**DATA236: Distributed Systems for Data Engineering  
Homework 3**

**Github Link:** <https://github.com/Vimalanandhan/DATA-236---Distributed-Systems-for-Data-Engineering/tree/main/Assignments/Assignment%203>

**Objective**:  
Create a Node.js application with Express that implements a simple user authentication system

for the Department of Applied Data Science at SJSU. The application should include:  
● User login and logout functionality.  
● Session management to keep users logged in.  
● Protected routes that only logged-in users can access.  
● Styling using Bootstrap to make the application visually appealing.

**Requirements:**

**1. Routes- Handled Separately in a router:**  
**Home Page (/):** Display a welcome message for ADS-SJSU. Show a link to the login page if the user is not logged in. Show a link to the dashboard and log out if the user is logged in.

**Routes Handled:**

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

**HomePage:**  
A screenshot of a computer

AI-generated content may be incorrect.

**Login Page (/login):** Display a login form with fields for username and password. Validate the credentials and log the user in if they are correct. Redirect to the dashboard on successful login.

A screenshot of a computer

AI-generated content may be incorrect.

If we give wrong credentials below is the image

A screenshot of a login form

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Dashboard Page (/dashboard):** Display a welcome message with the user’s name. Show a logout link. Protect this route so only logged-in users can access it.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Logout (/logout):** Destroy the session and redirect the user to the home page.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.  
**2. Session Management:** Use express-session to manage user sessions. Store the logged-in user’s information in the session. Ensure that the session cookie is secure

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

express-session to manage user sessions. Stores the logged-in user’s information in the session. Ensure that the session cookie is secure

**3. Styling with Bootstrap:** Explore and use Bootstrap to style all pages. Make the application responsive and visually appealing. Use Bootstrap components, such as the Navbar for navigation, Cards for forms and content, Buttons for actions, and Alerts for messages.

Views folder directory:

A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

**Express app.js code screenshot:**

A screen shot of a computer program

AI-generated content may be incorrect.

**Styling with Bootsrap:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Part 2:** **Compare Three LlamaIndex Chunking Techniques (Retrieval-Only RAG)**

Implement three chunking techniques in LlamaIndex on the Tiny Shakespeare, build  
in-memory vector indexes, and compare retrieval quality. You’ll print the embeddings and  
retrieval outputs for a shared query, then argue which technique is best and why.

Techniques to implement:  
1. Token-based chunking — TokenTextSplitter (LlamaIndex)  
2. Semantic chunking — SemanticSplitterNodeParser (LlamaIndex)  
3. Sentence-window chunking — SentenceWindowNodeParser (LlamaIndex)

**Dataset**

Use the same file as in class:

● **Tiny Shakespeare (raw text):**

https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt

**What you must build**

A. Environment & Setup Install: llama-index, llama-index-embeddings-huggingface,

sentence-transformers, faiss-cpu, numpy, pandas.

●Use a public sentence embedding model (e.g.,

sentence-transformers/all-MiniLM-L6-v2). (pull from huggingface )

B. One retrieval-only pipeline per technique

For each chunker (Token / Semantic / Sentence-window):

**1. Chunking**

○Token: set a token chunk\_size and chunk\_overlap (choose sensible values).

( LlamaIndex )

○Semantic: pick a buffer\_size and use your embed model for the splitter to find

semantically coherent boundaries. ( LlamaIndex )

○Sentence-window: split to single sentences and attach a window (neighbor

sentences) in metadata to keep surrounding context available. ( LlamaIndex )

**2. Indexing (in memory)**

○Build a VectorStoreIndex over your nodes with an in-memory vector store (e.g.,

SimpleVectorStore) to keep everything local and fast. ( LlamaIndex )

**3. Retrieval-only function**

Write a helper that, given a query and k, does the following:

○Compute the query embedding (show its dimension and the first 8 values ).

○Retrieve top-k nodes; for each, compute and print:

■ Store similarity score (if available from retriever).

■ Cosine similarity between the query embedding and the document

embedding (compute embeddings of the returned chunks explicitly).

■ Chunk length and a short text preview (first ~160 chars).

○Print the shapes of the query vector and the stacked doc vectors.

Your printed output should clearly identify the technique used and list a table with:

rank, store\_score, cosine\_sim, chunk\_len, preview.

Query to use

Use this one query to print outputs for all three techniques:

●Query: Who are the two feuding houses?

You may optionally add 1–2 more queries (like, “Who is Romeo in love with?” , “Which play

contains the line ‘To be, or not to be’?” ) to strengthen your comparison

What to compare (report section)

After you run the three pipelines:

1. Retrieval Quality:

* top-1 cosine (highest similarity among the top-k for that technique)
* mean@k cosine (average of top-k cosines)
* #chunks produced by the chunker and the avg chunk length (characters or

tokens)

* retrieval latency in milliseconds (time the similarity search took; simple timer is

fine)

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Dataset Information

-Dataset: Tiny Shakespeare

-Source: https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt

-Embedding Model: sentence-transformers/all-MiniLM-L6-v2

Technique Configurations

1. Token-based Chunking: chunk\_size=512, chunk\_overlap=50

2. Semantic Chunking: buffer\_size=1, breakpoint\_percentile\_threshold=95

3. Sentence-window Chunking: window\_size=3

**Chunking Statistics**

| Technique | Total Chunks | Avg Chunk Length (chars) |

|-----------|--------------|---------------------------|

| Token | 657 | 1879.4 |

| Semantic | 624 | 1787.5 |

| Sentence\_Window | 12453 | 89.6 |

**Retrieval Quality Metrics**

Query: "Who are the two feuding houses?"

| Technique | Top-1 Cosine | Mean@k Cosine | Retrieval Time (ms) |

|-----------|--------------|---------------|---------------------|

| Token | 0.3063 | 0.2822 | 1049.32 |

| Semantic | 0.3776 | 0.3038 | 23.63 |

| Sentence\_Window | 0.5126 | 0.4661 | 203.18 |

Query: "Who is Romeo in love with?"

| Technique | Top-1 Cosine | Mean@k Cosine | Retrieval Time (ms) |

|-----------|--------------|---------------|---------------------|

| Token | 0.5757 | 0.5575 | 323.43 |

| Semantic | 0.6302 | 0.6025 | 14.53 |

| Sentence\_Window | 0.8024 | 0.7892 | 120.37 |

Query: "Which play contains the line 'To be, or not to be'?"

| Technique | Top-1 Cosine | Mean@k Cosine | Retrieval Time (ms) |

|-----------|--------------|---------------|---------------------|

| Token | 0.4110 | 0.3852 | 40.55 |

| Semantic | 0.4095 | 0.3759 | 11.73 |

| Sentence\_Window | 0.5407 | 0.4900 | 115.38 |

**Code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**