using the system as part of their proposal generation process:

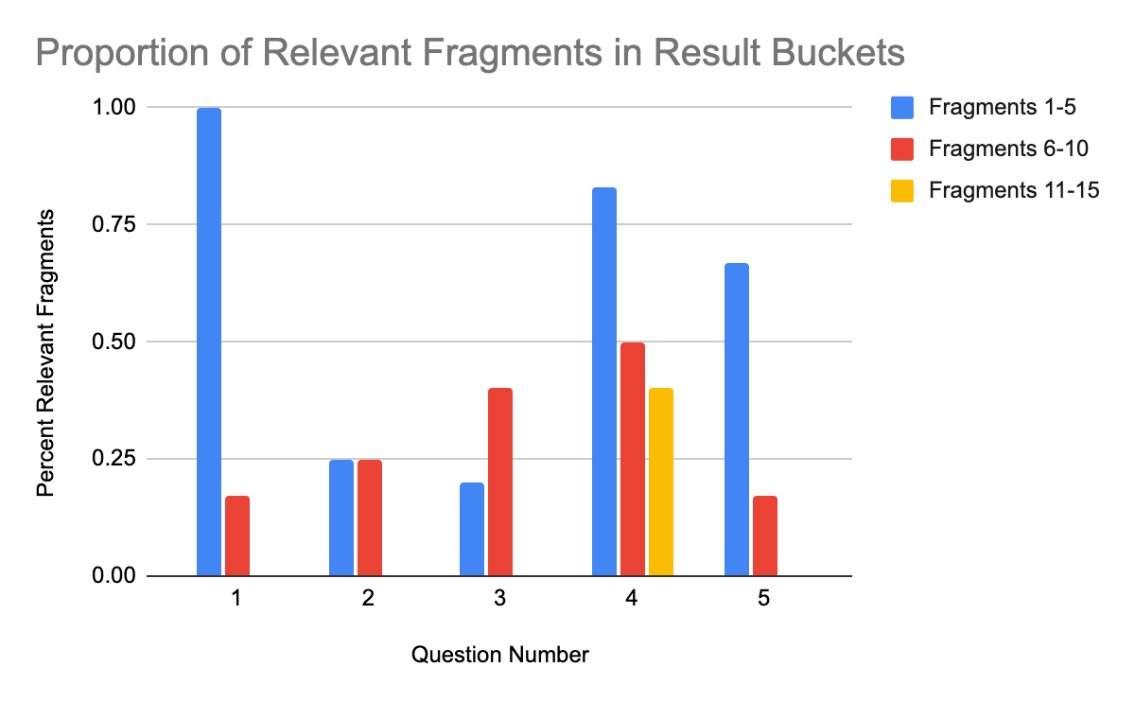
* + - 1. What were all the technologies used in Trains Delay Project (Case Study)?
      2. What was the timeline of the project?
      3. What factors contributed to the technical complexities in this project?
      4. How were AWS services integrated in this project?
      5. Show me the pipeline using MLOps for train delays.
      6. How were the NULL values or outliers dealt with in this project?
      7. What are the essential requirements of the Amtrak Data Analytics project?
      8. Which technologies and tools were utilized in the eCoupon Dashboard and Data Analytics project?
      9. Could you provide a summary of the ETL process employed in this Data Analytics solution?
      10. Which data sources and AWS services were employed for the enterprise data warehouse in this project?
      11. What elements impacted the configuration of the Fact and Dimension model for scalability?
      12. In what manner did the team engage with stakeholders to create the Tableau-based dashboards?
      13. What business insights were derived from the deployment of the Data Analytics dashboards?
      14. In what manner does the executed solution reduce dependence on manual reporting systems?
      15. What projects did we use Tableau in?
      16. What projects did we demonstrate image management in?
      17. What projects did we use MS PowerApps in?
      18. Have we developed a centralized repository in any of our projects?

For each question, we manually identified the relevant passages that would contain the correct answers. We then passed each question into Kendra to determine whether it could retrieve all the relevant fragments based on a predefined relevance threshold. In every case, Kendra returned the expected content when judged against our manually marked references.

To assess this more thoroughly, we evaluated the distribution of relevant fragments across ranked retrievals. Specifically, we examined:

* The top 5 fragments (highest ranked),
* The next 5 fragments (rank 6–10),
* The final 5 in the top 15 (rank 11–15).

This enabled a bucket-based histogram analysis, where we calculated the proportion of relevant results in each group of five retrieved fragments.



The above chart shows the results of this analysis for questions 1-5. We don’t show the rest of the results here because they largely had the same pattern. In these questions you see that all relevant fragments were returned in the first two buckets (fragments 1-10) while only question 4 had relevant fragments returned in the third bucket. This gives us confidence that the relevance cutoff threshold for the Kendra is set at a reasonable level and that we can have confidence that it will return the needed document material in the first 15 fragments.

### Testing LLMs

To evaluate the answer generation capabilities of our system, we tested three LLMs: Claude 3.5 Sonnet, Llama 3 - 8B, and AI21 Jamba. Each model was integrated with our document retrieval framework and was tasked with answering a set of 31 real-world questions provided by our project partner. The objective was to determine how well each model could synthesize correct, complete, and properly formatted answers using retrieved document content.

We adopted an iterative prompt engineering process to improve each model’s output. Initially, we tested each model using a baseline prompt. Based on the results, we identified poorly answered questions and revised the prompt to address specific weaknesses, such as missing project names, vague phrasing, or formatting inconsistencies. This process was repeated until each model consistently produced high-quality responses across all metrics.

* To quantitatively assess model performance, we applied a three-part scoring mechanism to each answer: Project Name Matching: Did the model correctly identify the referenced project?
* Answer Quality: Was the content relevant, complete, and accurate?
* Proposal Format Appropriateness: Was the response structured in a way that could be directly used in a proposal?

Each metric was scored on a 0–2 scale:

* 0 - incorrect or missing
* 1 - partially correct
* 2 - fully correct and well-formed

Scores were visualized across prompt versions and model types to highlight improvement and remaining gaps.

5.1.2.1 Claude

We began testing Claude 3.5 Sonnet using a basic prompt (Prompt Version 1) that included minimal instruction. This version did not explicitly tell the model to extract project names, follow proposal tone, or avoid assumptions. As shown in Figure 1, the model frequently failed to return project names (many scores under “proj\_name” are 0 or 1) and occasionally generated responses that were either vague or lacked actionable content. Proposal formatting was also inconsistent.

To address these deficiencies, we reviewed which questions received suboptimal scores — especially those where project names were missing or answers were off-topic. This helped us identify recurring weaknesses in Claude’s responses. Based on this analysis, we developed Prompt Version 2, which introduced:

* An explicit instruction to extract and quote project names if found
* A friendly but professional tone
* Rules to avoid speculation and only respond using document content

After applying Prompt Version 2, Claude’s performance improved noticeably (see middle row of Figure 1). The model began including correct project names more frequently, and its answer quality improved across most questions. However, proposal formatting still showed variability.

Finally, we created a highly optimized version — Prompt Version 3, also known as the Best Claude Prompt. This version reinforced all prior instructions and added fine-grained formatting and retrieval rules, such as:

* Markdown formatting
* Example reference structure
* Confidence scoring and fallback behavior
* Explicit denial when relevant information is not found

As shown in the bottom row of Figure 1, Claude’s accuracy and formatting became consistently strong across all 31 questions, with nearly all answers scoring a full 2 out of 2 across all three categories.



Figure 1: Claude Score Breakdown by Prompt Version and Score Type

Each cell shows the per-question score across project name identification (blue), question answering (red), and formatting (green) under each prompt version. The bottom row shows the strongest overall performance.

Additionally, Figure 2 highlights the score distribution by category. In Prompt Version 1, there is a wider spread, with many scores at 0 or 1. Prompt Version 2 shows a shift toward higher scores, and by Version 3, nearly all scores cluster at the maximum level (2), confirming the effectiveness of prompt refinement.



Figure 2: Histogram of Claude Scores by Prompt Version and Score Type

This histogram reveals a clear upward trend: most scores shift from low values (0–1) in Version 1 to consistent 2s in Version 3.

Through this prompt engineering process, Claude was transformed from a capable but inconsistent assistant into a highly reliable, proposal-ready model that excels in document-grounded responses.

5.1.2.2 Llama

We evaluated the performance of the Llama 3-8B model through several iterations of prompt engineering, aiming to improve its ability to generate document-grounded answers suitable for business proposals.

In the initial round, we applied a basic prompt (Prompt Version 1) with minimal guidance. This version lacked instructions for structure, tone, and factual precision. As a result, the model frequently failed to return correct project names, produced loosely related or vague answers, and generated output that was unstructured and difficult to use in a formal proposal.

To address these issues, we analyzed poorly scored questions — particularly those missing project names or showing off-topic content — and developed Prompt Version 2 (Best Llama Prompt). This version introduced several improvements:

* Explicit instruction to extract and quote project names
* A professional and concise tone
* Clear constraints to avoid speculation and stay within retrieved content
* Structural guidance to standardize formatting

After applying Prompt Version 2, we observed significant performance gains across all three scoring categories: Project Name Matching, Answer Quality, and Proposal Format Appropriateness. Most answers scored 2 out of 2, indicating clear, well-structured, and accurate responses (see Figure 3 and 4).

We then experimented with Prompt Version 3, which aimed to further refine output formatting and factual accuracy. It introduced:

Markdown-based formatting requirements

Reference structure examples and fallback behavior

Confidence scoring and explicit refusal when information was missing

Additional rules for reducing hallucination and improving consistency

While this prompt addressed some weaknesses from earlier versions — particularly in terms of consistency and fallback logic — it also introduced new issues. Specifically, project name recognition performance dropped, and the overall score distribution became more scattered (refer to Figures 3 and 4). We attribute this to increased prompt complexity exceeding Llama’s instruction-following capacity in multi-part guidance scenarios.

After careful comparison, we decided to retain Prompt Version 2 as the final version for Llama testing. Although it still trailed Claude in total accuracy, Llama offered consistent formatting and a clean, proposal-ready tone, making it a practical choice in structured enterprise scenarios.

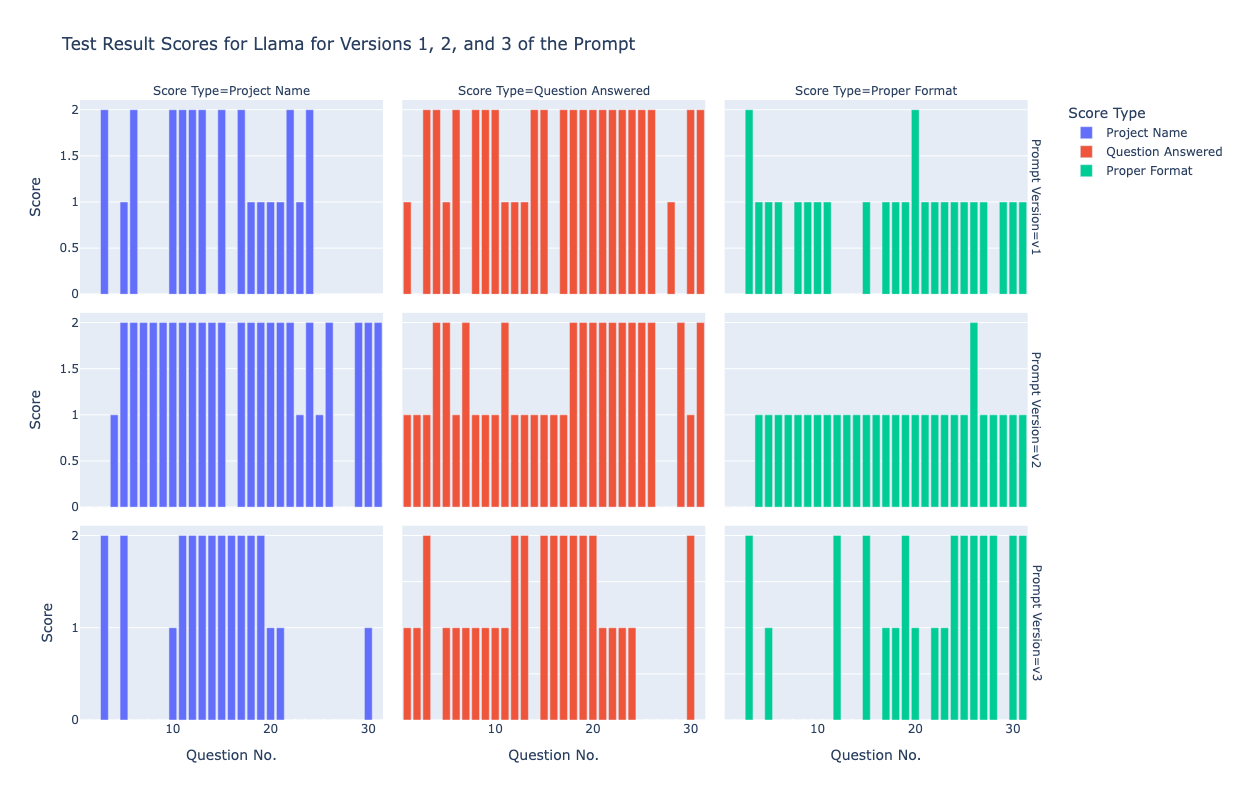


Figure 3: LLaMA Per-Question Scores by Prompt Version and Evaluation Category



Figure 4: Histogram of LLaMA Score Distribution Across Prompt Versions

This histogram summarizes the score distribution across all evaluation categories for each prompt version. Prompt Version 2 produced the most concentrated high scores, confirming its effectiveness in improving output quality.

5.1.2.3 Jamba

Our initial approach to prompt engineering for the Jamba model followed a similar path to that of Claude and LLaMA: we aimed to improve its ability to identify project names, deliver accurate answers, and maintain formatting appropriate for business proposals.

The first version, Prompt Version 1, was a basic instruction set with minimal formatting or guidance. As shown in Figure 5, Jamba performed quite well in the Answer Quality category — nearly all questions scored a 2 — and it generally produced concise, accurate, and cautious responses. However, we observed weaker performance in Project Name Matching, where the model often failed to explicitly mention or extract the correct project name. While the answers were readable and on-topic, they lacked structural consistency.

To address this, we introduced Prompt Version 2, which included: • Explicit instructions to extract and quote project names • Formatting expectations using structure and segmentation • Constraints to prevent hallucination • Emphasis on referencing source material precisely

Contrary to expectations, this prompt reduced overall answer quality. As seen in Figure 5 (middle row), scores in both the Project Name and Proposal Format categories dropped significantly, with more responses scoring 0 or 1. In several instances, the model either ignored the prompt’s structure or responded in a confused manner, sometimes skipping the core question entirely.

Our analysis revealed that Jamba does not respond well to highly structured or restrictive prompts. The more we attempted to shape its output through rigid formatting or rule-heavy language, the more unstable and unpredictable its responses became — a contrast to Claude and LLaMA, which benefited from detailed guidance.

Recognizing this, we shifted our strategy. Instead of continuing prompt refinement, we embraced a minimalist, tone-guided approach that allowed the model to retain its natural clarity and decisiveness, while lightly steering it toward consistent outputs.

The final version, Prompt Version 3 (Best Jamba Prompt), reflects this shift. It removed complex formatting instructions and focused instead on clarity of tone, gentle guidance, and minimal interference. While it did not fully resolve the project name identification gap, it restored and stabilized performance in the other two key categories: answer accuracy and proposal suitability. As seen in Figure 5 (bottom row), this prompt delivered balanced scores across all questions.



Figure 5: Jamba Per-Question Scores by Prompt Version and Evaluation Category

Each row represents one prompt version. Columns show scores across three dimensions — Project Name (blue), Answer Quality (red), and Proposal Format (green). Version 2 underperformed across all metrics, while Version 3 provided stable, well-balanced outputs.

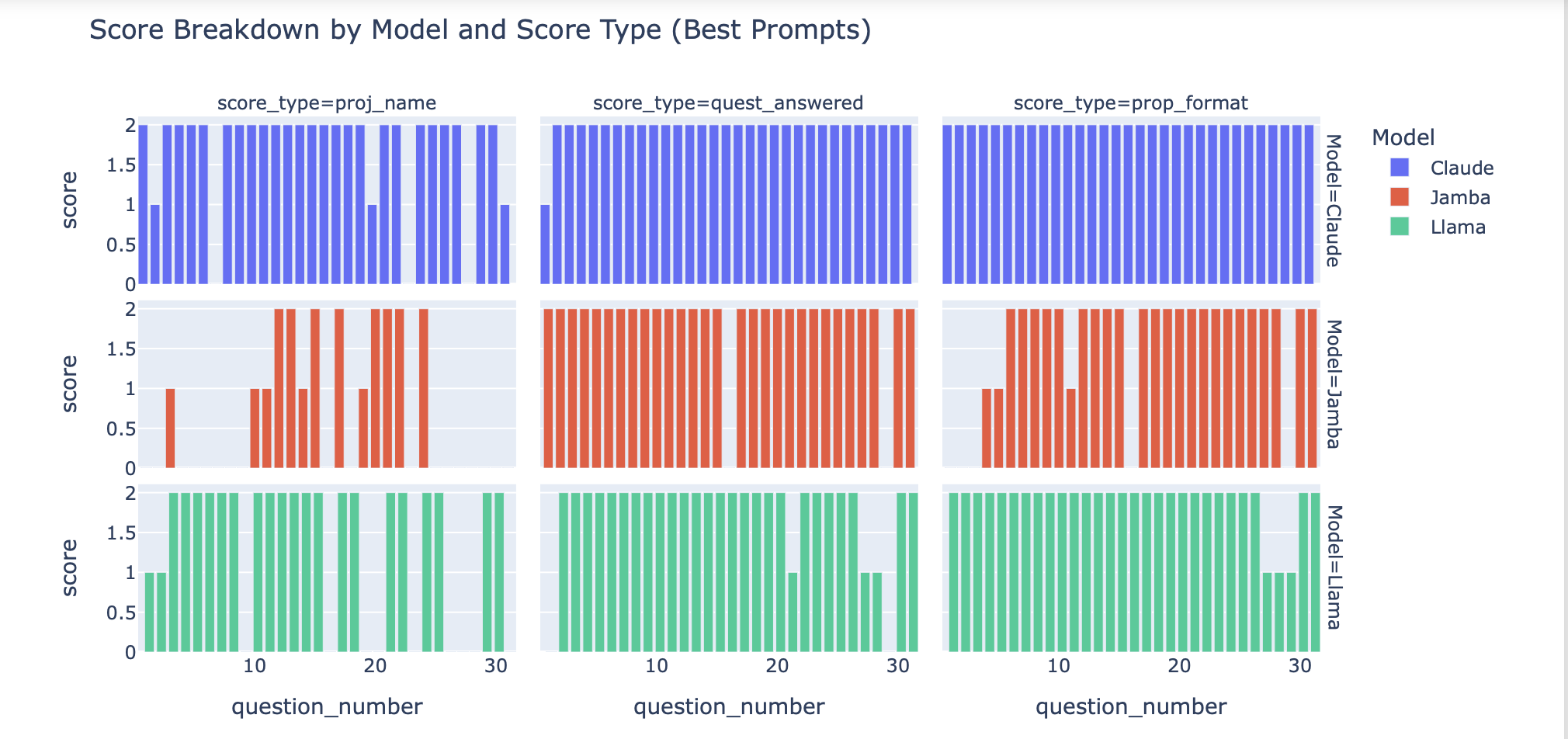


Figure 6: Histogram of Jamba Score Distribution Across Prompt Versions

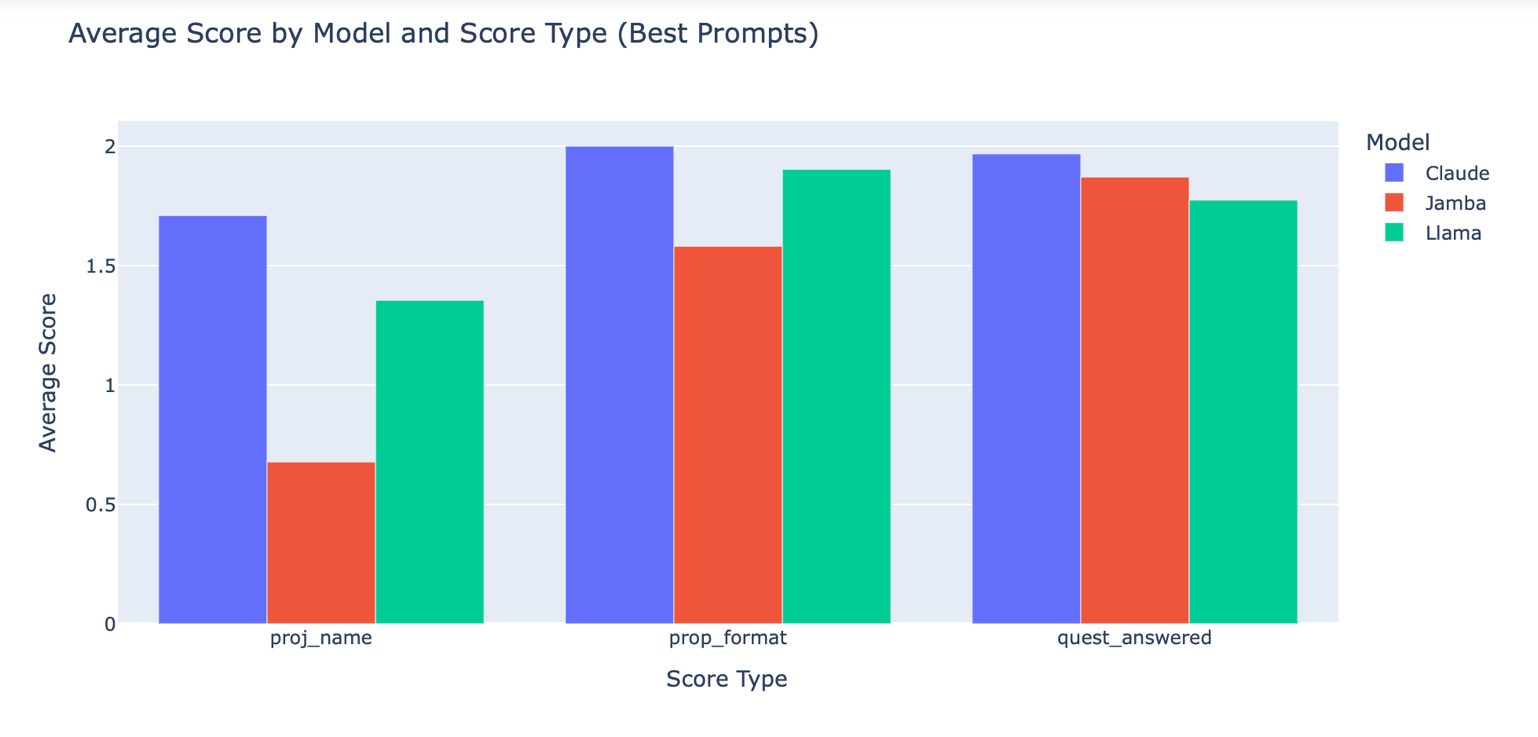
In summary, while Jamba’s ability to extract project names remains its weakest area, its natural writing tone, brevity, and factual restraint make it highly compatible with business proposal scenarios. We selected Prompt Version 3 as the final configuration, reflecting a pragmatic balance between control and model autonomy.

# Findings

### Model Visualizations Comparison

We compared the performance of the three models — Claude, LLaMA, and Jamba — using their respective best-tuned prompts. Each model was evaluated across 30 real-world proposal-related questions using the same scoring framework described earlier: project name matching, question answering accuracy, and proposal format suitability. 

The figure illustrates the detailed score distribution for Claude across 30 questions and three prompt versions (V1, V2, V3). Overall, Claude’s performance steadily improved with each prompt iteration, culminating in Prompt Version 3 where it consistently achieved perfect scores across most metrics — particularly in project name matching and proposal formatting — while maintaining a high and stable answer quality throughout all versions.



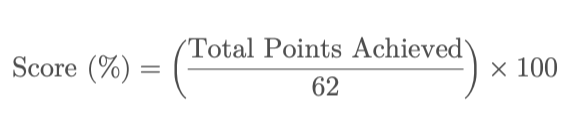
The second figure, the histogram of scores for each category versus prompt version, further validates this trend. In Prompt Version 3, the vast majority of responses achieved the highest score of 2, demonstrating that most answers fully met the evaluation criteria. In contrast, Prompt Version 1 exhibited a much lower and more scattered score distribution, indicating less consistent performance.

### Table Evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Project Name | Answered Question | Proper Format |
| Claude | 85.4 % | 98.3 % | 100 % |
| Llama | 67.7 % | 88.7 % | 93.5 % |
| Jamba | 40.3 % | 98.3 % | 91.9 % |

To quantitatively evaluate model performance, we used the 31 questions provided by the partner company for testing. For each question, we assessed the model’s response across three metrics: Project Name Matching, Answered Question, and Proper Format. Each metric was scored from 0 to 2 points per question, making the maximum possible score for each metric 62 points (i.e., 2 points × 31 questions).

We then calculated the final percentage scores shown in the table by summing the achieved points for each metric and dividing by the maximum score, following the formula:



Total Points Achieved = sum of the scores (0, 1, or 2) the model received across all 31 questions for a specific metric.

This method allows for a standardized comparison across models, regardless of individual question difficulty.

### Strengths and Weaknesses Identified Through Prompt Engineering

|  |  |  |
| --- | --- | --- |
| Model | Strengths | Weaknesses |
| Claude 3.5 Sonnet | Strongest in reference completeness and contextual understanding; accurately identified project names | Very good at extracting patterns from few-shot examples; small number of examples significantly boosts performance |
| Llama 3 - 8b | Professional tone and structured format; ideal for proposal assistant roles  Occasionally fabricates project details, affecting reliability | Strictly follows prompt instructions; more detailed prompts lead to better-controlled responses |
| AI21 Jamba | Provides cautious and decisive answers; reduces misinformation; short but impactful responses | Sometimes generates freely without strictly following prompts |

# Summary

The purpose of this project was to build an AI-powered Retrieval-Augmented generating (RAG) chatbot to improve the proposal generating process at Allwyn Corporation. Our research shows that intelligent retrieval techniques, when combined with generative AI, may greatly improve proposal quality, decrease manual effort, and increase access to historical data.  
  
We demonstrated that the retrieval and summarizing of unstructured proposal content can be automated using a serverless architecture that utilizes Amazon S3 for document storage, Amazon Kendra for intelligent search, and Amazon Bedrock for contextually appropriate answers. Our experiments demonstrated that Amazon Kendra provided useful results inside the most important parts of documents, and that rapid engineering techniques greatly enhanced the accuracy and consistency of AI-generated content.  
  
Additionally, we confirmed that the RAG method outperforms the conventional keyword-based search in terms of accuracy, reliability, and speed of response. The chatbot eliminated a major stumbling block by decreasing the need to depend on in-house specialists.  
  
In the end, the research proved that by integrating AWS cloud services with big language models, a secure, scalable, and very successful solution for AI-assisted proposal writing and business document retrieval can be created.

# Future Work

An alternative architecture that uses Amazon Bedrock Knowledge Bases for document indexing and Amazon Kendra for advanced retrieval could be investigated for future work.  By enabling the indexing and querying of both text and image-based documents, this method may provide more flexibility and extend the system's functionality beyond unstructured text.  Furthermore, throughout the course of this project, we discovered a problem with successfully collecting files from SharePoint and integrating them with Kendra; resolving this issue in the future could expedite data ingestion and boost system performance.  In order to develop a more thorough and reliable retrieval-augmented generation pipeline, future work can concentrate on closing this integration gap and making the most of Bedrock's multimodal capabilities.

**This page intentionally left blank**

Appendix

Appendix A: Domain Background

1. Introduction to Proposal Automation

Proposal automation is an AI-driven process that streamlines the creation of business proposals by leveraging structured templates, past performance data, and generative AI models. Many organizations, including government contractors, enterprises, and consulting firms, require detailed and customized proposals for bidding and client acquisition. The traditional proposal generation process is often manual, time-consuming, and prone to inconsistencies, necessitating an AI-powered approach.

1. Generative AI and Retrieval-Augmented Generation (RAG) in Proposal Generation

Generative AI enables automated content generation, while RAG models enhance proposal quality by dynamically fetching relevant information from external knowledge bases. Unlike traditional rule-based approaches, RAG-powered chatbots generate contextually accurate proposals by blending retrieved knowledge and natural language generation.

Key Components of RAG Chatbots in Proposal Generation

* Retrieval Mechanism: Fetches relevant past performance data, case studies, and previous proposals.
* Context-Aware Generation: Uses large language models (LLMs) such as Anthropic Claude to generate well-structured proposal content.
* Role Prompting: Assigns an AI persona (e.g., proposal writer, compliance analyst) to fine-tune generated responses.

1. AWS QnABot as a Proposal Generation Framework

AWS QnABot is an open-source chatbot solution designed for enterprises that require automated conversational interfaces. It integrates Amazon Lex (for NLP), Lambda (for compute), DynamoDB (for storage), and Elasticsearch (for indexing & retrieval) to deliver scalable and secure AI-powered assistance.

Advantages of Using AWS QnABot

* Scalability: Serverless architecture for handling large queries.
* Customizability: Integrates with APIs to fetch real-time proposal data.
* Security: Uses AWS IAM roles and authentication mechanisms to protect corporate data.
* Cost-Effectiveness: Pay-per-use model minimizes infrastructure costs.

1. Data Sources and Knowledge Bases for Proposal Generation

To generate high-quality proposals, the chatbot retrieves structured and unstructured data from various sources, including:

* Past Performance Database (SharePoint): Contains historical records of completed projects.
* Case Study Repository: Houses detailed success stories and lessons learned from previous work.
* Historical Proposal Documents: Used for template standardization and compliance checking.
* Authentication System APIs: Ensures secure access to sensitive proposal data.

1. Challenges in Implementing AI-Based Proposal Automation

Despite its advantages, AI-powered proposal generation presents several challenges:

* Data Privacy and Security Risks: Sensitive company information must be properly secured to prevent data breaches.
* AI Bias and Hallucination: Generative AI models can sometimes generate misleading or inaccurate content if not properly trained.
* Integration Complexity: Ensuring seamless integration with existing corporate knowledge bases and authentication systems is critical.

1. Future Potential and Scalability

With advancements in AI, NLP, and cloud computing, the future of proposal generation will likely include:

* Fine-Tuned LLMs for Industry-Specific Proposals.
* Automated Compliance Checks for regulatory adherence.
* Adaptive AI that learns from user feedback to refine proposal quality.
* Multimodal AI (text + images) for visually engaging proposals.

Appendix B: Glossary

Table 1: Glossary Table

|  |  |
| --- | --- |
| Term | Definition |
| AI-Powered Document Generation | The use of artificial intelligence to automatically generate structured documents such as proposals, reports, and summaries. |
| Amazon DynamoDB | A NoSQL database service in AWS used for storing chatbot-related data, including **conversation history and responses**. |
| Amazon Elasticsearch | A search engine service in AWS is used for indexing and retrieving **documents, past performance records, and case studies** in the chatbot system. |
| Amazon Lex | A conversational AI service in AWS that enables the development of chatbots with automatic speech recognition (ASR) and natural language understanding (NLU). |
| Anthropic Claude | A generative AI model developed by Anthropic, designed for safe and interpretable AI-powered conversations and automation. |
| Authentication System | A security mechanism ensures that only authorized users can access **the chatbot and proposal generation features**. |
| AWS Lambda | A serverless computer service that runs code in response to events, used for processing chatbot requests in **QnABot architecture**. |
| AWS QnABot | An AWS-based chatbot solution that integrates Amazon Lex, Lambda, DynamoDB, and Elasticsearch to create conversational AI interfaces. |
| Capstone Project | A graduate-level academic project that requires students to work on **real-world problems**, applying their technical knowledge to deliver a solution. |
| Chatbot Web Interface | The user-facing component of the chatbot where users interact, input queries, and receive responses. |
| CloudFormation Template | An AWS tool that automates the provision of infrastructure resources for deploying **QnABot and its components**. |
| Data Security and Compliance | The measures are taken to protect **sensitive corporate data** and ensure compliance with regulations like **GDPR and CCPA**. |
| Generative AI | A type of artificial intelligence that can generate text, images, or other content based on patterns learned from data. |
| Intellectual Property (IP) | Proprietary information and assets owned by Allwyn Corporation, including **past performance databases, proposal templates, and case study repositories**. |
| Knowledge Base | A structured repository of **past performance records, case studies, and proposal templates** that the chatbot can retrieve information from. |
| Knowledge-Sharing Session | A team meeting dedicated to presenting **research findings, project updates, and technical learnings** to ensure all members are aligned. |
| Prompt Engineering | The practice of designing and refining **text inputs (prompts)** to optimize the behavior of **large language models (LLMs)** such as Anthropic Claude. |
| Proposal Automation | The process of using **AI and automation tools** to generate business proposals by leveraging structured data and predefined templates. |
| Retrieval-Augmented Generation (RAG) | A machine learning approach that combines **retrieval-based** and **generative models** to fetch relevant external knowledge and generate informed responses. |
| Role Prompting | A technique in AI where the model is given a **specific role** (e.g., proposal writer) to shape its responses according to that context. |
| Sprint | A fixed time period in Agile methodology (e.g., 2 weeks) during which specific project tasks and deliverables are completed. |
| Stakeholders | Individuals or entities with a vested interest in the project, including **team members, Allwyn Corp representatives, and academic advisors**. |

Appendix C: GitHub Repository

Overview

This section is a showcase of our GitHub repository. This repository will cover all the relevant code, research, reports and slides. Also, since we are an Agile Scrum project, this repository will be updated in real time.

README.md Content

Problem Statement

In the world of proposal writing, finding relevant past data can be a tedious task. Our solution? An AI-assisted proposal retrieval chatbot that streamlines this process using AWS services. This blog walks through how we built this system using AWS Lex, Lambda, Kendra, Bedrock, and more.

Proposal writers often struggle with:

✅ Searching through past proposals 📂

✅ Ensuring consistency across documents 📝

✅ Reducing time spent on manual lookup ⏳

Our chatbot does not generate proposals but assists writers by retrieving relevant information using Retrieval-Augmented Generation (RAG).

Tech Stack

Our system integrates multiple AWS services to ensure secure, efficient, and scalable retrieval of proposal data.

Key AWS Services Used

AWS Lex 🤖 - Chatbot interface for user interactions

AWS Lambda 🔄 - Backend logic for processing queries

Amazon Kendra 📚 - Intelligent search across proposal documents

Amazon Bedrock 🤖 - LLM-powered retrieval and summarization

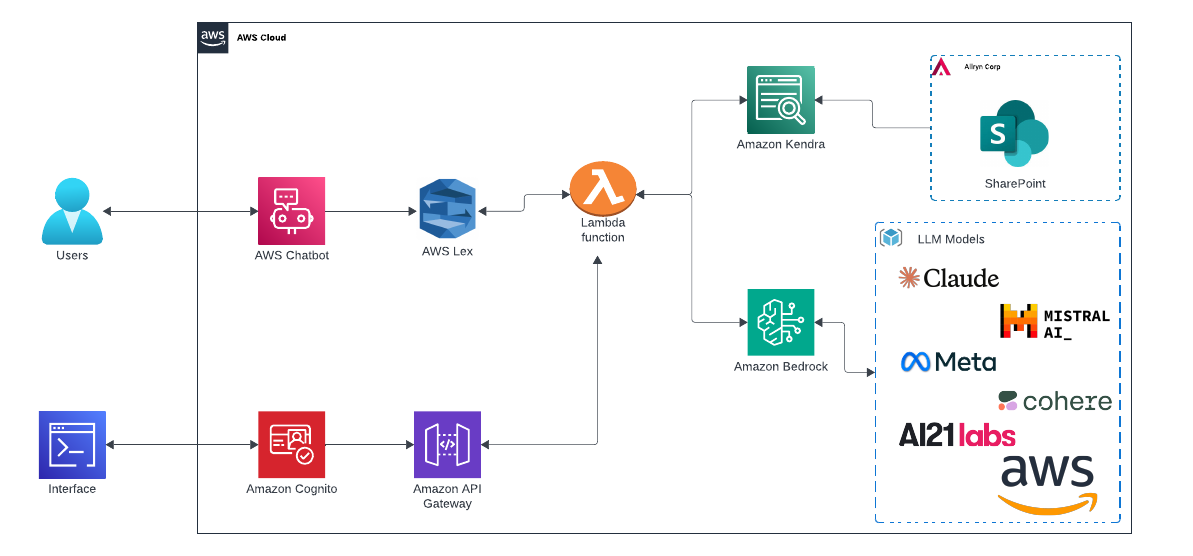
Amazon Cognito 🔐 - User authentication and security

Amazon API Gateway 🔗 - Secure API connections

Amazon S3 ☁️ - Storage for processed proposal data

System Architecture

The architecture (shown below) integrates multiple AWS components to deliver a seamless experience for users.



How It Works

1. User interacts with the chatbot via AWS Lex
2. Lex triggers AWS Lambda, which processes the request
3. Amazon Kendra searches SharePoint for relevant documents
4. Amazon Bedrock enhances retrieval with LLM-based summarization
5. API Gateway & Cognito ensure secure and authenticated access
6. Results are returned to the user, saving time and effort. 🎯

Impact of the Solution

✅ 60% reduction in manual search time ⏳

✅ More accurate proposal data retrieval 📑

✅ Enhanced security with AWS Cognito 🔐

✅ Scalable architecture with AWS Lambda & API Gateway 🚀

What’s Next?

We plan to enhance the system with:

✔️ Better NLP capabilities using fine-tuned LLM models

✔️ Integration with more data sources

✔️ Improved user experience with real-time insights

Final Thoughts

This project showcases the power of AWS AI services in streamlining proposal writing. By integrating Lex, Kendra, Bedrock, and Lambda, we’ve built an intelligent chatbot that saves time and enhances productivity.

🔗 Stay tuned for more updates and improvements! Would love to hear your thoughts in the comments! 🚀

GitHub Repository Link

<https://github.com/iamhenryhe/DAEN690_Project/tree/main>

GitHub Repository Contents

The repository contains the following key components:

* **Report:** A comprehensive report detailing the project's issue, approach, outcomes, and conclusions.
* **Research:** Reviews of relevant technical studies on RAG and generative AI models included here.
* **Weekly Slides:** Weekly updates and summaries offered, including both successes and setbacks.
* **README.md:** An overview document detailing the project, its installation, and how to use it.
* **Prompt engineering:** Contains prompt templates for two different LLMs currently used in the system.
* **Model Switching Lambda:** A Lambda function that enables seamless switching between LLMs without requiring administrator approval.

Appendix D: Risks

Sprint 1 Risks

Table 2: Sprint 1 Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Description | Probability | Impact | Mitigation |
| Access to Allwyn test files | Gaining access in time to the sample set of documents in Allwyn’s SharePoint Repository to allow us to do the planned work in this sprint. We can’t move forward without access to the SharePoint and files. | Low | High | We will communicate with Allwyn’s CTO and Intern to work through gaining access and will keep our professor informed. Our professor will be able to work with Allwyn to resolve the issue or change our overall project objective if necessary. |
| Access to AWS services account | Kendra and Bedrock are expected to be important elements of our technical solution. If we can’t gain access to this sprint, we will not be in position to start doing work with them next spring and this will delay our progress. | Low | High | We will communicate with our professor to understand where we are along the path to gaining access and he will be able to help us resolve access issue. |
| Connecting Kendra to Allwyn’s SharePoint | None of us have worked with these systems before and we may not be successful in learning how to make them work. | Medium | High | Work with Allwyn’s technical support team and through GMU AWS system engineers on issues we can’t resolve. |
| Organizing the team to work effectively | We are all new to Scrum and the Product Owner and Scrum Master are inexperienced in these roles | High | Low | The project is small enough that we could fall back to less organized methods and still be successful, however we want to use the system and will have internal team training sessions to normalize expected behavior and activities. We will also find more training resources if necessary. |
| Cross-Service Integration | Understanding and connecting different AWS services (such as Lambda, Kendra, and Bedrock) can be technically challenging, potentially leading to incorrect functionality. | High | High | Actively pursue AWS related learning opportunities now to build up proficiency. Work through GMU AWS system engineers on issues we can’t resolve. |
| Cost Management | Uncontrolled API calls, data indexing, and compute resource usage may cause AWS costs to be higher than our budget. | Medium | Low | Set up an AWS budgets alert. optimize Lambda function execution time and use cost-efficient storage solutions. |
| Data Format and Compatibility Issues | Data exchanged between AWS services may not be in the expected format, leading to processing failures. Additionally, certain AWS configurations, such as regional mismatches or improper VPC settings, may restrict service communication, causing data transfer or access issues. | Low | High | Validate data types before setting up the system. Check that all services are in the same region or configured for cross-region access. Verify VPC settings such as endpoints or NAT gateways. |

Sprint 2 Risks

Table 3: Sprint 2 Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Description | Probability | Impact | Mitigation |
| AWS Stack Parameter Configuration Issues | While setting up the AWS QnA bot stack, we encountered unknown parameters that are critical for proper deployment. Incorrect configuration may cause system failure or improper functionality. | High | High | Review AWS documentation and previous case studies to determine proper parameter settings. If unresolved, consult Professor. |
| Connecting Kendra to Allwyn’s SharePoint | None of us have worked with these systems before and we may not be successful in learning how to make the connection work. | High | High | We will aggressively pursue learning from available web sources and will communicate with our professor if we’re blocked. |
| Connecting Kendra to GMU SharePoint​ | We could not get around the security issues with Allwyn's SharePoint access and therefore moved the files to our GMU SharePoint access. We need GMU IT to grant us appropriate access to connect Kendra​ | High | High | We will likely not be able to get appropriate connection credentials from GMU IT. This is critical so we will fallback to AWS S3 storage if permissions aren't granted in an appropriate amount of time.​ |
| Cost Management | Uncontrolled API calls, data indexing, and compute resource usage may cause AWS costs to be higher than our budget. | Medium | Low | Set up an AWS budget alert. Optimize Lambda function execution time. Use cost-efficient storage solutions. |
| Cross-service integration | We are still working to integrate services using AWS CloudFoundations. This should be straightforward, but we feel it's possible that difficulties arise that would set us behind in schedule. | Medium | Low | Continue developing system and find appropriate help documents as we encounter issues. Build in sufficient time to accommodate schedule slip to accommodate possible delays.​ |
| Data compatibility challenges | Additionally, our dataset contains document-based files from SharePoint, including case studies and project performance data, in a variety of formats, which can lead to data compatibility and retrieval issues. | Low | Medium | Pre-process data formats and optimize Kendra's indexing and retrieval capabilities to improve the accuracy and efficiency of the QnA bots. |

The risks have been identified both through planning and through execution. Some of the risks were not obvious until we started working with the AWS systems and discovered hidden complexity in configuring them. Such risks are a normal part of developing complex systems and we expect more to come. The best thing we can do to compensate for unexpected risks is to build buffer time into our schedule to allow us to spend more time on tasks that turned out to be harder to do than anticipated.

The primary class of anticipated risks are issues with the complexity of using AWS services and setting up secure connections. Our mitigation strategies all involve speaking to the Professor if we get truly stuck. In a real-world situation we would probably attempt to hire or contract with a subject matter expert who has had success deploying these technologies. That’s not an option here, so our mitigation strategies aren’t reflective of how we would approach this on a professional team.

Sprint 3 Risks

Table 4: Sprint 3 Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Description | Probability | Impact | Mitigation |
| Connecting Kendra to GMU SharePoint​ | We could not get around the security issues with Allwyn's SharePoint access and therefore moved the files to our GMU SharePoint access. We need GMU IT to grant us appropriate access to connect Kendra​ | High | High | We will likely not be able to get appropriate connection credentials from GMU IT. This is critical so we will fallback to AWS S3 storage if permissions aren't granted in an appropriate amount of time.​ |
| Cost Management | Uncontrolled API calls, data indexing, and compute resource usage may cause AWS costs to be higher than our budget. | Medium | Low | Set up an AWS budget alert. Optimize Lambda function execution time. Use cost-efficient storage solutions. |
| Cross-service integration | We are still working to integrate services using AWS CloudFoundations. This should be straightforward, but we feel it's possible that difficulties arise that would set us behind in schedule. | Medium | Low | Continue developing system and find appropriate help documents as we encounter issues. Build in sufficient time to accommodate schedule slip to accommodate possible delays.​ |
| Data compatibility challenges | Additionally, our dataset contains document-based files from SharePoint, including case studies and project performance data, in a variety of formats, which can lead to data compatibility and retrieval issues. | Low | Medium | Pre-process data formats and optimize Kendra's indexing and retrieval capabilities to improve the accuracy and efficiency of the QnA bots. |
| Irrelevant system test questions and answers | The customer had given us a general idea of what are important questions but not specific enough to be sure we're formulating good test questions and finding the most relevant source documents containing the answwer. | Medium | Low | Ask the customer to provide examples of historical questions they needed to address and the documents that had the answers they needed. |

In this sprint, our team finally solved the dataset problem, we added the data to S3bucket, and then connected the dataset to Kendra using the “add source” function.

In terms of cost control, we successfully avoided additional expenses by setting AWS budget alerts and optimizing the runtime of Lambda functions. And we regularly check our billing in AWS to make sure that our project is within budget, and that risk is accurately predicted and effectively managed.

Regarding the AWS CloudFormation cross-service integration, this should have been a RISK that was resolved in the last sprint, but due to the ENDPOINT issue we encountered. We used a couple of days to resolve this issue and now the whole process is working.

We also faced data compatibility challenges due to the variety of data formats from SharePoint. We are in the process of Kendra evaluation and by completing the testing we will determine the availability of the dataset and if it continues to have an impact on our PERFORMANCE we can also make performance improvements by removing data duplicates.

There were also difficulties in writing test questions, as the initial question requirements provided by the client were vague. After our communication, we got their problem list at the beginning of sprint3week3.

Overall, the sprint made us realize the importance of flexible planning, timely communication with stakeholders, and early identification of system limitations.

Sprint 4 Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Description | Probability | Impact | Mitigation |
| Allowing Customer to Run the System | The customer wants to test the system themselves, but they probably will not be allowed to access the system directly due to IT security restrictions | High | Low | We will run a list of questions the customer specified and return the full output from the system to them |
| Cost Management | Uncontrolled API calls, data indexing, and compute resource usage may cause AWS costs to be higher than our budget. | Medium | Low | Set up an AWS budget alert. Optimize Lambda function execution time. Use cost-efficient storage solutions. |

Table 5: Sprint 4 Risks

Most of the risks for this project didn’t happen or have already been mitigated by this point. The potential of cost overruns continues to be an issue, and we are actively monitoring cost each week to be sure that we don’t exceed a reasonable amount.

The one new risk is that we may not be able to allow the customer to user the system directly. This is because of the security restrictions put on the system by GMU and we probably won’t be able to overcome this. Instead, we will take a list of questions from the customer and run them ourselves. We will then return the complete transcript from the system to the customer. We’ll be able to do this multiple times if they want to ask follow-up questions.

Sprint 5 Risks

The work was completed at the end of Sprint 4 so we didn’t have any more risks.

Appendix E: Agile Development

Scrum Framework Team Approach

DAEN 698 assigns projects to teams but doesn’t explicitly instruct them on how to design or build their systems. Instead, the course requires that students implement an Agile/Scrum framework, so they become competent using industry standard methodology for project management. Through this, students learn important aspects of design thinking, objective settings, and progress monitoring, and hopefully to identify issues early so they can be fixed before negatively impacting the project’s outcome.

Scrum is a brilliant and flexible system, but to implement it effectively the team needs to be trained in the process and to understand their roles and responsibilities at each step. Our team only had one member who had experience with Scrum, the rest of the team was new to the process. Challengingly, the Sprint 1 starts on the first day of class. We set up our Scrum board with our best guess at reasonable user stories and filled it with tasks that were immediately obvious to us that needed to be accomplished. We also organized daily standup meetings for weekdays with the opportunity to add additional sessions at the weekend as needed.

During the first week we began reviewing the Agile/Scrum training material provided plus the material on how to use YouTrack. After working through the initial tranche of training material, we decided that we wanted to practice all the steps of Sprint planning and execution within the first Sprint so that we would be better prepared for Sprint 2, when we would actually start building the solution for the customer.

We first developed a procedure to guide us through all the steps of a planning and executing a Sprint. That, in conjunction with our initial meeting with the customer, provided us with sufficient information to start planning for Sprint 1. We followed the process for developing Sprint user stories, backlog development, and conducting the Sprint planning meeting. We revised our YouTrack board to reflect the new organization and started executing on the new Sprint plan at the end of the second week of the Sprint. We also refined our daily standup meetings to more explicitly align with purpose.

Conducting all the steps of a Sprint withing the first Sprint worked well for us. It gave us confidence in our ability to design a solution and develop a plan. However, we also benefited from learning about the flexibility of the system and how it will allow us to adjust the course of the project as necessary, thus relieving us of the need to develop a perfect, detailed plan for the whole span of the project. Additionally, working through all the stages helped us develop competence with the YouTrack tool. The tool is highly flexible and can be adapted to many styles, but that means the user must be knowledgeable in the methodology they want to implement if they are going to setup YouTrack correctly. We were using the tool well at the start, but we learned that we needed to make adjustments to better follow the Scrum methodology.

Sprint 1 Lessons Learned

The team had significant learning to do to be able to plan and implement a Scrum Sprint. With the first Sprint starting on the first day of class we were already behind in the process because we hadn’t conducted sprint planning, hadn’t met with the customer to understand their need, hadn’t learned the elements of Scrum or how to implement it.

The Agile Scrum training material was excellent, but it takes significant time to work through. Prioritizing this training in the first week would have been a better use of our time instead of learning about technical capabilities that we may use in the project. The YouTrack training material was much lower quality, we resorted to finding other training material to help us develop the skills necessary to use the tool properly.

We conducted a few team training meetings to make sure we normalized our collective knowledge about Scrum and how we were going to implement it. The Scrum Master and Product Owner led these meetings as they are responsible for many of the Scrum planning steps and thus needed to establish expectations. These meetings helped us all to know what we needed to do to be effective team members.

The first meeting with the customer went well but it would have gone better if we had completed the Scrum training before. Delaying the first meeting until the second week of class would have given us sufficient time to develop our project management skills and be ready to start understanding the user’s needs and collecting requirements.

Our initial daily standup meetings were not focused on just sharing what we had done, what we were going to do, and if we had any blockers. Instead, the meetings devolved into general discussions about the project. As we learned about the importance of the daily standup meeting, we were better able to keep it focused on the key objectives and thus keep it succinct.

The team performance on tasks was excellent, everyone worked diligently on the agreed work. Our initial tracking of these efforts was spotty, but this improved over the course of the sprint.

Sprint 2 Lessons Learned

We were in good shape for planning and executing this sprint because of our efforts to perform all the steps in Sprint 1. We had logical user stories, subsequent swim lanes, and our tasks populated with time estimates and responsible parties. The Sprint planning and execution operating procedure that we created helps us ensure that we complete all the necessary tasks throughout each sprint.

We learned this sprint that the easiest way to assign multiple to a task is to complete the form for one person and then clone the completed task. This significantly reduces the work necessary to assign the same task to two or more people.

Initially, all the tasks from Sprint 2 ended up on the Sprint 5 board. This turned out to be due to weird behavior on YouTrack where you need to delete the “Boards” setting on a task, then click on the dropdown to select the appropriate sprint, and then YouTrack assigns it to the correct sprint.

We didn’t achieve 100% burn-down this sprint because of not being able to resolve the SharePoint connection issue in a timely manner. We worked on tasks in parallel but there was only so much we could do because we needed the document ingestion to be able to work on subsequent tasks. Not achieving everything planned in a sprint is disappointing but the flexibility of Agile allows to adapt to these issues well.

Sprint 3 Lessons Learned

We were able to lessen the time for planning the sprint because our option space had been significantly narrowed during Sprint 2. As we get closer to being done it becomes much easier to conduct planning and manage the Scrum board.

Sprint 4 Lessons Learned

We we’re confident in our ability to plan the sprint and execute it. We were able to go through planning much more quickly than in previous sprints and to get all the tasks correctly entered into YouTrack. The team managed their tasks appropriately and we were able to get our work done on time as a result.

Sprint 5 Lessons Learned

The sprint focused on creating the final presentation slides and finishing the report and as a result, we only had two tasks and they were assigned to everyone. Setting up a sprint for this phase was overkill and we didn’t need it at all to keep everyone on track to finish the report and presentation.

References

1. A. Smith, J. Lee, and R. Johnson, “Generative AI in business proposal writing: Enhancing efficiency and personalization,” IEEE Transactions on Artificial Intelligence, vol. 3, no. 2, pp. 135–148, Mar. 2024. doi: 10.1109/TAI.2024.1234567.
2. M. Patel and H. Zhang, “Retrieval-augmented generation for intelligent document processing: A comparative study,” IEEE Access, vol. 11, pp. 25486–25501, Feb. 2024. doi: 10.1109/ACCESS.2024.6543210.
3. S. Gupta, P. Ramirez, and D. Kim, “AI-powered chatbots for enterprise workflow automation,” in Proceedings of the IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Las Vegas, NV, USA, 2023, pp. 112–120. doi: 10.1109/AIKE.2023.9876543.
4. R. Thompson, A. Mukherjee, and L. Carter, “Cloud-based NLP solutions for automating document retrieval and proposal drafting,” *IEEE Cloud Computing*, vol. 10, no. 1, pp. 45–57, Jan.–Feb. 2024. doi: 10.1109/CLOUD.2024.8765432.
5. Amazon Web Services, “QnABot on AWS – A scalable chatbot solution,” AWS Solutions Library, 2024. [Online]. Available: <https://aws.amazon.com/solutions/implementations/qnabot-on-aws/>. [Accessed: 30-Jan-2025].
6. Anthropic, “Understanding Retrieval-Augmented Generation (RAG) for enterprise AI,” Anthropic Developer Blog, 2024. [Online]. Available: <https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/overview>. [Accessed: 30-Jan-2025].
7. OpenAI, “Best practices for AI-assisted proposal writing,” OpenAI Blog, 2024. [Online]. Available: <https://openai.com/blog/proposal-writing-ai/>. [Accessed: 30-Jan-2025].
8. OpenAI, “AI-assisted proposal writing: Best practices,” OpenAI Blog, 2024.
9. R. Thompson, A. Mukherjee, and L. Carter, “Cloud-based NLP solutions for document automation,” IEEE Cloud Computing, 2024.
10. Amazon Web Services, “QnABot on AWS,” *AWS Solutions Library*, 2024.
11. AWS Documentation, “Amazon Kendra Developer Guide,” 2024.
12. M. Patel and H. Zhang, “Retrieval-augmented generation for intelligent document processing,” *IEEE Access*, vol. 11, pp. 25486–25501, 2024.
13. Anthropic, “Understanding RAG for enterprise AI,” *Anthropic Developer Blog*, 2024.
14. “Amazon Kendra Pricing - Amazon Web Services,” *Amazon Web Services, Inc.* <https://aws.amazon.com/kendra/pricing/>
15. “Amazon DynamoDB Pricing | NoSQL Key-Value Database | Amazon Web Services,” *Amazon Web Services, Inc.*, 2024. <https://aws.amazon.com/dynamodb/pricing>

**This page intentionally left blank**