

# **BLOOMQUEST: A PERSONALIZED LEARNING PLATFORM**

Wijayasena W.D.N.D.T

(IT20133368)

BSc (Hons) in Information Technology  
Specializing in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology  
Sri Lanka

February 2023

# **BLOOMQUEST: A PERSONALIZED LEARNING PLATFORM**

Wijayasena W.D.N.D.T

(IT20133368)

Dissertation submitted in partial fulfillment of the requirements for the Bachelor of  
Science in Information Technology Specializing in Data Science

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

February 2023

## **Declaration**

“I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)”.



10.09.2023

Signature:

Date:

The above candidate has carried out research for the bachelor's degree Dissertation under my supervision.

Signature of the supervisor:

Date

## **Abstract**

University students face enormous pressure and stress when it comes to their academics today. Many students find themselves struggling to keep up with a wide range of subjects to cover and a huge number of assignments and exams. Some students do not have enough time to refer to all lectures, some students may not know how to study efficiently in order to achieve good grades. Many students try to do self-study to stay on the correct track. But unfortunately, students don't have a good idea about how to study effectively on their own and how they want to focus on each and every section to obtain higher results. This can lead to poor performance and even more stress. So, students need to be able to develop good self-study habits, but that can be tough to do without help.

There are different types of self-study methods. Self-reading is the main part of self-study since it allows students to grab and process information independently at their own rate and depth. While performing self-reading, students may encounter obstacles such as difficulty understanding the reading content or the need for additional clarification. In these cases, learners will look for extra study materials on the internet.

However, it would be impossible for them to locate the most appropriate materials for both the learner and the query passage that user given to the system. To overcome this problem, this strategy encourages enhanced self-directed learning by providing students with tailored learning opportunities based on the difficulty level of user's study material that they refer to and employing modern ranking systems.

## **Acknowledgement**

Embarking on this research journey, I found myself surrounded by an immense pool of support, wisdom, and encouragement. It is with profound gratitude that I acknowledge those who played a pivotal role in shaping my thesis.

First and foremost, I express my deepest appreciation to my supervisor, Mr. Prasanna Sumathipala, whose unyielding guidance, patience, and expertise steered me through the complexities of my research. Your insightful feedback was the cornerstone of my journey. To my co-supervisor, Mr. Sathira Hettiarachchi, for all the invaluable advice.

To all the members of my research group, I extend my heartfelt gratitude. Your camaraderie, discussions, brainstorming sessions, and collaborative spirit made the intricate research process an enriching and enjoyable one.

I am also deeply indebted to the Research Panel for their time, expertise, and invaluable feedback. Your keen observations and suggestions have immensely contributed to refining my research work.

## Table of Contents

Declaration .....	i
Abstract .....	ii
Acknowledgement.....	iii
List of Figures .....	vi
List of Tables .....	vii
1. INTRODUCTION .....	1
1.1. Background Literature.....	4
1.2. Research Gap.....	7
1.3. Research Problem.....	8
1.4. Research Objectives .....	10
1.5. Sub Objectives.....	10
1.6. Specific Objectives.....	11
2. METHODOLOGY .....	12
2.1. Requirement Gathering .....	12
2.2. Methodology of Online Resources Recommendation Component.....	12
2.3. Implementation.....	17
2.4. Testing .....	24
2.5. Commercialization Aspect of the Product.....	27
3. RESULTS AND DISCUSSION.....	29
3.1. Results .....	29
3.2. Research Findings .....	30
3.3. Discussion .....	31
4. SUMMARY OF EACH STUDENT’S CONTRIBUTION.....	34
4.1. Generate Mind-Maps Utilizing The Student’s Study Materials. (Member1 : IT20133504).....	34
4.2. Generate Questions Mapped To Bloom’s Taxonomy Levels With Answers Utilizing The Student’s Study Materials. (Member2 : IT20126438) .....	35
4.3. Track And Predict Student Performance. (Member3: IT20123468) .....	36
4.4. Recommend Extra Study Recourses (Documents, Videos). (Member4 : IT20133368 – My Self).....	37
5. CONCLUSION.....	38
6. REFERENCES.....	39

7. APPENDICES.....	41
--------------------	----

## List of Figures

Figure 1: BloomQuest Overall System .....	2
Figure 2: User Response for having for High-quality Resource Recommendation System.....	3
Figure 3: User responses for having issue in existing recommendation systems. ....	9
Figure 4: User response for having a self-learning system which consists of our main objectives.....	10
Figure 5: Resource Recommendation Component Diagram. ....	13
Figure 6: Preprocessed Paragraph.....	18
Figure 7: LDA Model Basic Architecture .....	18
Figure 8: Generated Set of Topics.....	19
Figure 9: Content Bucket .....	20
Figure 10:Flesch Reading Ease Formula .....	21
Figure 11: YouTube Video Metrics Score .....	22
Figure 12: Combine Formula of Relevance Score and Metrics Score.....	23
Figure 13: Overview of mind-map component.....	34
Figure 14: Overview of quiz component. ....	35
Figure 15: Overview of performance predicting component.....	36
Figure 16: Overview of resource recommender component.....	37



## **List of Tables**

Table 1: Difference Between Term-Based and Topic-Based Ranking .....	15
Table 2: Tools and Libraries.....	17
Table 3: FRE Score Ranges .....	21
Table 4: Results Summary .....	29

## 1. INTRODUCTION

In our research (Bloom Quest) initially our main focus is on university students. University students face enormous pressure and stress when it comes to their academics today. Many students find themselves struggling to keep up with a wide range of subjects to cover and a huge number of assignments and exams. Some students do not have enough time to refer to all lectures, some students may not know how to study efficiently in order to achieve good grades. Many students try to do self-study to stay on the correct track. But unfortunately, students don't have a good idea about how to study effectively on their own and how they want to focus on each and every section to obtain higher results. This can lead to poor performance and even more stress. So, students need to be able to develop good self-study habits, but that can be tough to do without help.

This document addresses the pressing need to enhance undergraduate learning through a more personalized and self-directed approach. While there are existing systems that facilitate self-learning by allowing students to search for topics of interest, our system goes a step further by empowering students to take control of their learning experience. Here, we are researching a self-learning system by adding modern and advanced techniques to traditional self-learning systems. The project is based on a personalized self-learning system that helps students to understand and apply Bloom's Taxonomy in their own learning process. The system gets the student's own study material as an input and generates questions and answers according to Bloom's Taxonomy from that material, system can measure student performance and show it in a dashboard. Another option is the system provides extra study resources based on student's study material or given content paragraph. Also, the system generates a comprehensive mind map to understand the uploaded material in a convenient way. Figure 1 depicts the overall system of the BloomQuest project.

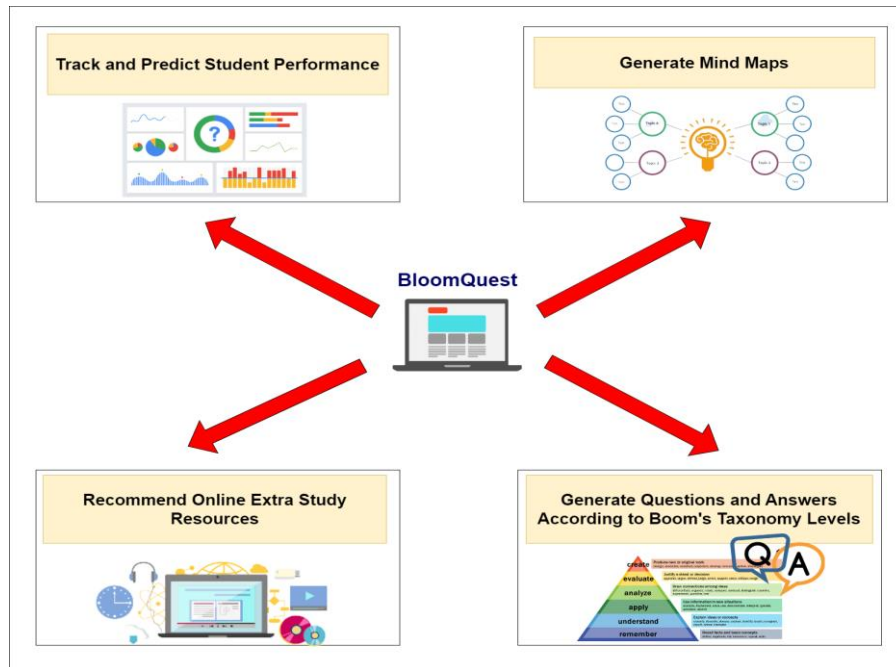


Figure 1: BloomQuest Overall System

The major goal of this project component is to provide an online extra resource recommending system that suggests appropriate and engaging resources based on the content and reading level of the user . This approach promotes self-directed learning by giving students individualized learning opportunities based on their level and using modern ranking methods. The aim of this project is to develop an online resource recommendation system that uses advanced technologies to recommend the most relevant and engaging resources to enhance the students' learning experience. This research component's final product is a fully completed web application that recommends extra study resources for students. Figure 2 shows how much students like to have a first-rate system for recommending online resources.

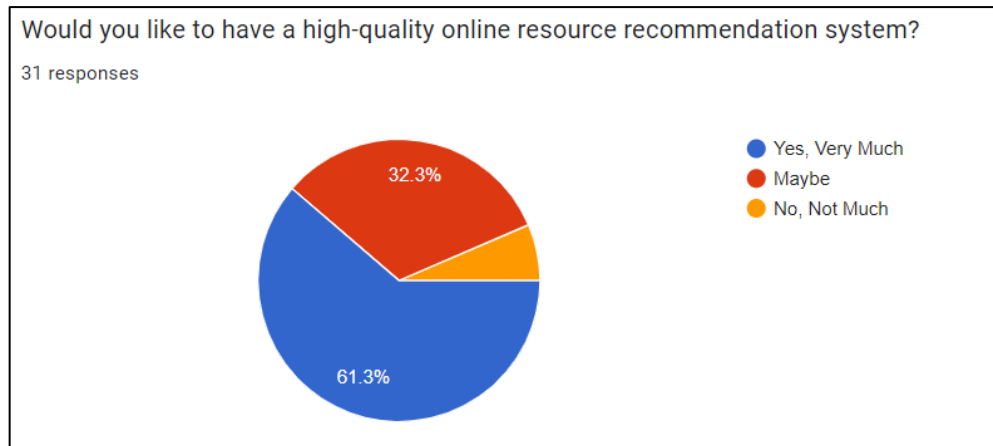


Figure 2: User Response for having for High-quality Resource Recommendation System.

The system utilizes a paragraph that in the student's study material as input, The system processes this input through a query processing section that conducts preprocessing, topic generation, and topic compression. It uses the Latent Dirichlet Allocation (LDA) topic modeling algorithm for topic generation and the cosine similarity method for topic compression. A topic model is based on the premise that when a document is about a specific topic, certain words should occur more frequently. Documents are topic mixtures, with each subject being a probability distribution over words. The generated topics are then used to retrieve relevant resources, considering the top k words in each topic. The system retrieves URLs of resources and ranks these resources based on their relevancy and difficulty level. The ranking system leverages the relevance ranking method and the Flesch Reading Ease method for assessing the difficulty of documents, while videos are evaluated using semantic analysis. These ranking scores are combined to provide the most relevant resources for the user. This research component's final product is a fully completed web application that recommends extra study resources for students.

### **1.1. Background Literature**

Online resource recommendation systems have drawn a lot of attention from researchers in recent years who are investigating various technologies to propose the best resources to users. The methodologies and tools that researchers used to create online resource recommendation systems are described in the section that follows.

In research done by Xin Wei et al in 2021 investigate the creation of a personalized extra learning resource recommendation system that enhances students' learning results by utilizing educational psychology theory and artificial intelligence technologies [1]. It discusses the difficulties in delivering relevant learning resources in online learning and suggests an algorithm for recommending learning resources based on LinUCB. The algorithm modifies the proportion of exploration and exploitation during recommendation using a unique exploration coefficient.

In research done by Hongtao Sun et al in 2022 proposed a learner-model-based collaborative filtering recommending approach for online study resources, which takes into account the traits and actions of learners when making recommendations in order to improve the accuracy of the content that is suggested [2]. In order to build learner models and to calculate the user similarity this paper proposed methods such as machine learning algorithms and learner modeling algorithms. The learner's evaluation of resources is revealed by how they use online learning resources, and the algorithm used to generate suggestions is based on how similar the learners are to one another and well rated regarded the resources are.

Another research done by Danyang Shen in 2020 proposed a neural network-based recommendation model that uses concept maps to represent knowledge points, learning behavior based on learning style scales, and Bloom's taxonomy to rank educational materials [3]. Then the score is calculated using the multilayer perceptron network based on the created embedding vectors. Studies carried out for this study's experiments showed that this recommendation model produced excellent results.

In research done by Raghad Obeidat, Rehab Duwairi and Ahmad Al-Aiad in 2019 also done research reading this topic [4]. In here, depending on commonalities in their prior course experience, a collaborative recommendation system for online courses has been developed. To identify trends among the courses, this system uses data mining techniques. Then, based on the students' behavior, clustering algorithms for the datasets use to put similar students into the same cluster.

Another research done by Gina George and Anisha M. Lal in 2019 proposed an ontology-based resource recommender system [5]. Web Ontology Language (OWL) can be use to represent ontology in a model. An ontology can serve as a database's equivalent of a XML schema. There, feature extraction, pattern classification utilizing ontology mapping, and ontology merging are used as methodologies to uncover and enhance the discovery of sets of related learning materials.

In research done by Bhaskar Mondal et al in 2020 suggest a machine learning strategy for recommending relevant online courses depending on student performance history [6]. A new student is initially categorized using the k-means clustering technique by the framework based on their prior performance. The cluster will use collaborative filtering to suggest appropriate courses.

Another research done by Jun Xiao, Minjuan Wang, cBingqian Jiang and Junli Li in 2017 developed a personalized resource recommending system for online learning that makes use of association rules, content screening, and cooperative filtering to make suggestions for relevant learning resources to students taking online courses [7]. The authors highlight the value of personalized learning, which has grown more significant with the quick growth of online and mobile technology.

In research done by Honggang Wang & Weina Fu in 2020 developed a dynamic collaborative filtering algorithm-based strategy for recommending unique learning resources [8]. The authors suggest using dynamic k-nearest-neighbor and the Slope One method to optimize collaborative filtering algorithms in order to address the issues of sparse data and low scalability. They also examine the network's learning resource data sparsity in considering the outcomes of neighbor selection.

In research done by Ronghua Shi, Lei Mao, Chao Hu and Sixiang Lire in 2018 provides a novel method for recommending educational resources that is based on the learner's current knowledge structure [9]. The suggested algorithm seeks to offer students individualized recommendations that are in sync with their interests while also considering their current knowledge base. The proposed method was tested in a controlled experiment by the authors, and the findings indicate encouraging gains in both the correlation score and learning process score.

Another research done by Roshan Bhanuse and Sandip Mal in 2021 focuses on a thorough analysis of deep learning-based e-learning recommendation systems (RS) and the problems with their accuracy, scalability, cold-start, and data scarcity [10]. The authors have noted that as the use of online learning materials has increased, it has become harder for students to sift through vast amounts of data for specific knowledge. The authors emphasize the significance of learning analytics (LA) and educational data mining (EDM) in the development of e-learning RS and contend that deep learning algorithms have the potential to improve RS's efficacy in individualized instruction.

## **1.2. Research Gap**

Referring to the research papers that we found, we learned that this research has been the subject of active research in fields where researchers have worked harder to suggest quality research. A personalized online resource recommending system provides many features to improve student's knowledge based on their individual needs and preferences. Developing a web application for this kind of system could have a commercial advantage. As a research team our team members come up with novel ideas which are not included in previous research papers. We try to give a comprehensive personalized self-study system with many novel options and features.

For this research, the component that discussed in this paper is recommending online extra resources to the users according to the given content passage. According to our experience we know that when we search a lengthy query in many existing resource recommendation systems it will not support well and not give proper output as our expectation. To answer this problem, the proposed system suggests dividing long queries into small topics and executing those small query topics to retrieve resources.

Many resource recommendation systems that are currently available do not give both document type and video type resources for users separately. This proposed system provides both documents type and video type resources and users can select when they input the passage whether they want document type resources or video type resources.

The primary difference between the proposed study and previous research initiatives is that there are many of them that use common information retrieval models to measure the relevance such as bag of words. The proposed system's approach to measuring relevance involves analyzing the topics that are present in both the query and the available resources. Another main gap in our system is we analyze the reading difficulty of the user-submitted passage and suggest resources with similar reading difficulty.



### **1.3. Research Problem**

It is essential to create well-balanced and high-quality exams for undergraduate students, and one popular framework that many lecturers use to assess students' intellectual abilities and skills is Bloom's Taxonomy cognitive domain. However, many students are not aware of Bloom's Taxonomy and how it affects their learning experience. Because of this, students may miss important opportunities for growth and development in their academic pursuits. To address this challenge, a personalized self-learning system could be developed to assist students in comprehending and applying Bloom's Taxonomy to their own learning process.

Self-reading has become a major part of self-studying. Students may have trouble when understanding content while self-reading or possibly wish to understand more about the topics it covers. In such case people use recommending systems to search and retrieve related details. However, Students often struggle to find the right resources for self-study and may be discouraged by materials that are either too difficult or not relevant to their needs. During my investigation, I discovered that several of the existing recommendation systems are not capable of handling lengthy queries effectively, leading to erroneous or inadequate search outcomes. Additionally, when these systems receive a large number of resources, they may not rank them appropriately, causing students to refer to resources that do not contain the information they require. Figure 3 depicts how many users experience issues in accuracy or relevance when using existing resource recommendation systems.

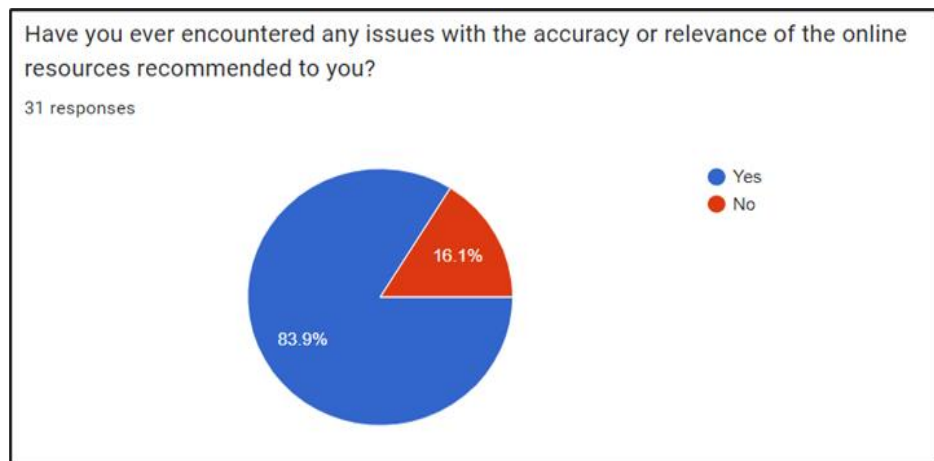


Figure 3: User responses for having issue in existing recommendation systems.

Since there are some doubts in existing recommendation systems, this paper raises the question “How to provide a method for students to retrieve more relevant and important extra online resources?”. This research component will present several solutions to address these issues and establish an efficient online resource recommendation system for users.

#### 1.4. Research Objectives

The main objective of this research is to create a customized self-study platform to support undergraduate students. Unlike a typical self-study system, this platform distinguishes itself by incorporating the students' study materials and helping them evaluate their performance in a specific subject based on the various levels of Bloom's Taxonomy.

#### 1.5. Sub Objectives

To achieve this objective, it is further divided into four sub-objectives. They are,

1. Generate a comprehensive mind map for a given study material.
2. Generate a set of questions and answers from the study material and categorize them according to Bloom's taxonomy.
3. Track and predict student performance in a specific subject.
4. Provide online extra study resources related to the uploaded study material.

Figure 4 shows user feedback for having a fully fetched self-learning system consisting above objectives.

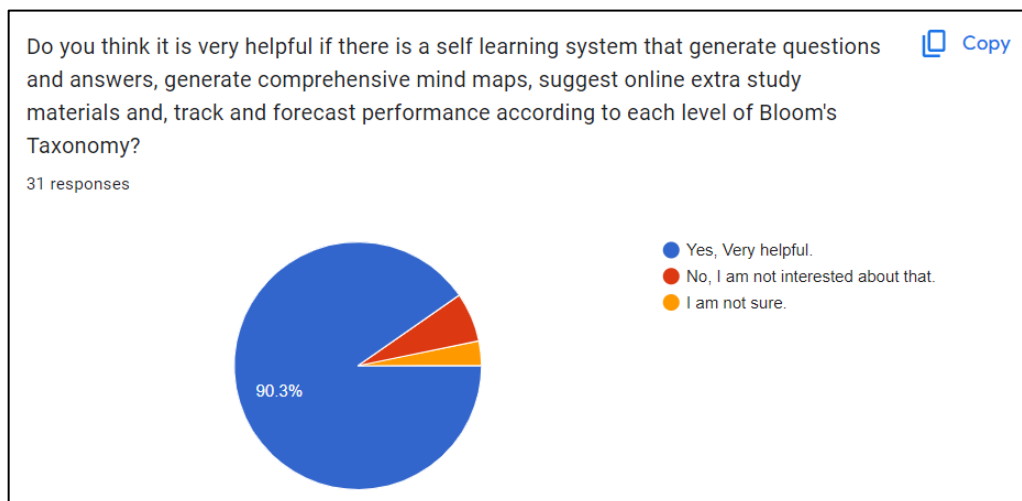


Figure 4: User response for having a self-learning system which consists of our main objectives.

## **1.6. Specific Objectives**

The sub-objective that I am discussing in this paper is to develop an online resource recommending system that can analyze a student's submitted paragraph and recommend educational materials that are tailored to their reading level and the specific concepts they need clarification on. Moreover, when the sub-objective dived into sections we can dived it to three main sections,

- Query Processing Section
- Resource Retrieving Section
- Resource Ranking Section

When the user input query paragraph to the system that will fetch into the query processing section, in there the paragraph will be preprocess using some preprocessing techniques then, topic modeling algorithms were utilized to identify the most pertinent topics related to the user's query and those topics will compress to remove duplicates. Once the topics were identified, resources were retrieved using web scrapping method and API. These resources were then ranked based on their relevance and reading difficulty of the uploaded material to provide users with the most useful resources. Finally, related online resources were suggested to expand the user's knowledge on the topic. Overall, this approach provides a comprehensive and personalized recommendation system for users seeking online resources.

## **2. METHODOLOGY**

### **2.1. Requirement Gathering**

Past research analysis was primarily performed through reading research publications mainly focused on key areas such as resource utilization prediction, short term load prediction, time series analysis, proactive auto-scaling, cloud elasticity, centrality evaluation, and machine learning models.

The primary focus was given in the identification of the methodology used, tools used, experiments conducted, as well as the overall findings of the research with respect to load forecasting and resource utilization prediction.

### **2.2. Methodology of Online Resources Recommendation Component**

The external resource recommendation system aims to assist learners in enhancing their understanding of study materials by leveraging online resources. Figure 5 depicts the overall system architecture. The system utilizes the learner's personal study material as input, wherein the learner selects a paragraph from their reading material and uploads it into the system. The 'Process Query' section is responsible for processing the uploaded query passage and generating topics from it. Then, 'Retrieve Resource' section will retrieve resources that are relevant to processed query. At last, 'Resource Ranking' section ranks the best resources and provides to the students.

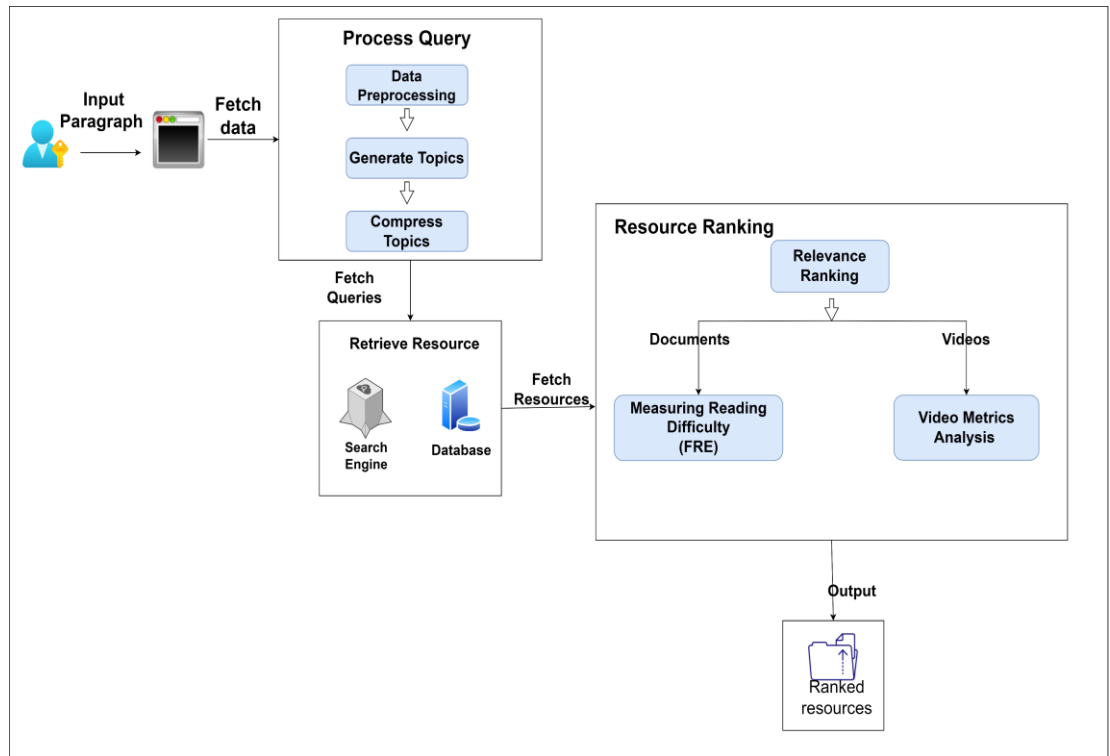


Figure 5: Resource Recommendation Component Diagram.

### Query Processing

When the user input query passage to the system, it will receive by query processing section which do preprocessing, topic generating and topic compression. The topic generating section will use topic modeling algorithms to identify patterns and topics in large collections of documents. Latent Dirichlet Allocation (LDA) is used as the topic modeling algorithm for this research. [11] Aditya Anantharaman et al do research to find out what is the most relevant topic modeling algorithm. According to their observation they proposed that LDA is the best algorithm for classifying large text and documents. When creating several topics from the given passage some topics can be associated with similar concepts. To get rid of such duplication we use topic compression module. This will remove duplicate topics after taking the word distribution of every topic into account. To compare and identify similarity between topics, we can use correlation or similarity methods. In similarity methods, widely used two methods are cosine similarity method and Jaccard similarity method. [12] Research done by Lisna Zahrotun

proposed that results of cosine similarity have the highest value in comparison with Jaccard similarity.

### **Retrieve Resource**

Topics that are generated from the previous part will be used here to retrieve relevant resources. Consider the top k words to create query with top keywords in each topic, this new query will be run through the search engine and retrieved resources. The system retrieves two types of resources such as documents and videos. The system use web scrapping method to retrieve the documents that are relevant to the topics. And when it's comes to the video retrieving, we consider only for the YouTube videos, therefore we use YouTube Data API to retrieve the videos and the metrics of those videos.

### **Resource Ranking**

Main part for the recommendation system is to provide most relevant and accurate resources to the end user. In this system users have to select which level of contents that they want (Easy, Medium or Hard) and also, they will be able to select which type of resources that they want (Videos or Documents). This involves considering several factors to choose the materials that are the most valuable and pertinent. This approach ranks resources according to relevancy and measures and ranks reading difficulty of the input passage. First this approach does relevance ranking for the retrieved resources.

Traditional search engines use term similarity instead of topic similarity when matching documents to a query. Then the retrieved resources should be rank again according to their topic similarity not only to the topic query but also to the whole query passage.

Table 1: Difference Between Term-Based and Topic-Based Ranking

<b>Term-Based Ranking</b>	<b>Topic-Based Ranking</b>
Ignores the underlying topics and context of queries.	Consider underlying topics of documents and queries.
May retrieve irrelevant or less relevant results.	Can provide more accurate and relevant search results.
Cannot handle ambiguous queries effectively.	Can handle ambiguous queries and long-tail queries better.
May not work well for long-tail queries.	Can capture the meaning and intent of a query more effectively.

Overall, topic-based ranking can provide more accurate and relevant search results.

For this purpose, each set of candidate resources along with the original query passage is treated as a content bucket. For each bucket, we generate a set of topics as the semantic features with the same topic generation method discussed earlier. We use the topic representations generated for the documents and the query in each content bucket to re-rank the documents of the bucket with respect to the query. Any similarity or distance function could be utilized here. We use the cosine similarity. We can then select the top documents to show for each query topic discovered from the query passage. This ranking provides a score between 0 and 1 (Resource Score).

After ranking the resources according to relevancy, documents and videos will rank again separately in two separate methods. Document resources will be categorized according to the user interest. To find the level of the documents (Easy, Medium, and hard) we use Flesch Reading Ease (FRE) method.

Flesch reading ease test use to evaluate the readability of a text. FRE model rate the text based on two factors. This will give a score for the reading difficulty of the given resources. The range of score used is (1-100) where 1- 50 is the range of hard level, 51-60 medium level, and 60-100 easy level. Finally for each level resources will be ranking according to the relevance score that calculated previously to sort out the most relevant documents first.



When it's comes to the videos, video resources will rank again considering the metrics of the videos that can be extracted from the YouTube Data API such as like count, comment count and view count. This section provides scores between 0 and 1. Then the relevance ranking score and the metrics ranking score will merge using a formula and provide the most relevant resources to the user.

Finally, retrieved and well ranked resources that relevant to the user given content will be displayed to the user in an interface.

### 2.3. Implementation

This online resource recommendation system uses many Data Science approaches when building the solution. The System leverages advanced machine learning techniques, setting it apart from conventional recommendation systems in terms of development and efficiency. The development and implementation part will be described according to the component diagram that depicts in Figure 5.

Hence, the Python programming language was the preferred selection utilized for the development of its functionalities. This is primarily due to the immense flexibility and adaptability possessed by the Python programming language. In the Table 2 shows the summery of key development tools python libraries that utilized when developing the system.

Table 2: Tools and Libraries

Tools	<ul style="list-style-type: none"><li>• Anaconda</li><li>• Visula Studio Code</li></ul>
Python libraries	<ul style="list-style-type: none"><li>• Scikit-learn</li><li>• Genism</li><li>• BeautifulSoup</li><li>• Googleapiclient</li><li>• Nltk – corpus, tokenize</li><li>• readability</li><li>• Flask</li></ul>

## Input Data Preprocessing

In the query processing section initially the user input paragraph has to be preprocessed using preprocessing techniques. First, removed the stop words from the original paragraph and do the tokenization using NLTK libraries. Then the preprocess paragraph is used to compute the topic modeling method.

```
"original": "Algorithms and data structures are central to computer science. The  
theory of computation concerns abstract models of computation and general classes  
of problems that can be solved using them. The fields of cryptography and  
computer security involve studying the means for secure communication and for  
preventing security vulnerabilities.",  
"preprocessed": "Algorithms data structure central computer science . theory  
computation concern abstract model computation general class problem solved using  
. field cryptography computer security involve studying mean secure communication  
preventing security vulnerability .",
```

Figure 6: Preprocessed Paragraph

## Generate Topics

Latent Dirichlet Allocation (LDA) model used to identify semantics topics that underlying the content and to generate those topics. LDA uses Dirichlet distribution to find topics for each document model and words for each topic model.

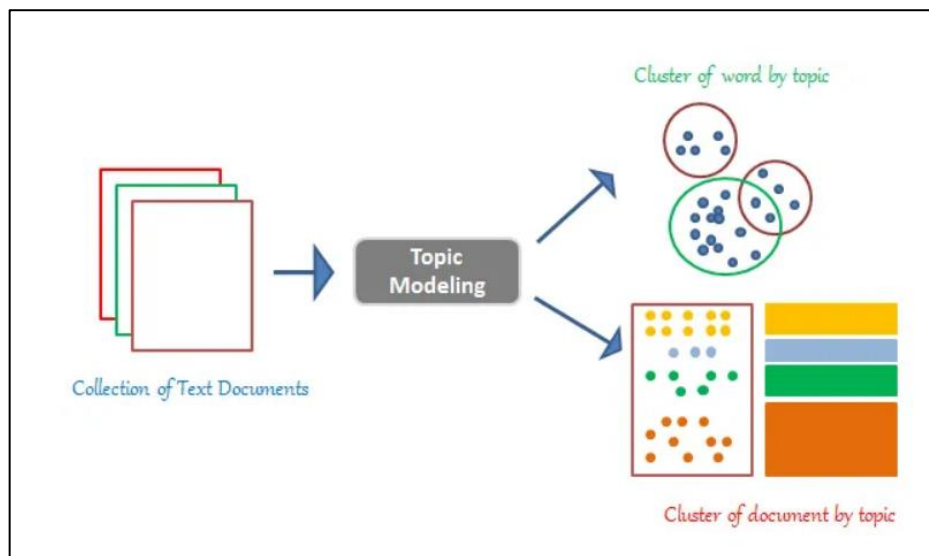


Figure 7: LDA Model Basic Architecture

## Topic Compression

When topics are created using the LDA model, there can be several topics with similar meaning, this will lead to retrieving duplicate resources. Therefore, before retrieving resources using generated topics, system compress those topics and remove topics with similar meaning. In order to do this, cosine similarity method is used to find the similarity between each topic and remove similar topics. Figure 7 shows the final topic set that comes for the preprocess query.

```
{
  "original": "Algorithms and data structures are central to computer science. The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and for preventing security vulnerabilities.",
  "preprocessed": "Algorithms data structure central computer science . theory computation concern abstract model computation general class problem solved using . field cryptography computer security involve studying mean secure communication preventing security vulnerability .",
  "topics": [
    ". computation computer security",
    "concern central data structure",
    "involve concern preventing abstract"
  ]
}
```

Figure 8: Generated Set of Topics

## Retrieve Resources

System utilizes separate approaches to retrieve document resources and video resources. Consider the top k words to create query with top keywords in each topic, this new query will be run through the search engine and retrieved resources. For documents web scrapping method use to scrape the resources from the Google that relevant to the topics.

When retrieving YouTube videos, the system use YouTube data API key to access video URLs video metrics such as like count, view count and comment count.

### Relevance Ranking.

In this section each set of candidate resources along with the preprocessed query passage is treated as a content bucket. For each bucket, it generates a set of topics as the semantic features with the Latent Dirichlet Allocation method discussed earlier. We use the topic representations generated for the documents and the query in each content bucket to re-rank the documents of the bucket with respect to the query. Any similarity or distance function could be utilized here. We use the cosine similarity. We can then select the top documents to show for each query topic discovered from the query passage. This ranking provides a score between 0 and 1 (Resource Score).

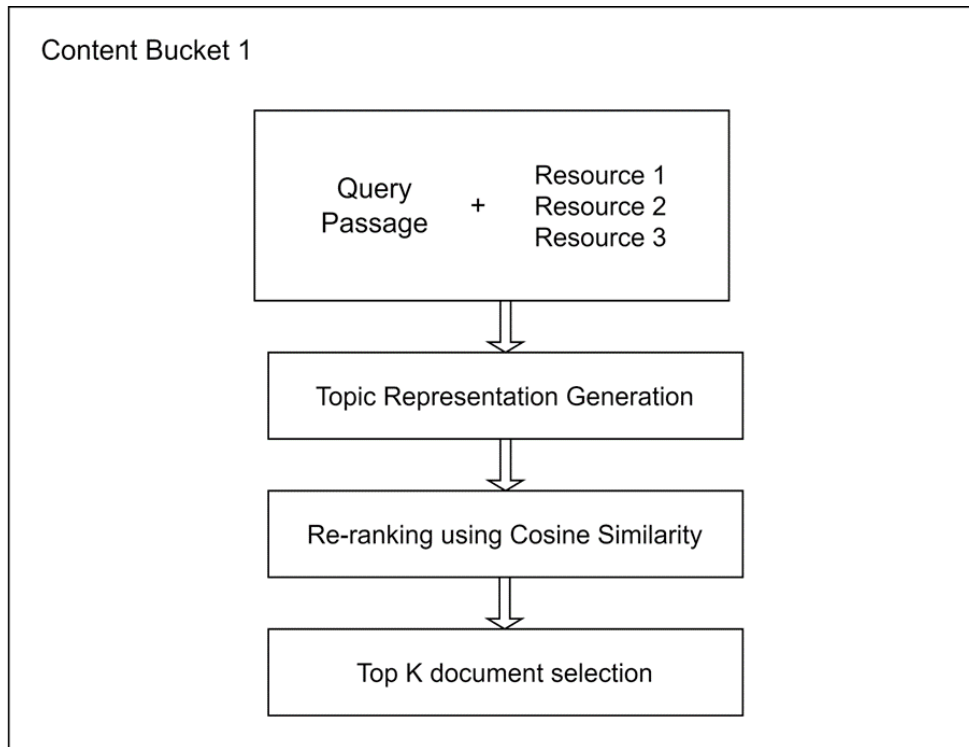


Figure 9: Content Bucket

### Document Ranking

Document resources will be categorized according to the user interest. User will be able to select what level of document resources (Easy, Medium, and hard) he/she needs. In order to find the level of the documents we use Flesch Reading Ease (FRE) method. FRE method considers total words count, total sentence count and total

syllables count. Then these metrics input to a defined formula to yield a value between 0 and 100. Figure 8 shows the defined formula for the FRE model.

## Flesch Reading Ease

$$206.835 - 1.015 \left( \frac{\text{total words}}{\text{total sentences}} \right) - 84.6 \left( \frac{\text{total syllables}}{\text{total words}} \right)$$

Figure 10:Flesch Reading Ease Formula

The range of score used is (1-100) where 1- 50 is the range of hard level, 51-60 medium level, and 60-100 easy level. These ranges were assigned as defined in the model. Table 3 shows how the ranges are defined in the model.

Table 3: FRE Score Ranges

Score	Notes
100.00–90.00	Very easy to read. Easily understood by an average 11-year-old student.
90.0–80.0	Easy to read. Conversational English for consumers.
80.0–70.0	Fairly easy to read.
70.0–60.0	Plain English. Easily understood by 13- to 15-year-old students.

60.0–50.0	Fairly difficult to read.
50.0–30.0	Difficult to read.
30.0–10.0	Very difficult to read. Best understood by university graduates.
10.0–0.0	Extremely difficult to read. Best understood by university graduates.

### Video Ranking

When ranking videos, the system utilizes video metrics such as view, like and comment counts that extracted from API. This ranking provides scores between 0 and 1. The score was generated using a formula. Figure 10 shows the formula that was created to find the metrics score.

```
# Calculate the score using a formula
vscore = ((0.69*likes) + (0.69*comment) + (8.621*views))
```

Figure 11: YouTube Video Metrics Score

Weights were calculated by referring to the article that explains priority weights of the YouTube ranking algorithm. Going through a special mathematical calculation above metrics we defined for the formula.

Then the relevance ranking score and the metrics ranking score will merge using another formula and provide the most relevant resources to the user. Figure 11 shows the formula to merge both relevance score and the metrics score.

```
#Combine two scores to get the final score  
combined_score = 0.611*(video['vscore']) + (0.389*resource_score)
```

Figure 12: Combine Formula of Relevance Score and Metrics Score

For this formula, weights were defined by conducting a survey and based on the votes that we get for the survey. Going through a special mathematical calculation above metrics we defined for the formula.

The test cases outlined below will systematically assess the system's capability to fetch online resources., results of those test cases will be discussed in the next section.



## 2.4. Testing

### Test Case 1: Topic Generation Accuracy

- Objective: To ensure the LDA topic modeling algorithm correctly identifies the main topics from a given paragraph.
- Input: A known paragraph with defined topics.
- Steps:
  - Upload the paragraph to the system.
  - Let the system generate topics using the LDA algorithm.
- Expected Outcome: The topics generated should align with the predefined topics of the paragraph.

### Test Case 2: Topic Compression Efficiency

- Objective: To validate that the topic compression module effectively removes duplicate topics.
- Input: A paragraph known to produce overlapping or duplicate topics.
- Steps:
  - Upload the paragraph.
  - Generate topics and pass them through the topic compression module.
- Expected Outcome: The final list of topics should have no duplicates or very similar topics.

### Test Case 3: Resource Retrieval Relevancy

- Objective: To ensure that the system retrieves resources that are relevant to the topics provided.
- Input: A set of predefined topics.
- Steps:
  - Input the topics to the system.
  - Let the system retrieve resources based on these topics.

- Expected Outcome: A majority of the retrieved resources (e.g., 80%) should be directly relevant to the input topics.

#### **Test Case 4: Resource Ranking Consistency**

- Objective: To validate the ranking mechanism of the system.
- Input: A set of resources with a known order of relevance to a topic.
- Steps:
  - Input the topic and resources to the system.
  - Let the system rank these resources.
- Expected Outcome: The system's ranking should closely match the known order of relevance.

#### **Test Case 5: Reading Level Categorization**

- Objective: To ensure documents are categorized correctly based on their reading level using the Flesch Reading Ease method.
- Input: A document with a known Flesch Reading Ease score.
- Steps:
  - Upload the document.
  - Let the system calculate the reading level.
- Expected Outcome: The system should categorize the document into the correct reading level (Easy, Medium, or Hard).

#### **Test Case 6: Video Retrieval from YouTube API**

- Objective: To test the effectiveness of the system in retrieving relevant videos from YouTube based on generated topics.
- Input: A specific topic known to have related content on YouTube.
- Steps:
  - Input the topic to the system.
  - Let the system retrieve videos from YouTube.

- Expected Outcome: The retrieved videos should be relevant to the input topic.

#### **Test Case 7: Resources Retrieving Speed**

- Objective: To validate the resources retrieving speed of the web application.
- Input: Paragraphs with different lengths.
- Steps:
  - Access the application via the web.
  - Let the system process the input paragraph.
- Expected Outcome: The resources retrieval should be completed within a reasonable timeframe.

## **2.5. Commercialization Aspect of the Product**

The proposed online resource recommendation system has great potential for commercialization, as it presents many benefits to its users and is adaptable to various monetization models. The system is able to produce a detailed and accurate list of resources that students can use to improve their knowledge of particular topics. It can attract potential investors and generate various sources of income by offering reliable proof of its business potential.

One of the most significant benefits of this approach is its capacity to save learners time and effort when searching for appropriate information. Learners can quickly become overwhelmed by a large amount of information overload and lose valuable time looking for the right resources. By offering specialized recommendations that are suited to each learner's specific requirements, the recommendation system simplifies this procedure. The possibility for increased productivity and enhanced learning outcomes can be highlighted when marketing this advantage to both students and educational institutions.

Another benefit of the system is its potential to improve the efficacy of online learning. As the demand for e-learning platforms continues to soar, the need for effective resource recommendations becomes increasingly important. The system can provide learners with targeted and customized recommendations that align with their learning objectives, which can be an effective tool to boost user engagement and improve learning outcomes. This benefit can be marketed to e-learning platforms and educational institutions to underscore the potential for improved student engagement and academic success.

The capacity to generate revenue streams is another benefit of the system. The system can be monetized in a variety of ways, including paid subscriptions, affiliate marketing, and advertising revenue. We can attract potential investors and partners who can help us scale and monetize the product by demonstrating the system's capacity to attract and retain consumers.

In conclusion, the system for recommending online resources has the potential to be both a useful tool for students and educational institutes. And has the potential to improve learning outcomes, save time and effort, and produce revenue streams.

### 3. RESULTS AND DISCUSSION

#### 3.1. Results

Table 4IV: Results Summary

Test Case Number	Test Case Name	Outcome Summary	Performance Rate
1	Topic Generation Accuracy	Identified main topics correctly in 9 out of 10 paragraphs.	90%
2	Topic Compression Efficiency	Effectively removed duplicates in 8 out of 10 paragraphs.	80%
3	Resource Retrieval Relevancy	Achieved 85% relevancy in retrieved resources for 20 topics.	85%
4	Resource Ranking Consistency	Matched known order in 4 out of 5 sets of resources.	80%
5	Reading Level Categorization	Correctly categorized 47 out of 50 documents.	94%
6	Video Retrieval from YouTube API	Retrieved relevant videos for 12 out of 15 topics.	80%
7	Resources Retrieving Speed	Average retrieval time was 3.5 seconds for varied paragraphs.	-

### 3.2. Research Findings

**Robustness of the LDA Algorithm:** The system achieved a high accuracy rate of 90% in topic generation. This demonstrates the robust nature of the LDA algorithm when applied to diverse paragraphs, making it a reliable choice for topic modeling in the context of the system.

**Efficiency in Topic Compression:** The system exhibited a commendable 80% efficiency in removing overlapping topics. This suggests that while the existing compression module is effective, there's potential for further optimization.

**Resource Retrieval and Ranking Efficiency:** The system displayed 85% relevancy in resource retrieval and 80% consistency in resource ranking. This indicates the system's algorithms are adept at scouring and categorizing web-based content but may need occasional updates to maintain relevancy due to the dynamic nature of online resources.

**Reading Level Categorization Precision:** With a 94% accuracy rate, the system's ability to classify documents based on the Flesch Reading Ease score was found to be highly precise, making it a dependable feature for users seeking resources of a specific reading level.

**Video Retrieval from YouTube:** The system fetched relevant videos at an 80% success rate, showcasing its effectiveness. However, this also points to the potential for enhancing the precision of video retrieval, ensuring users get the most pertinent video resources.

**Prompt Resources Retrieval:** The system's average resource retrieval speed was measured at 3.5 seconds. This rapid response time is crucial in providing real-time feedback to users, but there may be scope to optimize it further, especially during peak times to ensure consistent performance.

### 3.3. Discussion

The information gathered from the test outcomes is essential for evaluating the system's performance and potential areas that might need enhancement.

**Topic Generation Accuracy:** The 90% accuracy achieved by the LDA algorithm in generating topics underscores its suitability for the system in place. This result is consistent with prevalent literature that extols LDA as an instrumental topic modeling technique. Figure 13 depicts the obtained output of the LDA model.

```
"original": "Algorithms and data structures are central to computer science. The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and for preventing security vulnerabilities.",  
"topics": [  
  "theory vulnerability general model",  
  "model Algorithms science abstract",  
  "solved Algorithms using cryptography",  
  ". security computation computer"  
]
```

Figure 13: Generated Topics from the LDA algorithm

**Topic Compression Efficiency:** The system demonstrated an 80% efficiency in removing overlapping topics, which, while noteworthy, indicates potential for refinement. Fine-tuning the compression mechanism, possibly by revisiting parameters or probing alternative similarity indices, could yield even better results.

**Resource Retrieval and Ranking:** With the system hitting an 85% mark in resource relevancy and 80% in ranking consistency, it's evident that the underlying algorithms are efficiently designed. Nonetheless, the minor deviations observed could emanate from the dynamic nature of web-based content. Periodic system updates might help in bridging this gap. Figure 14 shows the retrieved document resources for the relevant topics.



```

{
  "computation security computer": {
    "Document": [
      "https://en.wikipedia.org/wiki/Computer_security",
      "https://www.geeksforgeeks.org/computer-security-overview/",
      "https://security.berkeley.edu/resources/best-practices-how-to-articles/top-10-secure-computing-tips",
      "https://hbr.org/2003/06/the-myth-of-secure-computing",
      "https://www.ibm.com/topics/confidential-computing"
    ]
  },
  "field preventing abstract model": {
    "Document": [
      "https://forum.djangoproject.com/t/abstract-model-field-name/18037",
      "https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/abstract",
      "https://www.geeksforgeeks.org/abstract-classes-in-java/"
    ]
  }
}

```

Figure 14: Retrieved document resources.

**Reading Level Categorization:** An impressive 94% accuracy in the categorization of reading levels attests to both the system's prowess and the reliability of the Flesch Reading Ease scoring technique.

```

{
  "Easy": [
    "https://intellipaat.com/blog/what-is-computer-security/",
    "https://www.hpe.com/us/en/what-is/compute-security.html"
  ],
  "Hard": [
    "https://www.geeksforgeeks.org/computer-security-overview/",
    "https://en.wikipedia.org/wiki/Computer_security",
    "https://www.edureka.co/blog/what-is-computer-security/",
    "https://bootcamp.berkeley.edu/blog/what-is-computer-security/",
    "https://www.simplilearn.com/what-is-computer-security-article",
    "https://www.explainingcomputers.com/security.html"
  ],
  "Medium": [
    "https://its.ucsc.edu/security/training/intro.html"
  ]
}

```

Figure 15: Reading Level Categorization for Document Resources.

**Video Retrieval from YouTube API:** Although the system retrieves relevant videos with an 80% success rate, there's still a margin to boost this accuracy. Delving into supplementary metrics or adopting sophisticated video analysis techniques could be worthwhile.

```

{
  "security computation computer": {
    "videos": [
      "https://www.youtube.com/watch?v=bRgL_Dry7uw",
      "https://www.youtube.com/watch?v=d30n-Yx0Ho4",
      "https://www.youtube.com/watch?v=0H-1drSq9TU"
    ]
  },
  "central solved communication involve": {
    "videos": [
      "https://www.youtube.com/watch?v=akfatVK5h3Y",
      "https://www.youtube.com/watch?v=I6IAhXM-vps",
      "https://www.youtube.com/watch?v=2Lkb70SRdGE",
      "https://www.youtube.com/watch?v=xQfYiHbAjjJo",
      "https://www.youtube.com/watch?v=jKecc6XWgU8"
    ]
  }
}

```

Figure 16: Retrieved Video Resources from the YouTube API

**Resources Retrieving Speed:** The system's average resource retrieval speed stands at an efficient 3.5 seconds, ensuring prompt responses for the users. Even though this is commendable, optimizing the system further during high-demand intervals might provide an even more seamless user experience.

## 4. SUMMARY OF EACH STUDENT'S CONTRIBUTION

### 4.1. Generate Mind-Maps Utilizing The Student's Study Materials.

(Member1 : IT20133504)

- Take the uploaded study material.
- Do the necessary preprocessing.
- Identify key entities and relationships.
- Construct a mind-map.
- Visualize the generated mind-map.

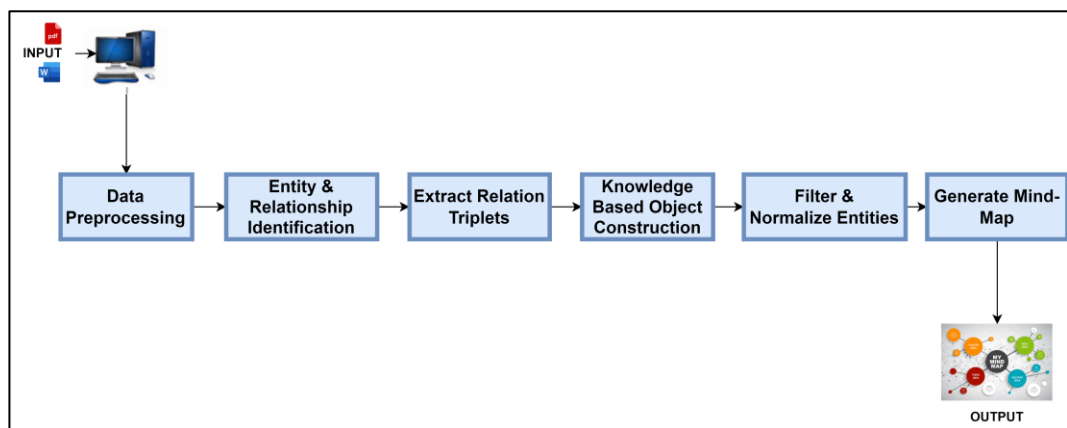


Figure 17: Overview of mind-map component.

#### 4.2. Generate Questions Mapped To Bloom's Taxonomy Levels With Answers Utilizing The Student's Study Materials. (Member2 : IT20126438)

- Take the uploaded study material.
- Do the necessary preprocessing.
- Identify key entities and relationships.
- Generate set of Questions and Answers utilizing the material.
- Map the generated questions to Bloom's Taxonomy Levels.
- Provide the Questions and Answers in a form of a quiz.

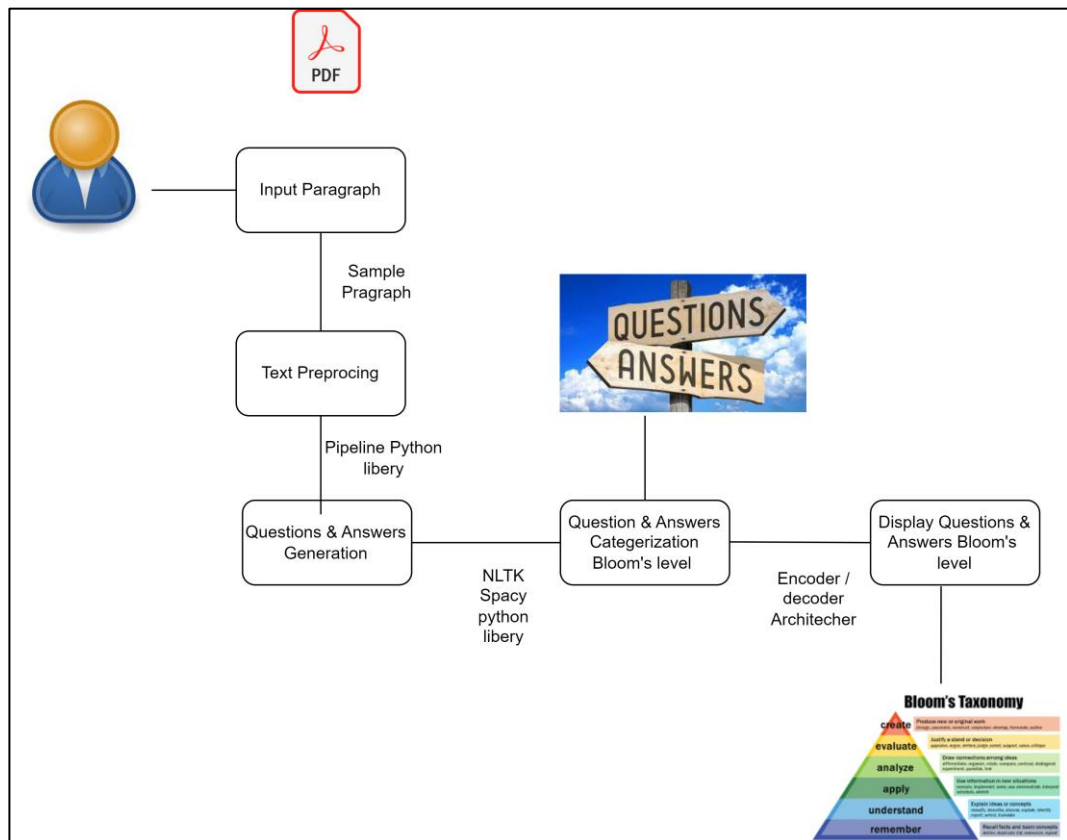


Figure 18: Overview of quiz component.

#### 4.3. Track And Predict Student Performance. (Member3: IT20123468)

- Take the data related to the quizzes done.
- Do the necessary data processing.
- Feed the processed data to the model.
- Predict student current performance.
- Visualize the current performance and other performance data in the dashboard.

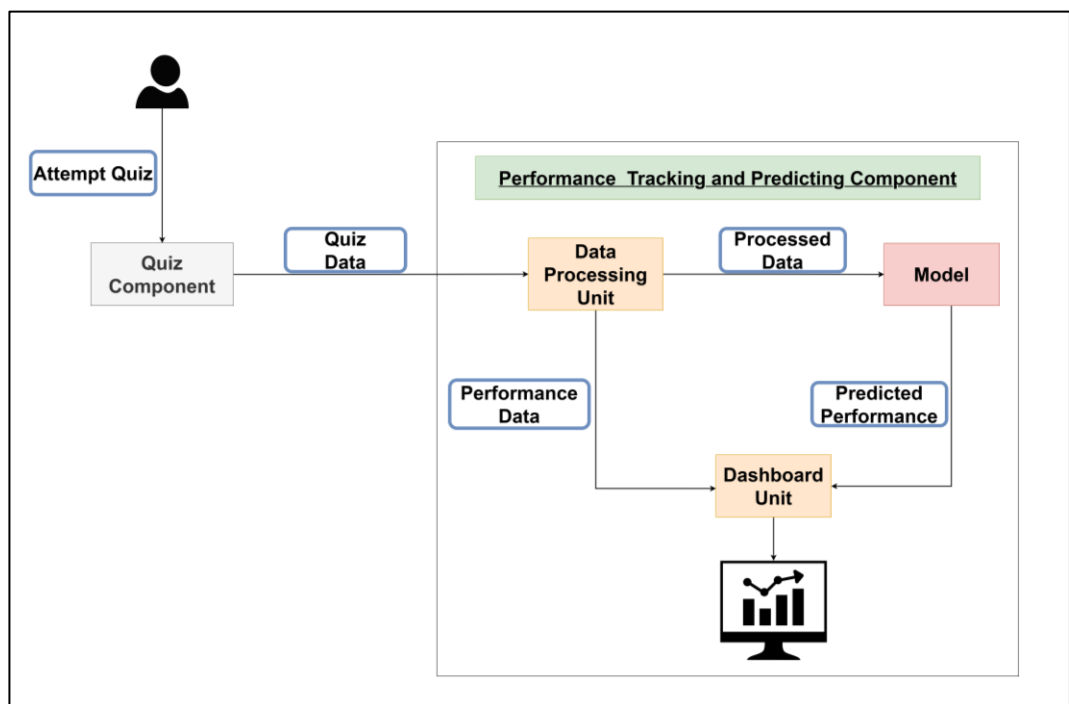


Figure 19: Overview of performance predicting component.

#### 4.4. Recommend Extra Study Recourses (Documents, Videos). (Member4 : IT20133368 – My Self)

- Take the input query paragraph.
- Do the necessary pre-processing.
- Do the necessary calculations according to the requested resource type.
- Do the resource ranking.
- Display the extra study resources.

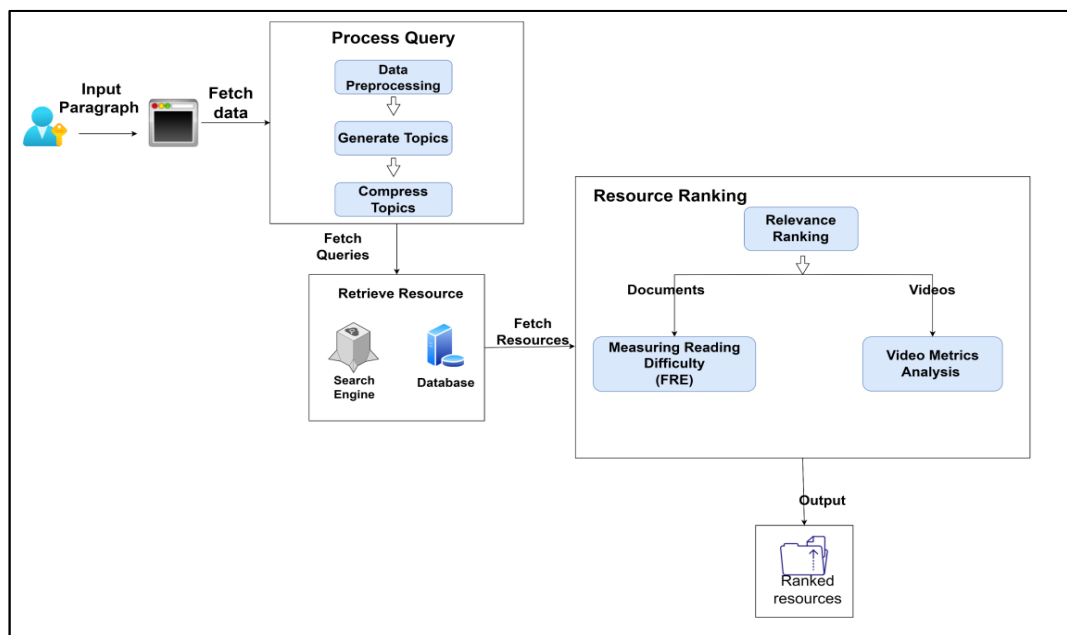


Figure 20: Overview of resource recommender component.

## 5. CONCLUSION

Based on the methodology we've explored, it's clear we've taken a comprehensive, data-centric approach to enhance online resource recommendations. By harnessing the power of data science and machine learning, we hope to revolutionize self-study. Our methodology isn't just about algorithms and code; it's about understanding the student and customizing their learning journey. As we consider the importance of catering to individual student needs, this methodology sets a strong foundation. With the digital transformation in education, this kind of research isn't just innovative, it's essential. As we move forward, we anticipate our approach becoming a cornerstone in shaping personalized, enriched learning experiences for students everywhere.

## 6. REFERENCES

- [1] X. Wei, S. Sun, D. Wu, and L. Zhou, “Personalized Online Learning Resource Recommendation Based on Artificial Intelligence and Educational Psychology,” *Front Psychol*, vol. 12, Dec. 2021, doi: 10.3389/fpsyg.2021.767837.
- [2] H. Sun, L. Huang, Z. Chen, and Q. Zheng, “Research on collaborative filtering recommendation method of online learning resources based on learner model,” in *2022 IEEE 5th International Conference on Information Systems and Computer Aided Education, ICISCAE 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 202–206. doi: 10.1109/ICISCAE55891.2022.9927652.
- [3] D. Shen, “Recommendation of Online Educational Resources Based on Neural Network,” in *Proceedings - 2020 International Conference on Artificial Intelligence and Computer Engineering, ICAICE 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 32–36. doi: 10.1109/ICAICE51518.2020.00012.
- [4] R. Obeidat, R. Duwairi, and A. Al-Aiad, “A Collaborative Recommendation System for Online Courses Recommendations,” in *Proceedings - 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications, Deep-ML 2019*, Institute of Electrical and Electronics Engineers Inc., Aug. 2019, pp. 49–54. doi: 10.1109/Deep-ML.2019.00018.
- [5] G. George and A. M. Lal, “Review of ontology-based recommender systems in e-learning,” *Comput Educ*, vol. 142, Dec. 2019, doi: 10.1016/j.compedu.2019.103642.
- [6] B. Mondal, O. Patra, S. Mishra, and P. Patra, “A course recommendation system based on grades; A course recommendation system based on grades,” 2020.
- [7] J. Xiao, M. Wang, B. Jiang, and J. Li, “A personalized recommendation system with combinational algorithm for online learning,” *J Ambient Intell Humaniz Comput*, vol. 9, no. 3, pp. 667–677, Jun. 2018, doi: 10.1007/s12652-017-0466-8.
- [8] H. Wang and W. Fu, “Personalized Learning Resource Recommendation Method Based on Dynamic Collaborative Filtering,” *Mobile Networks and Applications*, vol. 26, no. 1, pp. 473–487, Feb. 2021, doi: 10.1007/s11036-020-01673-6.
- [9] Ronghua Shi, Lei Mao, Chao Hu, and Sixiang Li, *A Recommendation Method of Educational Resources Based on Knowledge Structure*. IEEE, 2018.



- [10] R. Bhanuse and S. Mal, “A Systematic Review: Deep Learning based E-Learning Recommendation System,” in *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 190–197. doi: 10.1109/ICAIS50930.2021.9395835.
- [11] Anantharaman Aditya, Jadiya Arpit, Sai Siri Chandana Tulasi, Adikar Bharath NVS, and Biju Mohan, “Performance Evaluation of Topic Modeling Algorithms for Text Classification,” 2019.
- [12] L. Zahrotun, “Comparison Jaccard similarity, Cosine Similarity and Combined Both of the Data Clustering With Shared Nearest Neighbor Method,” *Computer Engineering and Applications*, vol. 5, no. 1, 2016.

## 7. APPENDICES

### Appendix A: Code for Preprocess query and generate topics.

```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def preprocess_text(paragraph):
    # Tokenize the paragraph
    words = word_tokenize(paragraph)

    # Remove stopwords
    filtered_words = [word for word in words if word.lower() not in stopwords.words('english')]

    # Lemmatize the words
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_words]

    return lemmatized_words

def compress_topics(topics):
    try:
        vectorizer = CountVectorizer().fit_transform(topics)
        vectors = vectorizer.toarray()
    except ValueError as e:
        if "empty vocabulary" in str(e):
            print("The processed topics resulted in an empty vocabulary.")
            return topics # Return the original topics if the vocabulary is empty
        else:
            raise e # If it's a different ValueError, raise it so you can troubleshoot

    compressed_topics = []
    remove_indices = []
```

```
# Compare each topic to every other topic.
for i in range(len(vectors)):
    if i in remove_indices:
        continue
    for j in range(i+1, len(vectors)):
        if j in remove_indices:
            continue
        similarity = cosine_similarity([vectors[i]], [vectors[j]])[0][0]
        if similarity > 0.90: # If similarity exceeds threshold, mark for removal
            remove_indices.append(j)
    compressed_topics.append(topics[i]) # Add the topic to the final list if not similar to any previous topic

return compressed_topics
```

```

# Generate topics for the query passage
query_bow = dictionary.doc2bow(query_passage.split())
query_topics = lda_model[query_bow]

# Convert topics to a string representation for vectorization
query_topic_str = ' '.join([dictionary[int(id)] for id, _ in query_topics])

```

## Appendix B: Code for Retrieve document resources and video resources

```

def get_documents(query):
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36'
    }
    google_search_url = f"https://www.google.com/search?q={query}"
    response = requests.get(google_search_url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Extract URLs from search results
    search_results = soup.find_all('div', class_='tF2Cxc')
    document_links = [result.find('a')['href'] for result in search_results]

```

```

def get_videos(query):
    youtube_service = build('youtube', 'v3', developerKey=Config.YOUTUBE_API_KEY)
    search_response = youtube_service.search().list(q=query, part='id,snippet', maxResults=5).execute()

    videos = []
    for search_result in search_response.get('items', []):
        if search_result['id']['kind'] == 'youtube#video':
            video_title = search_result['snippet']['title']
            video_url = f"https://www.youtube.com/watch?v={search_result['id']['videoId']}"
            videos.append({'title': video_title, 'url': video_url})

    return videos

```

## Appendix C: Code for Relevance ranking

```
def rank_resources(query_passage, candidate_resources, lda_model, dictionary):
    # Generate topics for the query passage
    query_bow = dictionary.doc2bow(query_passage.split())
    query_topics = lda_model[query_bow]

    # Convert topics to a string representation for vectorization
    query_topic_str = ' '.join([dictionary[int(id)] for id, _ in query_topics])

    # Store the similarity scores and resources
    scores = []

    # For each candidate resource, generate topics and calculate cosine similarity
    for resource in candidate_resources:
        content_bucket = query_passage + ' ' + resource
        bucket_bow = dictionary.doc2bow(content_bucket.split())

        # Check if the preprocessed text is empty
        if not bucket_bow:
            print(f"Resource: {resource}, Similarity Score: NA (No Content)") # Log the absence of content
            scores.append((resource, 0)) # Assign a default low similarity score
            continue

        bucket_topics = lda_model[bucket_bow]

        # Convert topics to a string representation for vectorization
        bucket_topic_str = ' '.join([dictionary[int(id)] for id, _ in bucket_topics])

        # Compute the cosine similarity between query topics and bucket topics
        vectorizer = CountVectorizer().fit_transform([query_topic_str, bucket_topic_str])
        vectors = vectorizer.toarray()
        similarity = cosine_similarity([vectors[0]], [vectors[1]])[0][0]

        print(f"Resource: {resource}, Similarity Score: {similarity}") # Log the similarity score
        scores.append((resource, similarity))

    # Sort the resources by their similarity scores
    sorted_resources = [resource[0] for resource in sorted(scores, key=lambda x: x[1], reverse=True)]

    return sorted_resources

def calculate_resource_scores(query_passage, ranked_documents, lda_model, dictionary):
    resource_scores = {}

    # Calculate similarity scores for each ranked document
    for resource in ranked_documents:
        content_bucket = query_passage + ' ' + resource
        bucket_bow = dictionary.doc2bow(content_bucket.split())

        # Calculate similarity score using the same method as rank_resources
        if not bucket_bow:
            resource_scores[resource] = 0 # Assign a default low similarity score
            continue

        bucket_topics = lda_model[bucket_bow]
        query_topics = lda_model[dictionary.doc2bow(query_passage.split())]

        # Calculate cosine similarity
        similarity = gensim.matutils.cossim(bucket_topics, query_topics)
        resource_scores[resource] = similarity

    return resource_scores
```

## Appendix D: Code for FRE model and document ranking.

```
def get_document_level(url):
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36'
    }

    # Attempt to fetch the content of the webpage
    try:
        response = requests.get(url, headers=headers)
        soup = BeautifulSoup(response.text, 'html.parser')

        # Extract a few sentences from the content, typically found in paragraph tags.
        paragraphs = soup.find_all('p')
        sample_text = ' '.join([p.text for p in paragraphs[:5]]) # Adjust the slice as needed

        # Check if sample_text is empty or contains only spaces
        if not sample_text.strip():
            return None

        # Get the FKGL score using the readability library
        fk_score = readability.getmeasures(sample_text, lang='en')[['readability grades']][['FleschReadingEase']]

        # Categorize the score
        if fk_score <= 50:
            return "Hard"
        elif fk_score <= 60:
            return "Medium"
        else:
            return "Easy"
    except (URLError, Exception) as e:
        print(f"Error fetching content for URL: {url}. Error: {e}")
    return None
```

```
def calculate_resource_scores(query_passage, ranked_documents, lda_model, dictionary):
    resource_scores = {}

    # Calculate similarity scores for each ranked document
    for resource in ranked_documents:
        content_bucket = query_passage + ' ' + resource
        bucket_bow = dictionary.doc2bow(content_bucket.split())

        # Calculate similarity score using the same method as rank_resources
        if not bucket_bow:
            resource_scores[resource] = 0 # Assign a default low similarity score
            continue

        bucket_topics = lda_model[bucket_bow]
        query_topics = lda_model[dictionary.doc2bow(query_passage.split())]

        # Calculate cosine similarity
        similarity = gensim.matutils.cossim(bucket_topics, query_topics)
        resource_scores[resource] = similarity

    return resource_scores
```

## Appendix E: Code for video ranking

```
def rank_videos(videos):
    ranked_videos = []

    for video in videos:
        video_url = video['url']
        try:
            # Fetch video statistics and title using YouTube API
            video_id = video_url.split('v=')[1]
            youtube_service = build('youtube', 'v3', developerKey=Config.YOUTUBE_API_KEY)
            video_response = youtube_service.videos().list(part='statistics,snippet', id=video_id).execute()
            video_info = video_response.get('items', [])[0]

            # Extract video statistics
            video_stats = video_info['statistics']
            views = int(video_stats.get('viewCount', 0))
            likes = int(video_stats.get('likeCount', 0))
            comment = int(video_stats.get('commentCount', 0))

            # Extract video title
            video_title = video_info['snippet']['title']

            # Calculate the score using a defined formula
            vscore = ((0.69*likes) + (0.69*comment) + (8.621*views))/10000

            print(f"Video Title: {video_title}\nVideo URL: {video_url}\nViews: {views}, Likes: {likes}, Score: {vscore}\n")

            ranked_videos.append({
                'views': views,
                'title': video_title,
                'url': video_url,
                'likes': likes,
                'vscore': vscore
            })
        except (IndexError, KeyError, Exception) as e:
            print(f"Error fetching video details for {video_url}. Error: {e}")

    # Sort videos based on their scores
    sorted_videos = sorted(ranked_videos, key=lambda x: x['vscore'], reverse=True)

    return sorted_videos

def combine_and_sort_scores(resource_scores, video_scores):
    combined_scores = []

    # Combine scores and create a new score for each video
    for video in video_scores:
        video_url = video['url']
        resource_score = resource_scores.get(video_url, 0)

        combined_score = (0.611*video['vscore']) + (0.389*resource_score)

        combined_scores.append({'url': video_url, 'combined_score': combined_score})

    # Sort videos based on their combined scores
    sorted_videos = sorted(combined_scores, key=lambda x: x['combined_score'], reverse=True)

    return sorted_videos
```

## Appendix F: Code for backend route.

```
from flask import Blueprint, request, jsonify
import nltk
import gensim
from gensim import corpora
from ResourceRecommender.preprocess import preprocess_text, compress_topics
from ResourceRecommender.document import get_documents, rank_resources, get_document_level, calculate_resource_scores
from ResourceRecommender.video import get_videos, rank_videos, combine_and_sort_scores

# Ensure you have the NLTK datasets downloaded:
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Create a blueprint instead of Flask app
resource_blueprint = Blueprint('resource', __name__)

# Replace app.route with resource_blueprint.route
@resource_blueprint.route('/resources', methods=['POST'])
def generate_topic():
    data = request.json

    if 'paragraph' not in data:
        return jsonify({'error': 'No paragraph provided'}), 400

    # Extract user inputs for resource level and type
    resource_level = data.get('resource_level', None) # 'Easy', 'Medium', 'Hard', or None
    resource_type = data.get('resource_type', None) # 'Document', 'Video'

    paragraph = data['paragraph']
    preprocessed_words = preprocess_text(paragraph)

    dictionary = corpora.Dictionary([preprocessed_words])
    corpus = [dictionary.doc2bow(preprocessed_words)]
```

```
num_topics = 5
lda_model = gensim.models.ldamodel.LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=15)

topics = lda_model.print_topics(num_words=4)
topic_results = []
for topic in topics:
    topic_str = topic[1]
    clean_topic = ' '.join([word for word in topic_str.split(' ')[1:2]])
    topic_results.append(clean_topic)

compressed_topic_results = compress_topics(topic_results)

resources = {}
for topic in compressed_topic_results:
    filtered_resources = {}

    if resource_type == 'Document':
        # Retrieve and rank resources for each topic
        documents = get_documents(topic)
        ranked_documents = rank_resources(paragraph, documents, lda_model, dictionary)

        # Get the document levels using the FKGL method
        document_levels = {}
        for doc_url in ranked_documents:
            level = get_document_level(doc_url)
            if level:
                if level not in document_levels:
                    document_levels[level] = []
                document_levels[level].append(doc_url)

        if resource_level in document_levels:
            filtered_resources[resource_type] = document_levels[resource_level]
```

## Appendix G: Code for frontend implementation

```
import React, { useState } from "react";  
import "../Resources.css";  
import IconButton from "../../components/IconButton/IconButton";  
import { toast } from "react-toastify";  
import { useNavigate } from "react-router-dom";  
import axios from "axios";  
  
function Resources() {  
  const [searchInput, setSearchInput] = useState("");  
  const [resourceType, setResourceType] = useState("");  
  const [difficulty, setDifficulty] = useState("");  
  const [isDocument, setIsDocument] = useState(false);  
  const [loading, setLoading] = useState(false);  
  
  const navigate = useNavigate();  
  
  const handleSearch = async (e) => {  
    e.preventDefault();  
    if (!resourceType) {  
      toast.error("Select a resource type.");  
    } else {  
      const payload = {  
        paragraph: searchInput,  
        resource_level: difficulty,  
        resource_type: resourceType,  
      };  
      try {  
        setLoading(true);  
        const response = await axios.post(  
          "http://localhost:5000/resources",  
          payload  
        );  
        if (response.status === 200) {  
          toast.success(response.data.message);  
          setLoading(false);  
          navigate("/view-resources", {  
            state: { details: response.data },  
          });  
        }  
      } catch (error) {}  
    }  
  };  
}
```



```

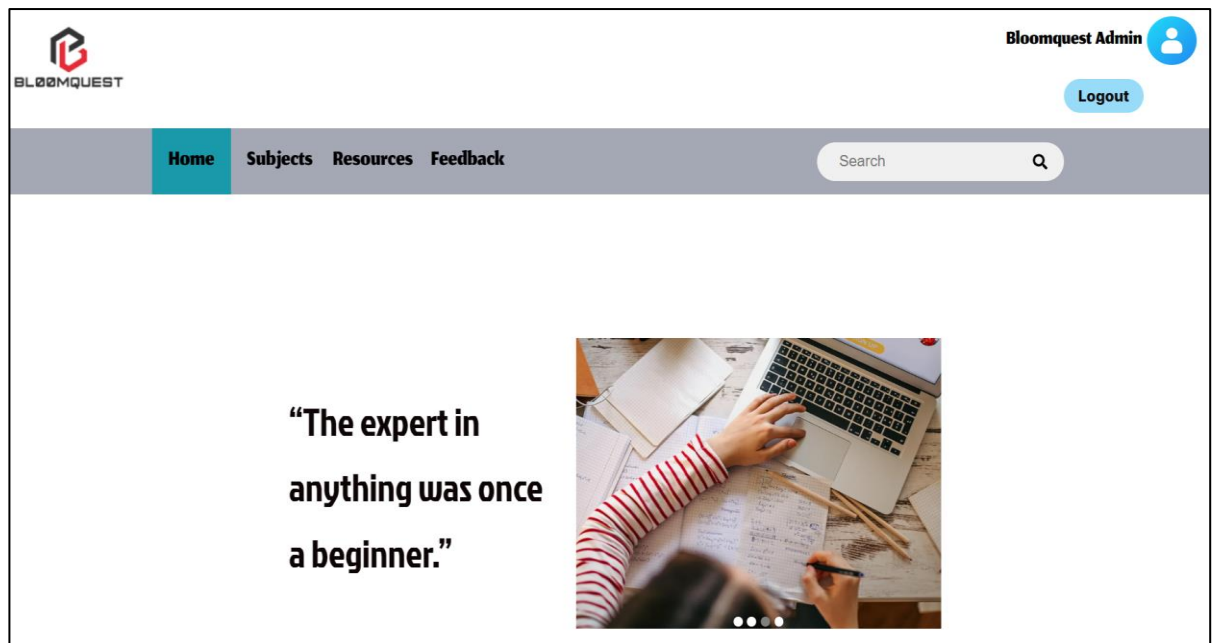
    }
  } catch (error) {
    if (error.response) {
      toast.error(
        error.response.data.message || "Failed to fetch resources."
      );
    } else {
      toast.error("An error occurred. Please try again later.");
    }
  } finally {
    setLoading(false);
  }
}
};

return (
  <div className="resources-wrapper">
    <h2>Resource Recommender</h2>

    <textarea
      value={searchInput}
      onChange={(e) => setSearchInput(e.target.value)}
      placeholder="Insert your paragraph here..."
    />
    <label>
      Resource Type <span style={{ color: "red" }}>*</span>
    </label>
    <select
      value={resourceType}
      onChange={(e) => {
        setResourceType(e.target.value);
        if (e.target.value === "Document") {
          setIsDocument(true);
        } else {
          setIsDocument(false);
        }
      }}
    >

```

## Appendix H: Screenshot of the Final App



### Resource Recommender

Algorithms and data structures are central to computer science. The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and for preventing security vulnerabilities.

**Resource Type \***

Video

**Search**

## Resource Recommender

Algorithms and data structures are central to computer science. The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and for preventing security vulnerabilities.

**Resource Type \***


Document



**Difficulty Type \***

Medium



 Search

## Extra Study Resources

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>

 <https://www.youtube.c...>


 <https://www.youtube.c...>

## Extra Study Resources

 <https://www.hpe.com/us...>

 <https://www.geeksforge...>

 <https://en.wikipedia.org/...>

 <https://bootcamp.berkel...>

 <https://security.berkeley...>

 <https://its.ucsc.edu/sec...>

 <https://www.explainingc...>

 <https://www.techtarget...>

 <https://www.imperva.co...>

 <https://scoutapm.com/b...>

 <http://vaidehijoshi.githu...>

## Appendix I: Mathematical process to calculate metrics score.

Weights that get from the article:

Session Starts: 38%

View: 25%

Upload Frequency: 12%

Session Duration: 4%

Session Ends: 4%

Video Is New: 3%

Playlist Adds: 3%

Closed Captions: 3%

Title: 2%

Comments: 2%

Likes: 2%

Tags: 1%

Description: 1%

Total Weight:  $38\% + 25\% + 12\% + 4\% + 4\% + 3\% + 3\% + 3\% + 2\% + 2\% + 2\% + 1\% + 1\% = 100\%$

### 2. Calculate Proportional Weights for Likes, Views, and Comments

Now, to get the proportional weight of each factor (likes, views, and comments) considering only these three:

#### a. Likes

Weight of likes =  $(\text{Given weight of likes} / \text{Total weight of likes, views, and comments}) * 100$

#### b. Views (Average View Duration)

Weight of views =  $(\text{Given weight of views} / \text{Total weight of likes, views, and comments}) * 100$

c. Comments

Weight of comments = (Given weight of comments / Total weight of likes, views, and comments) \* 100

Total weight of likes, views, and comments = 2% + 25% + 2% = 29%

a. Likes

Weight of likes = (2% / 29%) \* 100 = 6.90%

b. Views (Average View Duration)

Weight of views = (25% / 29%) \* 100 = 86.21%

c. Comments

Weight of comments = (2% / 29%) \* 100 = 6.90%

3. Results

Likes: 6.90%

Views: 86.21%

Comments: 6.90%

## Appendix J: Mathematical process to calculate combined score of resource relevance score metrics score.

### Weighting Two Scoring Metrics- video['vscore'] vs resource\_score

#### Questions that asked in the survey and the results.

1. On a scale of 1-5, how familiar are you with YouTube metrics (like count, share count, view count)?  
1 - 2  
2 - 1  
3 - 5  
4 - 10  
5 - 12
2. On a scale of 1-5, how familiar are you with relevance calculations like cosine similarity?  
1 - 11  
2 - 5  
3 - 5  
4 - 5  
5 - 4
3. In your opinion, how much more important is the video['vscore'] compared to the resource\_score?  
Both are equally important. - 6  
video['vscore'] is slightly more important. - 8  
resource\_score is slightly more important - 4  
video['vscore'] is strongly more important. - 11  
resource\_score is strongly more important. – 1

$$Q1W = \frac{\sum (\text{Score}_i * \text{Vote}_i)}{\sum \text{Votes}} = \frac{(1*2 + 2*1 + 3*5 + 4*10 + 5*12)}{30} = 3.97$$

$$Q2W = \frac{\sum (\text{Score}_i * \text{Vote}_i)}{\sum \text{Votes}} = \frac{(1*11 + 2*5 + 3*5 + 4*5 + 5*4)}{30} = 2.53$$

For Q3, consider the Pairwise comparison, it follows scale suggested by Thomas Saaty:

Both are equally important – 0.5

video['vscore'] is slightly more important – 1.25

resource\_score is slightly more important – 0.75

video['vscore'] is strongly more important. – 1.5

resource\_score is strongly more important. – 0.5

$$Q3W = \sum (\text{weights}_i * \text{votes}_i) / \sum \text{votes} = 1.2$$

Final weights:

$$W\_vscore = Q1W * Q3W = 4.764$$

$$W\_resource = Q2W * Q3W = 3.036$$

Normalized weights:

$$W\_vscore\_nor = 4.764 / (4.764 + 3.036) = \underline{0.611}$$

$$W\_resource\_nor = 3.036 / (4.764 + 3.036) = \underline{0.389}$$