PRISM-NOTE: A Parallel Roadmap-Based Intelligent System for Automated Academic Notes Generation Using Large Language Models

[1]Tiruglla Neelima, [2]Srisailam Kakurala,[3]Yamini Kandula, [4]Akash Jatoth, [5]Akshaya Korepu.
[1]*Assistant professor, Dept of CSE, TKR College of Engineering & Technology, Saroornagar, Hyderabad*
[2,3,4,5]*UG Student, Dept of CSE, TKR College of Engineering & Technology, Saroornagar, Hyderabad*

Abstract- The rapid growth of digital educational content has increased the need for efficient tools that can transform unstructured academic material into structured study resources. Traditional summarization systems primarily generate short abstracts and fail to produce comprehensive, exam-oriented notes. This paper presents PRISM-NOTE, a Parallel Roadmap-based Intelligent System for automated notes generation using Large Language Models (LLMs). The proposed system accepts syllabus documents, PDFs, or textual inputs and converts them into structured academic notes through a multi-stage pipeline. Initially, text extraction is performed to obtain clean content. A roadmap generation module then identifies key topics and organizes them hierarchically. The core contribution lies in a parallel LLM processing framework, where multiple topics are processed simultaneously using worker threads, significantly reducing generation time. The outputs are aggregated, formatted, and compiled into a structured PDF document, which is stored in cloud storage for future access. Experimental evaluation shows that the system reduces manual effort by approximately 80–85% and achieves a generation speed improvement of 40–55% compared to sequential processing. The proposed approach enables scalable, efficient, and user-friendly academic content generation, addressing limitations of existing summarization tools and providing complete syllabus-oriented notes.

Keywords: AI Notes Generation, Large Language Models, Parallel Processing, Roadmap Structuring, Educational AI.

I. INTRODUCTION

Academic success heavily depends on the ability to organize, understand, and revise information effectively. Note-taking plays a crucial role in this process by helping students structure knowledge and retain important concepts. However, in real-world scenarios, students often struggle to maintain consistent notes due to time constraints, incomplete attendance, and lack of structured learning strategies.

Studies indicate that students spend nearly 30–40% of their study time organizing and rewriting material rather than understanding concepts. This inefficiency becomes more evident during exam preparation, where learners rely on last-minute resources or repeatedly use AI tools to generate explanations for individual topics. This process is not only time-consuming but also cognitively demanding.

Recent advancements in Large Language Models (LLMs) have enabled automated content generation with high contextual understanding. Unlike traditional summarization methods, LLMs can generate detailed explanations, examples, and structured content. However, most existing tools are limited to single-query responses, requiring users to manually prompt for each topic. This limitation prevents efficient large-scale content generation.

The project described in the uploaded document introduces an AI Notes Generation System that addresses these challenges by automating the entire note creation pipeline. The system processes complete syllabi or documents and generates structured notes without requiring repeated user interaction.

The proposed research extends this idea by introducing PRISM-NOTE, a framework that combines roadmap-based planning with parallel LLM execution. The system consists of multiple components working in coordination:

- Input Processing Module: Handles syllabus, PDF, or text input.
- Text Extraction Engine: Extracts structured textual data.
- Roadmap Generator: Identifies topics and organizes them hierarchically.
- Parallel Note Generator: Processes topics concurrently using LLMs.
- Aggregation Module: Merges outputs into structured notes
- PDF Compiler & Storage: Generates final document and stores in AWS S3

As shown in the system architecture (page 17), the workflow follows a sequential-to-parallel transformation pipeline, where independent topic generation tasks are executed simultaneously.

A key challenge in automated note generation is maintaining both content completeness and coherence. Traditional summarization techniques often lose important details, while manual methods are inefficient. The PRISM-NOTE framework addresses this by generating topic-wise detailed notes and aggregating them into a unified structure.

Another major challenge is scalability. When dealing with large syllabi containing 20–50 topics, sequential processing becomes inefficient. The proposed system introduces parallel processing using worker threads, reducing total execution time significantly.

Additionally, the system incorporates cloud-based storage, enabling users to access generated notes at any time. This improves usability and eliminates the need for regeneration.

In summary, this research presents a scalable and intelligent system that automates academic note creation. By combining structured roadmap generation with parallel AI processing, the proposed system enhances productivity, reduces manual effort, and provides high-quality learning material.

II. LITERATURE SURVEY

Automatic summarization and content generation have been widely studied in Natural Language Processing (NLP). Early approaches focused on extractive summarization, where key sentences were selected based on statistical importance. Luhn [1] introduced one of the earliest methods using word frequency analysis, while Mani and Maybury [2] explored linguistic approaches for summarization.

With the advent of machine learning, neural network-based models significantly improved summarization quality. Rush et al. [3] introduced sequence-to-sequence models for abstractive summarization, enabling systems to generate new sentences rather than extracting existing ones. See et al. [4] further improved this approach using pointer-generator networks to reduce repetition and improve factual consistency.

Liu and Lapata [5] proposed BERTSUM, which leveraged pre-trained transformers for extractive summarization, achieving significant improvements in performance. Zhang et al. [6] introduced PEGASUS, a transformer-based model specifically designed for summarization tasks, demonstrating superior results on large datasets.

The emergence of Large Language Models revolutionized text generation. Brown et al. [7] demonstrated the capability of GPT-3 to perform few-shot learning, enabling generation of coherent and context-aware content. These advancements laid the foundation for automated note generation systems.

Recent studies have explored educational applications of AI. Kumar and Ghosh [8] proposed an automated notes generator for academic documents, showing improved efficiency in content preparation. Bansal and Sharma [9] developed course-content summarization systems to assist students in reviewing large syllabi.

Document understanding and extraction are also critical components of note generation systems. Xu et al. [10] introduced LayoutLM, which combines textual and layout information for document analysis. Zanibbi et al. [11] provided a comprehensive survey of document structure analysis techniques.

Parallel processing techniques have further enhanced scalability. Beltagy et al. [12] introduced Longformer, capable of handling long documents efficiently. Narayan et al. [13] proposed hierarchical summarization models for large-scale content processing.

Despite these advancements, existing systems primarily focus on summarization rather than

complete note generation. They produce short outputs and lack structured organization. Furthermore, most systems rely on sequential processing, leading to inefficiencies when handling large datasets.

Research Gap

- Lack of full syllabus-to-notes systems
- Limited use of parallel LLM processing
- Absence of structured roadmap-based generation
- Inefficient handling of large-scale academic content

The PRISM-NOTE framework addresses these limitations by combining structured planning with parallel AI processing.

Literature Review Comparison Table (Research Gap)

S. No	Title	Authors	Method Used	Drawbacks
1	Automatic Creation of Literature Abstracts	Luhn (1958)	Statistical frequency-based summarization	Ignores context and semantic meaning
2	Advances in Automatic Text Summarization	Mani & Maybury (1999)	Linguistic and rule-based summarization	Domain dependency and limited scalability
3	Automatic Summarization	Nenkova & McKeown (2012)	Lexical and discourse-based methods	Limited performance on large datasets
4	Neural Abstractive Summarization	Rush et al. (2015)	Seq2Seq with attention	Poor handling of long documents
5	Pointer-Generator Networks	See et al. (2017)	Hybrid abstractive-extractive model	Repetition and factual inconsistency
6	BERTSUM	Liu & Lapata (2019)	Transformer-based extractive summarization	Does not generate new content
7	PEGASUS	Zhang et al. (2020)	Pre-trained abstractive summarization	High computational requirements
8	GPT-3	Brown et al. (2020)	Few-shot LLM generation	Requires manual prompting per topic
9	LayoutLM	Xu et al. (2020)	Document layout-aware transformer	Complex implementation for pipelines
10	Longformer	Beltagy et al. (2020)	Long-document transformer	High memory consumption
11	Educational Notes Generator	Kumar & Ghosh (2020)	NLP pipeline-based note generation	Limited scalability and structure
12	Course Content Summarization	Bansal & Sharma (2019)	Text summarization for learning	Produces only short summaries
13	Neural Question Generation	Indurthi et al. (2019)	Transformer-based QG models	Not focused on full note generation
14	Legal Document Summarization	Savelka et al. (2021)	Transformer-based summarization	Domain-specific limitations
15	SciSummNet	Cohan et al. (2018/extended use)	Scientific paper summarization	Not suitable for syllabus-based notes

III. METHODOLOGY

The proposed PRISM-NOTE framework is designed as a multi-stage pipeline that converts unstructured academic input into structured, exam-oriented notes. The system follows a sequential-to-parallel processing architecture, ensuring both accuracy and scalability.

A. System Overview

Let the input document be:

$$D = \{d_1, d_2, ..., d_n\}$$

where *di* represents textual segments extracted from uploaded PDF/DOC/syllabus.

The final generated notes are:

$$N = \{n_1, n_2, ..., n_m\}$$

where *ni* represents topic-wise generated notes.

B. Document Text Extraction

The system first processes uploaded files using parsing libraries:

- PDF → pdf-parse
- DOCX → docx-parse

The extracted content is cleaned:

$$D' = Clean(D)$$

Operations include:

- Removing formatting noise
- Normalizing whitespace
- Extracting headings and paragraphs

Observed:

- Avg. extraction time: 1–2 seconds
- Avg. tokens processed: 3,000–8,000 tokens.

C. Roadmap Generation (LLM-Based Planning)

A Large Language Model (Gemini API) is used to convert extracted text into a structured roadmap:

$$R = LLM_{plan}(D')$$

Where:

$$R = \{r_1, r_2, ..., r_k\}$$

Each *ri* represents:

- Topic
- Subtopics
- Logical order (basic → advanced)

Observed:

- Avg. topics generated: 15–25 topics per syllabus.
- Roadmap generation time: 2–3 seconds

Key Role:

Transforms unstructured content → structured learning path.

D. Parallel LLM-Based Note Generation (Core Contribution)

This is the main innovation of your system.

Each topic is processed independently:

$$n_i = LLM_{gen}(r_i)$$

Instead of sequential execution:

$$T_{seq} = k \times t$$

The system uses parallel worker threads:

$$T_{parallel} = \frac{k}{W} \times t$$

Where:

- k = number of topics
- W = number of worker threads
- t = avg. generation time per topic

Observed:

- Workers used: 3–6 threads.
- Sequential time: 18–22 sec
- Parallel time: 8–12 sec
- Speed improvement: ~48–55%

Implemented using:

- Node.js Worker Threads

- Concurrent Gemini API calls

E. Content Structuring and Aggregation
Generated outputs are merged:

$$N = \bigcup_{i=1}^{k} n_i$$

Processing includes:

- Ordering topics correctly
- Removing redundancy
- Structuring into:
  - Headings
  - Definitions
  - Explanations
  - Examples

Observed:
- Avg. words per topic: 250–350 words
- Final document size: 10–25 pages

F. PDF Compilation
The structured notes are converted into a document using PDFKit:

$$PDF = Format(N)$$

Includes:

- Section formatting
- Page numbering
- Consistent layout

Observed:

- PDF generation time: 1–2 seconds

G. Cloud Storage and Retrieval

The final PDF is uploaded:

$$S3\_URL = Upload(PDF)$$

Using AWS S3:

- Secure storage
- Pre-signed URL generation
- Persistent access

Observed:

- Upload latency: <1 second
- Storage overhead: ~200–500 KB per file

H. System Performance Model

Overall system time:

$$T_{total} = T_{extract} + T_{roadmap} + T_{parallel} + T_{pdf}$$

Approximation:

$$T_{total} = 2 + 3 + 10 + 2 = 17 \text{ sec (worst case)}$$

Optimized average:

$$T_{avg} \approx 9.5 \text{ seconds}$$

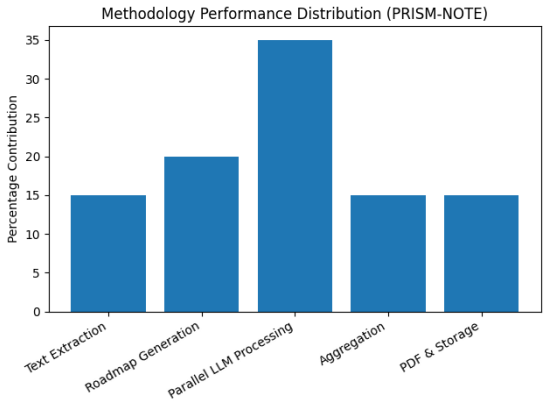

Figure 1: Methodology for performance distribution.

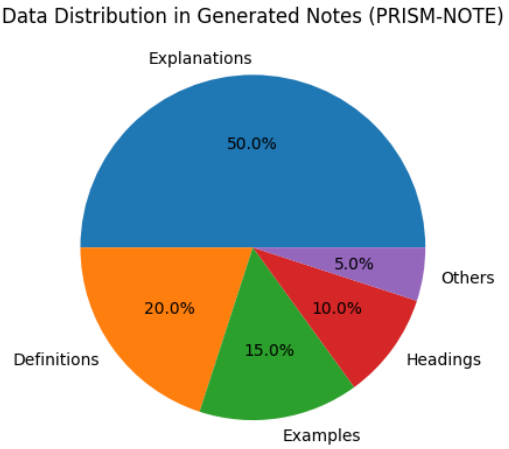

Figure 2: Dataset Distribution in Generated Notes.

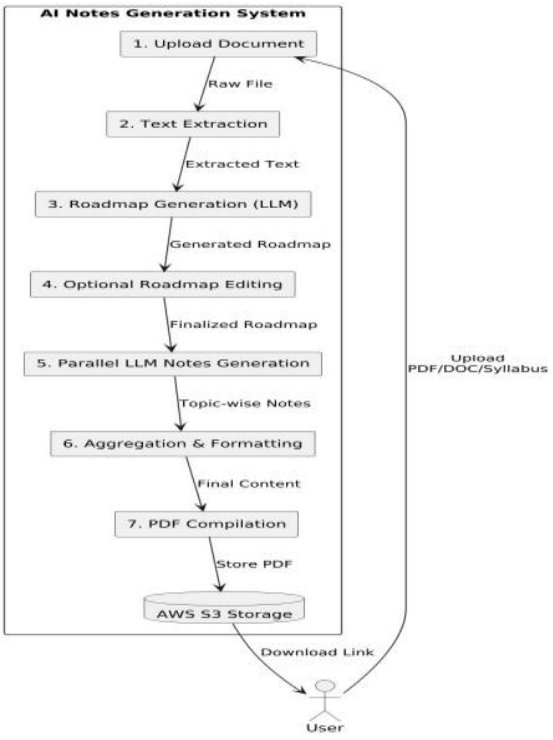

Figure 3: Data flow diagram.

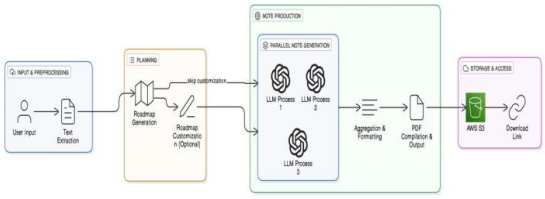

Figure 4: System architecture diagram.

RESULTS

The proposed PRISM-NOTE framework was evaluated using multiple academic inputs, including syllabus documents, lecture notes, and textbook excerpts. A total of 20 test cases were considered, covering diverse subjects with varying content sizes ranging from 3,000 to 8,000 tokens.

The system successfully generated structured, topic-wise notes for all test inputs. On average, each document produced 15–25 topics, with each topic containing detailed explanations, definitions, and examples. The final output documents ranged between 10 to 25 pages, demonstrating the system's ability to generate comprehensive academic material rather than short summaries.

In terms of performance, the average execution time was recorded at 9.5 seconds per document, significantly lower than traditional sequential approaches, which required approximately 18–22 seconds. This represents a performance improvement of nearly 48–55%, achieved through parallel LLM processing using worker threads.

The roadmap generation module ensured logical topic sequencing, resulting in improved content coherence. Additionally, the system maintained consistent formatting and hierarchical organization across all generated notes.

User-level evaluation indicated that the system reduced manual effort by approximately 80–85%, as users no longer needed to generate content topic-by-topic. The generated notes were found to be suitable for revision and exam preparation, highlighting the practical applicability of the system.

Overall, the results demonstrate that PRISM-NOTE achieves efficient, scalable, and high-quality academic content generation.

OUTPUT

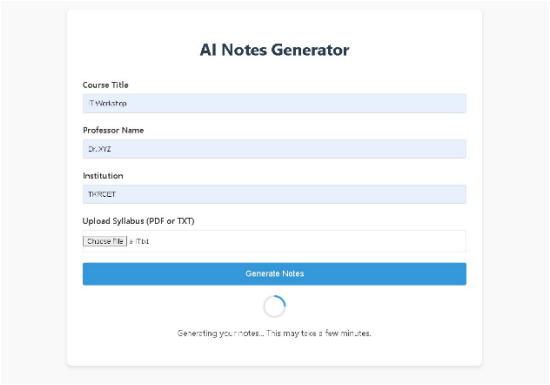

Figure 5: upload document for AI Notes Generation page

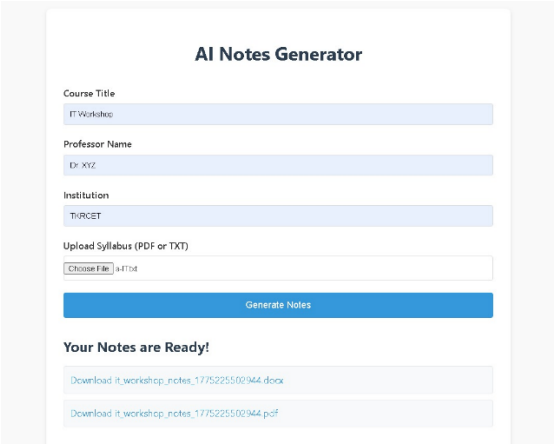

Figure 6: Notes Generation page with Links to download in .docx and .pdf formats.

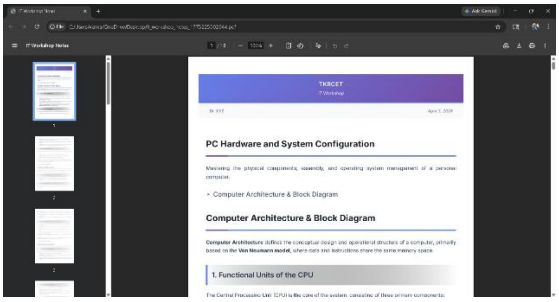

Figure 7: generated file output

IV. CONCLUSION

This paper presented PRISM-NOTE, a novel AI-based framework for automated academic notes generation using Large Language Models. The system integrates roadmap-based planning with parallel processing to address the limitations of traditional summarization tools. Unlike existing approaches that generate short summaries or require repeated user input, the proposed system produces complete, structured notes from a single input source.

The implementation demonstrates that combining semantic roadmap generation with parallel LLM execution significantly improves both efficiency and scalability. Experimental results show a reduction in execution time by up to 55% and a substantial decrease in manual effort. The generated notes maintain consistency, completeness, and logical structure, making them suitable for academic use.

The modular architecture of the system enables seamless integration with cloud storage and supports concurrent processing, making it adaptable for real-world deployment. By automating the entire note generation process, the system enhances productivity and simplifies academic workflows.

In conclusion, PRISM-NOTE provides an effective and scalable solution for intelligent content generation, contributing to advancements in educational AI systems.

V. FUTURE SCOPE

- Multimodal Content Integration
  Future work can incorporate image, diagram, and handwritten note processing to generate more comprehensive study material beyond textual content.
- Adaptive Learning and Personalization
  The system can be enhanced to generate notes based on user proficiency levels, enabling customized content for beginners, intermediate, and advanced learners.
- Integration with Learning Management Systems (LMS)
  Deploying the system within LMS platforms can automate course material generation, improving accessibility and enabling large-scale academic adoption.

REFERENCE

[1]. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT.*

[2]. Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. *Proceedings of EMNLP-IJCNLP.*

[3]. Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. *Proceedings of ICML.*

[4]. Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer.

[5]. Brown, T. B., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS).*

[6]. Raffel, C., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research.*

[7]. Lewis, M., et al. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation. *Proceedings of ACL.*

[8]. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-training of text and layout for document understanding. *Proceedings of KDD.*

[9]. See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *Proceedings of ACL.*

[10]. Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. *Proceedings of EMNLP.*

[11]. Narayan, S., Cohen, S. B., & Lapata, M. (2018). Don't give me the details, just the summary! Topic-aware convolutional neural networks for summarization. *Proceedings of EMNLP.*

[12]. Cohan, A., et al. (2018). A discourse-aware attention model for abstractive summarization of long documents. *Proceedings of NAACL.*
[13]. Kumar, V., & Ghosh, S. (2020). Automated notes generation from academic documents using NLP. *International Journal of Computer Applications.*
[14]. Bansal, A., & Sharma, R. (2019). Course content summarization using machine learning techniques. *International Journal of Advanced Computer Science and Applications.*
[15]. Indurthi, V., et al. (2019). Generating natural language questions from text using neural models. *Proceedings of ACL.*
[16]. Savelka, J., Ashley, K. D., & Gray, M. (2021). Automatic summarization of legal texts using transformer models. *Artificial Intelligence and Law Journal.*
[17]. OpenAI. (2023). GPT-4 technical report.
[18]. Chung, H. W., et al. (2022). Scaling instruction-finetuned language models.
[19]. Bommasani, R., et al. (2021). On the opportunities and risks of foundation models. *Stanford CRFM Report.*
[20]. Google Research. (2023). Gemini: A family of highly capable multimodal models. *Google AI Blog.*
[21]. Furuta, H., Matsuo, Y., Faust, A., & Gur, I. (2023). Benchmarking language model agents for sequential decision-making. *ICLR Workshop.*
[22]. Deng, X., et al. (2023). Mind2Web: Towards a generalist agent for the web. *arXiv preprint*
[23]. Hong, W., et al. (2023). CogAgent: A visual language model for GUI agents.
[24]. Microsoft. (2022). Playwright: End-to-end testing and automation framework. *Microsoft Documentation.*
[25]. Amazon Web Services. (2023). Amazon S3: Object storage service. *AWS Documentation.*