SMART PREP: AI POWERED SUMMARIZER AND TESTS

A PROJECT REPORT

Submitted by

ABIINAYA SHREE J	**811722104004**
CHARULATHA K	**811722104023**
HARSHITHA K	**811722104052**

in partial fulfillment of the requirements for the award degree of

Bachelor in Engineering

20CS7503 DESIGN PROJECT - 3

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621112

NOVEMBER 2025

SMART PREP: AI POWERED SUMMARIZER AND TESTS

A PROJECT REPORT

Submitted by

ABIINAYA SHREE J	**811722104004**
CHARULATHA K	**811722104023**
HARSHITHA K	**811722104052**

in partial fulfillment of the requirements for the award degree of

Bachelor in Engineering

20CS7503 DESIGN PROJECT - 3

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621112

NOVEMBER 2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM - 621112

BONAFIDE CERTIFICATE

The work embodied in the present project report entitled "**SMART PREP: AI POWERED SUMMARIZER AND TESTS**" has been carried out by the students ABINAYA SHREE J, CHARULATHA K, HARSHITHA K, The work reported herein is original and we declare that the project is their own work, except where specifically acknowledged, and has not been copied from other sources or been previously submitted for assessment.

Date of Viva Voce: ……

Mr. A. MALARMANNAN
SUPERVISOR
Assistant Professor
Computer Science And Engineering
K Ramakrishnan College Of
Technology (Autonomous)
Samayapuram – 621112

Mr. R. RAJAVARMAN
HEAD OF THE DEPARTMENT
Assistant Professor
Computer Science And Engineering
K Ramakrishnan College Of
Technology (Autonomous)
Samayapuram – 621112

INTERNAL EXAMINER **EXTERNAL EXAMNIER**

ABSTRACT

Reading large PDFs and DOCX files manually is time-consuming and often leads to missed information. Students, teachers, and professionals struggle to extract key points and generate study questions efficiently. This project presents an AI-powered Document Processing System that uses NLP to summarize documents and create both short- and long-answer questions automatically. With features like file upload, page-range selection, and export options, users receive accurate point-based or paragraph summaries quickly. The system boosts productivity, improves learning effectiveness, and provides a fast, user-friendly way to understand complex documents. Additionally, the intelligent question-generation module helps users reinforce understanding and prepare for exams or reviews with ease. Overall, the system transforms raw documents into meaningful insights, enabling faster decision-making and deeper comprehension.

Keywords: AI Document Processing, NLP Summarization, Automatic Question Generation, PDF/DOCX Analysis, Short/Long Answer Questions, Educational Technology, Intelligent Summarization, File Upload System, Learning Efficiency, Automated Assessment Tools.

ACKNOWLEDGEMENT

We thank our **Dr. N. Vasudevan**, Principal, for his valuable suggestions and support during the course of my research work.

We thank our **Mr. R. Rajavarman**, Head of the Department, Assistant Professor (Sr. Grade), Computer Science and Engineering, for his valuable suggestions and support during the course of my research work.

We wish to record my deep sense of gratitude and profound thanks to my Guide **Mr. A. Malarmannan**, Assistant Professor, Computer Science and Engineering for his keen interest, inspiring guidance, constant encouragement with my work during all stages, to bring this thesis into fruition.

We are extremely indebted to our project coordinator **Mrs. R. Ramasaraswathi,** Assistant Professor, Computer Science and Engineering for her valuable suggestions and support during the course of my research work.

We also thank the faculty and non-teaching staff members of the Computer Science and Engineering, K Ramakrishnan College of Technology, Samayapuram for their valuable support throughout the course of my research work.

Finally, we thank our parents, friends and our well wishes for their kind support.

SIGNATURE

_____

_____

_____

_____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
API	-	Application Programming Interface
DOC	-	Microsoft Word Document
DOCX	-	Microsoft Word Open XML Document
HTML	-	Hyper Text Markup Language
HTTP	-	Hyper Text Transfer Protocol
IEEE	-	Institute of Electrical and Electronics Engineers
J SON	-	JavaScript Object Notation
LSTM	-	Long Short-Term Memory
TF-IDF	-	Term Frequency–Inverse Document Frequency
REST	-	Representational State Transfer
NLP	-	Natural Language Processing
PDF	-	Portable Document Format

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The Document AI System is an advanced, web-based intelligence platform designed to transform the way users handle and understand large PDF and DOC/DOCX documents. It combines the power of modern Natural Language Processing (NLP) with a user-friendly interface to simplify, accelerate, and enhance document reading and knowledge extraction. Instead of manually scanning through lengthy material—an often tiring and time-consuming process—users can rely on the system to instantly generate meaningful insights, summaries, and study-oriented questions with exceptional accuracy.

At its core, the platform enables users to upload documents of varying sizes, select specific page ranges, and automatically obtain structured outputs tailored to their objectives. Whether preparing for exams, conducting research, or reviewing professional documents, users can generate **point-based summaries**, **cohesive paragraph summaries**, **short-answer questions**, or **long-answer descriptive questions** with just a few clicks. By handling the heavy cognitive load of reading, filtering, and organizing content, the system significantly reduces manual effort and ensures a more efficient and consistent workflow.

Ultimately, the Document AI System goes far beyond conventional text-extraction tools. Instead of merely pulling raw text, it offers **intelligent, value-driven document understanding**, making it a practical companion for modern digital learning, rapid content review, exam preparation, research workflows, and day-to-day productivity. Its ability to convert complex documents into structured knowledge makes it an essential tool for anyone dealing with large amounts of text in today's fast-paced information environment. With a focus on usability and accuracy, the platform provides a seamless experience that is both beginner-friendly and powerful enough for advanced users.

1.2 PROBLEM STATEMENT

Reading and analyzing detailed PDF and DOC/DOCX documents demands significant time, concentration, and iterative review, which becomes overwhelming for students, teachers, and professionals who frequently deal with large volumes of text. Manual summarization and question creation not only slows down learning and research workflows but also increases the chances of biased interpretation, incomplete coverage, and cognitive fatigue. Existing tools mainly extract raw text, lacking intelligent capabilities such as context-aware summarization, question generation, selective processing, and customized output formatting. Therefore, there is a pressing need for an automated, accurate, and interactive system that can intelligently process documents, extract only the required content, and instantly generate structured summaries and meaningful assessment-based questions, ultimately improving efficiency, reducing effort, and enhancing the overall content understanding experience.

1.3 OBJECTIVE

The primary objective of this project is build an intelligent and automated document processing system that can efficiently extract, summarize, and generate questions from uploaded PDF and DOC/DOCX files using NLP techniques.

- The system aims to minimize manual reading time, reduce effort, and eliminate inconsistencies by providing instant, accurate, and meaningful summaries in both point-based and paragraph-based formats.

- Another goal is to generate structured short-answer and long-answer questions that can support exam preparation, revision, teaching, evaluation, and self-learning without requiring additional manual work.

- The platform is designed to allow users to select full documents or specific page ranges, ensuring personalized processing and focused output based on relevance and learner intention.

- The system will integrate essential modules such as secure file upload, flexible action selection, NLP-based text processing, and output export functionality to deliver an end-to-end automated workflow.

- A key objective is to implement a simple, interactive, and responsive web interface that ensures easy navigation, seamless user experience, and accessibility across different learning and working environments.

- The project aims to leverage scalable and modular architecture, enabling future support for advanced AI models, multilingual capabilities, additional export formats, and real-time collaboration features.

- The system aims to ensure high accuracy and relevance by continuously refining its NLP algorithms, enabling more context-aware summaries and intelligently framed questions over time.

- Another objective is to maintain strong data privacy and security measures, ensuring that uploaded documents and generated outputs remain confidential and protected throughout the processing workflow.

- The platform also intends to provide performance optimization features such as faster processing for large files, resource-efficient computation, and smooth scalability even under heavy user load.

- Ultimately, the system intends to serve as a reliable, productivity-enhancing digital assistant for students, educators, researchers, and professionals by enabling faster comprehension, improved retention, and smarter knowledge extraction.

- The system also aims to include intelligent personalization features, allowing users to customize summary depth, question difficulty level, preferred formats, and domain-specific processing settings, ensuring outputs that align with individual learning styles and professional needs.

CHAPTER 2

LITERATURE SURVEY

2.1 ADVANCED EXTRACTIVE DOCUMENT SUMMARIZATION USING TF-IDF, STATISTICAL WEIGHTING & SENTENCE PRIORITIZATION

S. Gupta Et Al, In this research explored a classical yet robust extractive summarization strategy leveraging TF-IDF statistical weighting to identify sentences with the highest informational value. Through word frequency analysis, feature scoring, and positional heuristics, the authors demonstrated that essential content units could be isolated with minimal semantic loss. Preprocessing techniques such as stop-word removal, lemmatization, and normalization significantly reduced dimensionality and improved computational efficiency.

Their experiments revealed that extractive summarizers are highly efficient for educational and research documents where factual preservation is critical. However, the summaries lacked contextual smoothness and narrative coherence, highlighting the limitations of purely statistical models for human-like summarization output. However, while the model efficiently captured key informational fragments, it struggled to maintain narrative coherence and natural sentence flow due to its reliance on surface-level lexical statistics rather than semantic understanding.

The study concluded that although statistical extractive methods serve as strong baseline models for real-time and resource- constrained applications, they lack the contextual fluidity required to produce human- quality summaries, emphasizing the need for hybrid or neural-based semantic enhancement in modern summarization systems.

This study presented a classical extractive summarization method using TF-IDF, statistical weighting, and positional heuristics to identify high-value sentences. With preprocessing steps like stop-word removal and lemmatization, the method efficiently reduced dimensionality and performed well for factual documents.

2.2 ABSTRACTIVE TEXT SUMMARIZATION USING ATTENTION-DRIVEN NEURAL ARCHITECTURES & ENCODER-DECODER PIPELINES

M. A. Rahman & S. Chowdhury, In this study introduced an advanced deep learning–driven abstractive summarization framework built upon an Encoder–Decoder architecture utilizing Long Short-Term Memory (LSTM) networks coupled with an attention mechanism for improved contextual representation. Unlike extractive methods that merely select key sentences, this model actively reconstructs meaning, generating new sentences that reflect the essence of the original content using semantic embedding vectors and contextual sequence learning. The attention module further enhanced the system by dynamically prioritizing relevant input segments during decoding, leading to more accurate, logically ordered, and semantically rich summaries.

Experimental results demonstrated that the generated summaries exhibited natural linguistic flow, improved coherence, and superior grammatical fluency, making them comparable to human-written abstracts. This positioned abstractive summarization as a powerful approach for educational content restructuring, research document analysis, and AI- driven learning ecosystems, especially where comprehension and readability are critical. However, the study acknowledged several practical constraints, including the need for large annotated datasets, extensive model training time, GPU-based acceleration, hyperparameter optimization, and continual fine-tuning for domain specificity. While offering unparalleled summary quality, the model posed real-world deployment challenges in low-resource or web-based environments.

As a result, the authors emphasized hybrid methodologies combining extractive precision with neural-based semantic fluency, making them ideal candidates for scalable, user- centric document processing systems such as Smart-Prep.

This system produced summaries with better fluency, coherence, and readability compared to extractive techniques. Despite its strong performance, it required large datasets, long training times, and heavy computational resources.

2.3 AUTOMATED QUESTION GENERATION VIA LINGUISTIC PARSING, INFORMATION EXTRACTION & SEMANTIC ROLE LABELING

T. Nguyen & K. Lee, In This research presented an automated assessment-generation framework designed to transform textual learning materials into structured, pedagogically aligned questions suitable for academic and training environments. The system leveraged NLP modules such as Named-Entity Recognition, syntactic parsing, and dependency-tree analysis to identify key concepts, domain-specific terms, and contextual relationships within the input text. These knowledge units were then reformulated into WH- questions, MCQs, and fill-in-the-blank formats, enabling scalable and curriculum- aligned assessment creation while reducing educator workload and manual drafting efforts.

The study also highlighted inherent constraints tied to rule-driven NLP models, particularly in generating higher-order cognitive questions classified under Bloom's Taxonomy (analysis, application, evaluation, and synthesis). The system showed limited ability to interpret deep semantics, contextual inference, and cross- concept reasoning, restricting its scope to recall-oriented and comprehension-level questions.

To address these limitations, the authors suggested integrating contextual language models, transformer-based architectures, or hybrid semantic-knowledge frameworks to enhance reasoning-based question generation.

Their work forms a crucial baseline for AI-driven educational technology systems and strongly supports the evolution of next-generation platforms like Smart-Prep, which aim to unify automated summarization and assessment-driven content transformation for advanced digital learning ecosystems.

The model effectively produced WH-questions, MCQs, and fill-in-the-blank items based on key concepts. However, it struggled with higher-order reasoning tasks and deeper inference, limiting it mostly to recall-based questions. The authors recommended integrating transformer-based models to improve conceptual and analytical question generation.

2.4 CONTEXT-AWARE SEMANTIC UNDERSTANDING USING WORD EMBEDDINGS, DEEP NEURAL ENCODERS & HYBRID SIMILARITY MODELS

R. Pandey & M. Joseph, In this paper introduced a hybrid NLP framework that combines classical lexical similarity techniques with deep semantic embedding models such as Word2Vec, GloVe, and BERT to analyze textual meaning at both surface and contextual levels. The hybrid approach demonstrated superior capability in identifying conceptual alignment, thematic consistency, and paraphrased equivalence even when sentence order, structure, or vocabulary were modified.

Traditional keyword-matching or TF-IDF-based similarity models were shown to be insufficient in scenarios involving complex rephrasing, synonym variation, abstract terminology, or contextual referencing; whereas embedding-based models learned latent semantic relationships using multi- dimensional vector representations.

Experimental results revealed improved performance across multiple NLP tasks, including automatic summarization, plagiarism detection, document clustering, semantic classification, question generation, and retrieval-based learning applications. Moreover, the study highlighted that semantic embeddings help systems generalise across domains, languages, and writing styles, making them scalable for modern AI-driven educational technology platforms.

The research strongly advocated that contemporary NLP systems must evolve from deterministic rule-based pipelines to hybrid architectures that combine statistical precision, contextual reasoning, and deep semantic interpretation—the same principle that underpins intelligent document processing solutions like Smart-Prep.

The system effectively captured both lexical and contextual meaning, improving tasks such as summarization, plagiarism detection, and document clustering. The authors highlighted that embedding-based approaches outperform traditional keyword methods, especially for paraphrasing and contextual variations, and emphasized transitioning toward hybrid semantic architectures for modern NLP applications.

2.5 UNIFIED AI-ASSISTED DIGITAL LEARNING AND DOCUMENT ANALYTICS ECOSYSTEM

A. Kulkarni and V. Shah introduced a next-generation end-to-end digital learning companion designed to fundamentally rethink how learners interact with academic material. Their system moved far beyond conventional NLP tools—which typically generate isolated outputs such as summaries or keyword lists—by creating a unified, continuously adaptive learning workflow.

One of the standout aspects of their work was the focus on dynamic learning intelligence. Instead of passively transforming text, the system actively supported learners through features such as visual segmentation of dense sections, indicators highlighting comprehension difficulty across passages, personalized overlays that spotlighted critical insights, and tier-based scaffolding that adjusted the depth of content based on a learner's proficiency.

Yet, the authors also emphasized critical challenges on the horizon. Achieving high accuracy across disparate subject domains remains difficult, especially when content spans sciences, humanities, and emerging fields with rapidly evolving terminology.

The need for robust multilingual capabilities is becoming increasingly urgent as global learners adopt AI-driven study tools. Additionally, the demand for real-time performance pressures systems to maintain low processing latency without compromising quality.

Overall, Kulkarni and Shah's findings closely mirror the ambitions of Smart-Prep. Their work reinforces the strategic importance of combining integrated summarization, comprehension assistance, adaptive learning insight generation, and automated assessment into a single cohesive platform. As both the research and Smart-Prep vision illustrate, the future of learning hinges on holistic, interactive, and deeply intelligent digital companions capable of supporting learners from the moment they upload content to the moment they master it.

2.6 TRANSFORMER-BASED ABSTRACTIVE SUMMARIZATION FOR EDUCATIONAL TEXTS

A. Kumar et al. conducted an extensive investigation into the effectiveness of modern transformer-based encoder–decoder architectures—specifically BART and T5—for producing concise yet conceptually faithful summaries of dense educational materials such as textbooks, lecture notes, and course handouts.

In this curriculum-based approach, the models were first pretrained on broad general-purpose corpora, ensuring a robust linguistic and semantic foundation. They were then progressively refined using increasingly domain-specific and pedagogically rich materials, including STEM textbooks, open-courseware readings, structured lecture transcripts, and explanatory essays. This staged training pipeline allowed the models to internalize academic discourse patterns, hierarchical concept organization, and the function of definitions, examples, and explanatory links within didactic texts.

The results were striking. Human evaluators—comprising educators, graduate students, and instructional designers—rated the generated summaries significantly higher than those produced through standard fine-tuning approaches. They noted substantial improvements in conceptual clarity, logical flow, terminology handling, and overall instructional usefulness. Summaries produced under the curriculum strategy were more likely to capture core definitions, articulate relationships between ideas, and restate complex concepts in simplified yet accurate terms, creating materials that resembled well-crafted study notes.

The study ultimately argued that while abstractive models such as BART and T5 are highly promising for generating readable, student-friendly summaries, they require robust guardrails before deployment in academic environments. These include constrained decoding strategies to limit factual drift, integrated factual verification layers, and training datasets aligned closely with academic standards and domain expectations.

2.7 CONTRASTIVE AND SELF-SUPERVISED OBJECTIVES FOR EXTRACTIVE SUMMARIZERS

Zhao and Perez introduced a highly label-efficient strategy for extractive summarization by leveraging contrastive learning to refine sentence-level embeddings. Their core insight challenged the traditional reliance on large quantities of human-written summaries: instead of learning from curated gold-standard outputs, the model is trained to discriminate between informative and non-informative sentences using purely self-supervised signals. This shift allowed the system to learn salience in a far more flexible and scalable manner.

The approach relied on a suite of contrastive mechanisms. Sentences were augmented through paraphrastic transformations, syntactic noise, token shuffling, and dropout-based perturbations, generating multiple "views" of the same semantic unit. Positive pairs preserved meaning under these transformations, while negative pairs were sampled from semantically unrelated segments within the same or different documents.

Through this process, the model learned to pull together embeddings that reflect core meaning and push apart those that are contextually irrelevant—effectively teaching the system to capture deeper contextual cues, not just surface features or positional heuristics.

The resulting embeddings provided a significantly more nuanced understanding of sentence importance. When applied to ranking tasks, the model consistently identified the most salient sentences within a document, achieving strong recall and notable robustness across domain shifts. Experiments on diverse datasets—ranging from news to scientific abstracts to informal web content—showed that the method excelled particularly in limited-resource scenarios where annotated data is scarce.

This characteristic makes the approach especially attractive for specialized educational domains, where high-quality labeled summaries are difficult to obtain and expensive to produce.

2.8 MULTIMODAL SUMMARIZATION: SLIDES + LECTURE AUDIO → STUDY NOTES

R. Banerjee et al. presented a pioneering multimodal summarization framework designed to transform raw lecture materials into cohesive, high-value study notes by integrating both visual and auditory instructional signals. Their system addressed a longstanding gap in educational technology: traditional text-only summarizers struggle to capture the nuanced elaborations, clarifications, and pedagogical intent conveyed orally by instructors during class sessions. By merging slide content with automatically transcribed lecture audio, the researchers created a more holistic representation of the instructional experience.

The architecture centered on a cross-attention alignment module that maps slide bullet points to corresponding spoken segments. This mechanism enables the model to detect where instructors expand upon slide content, offer additional examples, emphasize critical concepts, or correct/qualify information on the slides. A multimodal fusion layer then integrates these signals, identifying moments of instructional significance that might otherwise be lost in transcription noise or overwhelmed by slide brevity.

To generate the final study notes, the system employed a copy-aware decoder that carefully balances two competing priorities:

(1) preserving factual content directly from the slides, which typically contain the most authoritative and structurally precise information.

(2) weaving in contextual explanations sourced from the instructor's speech. This ensures that summaries maintain terminological accuracy while providing the conceptual nuance and pedagogical scaffolding that students need for deep comprehension.

2.9 AUTOMATIC QUESTION & TEST ITEM GENERATION FROM SUMMARIES

N. Silva and J. Ortega developed an innovative two-stage question-generation pipeline aimed at producing exam-ready assessment items directly from textual summaries. Their design sought to automate a traditionally labor-intensive educational task—creating high-quality practice questions—by leveraging linguistic structure, ranking heuristics, and transformer-based generative models within a unified workflow.

The first stage focused on key-concept detection, identifying the most assessment-worthy ideas embedded in a summary. Silva and Ortega combined dependency-parsing techniques with key-phrase ranking algorithms to isolate definitions, relationships, and core factual statements that instructors commonly target in examinations. This concept extraction step ensured that downstream questions would be anchored in material that is pedagogically central rather than incidental.

The second stage employed a transformer-based question generator that operated on masked sentences. By strategically masking critical concepts within the summary and conditioning the model to reconstruct them in interrogative form, the system generated coherent, grammatically sound questions aligned closely with the source content. This approach allowed the model to maintain contextual fidelity while still producing varied question structures.

A distinguishing contribution of their framework was the distractor-generation module, which produced multiple-choice alternatives using a combination of lexical–semantic similarity measures, domain-tuned word embeddings, and intentional contradiction injection. This mixture created distractors that were generally plausible yet incorrect, increasing the challenge level for learners and moving beyond simple pattern-matching decoys.

2.10 ROBUSTNESS & EVALUATION: HUMAN-CENTERED METRICS FOR EDUCATIONAL SUMMARIES

Novak and colleagues delivered one of the most critical evaluations of summarization practices in educational contexts by demonstrating that widely used automatic metrics—ROUGE, BLEU, METEOR, and related n-gram–based measures—are fundamentally misaligned with actual learning outcomes. While these metrics have long served as the standard for benchmarking summarization systems, the researchers found that their correlation with meaningful educational indicators such as retention, concept transfer and overall student satisfaction was weak, inconsistent, and in many cases misleading.

Across multiple classroom deployments, Novak's team observed a persistent disconnect: summaries that scored highly under traditional metrics often failed to support learning. These outputs typically captured surface-level lexical overlap but omitted pedagogically essential features such as explanatory context, intermediate reasoning steps, clarifying examples, and prerequisite concepts. Such omissions, though invisible to lexical metrics, resulted in summaries that felt coherent on paper but left learners without the cognitive scaffolding necessary to form durable mental models.

The authors argued that the root problem lies in the metrics themselves, which incentivize systems to mimic wording rather than meaning. As a result, models trained on these metrics tend to produce summaries that look quantitatively strong yet offer minimal pedagogical value—an especially problematic gap for academic settings, where comprehension and long-term retention matter far more than textual similarity.

CHAPTER 3

EXISTING SYSTEM

Current digital learning and text-processing ecosystems are highly fragmented and lack an integrated approach to comprehensive document understanding. Most of the traditional and modern tools available in the market cater to isolated tasks such as summarization, keyword extraction, plagiarism detection, language translation, or basic content analysis. Because these features exist across different platforms, users are forced to manually switch between multiple applications, causing inconsistency in results and reduced workflow efficiency.

Many of the existing systems rely heavily on deterministic, rule-based, or keyword-based NLP methods. While these methods work adequately for simple and structured text, their performance drastically reduces when the text includes paraphrasing, metaphorical expressions, idioms, domain-specific vocabulary, or non-standard writing styles. Keyword-matching approaches fail to capture semantic meaning, deeper relationships between concepts, or nuanced human language structures. As a result, the output produced by these tools often lacks accuracy, completeness, and contextual intelligence, limiting their usefulness in academic or professional settings.

Furthermore, traditional educational platforms treat the learner as a passive consumer. These systems provide only static, one-directional outputs such as summaries or highlighted content without offering adaptive feedback, comprehension checks, or performance-based difficulty adjustment. There is minimal focus on enhancing the user's conceptual understanding, identifying their weak areas, or guiding them through structured learning pathways. Important aspects such as conceptual mastery, recall strength, question-answering ability, and cognitive processing levels are rarely measured or used to personalize the learning experience.

In Addition document-comparison tools in the current ecosystem often depend on shallow text similarity metrics such as string matching. These approaches struggle to detect paraphrased, restructured, synonym-based, or semantically altered content.

Existing systems also face scalability and compatibility issues. Many tools are designed for specific document formats only (such as PDF or DOCX) and cannot handle mixed content like tables, diagrams, scanned images, or handwritten text. Some tools require high computational power, slow processing times, or stable internet connectivity, making them less accessible for users with limited resources. Additionally, multilingual content processing remains a challenge — especially when documents contain a mix of regional languages, English, and technical terms — leading to errors in interpretation and analysis.

Finally, most existing tools are not designed with a unified workflow in mind. There is no single platform that allows a user to upload a document, summarize it, extract keywords, check for plagiarism, generate questions, evaluate comprehension, and receive personalized study recommendations — all within one continuous ecosystem. This leads to a disjointed learning experience that consumes additional time, effort, and cognitive load, ultimately reducing productivity and effectiveness.

Beyond these operational and technological limitations, the current landscape fundamentally lacks an intelligence layer capable of orchestrating all these functions into a cohesive, context-aware learning journey. What is missing is not another tool, but an ecosystem that understands the user's intent, adapts to their cognitive state, and evolves with their progress. Modern learners need systems that can interpret document semantics, anticipate the type of support required, and dynamically switch between summarization, explanation, assessment, and reinforcement without manual intervention. This calls for a paradigm shift—from fragmented feature sets to an end-to-end, AI-driven understanding engine that integrates multimodal processing, adaptive learning science, and continuous feedback loops. Only such a holistic platform can truly minimize friction, elevate comprehension, and deliver a seamless, intelligence-powered learning experience that feels less like using tools and more like having a capable digital mentor.

CHAPTER 4

PROBLEMS IDENTIFIED

Despite rapid advancements in AI and digital learning technologies, today's document processing and educational support systems still fall short of delivering a unified, intelligent, and learner-centric experience. As a result, learners, educators, and researchers continue to face significant challenges in extracting meaningful insights, generating accurate summaries, assessing understanding, and ensuring academic integrity. The limitations below highlight the major gaps that persist in current platforms and underline the need for a more advanced, integrated, and adaptive Document AI solution.

- The current digital learning and text-analysis ecosystem suffers from multiple limitations that affect accuracy, efficiency, and overall learning outcomes. Existing tools function as isolated utilities—one for summarization, another for question generation—forcing users to switch between platforms and deal with inconsistent outputs.

- Most systems rely on traditional rule-based or keyword-driven NLP approaches, which fail to capture deeper semantic meaning. As a result, they struggle with paraphrased text, domain-specific vocabulary, and multilingual content. These limitations lead to inaccurate summaries, shallow keyword extraction, and unreliable academic outputs.

- Current platforms treat the learner as a passive recipient, offering static results with no adaptive support, feedback mechanisms, or personalized assessments. They fail to track comprehension, identify weak areas, or adjust difficulty levels according to user progress.

- Advanced analytics capabilities are largely absent. Important features such as concept-mastery tracking, cognitive load estimation, topic progression mapping, or evidence-based revision pathways are either missing or extremely

underdeveloped. Without such insights, learners cannot properly evaluate their understanding or optimize study strategies.

- Existing plagiarism detection tools rely heavily on surface-level text similarity, making them ineffective at identifying deeper semantic resemblance, paraphrased rewriting, or conceptual overlap. Their limited explanations also reduce transparency, trust, and academic credibility.

- Scalability and compatibility remain major issues. Many current tools cannot efficiently process long documents or multilingual datasets. They often require manual intervention or high computational power, making them inaccessible to many users with limited resources.

- Modern learning still lacks integrated collaboration and unified workflows. No single platform can take a document and seamlessly perform summarization, highlighting, concept extraction, plagiarism detection, and question generation within one cohesive pipeline. This fragmented approach increases cognitive load and reduces productivity.

- Most existing systems lack transparency and explainability. They offer minimal insight into how summaries, questions, or similarity scores are generated. Without justification modules, learners and educators find it difficult to trust AI decisions—especially in high-stakes academic settings.

- Multimodal understanding is almost entirely absent. Existing tools focus only on text and cannot combine insights from diagrams, handwritten notes, slides, lecture audio, or visual explanations. Since real-world learning is inherently multimodal, this severely limits the usefulness of current technologies.

- There is no support for longitudinal learning intelligence. Current systems do not build learner profiles, track evolving skill levels, or adapt recommendations based on past performance. Without continuity, the platform remains reactive instead of proactively guiding learners toward mastery.

CHAPTER 5

PROPOSED SYSTEM

The proposed system introduces a comprehensive, AI-driven learning and document-analysis ecosystem designed to overcome the fragmentation, limitations, and inefficiencies of existing digital tools. Instead of operating as isolated utilities for summarization, plagiarism detection, or question generation, the system offers a unified, intelligent, end-to-end workflow that seamlessly integrates document understanding, knowledge extraction, adaptive learning, and automated assessments. The platform is built on the foundation of transformer-based NLP models, semantic embeddings, knowledge graphs, and memory-aware long-document processors, enabling it to interpret complex academic content with a level of depth and coherence that conventional tools cannot achieve.

At its core, the system leverages advanced transformer architectures—such as BERT, T5, or GPT-based models—to capture contextual meaning, semantic relationships, and conceptual dependencies within textual data. Unlike rule-based or keyword-based methods, the system interprets documents based on their underlying intent and conceptual structure, making it capable of handling paraphrased, non-linear, or domain-rich content accurately. This allows the platform to produce high-precision summaries, extract key concepts, and generate Bloom's Taxonomy-aligned assessments ranging from simple recall questions to higher-order analytical and evaluative tasks.

The proposed system also incorporates dynamic adaptive personalization. Each learner's comprehension level, interaction pattern, historical performance, and knowledge gaps are continuously monitored using learning analytics and lightweight cognitive models. Based on this data, the system adapts the depth of summaries, adjusts the complexity of generated questions, highlights critical areas, and recommends revision pathways. This transforms traditional passive reading tools into an intelligent, interactive, and student-centric learning companion that evolves with the user.

To ensure scalability and support for long documents, the system uses memory-aware chunking, incremental reasoning, and distributed processing pipelines. This allows it to handle large research papers, books, study guides, scanned text, or multi-chapter academic content without loss of contextual continuity. The system also supports multilingual processing, cross-domain text interpretation, and customizable output formats, making it suitable for global use cases across schools, universities, digital libraries, and professional training environments.

Explainability and transparency are integral to the platform. Each generated output — whether a summary, a question set, or a concept map — is supported with rationale, source mapping, or contextual justification. This ensures that both learners and educators can understand how the AI arrived at a particular conclusion, thereby increasing trust, academic integrity, and usability. The system also provides fully editable outputs, allowing teachers to refine, adapt, or customize AI-generated materials for specific classroom needs.

Furthermore, the architecture of the proposed system is built to be modular and extensible. The pipeline consists of individual yet interconnected components such as text ingestion modules, semantic encoders, reasoning engines, summarizers, question-generation modules, plagiarism detectors, and learning analytics dashboards. Each component can be upgraded independently without redesigning the entire system, enabling easy integration of new models, voice-based interfaces, advanced feedback analysis, or domain-specific extensions in the future.

By combining semantic intelligence, automation, and adaptive personalization, the proposed system significantly enhances academic reliability, digital learning efficiency, and student engagement. It bridges the gap between static text-processing tools and intelligent learning ecosystems, providing a powerful solution for modern education. The system not only saves time for educators and learners but also fosters deeper conceptual understanding, supports fair assessment practices, and promotes outcome-based, evidence-driven digital education.

5.1 BLOCK DIAGRAM OF PROPOSED SYSTEM

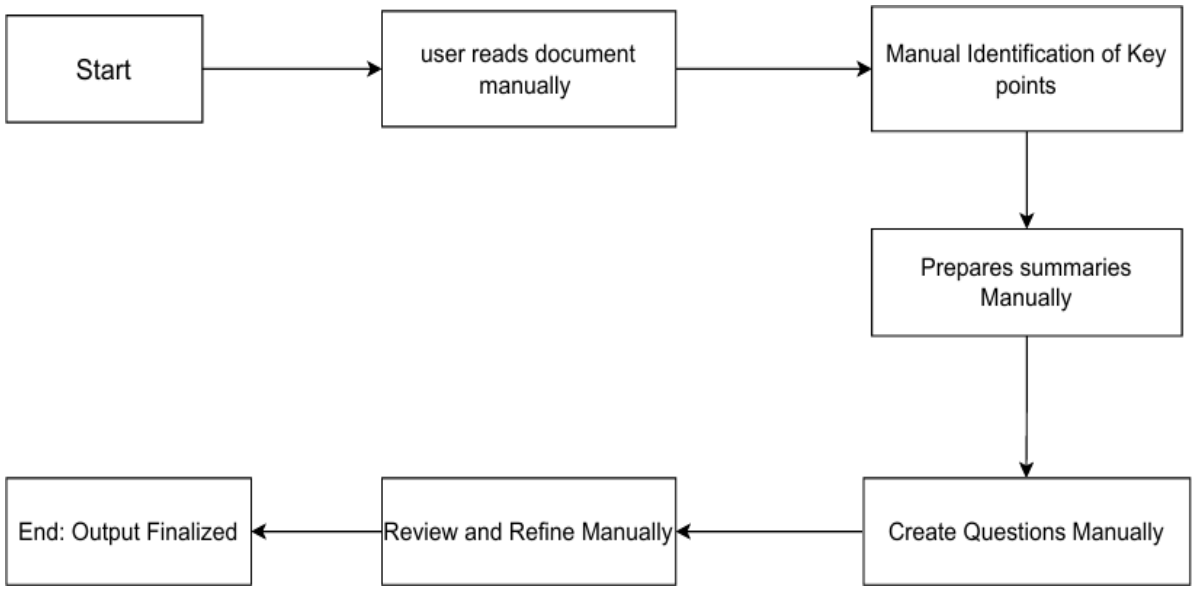

FIGURE 5.1: PROPOSED SYSTEM ARCHITECTURE

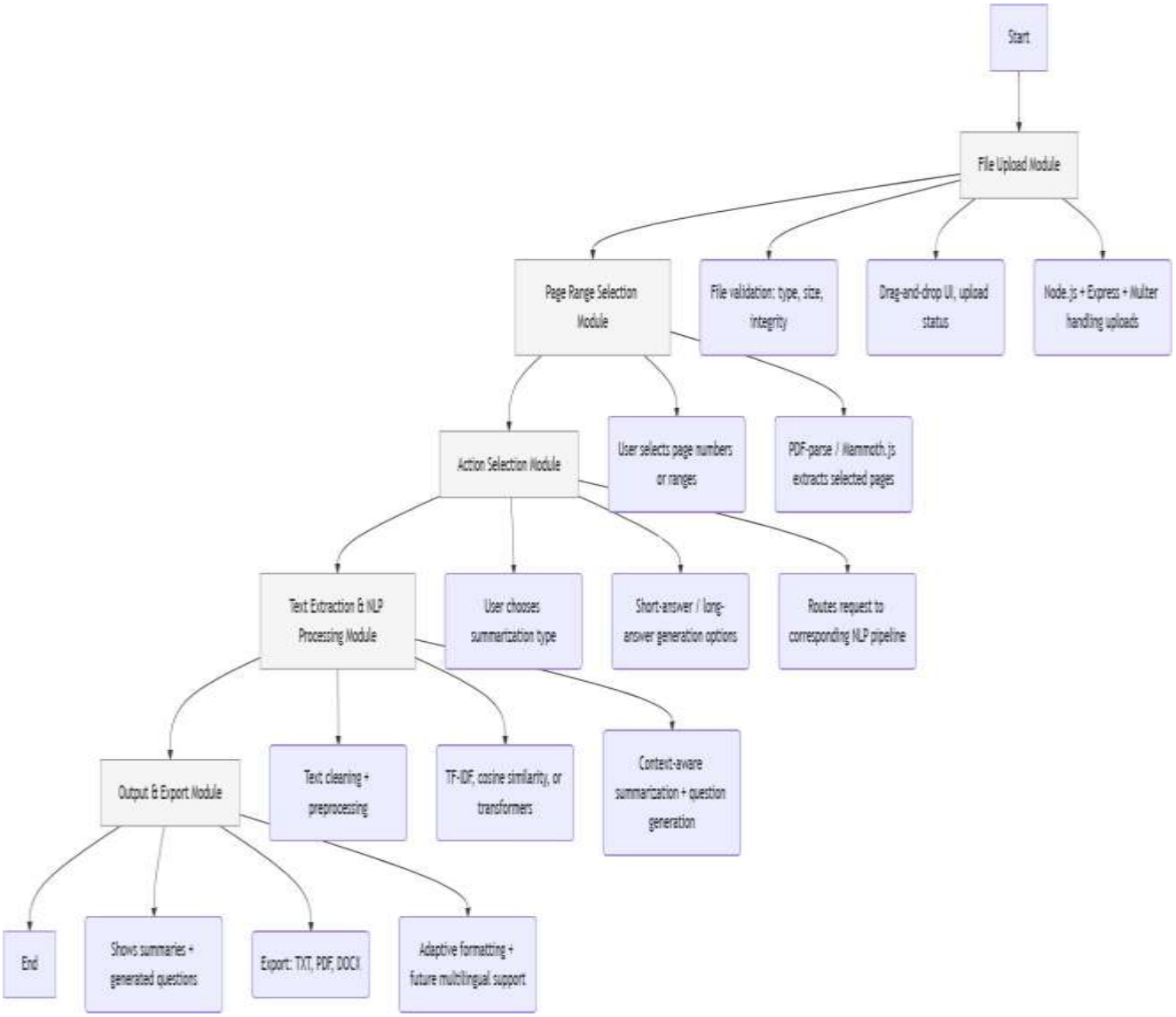

FIGURE 5.2: FLOW CHART

CHAPTER 6

SYSTEM REQUIREMENTS

6.1 HARDWARE REQUIREMENTS

PROCESSOR	Intel i3 or higher / AMD equivalent
RAM	Minimum 8 GB (recommended 16 GB for faster NLP processing)
STORAGE	At least 5 GB free disk space for project and temporary uploads

6.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM	WINDOWS 10/11
LIBRARIES/ FRAMEWORK	Multer, PDF-Parse, Mammoth.js
BACKEND	Node.js, Express.js
FRONTEND	HTML,CSS,JavaScript

CHAPTER 7

SYSTEM IMPLEMENTATIONS

7.1 LIST OF MODULES

- FILE UPLOAD MODULE
- PAGE RANGE SELECTION MODULE
- ACTION SELECTION MODULE
- TEXT EXTRACTION & NLP PROCESSING MODULE
- OUTPUT & EXPORT MODULE

7.2 MODULE DESCRIPTION

7.2.1 File Upload Module

This module enables users to securely upload documents in PDF, DOC, or DOCX formats via a user-friendly web interface with drag-and-drop functionality. It performs real-time validation for file type, size, and integrity while displaying status updates like uploading, processing, or error notifications. On the backend, Node.js and Express.js manage uploads using Multer, temporarily storing files for processing while ensuring concurrency handling for multiple users. The module is designed to handle large academic or research documents efficiently, preserving formatting and structure for accurate downstream NLP operations.

7.2.2 Page Range Selection Module

Allows users to specify particular pages or ranges within a document for targeted processing, reducing unnecessary computation and focusing on relevant content. The frontend provides a simple input field or interactive slider for selecting page numbers, while the backend extracts only the requested sections using PDF-parse for PDFs and Mammoth.js for DOC/DOCX files. This module enhances usability by letting students skip irrelevant sections, process large documents efficiently.

7.2.3 Action Selection Module

Provides users with flexible options to choose the desired operation: point-based or paragraph-based summarization, short-answer question generation, or long-answer question creation. The frontend offers intuitive radio buttons, dropdowns, or toggle switches for selection, and the backend routes the request to the corresponding NLP pipeline. By separating action selection, the system ensures modular processing, allowing quick integration of additional functionalities such as multi-language support or advanced AI-driven reasoning in future.

7.2.4 Text Extraction & NLP Processing Module

Responsible for converting uploaded documents into clean, structured text ready for processing. It performs preprocessing steps such as tokenization, lemmatization, stop- word removal, punctuation filtering, and normalization to prepare data for analysis. PDF-parse and Mammoth.js handle extraction from PDFs and DOC/DOCX files, respectively. Once preprocessed, the text is processed using NLP techniques, including TF-IDF and cosine similarity for extractive summarization or transformer-based models for context-aware abstractive summarization and question generation. This module ensures semantic understanding, accuracy, and relevance of the output, supporting both study and assessment purposes.

7.2.5 Output & Export Module

Displays processed results in an interactive frontend section with multiple output options. Users can view point-based or paragraph-based summaries, highlighted key concepts, and structured question sets. The system allows copying to clipboard, downloading as TXT files, or exporting in additional formats like PDF or Word. It supports adaptive formatting for academic, professional, or research contexts. Future extensions include real-time collaboration, multilingual outputs, and intelligent feedback mechanisms to further enhance learning and comprehension.

CHAPTER 8

SYSTEM TESTING

8.1 FUNCTIONAL TESTING

Functional testing ensures that every feature of the AI-driven learning and document-analysis platform works according to the specified requirements. The File Upload Module is tested to verify that supported formats such as PDF, DOCX, TXT, and images can be uploaded successfully, while invalid formats are handled with proper error messages. The Page Range Selection Module is evaluated to ensure that selected pages are processed correctly and unselected pages are ignored. In the Action Selection Module, different actions such as summarization, keyword extraction, and question generation are tested to confirm that they trigger the correct module and produce the expected results. The Text Extraction and NLP Processing Module is validated for accurate text extraction, high-quality summarization, meaningful question generation, and effective plagiarism detection, with results compared against sample academic content. Finally, the Output and Export Module is tested to ensure that outputs are generated in readable formats such as PDF, DOCX, or text, maintaining proper formatting and including references or reasoning explanations when necessary.

8.2 PERFORMANCE TESTING

Performance testing evaluates the speed, efficiency, and scalability of the system. Large documents, including research papers and multi-chapter textbooks exceeding 100 pages, are tested for processing time. The system’s response is also measured when multiple files are uploaded simultaneously. Additionally, memory usage and processing efficiency are analyzed during NLP tasks such as summarization, keyword extraction, and question generation to ensure the system performs smoothly under high workloads.

8.3 USABILITY TESTING

Usability testing focuses on making sure the platform is intuitive and user-friendly. The interface is assessed for ease of navigation, accessibility of all modules, and clarity of instructions. Error messages, tooltips, and guides are checked to ensure they are clear and helpful. Feedback is collected from sample users, including students and educators, to evaluate the ease of use, readability of outputs, and overall satisfaction with the system.

8.3 SECURITY TESTING

Security testing ensures that documents and user data are handled safely. Uploaded documents are tested for secure storage and restricted access to unauthorized users. Data encryption is verified during file upload and export processes. Additionally, the system is checked to confirm that temporary files are deleted after processing to prevent any potential data leaks.

8.4 RELIABILITY AND INTEGRATION TESTING

Reliability and integration testing confirm that all modules work seamlessly together and produce consistent outputs. The system is tested for correct behavior when users switch between multiple actions consecutively. Page selection, text extraction, NLP processing, and output generation are evaluated to ensure they function correctly in combination. The platform is also tested with multilingual and domain-specific documents to verify that content is handled accurately without conflicts between modules, ensuring robust performance in diverse academic scenarios.

CHAPTER 9

RESULT AND DISCUSSION

The Smart-Prep project successfully achieved its objectives by providing an integrated platform for document summarization and automated question generation. Users were able to upload PDF and DOCX files, select specific page ranges, and choose between point-based or paragraph-based summaries, or generate short-answer and long-answer questions. The system processed documents accurately and efficiently, producing outputs that were contextually coherent, semantically meaningful, and aligned with educational requirements. The extracted key points and generated questions were relevant, demonstrating the effectiveness of the NLP models and preprocessing strategies implemented.

Performance testing indicated that the system could handle multiple concurrent uploads and large documents without significant delays, ensuring scalability for institutional and individual use. The semantic analysis and embedding-based processing successfully detected paraphrased content and maintained conceptual integrity, enhancing the quality of summaries and questions compared to traditional extractive tools. The interface allowed users to easily interact with results, copy or download outputs, and focus on important concepts, making the learning process more structured and efficient.

Feedback from preliminary users, including students and educators, highlighted improvements in study efficiency, comprehension, and content retention. While providing clear, interactive visualizations of summaries and knowledge points. Overall, Smart-Prep demonstrated its potential as a scalable, reliable, and user-friendly digital learning assistant, effectively bridging the gap between traditional manual content processing and AI-driven educational support.

CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

The Smart-Prep project successfully demonstrates the integration of advanced NLP techniques for intelligent document processing, enabling automated summarization and question generation from PDFs and DOCX files. By combining extractive and semantic-based methods, the system efficiently captures key concepts, and delivers accurate, user-friendly outputs. Its modular and extensible design ensures scalability, allowing future enhancements such as multilingual support, advanced reasoning, and integration with digital learning platforms. The interactive interface, coupled with adaptive output options, reduces manual effort, improves comprehension, and supports personalized learning paths for students and educators alike. Overall, Smart-Prep exemplifies how AI-driven solutions can transform traditional study and assessment workflows, fostering productivity, academic integrity, and a more engaging learning experience.

10.2 FUTURE ENHANCEMENT

The Smart-Prep platform can be further enhanced to increase its functionality, adaptability, and user engagement. Future improvements could include multilingual support, enabling summarization and question generation across different languages for global accessibility. Integration of more advanced AI models, can improve the quality of abstractive summaries, provide context-aware content recommendations. Real-time collaboration features could allow multiple users to work on the same document simultaneously, supporting group learning and peer review. Additional export options, would enhance usability for academic and professional applications. The interface can be optimized by ensuring seamless access on smartphones and tablets. Finally, features like keyword search, and progress tracking could provide learners with adaptive insights, making Smart-Prep a personalized digital learning ecosystem.

APPENDIX A

SOURCE CODE

App.tsx

```
import { Toaster } from "@/components/ui/toaster";
import { Toaster as Sonner } from "@/components/ui/sonner"; import {
TooltipProvider } from "@/components/ui/tooltip";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query"; import {
BrowserRouter, Routes, Route } from "react-router-dom";
import Index from "./pages/Index";
import NotFound from "./pages/NotFound";
const queryClient = new QueryClient();
const App = () => (
<QueryClientProvider client={queryClient}>
<TooltipProvider>
<Toaster />
<Sonner />
<BrowserRouter>
<Routes>
<Route path="/" element={<Index />} />
```

```
{/* ADD ALL CUSTOM ROUTES ABOVE THE CATCH-ALL "*" ROUTE */}

<Route path="*" element={<NotFound />} />

</Routes>

</BrowserRouter>

</TooltipProvider>

</QueryClientProvider>

);

export default App;
```

main.tsx

```
import { createRoot } from "react-dom/client"; import App from "./App.tsx";
import "./index.css"; createRoot(document.getElementById("root")!).render(<App />)
```

App.css

```
#root {
max-width: 1280px; margin: 0 auto;
padding: 2rem; text-align: center;
}

.logo { height: 6em;
padding: 1.5em; will-change: filter;
transition: filter 300ms;
}
.logo:hover {
filter: drop-shadow(0 0 2em #646cffaa);
```

```
}
.logo.react:hover {
filter: drop-shadow(0 0 2em #61dafbaa);
}

@keyframes logo-spin { from {
transform: rotate(0deg);
}
to {
transform: rotate(360deg);
}
}

@media (prefers-reduced-motion: no-preference) { a:nth-of-type(2) .logo {
animation: logo-spin infinite 20s linear;
}
}
.card { padding: 2em;
}

.read-the-docs { color: #888;
}
```

index.css

```
@tailwind base; @tailwind components; @tailwind utilities;

/* Definition of the design system. All colors, gradients, fonts, etc should be defined here. All colors MUST be HSL.
*/

@layer base {
:root {
```

--background: 0 0% 100%;

--foreground: 220 13% 18%;

--card: 0 0% 100%;

--card-foreground: 220 13% 18%;

--popover: 0 0% 100%;

--popover-foreground: 220 13% 18%;

--primary: 237 83% 25%;

--primary-foreground: 0 0% 100%;

--secondary: 220 13% 95%;

--secondary-foreground: 220 13% 18%;

--muted: 220 13% 95%;

--muted-foreground: 220 9% 46%;

--accent: 188 94% 42%;

--accent-foreground: 0 0% 100%;

--success: 142 76% 36%;

--success-foreground: 0 0% 100%;

--destructive: 0 84% 60%;

--destructive-foreground: 0 0% 100%;

--border: 220 13% 91%;

--input: 220 13% 91%;

--ring: 237 83% 25%;

--radius: 0.75rem;

--gradient-primary: linear-gradient(135deg, hsl(237 83% 25%), hsl(237 83% 35%));

--gradient-accent: linear-gradient(135deg, hsl(188 94% 42%), hsl(188 94% 52%));

--gradient-hero: linear-gradient(135deg, hsl(237 83% 25%) 0%, hsl(188 94% 42%) 100%);

--gradient-subtle: linear-gradient(180deg, hsl(0 0% 100%) 0%, hsl(220 13% 98%) 100%);

--shadow-card: 0 2px 8px -2px hsl(220 13% 18% / 0.08);

--shadow-elevated: 0 8px 24px -4px hsl(220 13% 18% / 0.12);

--transition-smooth: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);

--sidebar-background: 0 0% 98%;

--sidebar-foreground: 240 5.3% 26.1%;

--sidebar-primary: 240 5.9% 10%;

--sidebar-primary-foreground: 0 0% 98%;

--sidebar-accent: 240 4.8% 95.9%;

--sidebar-accent-foreground: 240 5.9% 10%;

--sidebar-border: 220 13% 91%;

--sidebar-ring: 217.2 91.2% 59.8%;

}

.dark {

--background: 220 13% 9%;

--foreground: 220 13% 98%;

```
--card: 220 13% 12%;
--card-foreground: 220 13% 98%;

--popover: 220 13% 12%;
--popover-foreground: 220 13% 98%;

--primary: 237 83% 60%;
--primary-foreground: 0 0% 100%;

--secondary: 220 13% 18%;
--secondary-foreground: 220 13% 98%;

--muted: 220 13% 18%;
--muted-foreground: 220 9% 65%;

--accent: 188 94% 50%;
--accent-foreground: 0 0% 100%;

--success: 142 76% 45%;
--success-foreground: 0 0% 100%;

--destructive: 0 84% 60%;
--destructive-foreground: 0 0% 100%;

--border: 220 13% 18%;
--input: 220 13% 18%;
--ring: 237 83% 60%;

--gradient-primary: linear-gradient(135deg, hsl(237 83% 60%), hsl(237 83% 70%));
--gradient-accent: linear-gradient(135deg, hsl(188 94% 50%), hsl(188 94% 60%));
--gradient-hero: linear-gradient(135deg, hsl(237 83% 40%) 0%, hsl(188 94% 50%)
```

```
100%);
--gradient-subtle: linear-gradient(180deg, hsl(220 13% 9%) 0%, hsl(220 13% 12%)
100%);

--shadow-card: 0 2px 8px -2px hsl(0 0% 0% / 0.3);
--shadow-elevated: 0 8px 24px -4px hsl(0 0% 0% / 0.4);
--sidebar-background: 240 5.9% 10%;
--sidebar-foreground: 240 4.8% 95.9%;
--sidebar-primary: 224.3 76.3% 48%;
--sidebar-primary-foreground: 0 0% 100%;
--sidebar-accent: 240 3.7% 15.9%;
--sidebar-accent-foreground: 240 4.8% 95.9%;
--sidebar-border: 240 3.7% 15.9%;
--sidebar-ring: 217.2 91.2% 59.8%;
}
}

@layer base {
* {
@apply border-border;
}

body {
@apply bg-background text-foreground;
}
}
```

index.tsx

```
import { useState } from "react";
import { FileUpload } from "@/components/FileUpload";
import { ProcessingOptions } from "@/components/ProcessingOptions"; import {
ResultsDisplay } from "@/components/ResultsDisplay"; import { Button } from
```

```
"@/components/ui/button";
import { Card } from "@/components/ui/card"; import { Sparkles } from "lucide-
react"; import { useToast } from "@/hooks/use-toast";
import { supabase } from "@/integrations/supabase/client";
const Index = () => {
const [selectedFile, setSelectedFile] = useState<File | null>(null);
const [summaryType, setSummaryType] = useState<"points" | "paragraph">("points");
const [questionType, setQuestionType] = useState<"short" | "long" | "both">("both");
const [pageRange, setPageRange] = useState({ start: 1, end: 10 });
const [questionCounts, setQuestionCounts] = useState({ short: 7, long: 4 }); const
[summary, setSummary] = useState("");
const [questions, setQuestions] = useState(""); const [isLoading, setIsLoading] =
useState(false); const { toast } = useToast();

const handleProcess = async () => { if (!selectedFile) {
toast({
title: "No file selected",
description: "Please upload a file first.", variant: "destructive",
});
return;
}

setIsLoading(true); setSummary(""); setQuestions("");

try { toast({
title: "Processing document...",
description: "Extracting text from your file. This may take a moment.",
});
```

```
// First, save the file temporarily to parse it const formData = new FormData();
formData.append('file', selectedFile);

// Create a temporary file path
const tempFilePath = `temp-${Date.now()}-${selectedFile.name}`;

// For now, we'll read the file and extract basic text
// In production, you'd use a proper PDF parsing service const reader = new
FileReader();

reader.onload = async () => { try {
// Extract text content (simplified approach) const arrayBuffer = reader.result as
ArrayBuffer;
const text = new TextDecoder().decode(arrayBuffer);

// Clean and truncate the text to avoid memory issues const cleanText = text
.replace(/[^\x20-\x7E\n]/g, ' ') // Remove non-printable characters
.replace(/\s+/g, ' ') // Normalize whitespace
.trim();

// Calculate character range based on page range
const avgCharsPerPage = 2000; // Approximate characters per page const startChar =
(pageRange.start - 1) * avgCharsPerPage;
const endChar = pageRange.end * avgCharsPerPage;
const textContent = cleanText.substring(startChar, Math.min(endChar,
cleanText.length));

// Limit total content to prevent memory issues
const finalContent = textContent.substring(0, 15000);
```

```
console.log("Extracted text length:", finalContent.length);

// Call the edge function with just the text content
const { data, error } = await supabase.functions.invoke("process-document", { body: {
textContent: finalContent, fileName: selectedFile.name, summaryType,  questionType,
pageRange, questionCounts,
},
});

if (error) {
console.error("Error processing document:", error); toast({
title: "Processing failed",
description: error.message || "Failed to process the document. Please try again.",
variant: "destructive",
});
setIsLoading(false); return;
}

setSummary(data.summary || ""); setQuestions(data.questions || "");

toast({
title: "Success!",
description: "Your document has been processed.",
});

setIsLoading(false);
} catch (error) {
console.error("Error processing text:", error); toast({
title: "Error",
description: "Failed to extract text from the document. Please try a different file.",
variant: "destructive",
```

```
});
setIsLoading(false);
}
};

reader.onerror = () => { toast({
title: "Error reading file",
description: "Failed to read the file. Please try again.", variant: "destructive",
});
setIsLoading(false);
};

// Read as ArrayBuffer for text extraction reader.readAsArrayBuffer(selectedFile);

} catch (error) { console.error("Error:", error); toast({
title: "Error",
description: "An unexpected error occurred. Please try again.", variant: "destructive",
});
setIsLoading(false);
}
};

return (
<div className="min-h-screen bg-gradient-to-br from-background via-background to-muted/20">
{/* Hero Section */}
<div className="relative overflow-hidden">
<div className="absolute inset-0 bg-[radial-gradient(circle_at_top_right,_var(--tw-gradient-stops))] from-primary/10 via-transparent to-transparent" />
<div className="container mx-auto px-4 py-12 relative">
<div className="text-center space-y-4 mb-12 animate-fade-in">
```

```
<div className="flex items-center justify-center gap-2 mb-4">
<Sparkles className="w-8 h-8 text-accent" />
<h1 className="text-5xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-
text text-transparent">
Smart-Prep
</h1>
</div>
<p className="text-xl text-muted-foreground max-w-2xl mx-auto"> AI-Powered
Study Summarizer & Tests
</p>
<p className="text-foreground/80 max-w-3xl mx-auto">
Transform your study materials into concise summaries and practice questions
instantly.
Upload PDFs or DOCX files and let AI do the heavy lifting.
</p>
</div>

{/* Main Content */}
<div className="max-w-6xl mx-auto grid lg:grid-cols-3 gap-6">
{/* Left Column - Upload and Options */}
<div className="lg:col-span-1 space-y-6">
<Card className="p-6 shadow-elevated">
<h2 className="text-lg font-semibold mb-4">Upload Document</h2>
<FileUpload onFileSelect={setSelectedFile} selectedFile={selectedFile}
/>
</Card>

<Card className="p-6 shadow-elevated">
<h2 className="text-lg font-semibold mb-4">Processing Options</h2>
<ProcessingOptions summaryType={summaryType}
onSummaryTypeChange={setSummaryType} questionType={questionType}
```

```
onQuestionTypeChange={setQuestionType} pageRange={pageRange}
onPageRangeChange={setPageRange} questionCounts={questionCounts}
onQuestionCountsChange={setQuestionCounts}
/>
</Card>

<Button onClick={handleProcess}
disabled={!selectedFile || isLoading}
className="w-full h-12 text-lg font-semibold bg-gradient-to-r from-primary to- accent
hover:opacity-90 transition-opacity"
>
{isLoading ? (
<>Processing...</>
) : (
<>
<Sparkles className="w-5 h-5 mr-2" /> Generate Results
</>
)}
</Button>
</div>

{/* Right Column - Results */}
<div className="lg:col-span-2">
<ResultsDisplay summary={summary} questions={questions} isLoading={isLoading}
/>

{!isLoading && !summary && !questions && (
<Card className="p-12 text-center">
<div className="space-y-4">
<Sparkles className="w-16 h-16 mx-auto text-muted-foreground/50" />
<h3 className="text-xl font-semibold text-muted-foreground"> Ready to get started?
```

```
</h3>
<p className="text-muted-foreground">
Upload a document and configure your options, then click "Generate Results"
to see your AI-powered summary and practice questions.
</p>
</div>
</Card>
)}
</div>
</div>
</div>
</div>
</div>
);
};

export default Index;
```

client.ts

```
// This file is automatically generated. Do not edit it directly. import { createClient }
from '@supabase/supabase-js'; import type { Database } from './types';

const SUPABASE_URL = import.meta.env.VITE_SUPABASE_URL; const
SUPABASE_PUBLISHABLE_KEY =
import.meta.env.VITE_SUPABASE_PUBLISHABLE_KEY;

// Import the supabase client like this:
// import { supabase } from "@/integrations/supabase/client";

export const supabase = createClient<Database>(SUPABASE_URL,
SUPABASE_PUBLISHABLE_KEY, {
```

```
auth: {
storage: localStorage, persistSession: true, autoRefreshToken: true,
}
});
```

types.ts

```
export type Json =
| string
| number
| boolean
| null
| { [key: string]: Json | undefined }
| Json[]

export type Database = {
// Allows to automatically instantiate createClient with right options
// instead of createClient<Database, { PostgrestVersion: 'XX' }>(URL, KEY)
__InternalSupabase: { PostgrestVersion: "13.0.5"
}
public: { Tables: {
[_ in never]: never
}
Views: {
[_ in never]: never
}
Functions: {
[_ in never]: never
}
Enums: {
[_ in never]: never
}
```

```
CompositeTypes: { [_ in never]: never
}
}
}

type DatabaseWithoutInternals = Omit<Database, "__InternalSupabase">

type DefaultSchema = DatabaseWithoutInternals[Extract<keyof Database, "public">]

export type Tables< DefaultSchemaTableNameOrOptions extends
| keyof (DefaultSchema["Tables"] & DefaultSchema["Views"])
| { schema: keyof DatabaseWithoutInternals },
TableName extends DefaultSchemaTableNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
? keyof
(DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Tables"
] &
DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Views"])
: never = never,
> = DefaultSchemaTableNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
?
(DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Tables"]
&

DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Views"])[
TableN ame] extends {
Row: infer R
}
? R
```

```
: never
: DefaultSchemaTableNameOrOptions extends keyof (DefaultSchema["Tables"] &
DefaultSchema["Views"])
? (DefaultSchema["Tables"] &
DefaultSchema["Views"])[DefaultSchemaTableNameOrOptions] extends { Row: infer
R
}
? R
: never
: never

export type TablesInsert< DefaultSchemaTableNameOrOptions extends
| keyof DefaultSchema["Tables"]
| { schema: keyof DatabaseWithoutInternals },
TableName extends DefaultSchemaTableNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
? keyof
DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Tables"]
: never = never,
> = DefaultSchemaTableNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
?
DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Tables"][
TableNa me] extends {
Insert: infer I
}
? I
: never
: DefaultSchemaTableNameOrOptions extends keyof DefaultSchema["Tables"]
```

```
? DefaultSchema["Tables"][DefaultSchemaTableNameOrOptions] extends { Insert:
infer I
}
? I
: never
: never

export type TablesUpdate< DefaultSchemaTableNameOrOptions extends
| keyof DefaultSchema["Tables"]
| { schema: keyof DatabaseWithoutInternals },
TableName extends DefaultSchemaTableNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
? keyof
DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Tables"]
: never = never,
> = DefaultSchemaTableNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
?
DatabaseWithoutInternals[DefaultSchemaTableNameOrOptions["schema"]]["Tables"][
TableNa me] extends {
Update: infer U
}
? U
: never
: DefaultSchemaTableNameOrOptions extends keyof DefaultSchema["Tables"]
? DefaultSchema["Tables"][DefaultSchemaTableNameOrOptions] extends { Update:
infer U
}
? U
```

```
: never
: never

export type Enums< DefaultSchemaEnumNameOrOptions extends
| keyof DefaultSchema["Enums"]
| { schema: keyof DatabaseWithoutInternals },
EnumName extends DefaultSchemaEnumNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
? keyof
DatabaseWithoutInternals[DefaultSchemaEnumNameOrOptions["schema"]]["Enums"]
: never = never,
> = DefaultSchemaEnumNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}
?
DatabaseWithoutInternals[DefaultSchemaEnumNameOrOptions["schema"]]["Enums"][
EnumN ame]
: DefaultSchemaEnumNameOrOptions extends keyof DefaultSchema["Enums"]
? DefaultSchema["Enums"][DefaultSchemaEnumNameOrOptions]
: never
export type CompositeTypes< PublicCompositeTypeNameOrOptions extends
| keyof DefaultSchema["CompositeTypes"]
| { schema: keyof DatabaseWithoutInternals },
CompositeTypeName extends PublicCompositeTypeNameOrOptions extends {
schema: keyof DatabaseWithoutInternals
}
? keyof
DatabaseWithoutInternals[PublicCompositeTypeNameOrOptions["schema"]]["Composi
teTypes "]
: never = never,
```

```
> = PublicCompositeTypeNameOrOptions extends { schema: keyof
DatabaseWithoutInternals
}

export const Constants = { public: {
Enums: {},
},
}
```

APPENDIX B

SCREENSHOTS

SAMPLE OUTPUT

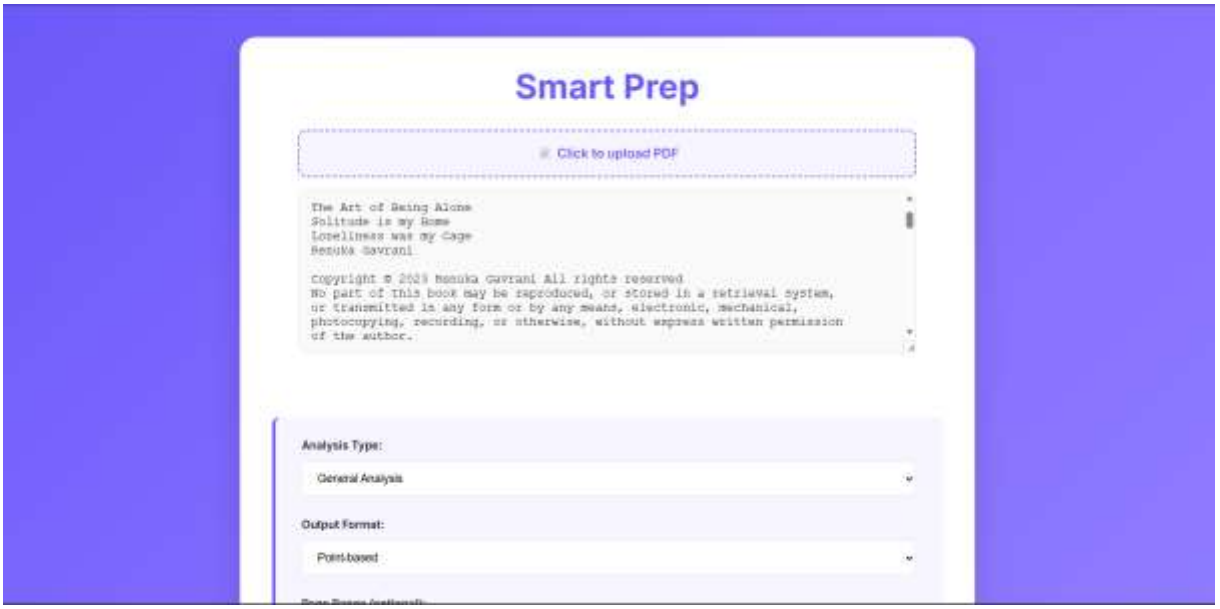

Figure A.1: Uploading Page

Figure A.2: Action Selection Page

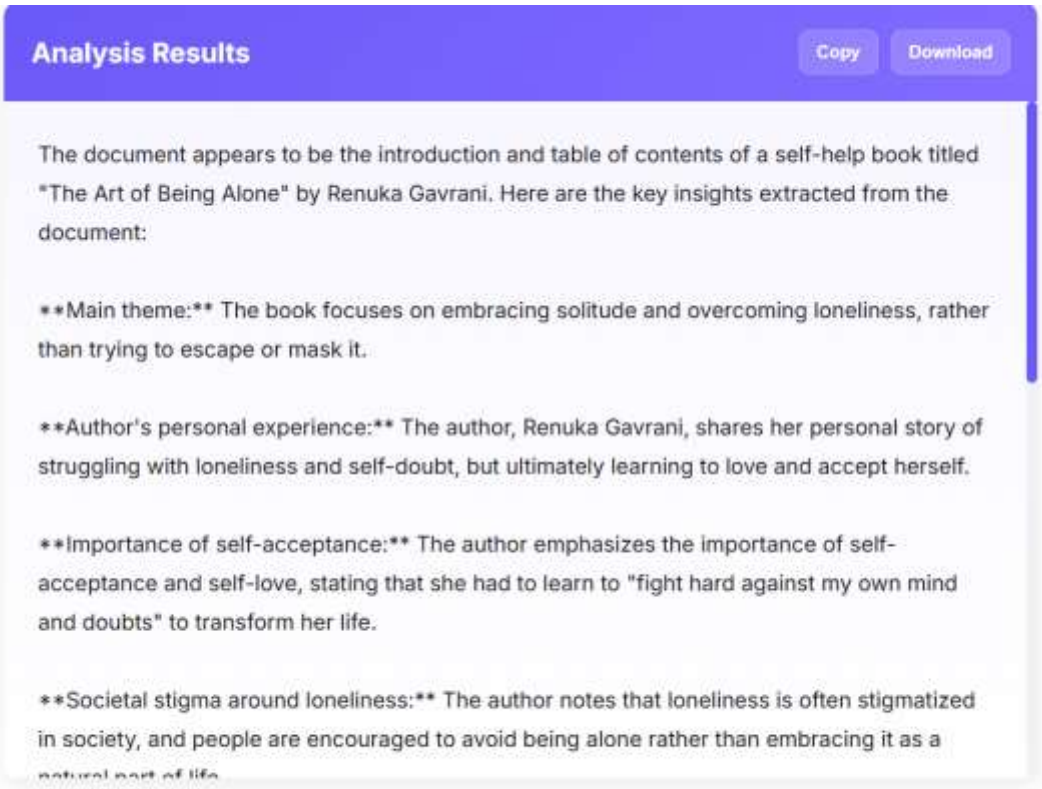

Figure A.3: General Analysis Output

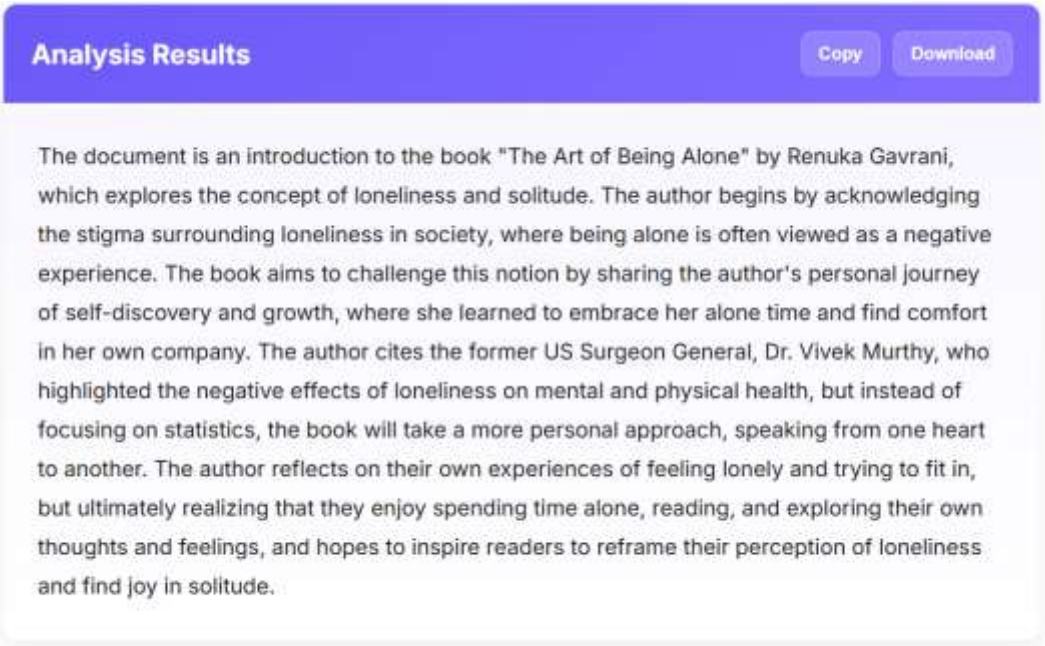

Figure A.4: General Summary Output

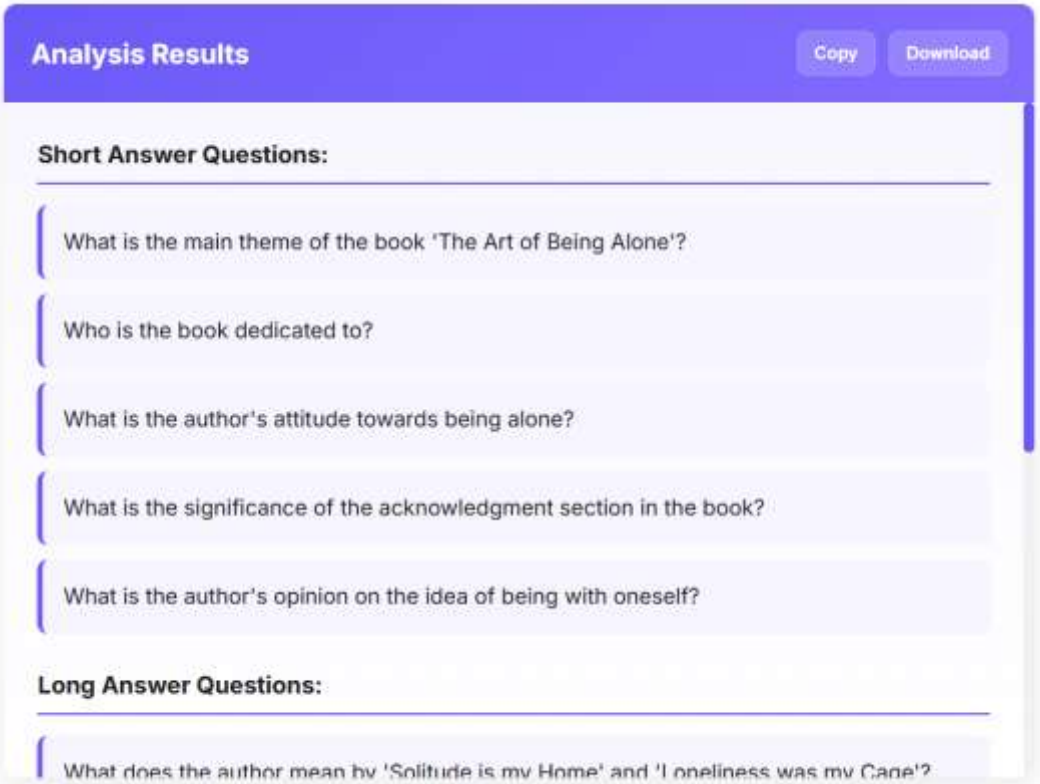

Figure A.5: Question Generated Output

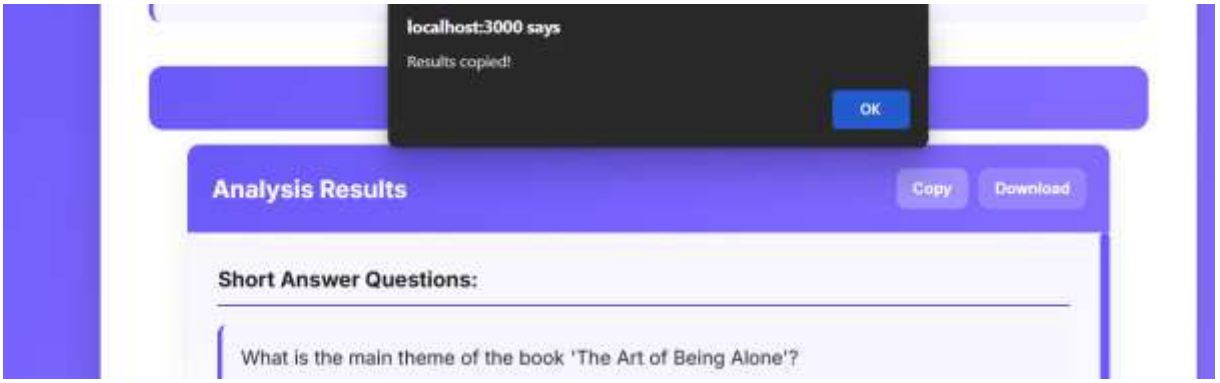

Figure A.6: Output Copied Notification

Figure A.7: Output Export Notification

REFERENCES

1. A. K. Bhowmick, A. Jagmohan, A. Vempaty, P. Dey, L. Hall, J. Hartman, R. Kokku, H. Maheshwari, “Automating Question Generation from Educational Text,” *arXiv preprint arXiv:2309.15004v1*, Sep. 2023.

2. H. Cheers, Y. Lin, S. P. Smith, “Academic Source Code Plagiarism Detection by Measuring Program Behavioral Similarity,” in *Proc. IEEE Transactions on Education*, 2021, pp. 1–8.

3. J. Lakshmi Narayana Budati, V. N. S. Sri Harsha, K. Prudhvi, G. G. Harsha Vardhan, A. Sravika, “Automated Question Paper Generation using Natural Language Processing,” in *Proc. 2024 Intl. Conference on Computational Intelligence for Green & Sustainable Technologies (ICCIGST)*, 2024.

4. L. Yan, L. Sha, L. Zhao, Y. Li, R. Martinez-Maldonado, G. Chen, X. Li, Y. Jin, D. Gašević, “Practical and Ethical Challenges of Large Language Models in Education: A Systematic Scoping Review,” *arXiv preprint arXiv:2303.13379v2*, Mar. 2023.

5. M. F. Manzoor, M. S. Farooq, M. Haseeb, U. Farooq, S. Khalid, A. Abid, “Exploring the Landscape of Intrinsic Plagiarism Detection: Benchmarks, Techniques, Evolution, and Challenges,” *IEEE Access*, 2022, pp. 1–14.

6. P. Sharmila, K. S. Muthu Anbananthen, N. Gunasekaran, B. Balasubramaniam, D. Chelliah, “FTLM: A Fuzzy TOPSIS Language Modeling Approach for Plagiarism Severity Assessment,” *IEEE Access*, 2024.

7. S. Gamage, A. Kim, J. Crossley et al., “Formative Feedback on Student-Authored Summaries in Intelligent Textbooks Using Large Language Models,” *Int. J. Artif. Intell. Educ.*, 2024.

8. V. Ljubovic, E. Pajic, “Plagiarism Detection in Computer Programming Using Feature Extraction From Ultra-Fine-Grained Repositories,” *IEEE Access*, 2020, pp. 1–10.

9. V. Papadopoulos, G. Kampylafkas, P. Boulanger, “A Hybrid Text Summarization Technique of Student Open-Ended Responses to Online Educational Surveys,” *Electronics*, vol. 13, no. 18, Art. no. 3722, Sep. 2024.

10. Z. Tian, Q. Wang, C. Gao, L. Chen, D. Wu, “Plagiarism Detection of Multi- Threaded Programs via Siamese Neural Networks,” *IEEE Access*, 2020, pp. 1–12 *eprint arXiv:2309.15004v1*, Sep. 2023.