WEBSUM - SUMMARY GENERATOR

Miniproject REPORT

In partial fulfilment of the requirements for the award of

Bachelor of Technology

In

Computer Science and Engineering (Artificial Intelligence and Machine Learning)

Of

A P J Abdul Kalam Technological University

April 2025

Submitted By

NEBU PLACID (CEK22AM022)

Under the Guidance of

Neethu Thomas
Assistant Professor

Department of Computer Science & Engineering
College of Engineering Kottarakkara
Kollam 691 531. Phone: 0474-2453300
`http://www.cekottarakkara.ac.in`
`cekottarakkara@ihrd.ac.in`

DECLARATION

I, **Nebu Placid (CEK22AM022)** declare the seminar of **WEBSUM - SUMMARY GENERATOR** is the result of original work done by me and to the best of my knowledge; a similar work has not been submitted to COLLEGE OF ENGINEERING KOTTARAKKARA, for fulfillment of the requirement of a course of study. This seminar report is submitted on partial fulfillment of the requirement for the B.Tech Computer Science and Engineering(Artificial Intelligence and Machine Learning).

Kottarakkara
09-04-2025 Nebu Placid (CEK22AM022)

College of Engineering Kottarakkara
Dept. of Computer Science & Engineering

Certificate

This is to certify that this report titled ***WEBSUM - SUMMARY GENERATOR*** is a bonafide record of the **CMD 334 MINIPROJECT** done by **NEBU PLACID (CEK22AM022)** Sixth Semester B. Tech Computer Science & Engineering(Artificial Intelligence & Machine Learning) students, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech.Computer Science and Engineering(Artificial Intelligence & Machine Learning) of **A P J Abdul Kalam Technological University uring the academic year 2024-2025.**

Prof. Neethu Thomas
(Project Guide)
Assistant Professor
Computer Science & Engineering
College Of Engineering Kottarakkara

Prof. Marina Glastin
(Project Coordinator)
Assistant Professor
Computer Science & Engineering
College Of Engineering Kottarakkara

Prof. Indu P K
Head of the Department
Computer Science & Engineering
College Of Engineering Kottarakkara

Acknowledgments

I take this opportunity to express my heartfelt gratitude to all those who have supported and guided me in the successful completion of this project.

I am deeply thankful to **Dr. Manoj Ray D**, Principal, College of Engineering Kottarakkara, for providing us with the necessary infrastructure and encouragement throughout the course of our work.

I extend my sincere thanks to **Prof. Marina Glastin**, Project Coordinator and Assistant Professor, Department of Computer Science and Engineering, and **Prof. Indu P. K**, Head of the Department, for their valuable support, insightful suggestions, and continuous encouragement.

I wish to place on record my deep sense of appreciation and gratitude to my project guide, **Prof. Neethu Thomas**, Assistant Professor, Department of Computer Science and Engineering, for her constant guidance, motivation, and mentorship, which were instrumental in the completion of this project.

Finally, I express my sincere thanks to my familiy, friends, and well-wishers for their unwavering support and encouragement throughout this journey.

Abstract

The exponential growth of online content has made information retrieval increasingly challenging, creating a strong demand for effective summarization tools. This project introduces a Web Summarizer powered by Meta's LLaMA (Large Language Model Meta AI), implemented locally using Ollama. It integrates LangChain and Streamlit to offer a seamless and interactive user experience. Unlike cloud-based models such as ChatGPT (GPT-4), Google Bard (Gemini), and Hugging Face Transformers, this solution runs entirely offline, ensuring improved data privacy, enhanced security, and zero API costs.

By leveraging state-of-the-art Natural Language Processing (NLP) techniques, the system delivers accurate and context-rich summaries from extensive online content. The combination of LangChain with the LLaMA model optimizes performance, while Streamlit enables real-time summarization through an intuitive graphical interface. This tool is particularly beneficial for professionals, researchers, and users who require concise overviews of long-form content such as articles, academic papers, and news reports.

Looking forward, the system is designed to evolve with features like multilingual summarization, real-time speech-to-text capabilities, and domain-specific fine-tuning to enhance performance in specialized fields such as medicine, finance, and law. This privacy-focused, cost-effective, and scalable approach to web summarization empowers users to extract insights efficiently while maintaining complete control over their data.

Contents

List of Figures

Chapter 1

Introduction

In the modern digital landscape, the overwhelming volume of online content makes it increasingly difficult to extract relevant information efficiently. Manually sifting through lengthy articles, research papers, and news reports is time-consuming, leading to a growing demand for automated web content summarization.

This project introduces a Web Summarizer that leverages Meta's Large Language Model Meta AI (LLaMA) for generating concise and informative summaries. It is implemented locally using Ollama and integrates LangChain and Streamlit to provide a seamless user experience. Unlike cloud-based summarization solutions such as OpenAI's ChatGPT (GPT-4), Google Bard (Gemini), and Hugging Face Transformers, which require an internet connection and rely on third-party APIs, this system runs entirely offline. As a result, it ensures enhanced data privacy, security, and cost efficiency.

The use of Ollama enables fast and efficient on-device execution of LLaMA, eliminating API costs and making LLM-driven summarization more accessible to a broader audience. LangChain facilitates smooth interaction between components, optimizing the processing pipeline for generating high-quality summaries. Meanwhile, Streamlit provides an intuitive graphical user interface (GUI), allowing users to input web content, receive structured summaries in real time, and adjust summarization parameters based on their needs.

Designed to serve a wide range of users, including researchers, journalists, business professionals, and everyday internet users, this system enables quick and accurate summarization of large text sources. By prioritizing offline execution, the solution is particularly useful for organizations handling sensitive data, ensuring that information remains within the user's device and complies with security regulations.

Furthermore, the Web Summarizer sets the foundation for future improvements, including multilingual summarization, real-time speech-to-text summarization for processing audio content, and domain-specific fine-tuning to enhance accuracy in specialized fields such as medicine, finance, and law. This project presents a scalable and privacy-conscious approach to automated summarization, addressing the limitations of cloud-based alternatives while providing a cost-effective and efficient way to process vast amounts of information.

1.1 Key Technologies

The Web Summarizer is built on a combination of three key technologies—**Ollama, LangChain, and Streamlit**. Each of these tools plays a crucial role in enabling efficient, user-friendly, and privacy-focused text summarization. Below is an in-depth exploration of these technologies and their specific contributions to the system.

1.1.1 Ollama – Running LLMs Locally with High Efficiency

Ollama is a lightweight and optimized framework designed for running **Large Language Models (LLMs) on local machines**, eliminating the need for cloud-based services. Traditional LLM-based applications often rely on cloud-based APIs (such as OpenAI's GPT-4 or Google Bard), which require internet connectivity, expose sensitive user data to third-party servers, and come with additional API costs. Ollama addresses these challenges by allowing models like **Meta's LLaMA (Large Language Model Meta AI)** to be executed entirely on local hardware, ensuring **data privacy, security, and cost efficiency**.

The core benefits of using Ollama in this Web Summarizer include:

- **Local Execution**: All processing happens on the user's machine, ensuring that sensitive information never leaves the device.
- **Performance Optimization**: Ollama is optimized for efficient inference, making it suitable even for personal computers and edge devices.
- **No API Costs**: Unlike cloud-based models that require users to pay for API access, Ollama allows unlimited local inference without recurring fees.
- **Customizability**: Users can fine-tune models for specific domains, improving summarization accuracy for specialized fields such as medicine, law, and finance.

Ollama significantly enhances the efficiency of the Web Summarizer by allowing **real-time text processing** without delays caused by network dependency. This makes the system particularly useful for **organizations with strict data security policies** and users who need a **cost-effective alternative to cloud services**.

1.1.2 LangChain – A Framework for Building LLM-Powered Applications

LangChain is a robust and highly modular framework specifically designed for building applications that leverage **Large Language Models (LLMs)**. It simplifies the process of integrating multiple AI components, allowing developers to create seamless workflows for complex natural language processing tasks. In the context of the Web Summarizer, LangChain plays a vital role in managing the flow of information between different components, ensuring that the summarization process is efficient and adaptable.

The major functionalities of LangChain in this project include:

- **Model Management**: LangChain efficiently connects the LLaMA model (running via Ollama) to the summarization workflow, ensuring smooth input-output processing.
- **Prompt Engineering**: Helps in structuring queries to optimize the quality of summaries, ensuring that responses are concise, informative, and contextually relevant.
- **Memory Mechanisms**: LangChain can maintain context across multiple interactions, making the summarizer capable of handling longer texts and multi-step queries effectively.
- **Workflow Automation**: By integrating various AI components, LangChain enables **automated text processing**, allowing the system to summarize large amounts of data quickly without requiring manual intervention.

Additionally, LangChain's flexibility allows future enhancements such as **multi-document summarization, document retrieval, and fine-tuned response generation**. This ensures that the Web Summarizer is not just a static tool but a **scalable and evolving application** that can adapt to diverse user needs.

1.1.3 Streamlit – A User-Friendly Interface for Real-Time Summarization

Streamlit is an open-source **Python-based web framework** designed for creating interactive and visually appealing web applications with minimal coding effort. It is particularly well-suited for AI-driven applications, as it allows developers to build intuitive user interfaces without requiring extensive knowledge of front-end development.

In the Web Summarizer, Streamlit serves as the **front-end interface**, allowing users to interact with the summarization system effortlessly. Its main advantages include:

- **Real-Time Interaction**: Users can enter text, URLs, or documents and receive instant summaries, making the application highly responsive.
- **Minimal Setup**: Unlike traditional web frameworks that require extensive HTML, CSS, and JavaScript coding, Streamlit allows **rapid development** using simple Python scripts.
- **Customization Options**: Users can tweak summarization parameters, such as summary length and abstraction level, to generate more tailored outputs.
- **Seamless Integration with LLMs**: Streamlit easily connects with Ollama and LangChain, providing a **smooth and interactive user experience**.

By incorporating Streamlit, the Web Summarizer ensures that **both technical and non-technical users can access advanced summarization features** without complex setup procedures. The interactive interface enhances usability, making AI-driven summarization accessible to a wide audience, from researchers and students to journalists and corporate professionals.

1.2 Motivation and impact

In an era where information is generated at an unprecedented rate, individuals and organizations face significant challenges in processing vast amounts of textual data efficiently. Whether in academic research, news media, or enterprise decision-making, the ability to quickly extract relevant insights from lengthy documents is crucial. Traditional manual methods of reviewing content are time-consuming and impractical, necessitating the development of automated solutions for text summarization. This project introduces a **Web Summarizer** utilizing **Meta's LLaMA, Ollama, LangChain, and Streamlit** to address these challenges. Unlike cloud-based solutions that rely on external APIs, this system operates **entirely offline**, ensuring data privacy, cost efficiency, and accessibility for a wider audience. The impact of this technology spans multiple domains, including academia, journalism, and business intelligence.

1.2.1 Research and Academia

Academics and researchers are often required to process large volumes of scholarly articles, research papers, and conference proceedings to stay updated with the latest advancements in their fields. Conducting literature reviews and synthesizing relevant information from various sources can be an overwhelming and time-intensive task. The traditional approach of reading full papers is inefficient, especially given the exponential increase in scientific publications.

The Web Summarizer enhances academic productivity by:

- **Accelerating Literature Reviews**: Researchers can quickly generate summaries of multiple papers, allowing them to focus on key findings and trends without reading each document in its entirety.
- **Improving Comprehension and Retention**: Concise summaries help students and scholars grasp essential concepts more efficiently, reducing cognitive load.
- **Enhancing Accessibility to Research**: By summarizing complex technical content into simplified, readable formats, the tool assists students, educators, and professionals in understanding dense academic material.
- **Ensuring Data Privacy**: Since all processing occurs locally, sensitive or unpublished research remains secure, addressing concerns over intellectual property and academic confidentiality.

By integrating this summarization tool into their workflow, researchers and students can optimize knowledge extraction, making the process of academic inquiry significantly more efficient.

1.2.2 News Aggregation

The rapid flow of information in today's digital world makes it increasingly difficult for individuals to stay informed about current events without investing substantial time in reading full-length news articles. Journalists, analysts, and media professionals must continuously monitor multiple news sources, synthesize information, and produce reports under tight deadlines. However, manually reviewing extensive news content is inefficient and often leads to information overload.

The Web Summarizer addresses these challenges by:

- **Providing Concise News Summaries**: Users can quickly obtain summarized versions of trending news articles, enabling them to stay informed without spending excessive time reading.
- **Filtering Relevant Information**: By automatically extracting key points, the system eliminates redundant details, presenting only the most critical aspects of a news story.
- **Enhancing Real-Time Decision Making**: Journalists and media analysts can quickly scan summaries to determine which stories require deeper investigation.
- **Protecting Sensitive Data**: Many investigative journalists handle confidential information. Since this tool runs locally, sensitive research remains private and secure.

By leveraging automated summarization, media professionals can streamline their information processing and focus on producing high-quality journalistic content.

1.2.3 Enterprise and Business Intelligence

In corporate environments, executives and professionals are required to analyze large volumes of business reports, market research, financial statements, and legal documents to make informed decisions. However, reading and extracting key insights from lengthy documents can be time-consuming and inefficient, particularly in fast-paced industries where rapid decision-making is crucial.

The Web Summarizer significantly improves business intelligence processes by:

- **Optimizing Decision-Making**: Executives can quickly generate summaries of business reports, allowing them to focus on essential insights without reading full documents.
- **Enhancing Productivity**: Employees can retrieve key points from meeting minutes, strategy reports, and corporate communications in a fraction of the time it would take to manually review them.
- **Facilitating Legal and Compliance Analysis**: Legal professionals can summarize lengthy contracts and regulatory documents, ensuring compliance without missing critical clauses.
- **Minimizing Costs**: By eliminating reliance on cloud-based summarization services, enterprises can reduce API costs and maintain greater control over proprietary information.

This tool enables businesses to enhance efficiency and maintain a competitive edge by streamlining the process of extracting actionable insights from complex corporate data.

1.2.4 Privacy-Preserving and Cost-Efficient Summarization

One of the most significant advantages of this Web Summarizer is its ability to function entirely offline, ensuring that user data never leaves the local system. Unlike cloud-based LLMs such as OpenAI's GPT-4, Google Bard, or Hugging Face Transformers, this system does not transmit sensitive information over the internet, making it particularly beneficial for industries and professionals who require strict data privacy.

Key benefits include:

- **No API Costs**: Many commercial LLM services charge fees per request, making large-scale summarization financially unsustainable for students, researchers, and businesses. By running locally, this tool eliminates those costs.
- **Enhanced Data Security**: Since all processing happens on the user's device, confidential information remains protected from potential data breaches or unauthorized access.
- **Independence from Internet Connectivity**: Unlike cloud-based solutions, this summarizer can function without requiring an internet connection, making it ideal for remote environments or users with limited access to online services.

By prioritizing privacy and cost efficiency, this system democratizes access to LLM-powered summarization, making it an accessible solution for a broader audience.

1.2.5 Future Prospects and Enhancements

As AI and natural language processing (NLP) technologies continue to evolve, the Web Summarizer has the potential for further enhancements, including:

- **Multilingual Support**: Expanding the model's capabilities to summarize content in multiple languages.
- **Speech-to-Text Summarization**: Enabling voice-based summarization for spoken content, making it useful for lectures, podcasts, and interviews.
- **Domain-Specific Optimization**: Fine-tuning the model for specialized fields such as healthcare, finance, and law to improve summarization accuracy.

- **Integration with Web Browsers**: Allowing real-time summarization of online articles and social media content directly within web browsers.

By incorporating these advancements, the Web Summarizer can further extend its utility, making information retrieval even more efficient and user-friendly.

1.3 Future Prospects

The development of the Web Summarizer represents a significant step toward automated, privacy-focused text summarization. However, the potential for further enhancements remains vast. As AI and Natural Language Processing (NLP) technologies continue to evolve, several improvements can be integrated into the system to expand its functionality, increase its accuracy, and cater to a broader range of users. The following are some key areas for future enhancements:

1.3.1 Multilingual Summarization

One of the most promising enhancements to the Web Summarizer is the integration of **multilingual support**. Currently, the summarizer primarily processes English-language text, limiting its usability for non-English speakers. Given the diverse linguistic landscape of the internet, enabling support for multiple languages would significantly broaden its accessibility.

Key benefits of multilingual summarization include:

- **Global Accessibility**: Users from different linguistic backgrounds will be able to generate summaries in their native languages, making the tool more inclusive.
- **Cross-Language Summarization**: The ability to summarize content written in one language and produce an output in another (e.g., summarizing a Spanish article in English) would be particularly useful for journalists, researchers, and international businesses.
- **Improved AI Model Adaptability**: Training the model on diverse language datasets would enhance its ability to handle complex sentence structures, idioms, and cultural nuances.
- **Support for Low-Resource Languages**: Many languages lack extensive AI models for NLP applications. Integrating multilingual support would help bridge this gap, providing summarization tools for underrepresented languages.

The implementation of multilingual capabilities could be achieved through fine-tuning LLaMA on a diverse dataset or integrating language translation models such as MarianMT or OpenNMT.

1.3.2 Real-Time Speech-to-Text Summarization

With the increasing popularity of podcasts, audiobooks, and recorded lectures, the ability to summarize **spoken content** in real-time is a valuable feature. By integrating speech recognition technology, the Web Summarizer can process audio input and generate concise text-based summaries.

Advantages of real-time speech-to-text summarization include:

- **Enhanced Productivity**: Users can quickly extract key insights from lengthy discussions, making note-taking during meetings or classes more efficient.
- **Accessibility for the Hearing-Impaired**: Individuals with hearing impairments can benefit from automatic summarization of spoken content, improving their access to information.
- **Content Indexing**: Journalists, researchers, and content creators can generate quick overviews of interviews, speeches, or panel discussions, allowing them to find relevant segments without manually reviewing entire recordings.
- **Real-Time Information Processing**: Users can listen to news broadcasts, business presentations, or educational webinars and receive instant summaries without having to pause the content.

To achieve this capability, the Web Summarizer can integrate Automatic Speech Recognition (ASR) models such as Whisper by OpenAI or DeepSpeech by Mozilla. These models can convert spoken words into text, which the summarizer can then process to generate a concise summary.

1.3.3 Domain-Specific Fine-Tuning

While general-purpose summarization is useful, different fields—such as **medicine, finance, and law**—require specialized handling of terminology and context. Domain-specific fine-tuning of the summarization model can significantly improve the accuracy and relevance of generated summaries.

1. Medical Summarization

- Doctors, medical researchers, and healthcare professionals often deal with complex clinical reports, research papers, and patient case studies.
- A medical fine-tuned summarizer could extract key findings from long documents, aiding in faster decision-making and research synthesis.

- Ensuring medical terminology is correctly interpreted is crucial for accurate summarization.

2. Financial Summarization

- Investors, analysts, and corporate executives need to process lengthy earnings reports, market research, and regulatory filings.
- A finance-specific summarizer could highlight crucial metrics, trends, and forecasts, streamlining financial decision-making.
- Summarizing financial news could provide real-time insights for stock traders and business strategists.

3. Legal Summarization

- Legal professionals frequently review lengthy contracts, case laws, and regulatory documents.
- A fine-tuned model for legal text could accurately summarize key clauses, obligations, and legal precedents, reducing the time required for legal analysis.
- This could benefit lawyers, compliance officers, and policymakers who need quick access to essential legal information.

To implement domain-specific fine-tuning, the summarizer can be trained on specialized corpora such as:

- Medical: PubMed, clinical trial reports, and healthcare journals.
- Finance: SEC filings, investor reports, and financial news archives.
- Law: Legal case databases, court rulings, and corporate law documents.

By tailoring the model to understand industry-specific jargon and context, the Web Summarizer can significantly enhance its effectiveness for professional use.

1.4 Problem Statement

To overcome the challenges posed by cloud-based summarization tools, this project introduces a **fully offline Web Summarizer**, leveraging **Meta's LLaMA (Large Language Model Meta AI)** for efficient and privacy-focused text summarization. Unlike traditional summarization models that depend on external APIs and cloud servers, this system runs locally on a user's device, eliminating concerns related to **data privacy, security, and cost inefficiencies**.

By utilizing **Ollama**, a lightweight framework specifically designed for running large language models locally, this solution ensures fast and effective text processing without the need for continuous internet connectivity. This approach significantly reduces the risk of data breaches, ensures full user control over sensitive information, and removes financial barriers associated with API-based services.

The integration of **LangChain** further enhances the summarization pipeline by introducing a structured and modular approach to **processing, managing, and retrieving text efficiently**. LangChain facilitates:

- **Memory mechanisms**
- **Retrieval-augmented generation (RAG)**
- **Prompt optimization**

These features make the summarization process more context-aware and adaptive to different content types. Unlike traditional extractive summarization methods, which merely pick out key sentences, LangChain enables **abstractive summarization**, where the system **generates human-like summaries that are concise yet contextually accurate**. This capability is particularly useful for users in specialized domains such as **medical research, legal analysis, and financial reporting**, where precise and contextually relevant summaries are crucial.

To ensure an interactive and user-friendly experience, the system utilizes **Streamlit**, a powerful Python-based web framework that allows users to generate summaries with just a few clicks. Streamlit provides an intuitive, lightweight web interface that requires **no prior programming knowledge**, making it accessible to a wide range of users, including:

- Researchers and students
- Business professionals
- Journalists

Users can simply input text, select summarization preferences, and receive **concise, accurate, and well-structured summaries** in real time. This eliminates the need for complex command-line operations or manual text processing, offering a **seamless and efficient summarization workflow**.

By combining these advanced technologies, the proposed Web Summarizer delivers a **secure, cost-effective, and accessible** solution to the problem of information overload. This system not only enhances productivity by allowing users to extract insights quickly but also ensures that **sensitive data remains secure** by running the entire process locally. Additionally, the **removal of API costs** makes LLM-powered summarization available to a broader audience, including:

- Individual users
- Educational institutions
- Small businesses

Future improvements may include:

- **Multilingual summarization**
- **Real-time speech-to-text capabilities**
- **Domain-specific fine-tuning**

These enhancements will further expand its practical applications across various industries.

Chapter 2

System Analysis

2.1 Current State of Web Summarization Tools

The current system for web summarization leverages state-of-the-art language models, particularly the **LLaMA** (Large Language Model Meta AI) architecture, to generate concise and contextually relevant summaries from online content. This system streamlines information consumption by extracting key points from lengthy web articles, blog posts, and other textual sources found on the internet.

Users interact with the system through a web-based interface where they can input a URL or paste text directly. The LLaMA-based model processes the input and generates a summary that retains the original context and meaning while significantly reducing the length. This feature makes the system highly useful for researchers, students, professionals, and general users who need quick insights without reading entire articles.

A notable feature of the current system is its ability to export the generated summaries as downloadable **PDF files**. This enhances usability by allowing users to save, share, and archive their summaries for offline access or future reference. Additional functionalities may include **language selection, tone adjustment** (e.g., formal, neutral, or casual), and **summarization length preferences**.

Overall, the current system represents a significant advancement in **natural language processing (NLP)** and AI-assisted content understanding, offering practical tools for efficient knowledge consumption in the digital age.

2.2 Limitations of Existing Systems

1. **Inaccurate and Generic Summaries**
   Most existing summarization tools rely on *extractive techniques*, which merely pick key sentences from the original text rather than generating a meaningful, condensed summary. This often results in disjointed or redundant summaries that fail to provide true contextual understanding.

2. **Limited Context Awareness**
   Traditional summarizers struggle to understand the broader context and intent of the content. As a result, they may miss critical nuances, especially in **technical, legal, or research-based articles**.

3. **Poor Handling of Complex and Long-Form Content**
   Many summarization systems fail to efficiently process **long articles, research papers, or multi-section documents**. They often truncate information improperly, leading to incomplete or misleading summaries.

4. **Lack of Customization and Personalization**
   Existing tools provide little to no control over **summary length, tone, or focus**. Users often receive fixed-length summaries without options for adjusting detail levels.

5. **No Multi-Language Support or Translation Issues**
   Many summarization tools support only a limited number of languages. Even when translation is available, the quality of summaries in **non-English languages** is often poor, with unnatural phrasing and loss of meaning.

6. **Inefficient Export and Storage Options**
   Most summarization tools lack support for structured document exports such as **PDF, Markdown, or HTML**. Even those that offer PDF exports often fail to preserve formatting, headings, and key structures.

7. **Dependence on Internet and Cloud Processing**
   Several existing systems require a continuous internet connection and rely on cloud-based processing, leading to **latency issues, slower response times**, and **privacy concerns** for users handling sensitive data.

8. **Inability to Process Multimedia Content**
   Current summarizers are text-focused and cannot process **images, charts, or embedded videos** within webpages. This limits their effectiveness for summarizing visually rich or interactive content.

9. **Ethical and Privacy Concerns**
   Some summarization tools store user data, raising concerns about **data security and unauthorized content usage**. This is particularly problematic when summarizing sensitive or private information.

10. **Scalability and Performance Limitations**
    As user demand increases, many systems experience **performance degradation and longer processing times**. This lack of scalability prevents them from handling high-traffic environments efficiently.

2.3 Comparison with Proposed System

Feature	**Existing Web Summarization Tools**	**Proposed LLaMA-Based Summarizer**
Summarization Approach	Mostly *extractive* (selects key sentences)	*Abstractive* (rewrites in a meaningful way)
Context Understanding	Limited, struggles with complex or long content	Strong context awareness, better summarization of technical and lengthy texts
Customization	Fixed-length summaries, no personalization options	Adjustable summary length, tone, and focus
Multi-Language Support	Supports few languages, lower quality in non-English texts	Supports multiple languages with high accuracy
Export Formats	Mostly plain text, limited or no PDF export	PDF, Markdown, HTML, and plain text export options
Processing Speed	May be slow due to cloud dependency	Optimized for fast summarization, even on large documents
Handling of Multimedia Content	Cannot summarize embedded *images, charts, or videos*	Future scope includes AI-driven extraction of key insights from multimedia
Privacy & Security	Some tools store user data, raising privacy concerns	Secure processing with no data retention
Scalability	May experience delays with heavy traffic	Efficient and scalable architecture for handling high user demand
Use Cases	General-purpose summarization, limited for specialized fields	Ideal for research, technical documents, news, business reports, and academic use

Table 2.1: Comparison of Existing Web Summarization Tools vs. Proposed LLaMA-Based Summarizer

2.4 Challenges and Considerations

1. **Computational Resource Requirements**
   Running large language models like LLaMA requires high computational power, including **GPUs or TPUs**, which can be expensive.

2. **Model Training and Fine-Tuning Complexity**
   Fine-tuning LLaMA for domain-specific summarization requires **large datasets and extensive computational resources**.

3. **Handling Long and Complex Documents**
   Summarizing long articles, research papers, or reports may exceed the model's token limit, leading to loss of context.

4. **Balancing Speed and Accuracy**
   Real-time summarization requires **optimization for speed** without compromising summary quality.

5. **Multi-Language Support Challenges**
   Some low-resource languages may have **lower summarization accuracy** due to limited training data.

6. **Privacy and Data Security Concerns**
   Ensuring **user data privacy** is crucial, especially when summarizing confidential information.

7. **Handling Bias in AI-Generated Summaries**
   AI models can inherit **biases from training data**, potentially leading to misleading summaries.

8. **Integration with External Systems**
   Developing seamless **browser extensions, APIs**, and cross-platform compatibility is a technical challenge.

Chapter 3

Literature Survey

Chapter 4

Methodology

This chapter outlines the methodology adopted for developing the web-based text summarization system using Langchain and Ollama LLM (Llama3 Instruct model). The implementation is structured into three main phases: Model Initialization, Input Processing, and System Deployment.

4.1 Model Initialization

The project utilizes the Langchain framework integrated with the Ollama LLM (Llama3 Instruct model) for performing text summarization tasks. The model is initialized using the `ChatOllama` class, where the local Ollama server is configured to run on `localhost:11434`.

The summarization logic is defined using a custom prompt template provided by `PromptTemplate` from Langchain. This template ensures that the model generates detailed, clear, and structured summaries strictly based on the input text, without relying on external data sources.

The model initialization is handled within the function `setup_summarization_chain()`, which defines summarization rules, constraints, and response formatting as shown in the source code.

4.2 Input Processing

The user provides a web URL via a Streamlit-based user interface. The system fetches the corresponding web content using the `requests` library, with a specified User-Agent header to bypass security restrictions.

The raw web content is cleaned using `BeautifulSoup`, removing all HTML tags to extract the readable text content. This functionality is encapsulated within the `load_document(url)` function.

After content extraction, the text is divided into smaller manageable chunks using the `chunk_text()` function. This is essential to manage the token limit constraints imposed by the LLM. Each text chunk is then processed sequentially through the LLM summarization chain, generating individual summaries for each segment, which are then combined to form the final output summary.

4.3 System Deployment

The complete system is deployed using the Streamlit framework, which provides an interactive and responsive web-based user interface.

Users can input the target URL, generate a detailed summary, and download the output as a PDF file using the `fpdf` library.

The main execution flow is handled in the `streamlit_mode()` function. The user interface consists of different sections such as:

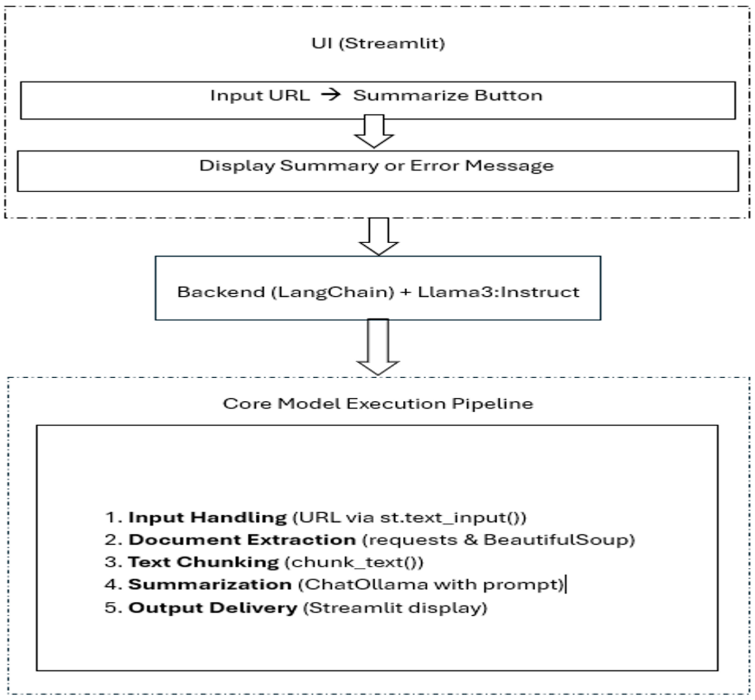
- URL input field for user interaction.
- Summarization output display area.
- Download button for exporting the generated summary as a PDF.

The system is tested and compatible with the Windows 11 environment, running through Windows PowerShell for execution. Furthermore, modularity is maintained throughout the project, with appropriate error handling and session management implemented using `st.session_state` to ensure a smooth user experience.

Chapter 5

System Architecture

5.1 WebSum Architecture



UI (Streamlit)
Input URL → Summarize Button
Display Summary or Error Message
Backend (LangChain) + Llama3:Instruct
Core Model Execution Pipeline
1. **Input Handling** (URL via st.text_input())
2. **Document Extraction** (requests & BeautifulSoup)
3. **Text Chunking** (chunk_text())
4. **Summarization** (ChatOllama with prompt)
5. **Output Delivery** (Streamlit display)

Figure 5.1: WebSum System Architecture

5.1.1 Core Components

- **User Interface (Streamlit):** Provides an interactive frontend where users can input URLs, view generated summaries, and download results as PDFs.
- **Web Content Fetcher (`requests.get`):** Retrieves raw HTML content from the input URL using custom headers to simulate real browser behavior, ensuring compatibility with different websites.
- **Content Extractor (BeautifulSoup):** Parses and cleans the fetched HTML to extract readable text, filtering out unwanted elements such as `<script>`, `<style>`, ads, and navigation elements.
- **Text Preprocessor (LangChain):** Processes large text bodies by splitting them into smaller, manageable chunks, ensuring efficient token usage when interacting with the LLM.
- **Prompt Engineering (`PromptTemplate` from LangChain):** Uses structured templates to provide clear, context-aware instructions to the LLaMA3 model, optimizing summarization quality.
- **Summarization Engine (LLaMA3 via Ollama):** Executes local summarization using Meta's LLaMA3 model within Ollama, ensuring high-quality, private, and offline text processing.
- **PDF Export Module (using `fpdf`):** Converts the generated summary into a well-formatted PDF document for offline storage or sharing. The module ensures high readability and structured formatting.

5.1.2 Key Features

- **Offline & Secure Processing** – The entire summarization pipeline runs locally, ensuring user privacy.
- **Modular System Design** – Each component operates independently, allowing easy updates or extensions.
- **LangChain Integration** – Efficient text chunking and prompt management enhance summarization accuracy.
- **User-Friendly Streamlit Interface** – Intuitive UI for URL input, summary visualization, and PDF export.
- **FPDF-Based PDF Export** – Lightweight PDF generation without relying on third-party services.

5.1.3 Workflow

1. **User Input:** The user enters a URL via the Streamlit interface.
2. **Fetching HTML Content:** The `requests` library downloads the webpage HTML with appropriate headers.
3. **Text Extraction:** BeautifulSoup extracts and cleans relevant text from the HTML document.
4. **Preprocessing:** LangChain processes the text into structured chunks for optimal model interaction.
5. **Prompting and Summarization:** The processed text is summarized using LLaMA3 via Ollama.
6. **Output Delivery:** The summary is displayed on Streamlit and available for download as a PDF.

5.1.4 Limitations

- **No JavaScript Rendering** – Dynamic content from JavaScript-heavy sites (React, Angular) may not be fully captured.
- **Dependent on Local Resources** – Performance varies based on system specifications.
- **Only Web-Based Input** – Currently does not support direct file uploads (e.g., PDF, DOCX).
- **Unable to Process Web Images** – Web images, which often form a major part of the information on a website, are completely ignored by the WEBSUM .

5.1.5 Technologies Used

Programming Language

- Python 3.13.2

Frontend

- Streamlit

Backend & Processing

- LangChain (prompt management and chunking)
- Ollama (LLM execution with LLaMA3)

Libraries used

- `requests` (Web content fetching)
- `BeautifulSoup` (HTML parsing)
- `fpdf` (PDF export)
- `streamlit` (User interface)

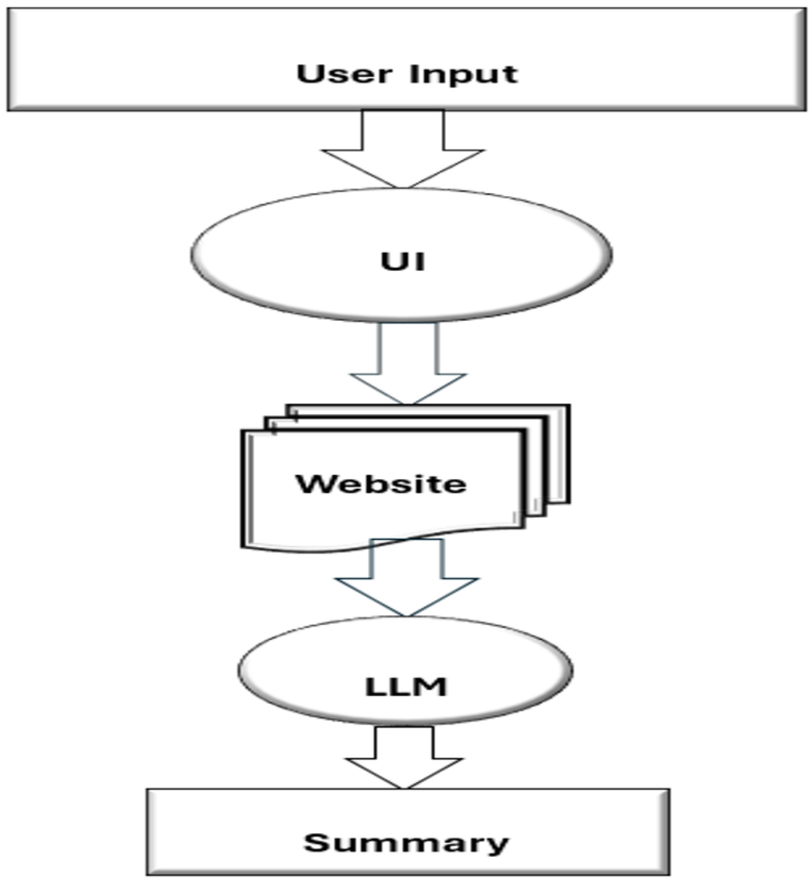


Figure 5.2: Data Flow Diagram of WebSum

Chapter 6

Implementation

6.1 Algorithm

The following steps outline the algorithm used in WebSum:

1. Initialize the Streamlit UI
2. Fetch and extract text from the URL using `requests.get()` and BeautifulSoup.
3. Process the extracted text using LangChain and LLaMA3.
4. Generate the summary.
5. Display the summary in the Streamlit interface.

6.2 Output

This section presents the output of our project, showcasing different stages of the summarization process.

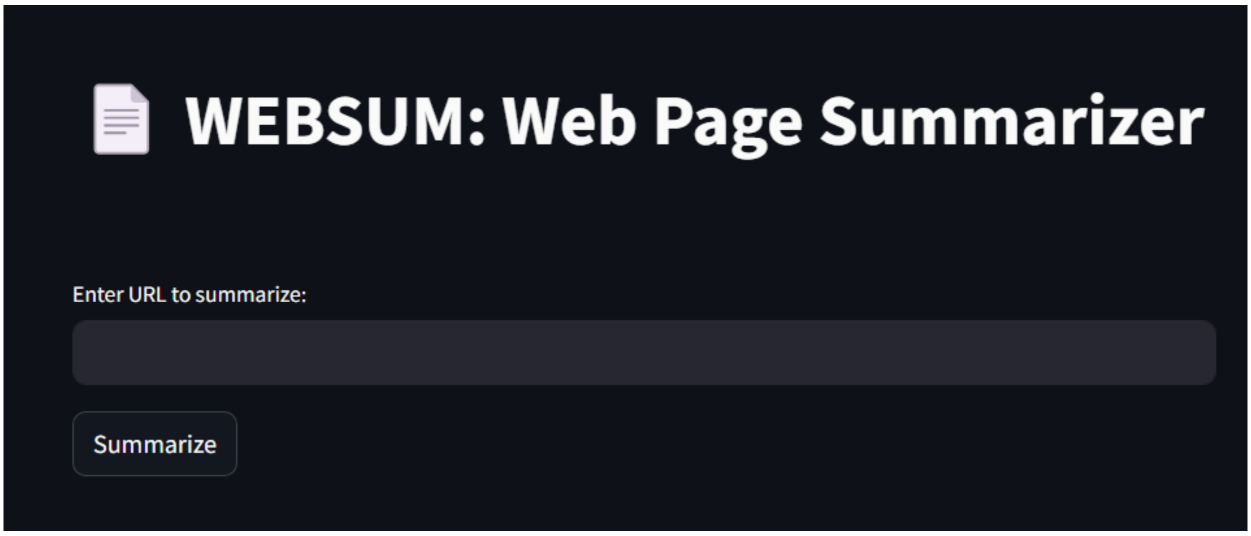
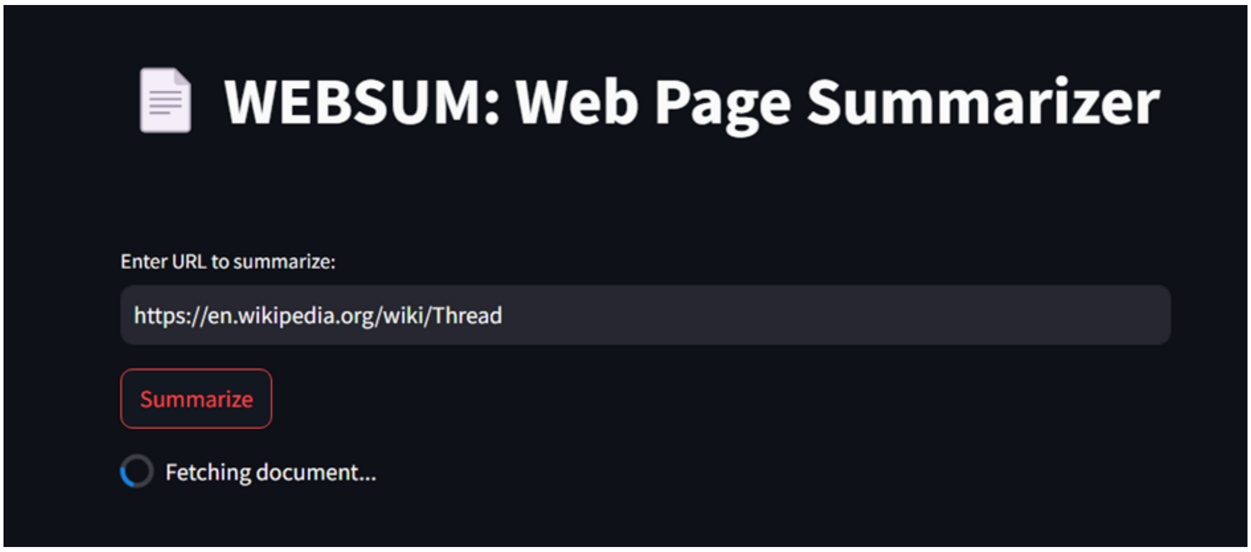


Figure 6.1: Output No: 1



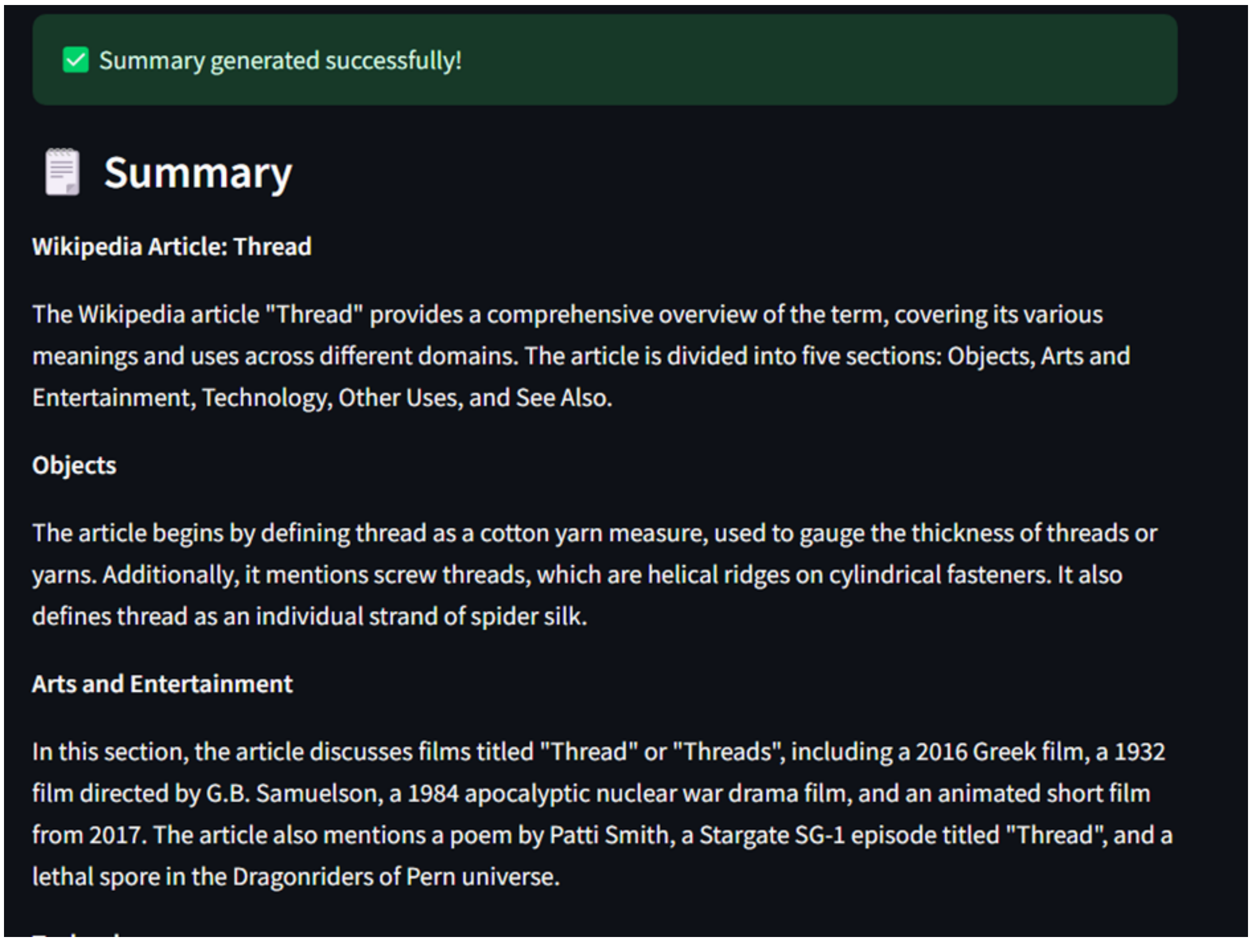
WEBSUM: Web Page Summarizer

Enter URL to summarize:

https://en.wikipedia.org/wiki/Thread

Fetching document...

Figure 6.2: Output No: 2



✅ Summary generated successfully!

Summary

Wikipedia Article: Thread

The Wikipedia article "Thread" provides a comprehensive overview of the term, covering its various meanings and uses across different domains. The article is divided into five sections: Objects, Arts and Entertainment, Technology, Other Uses, and See Also.

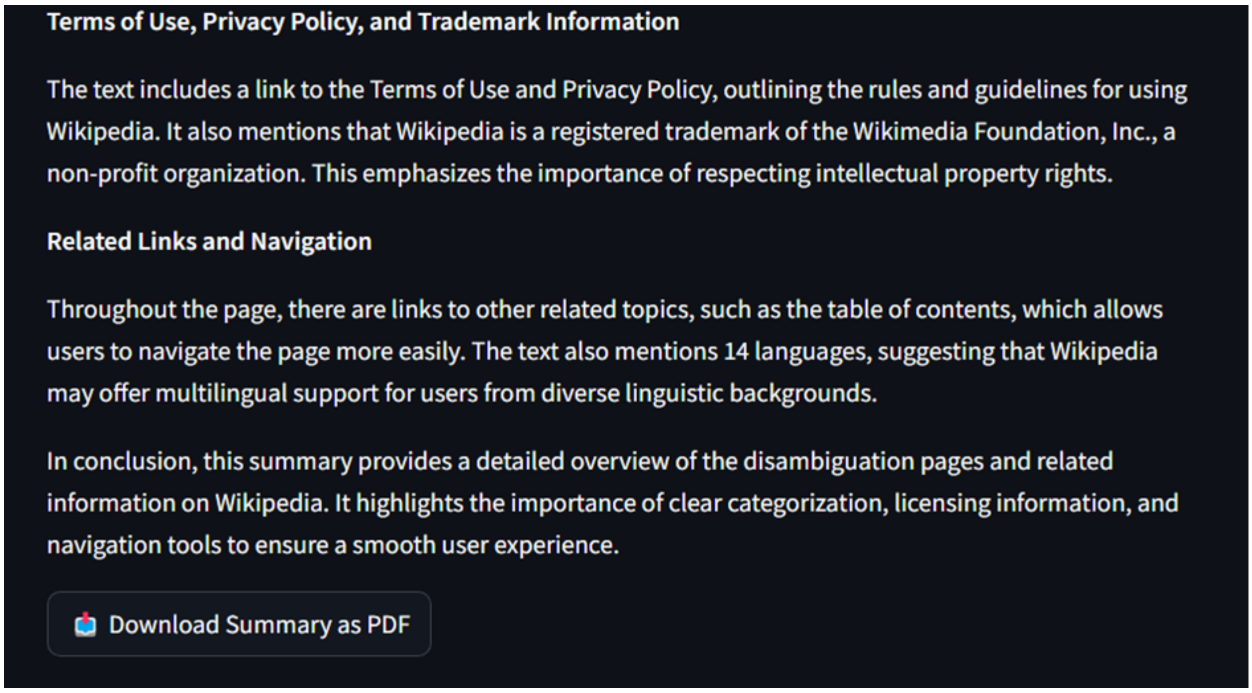
Objects

The article begins by defining thread as a cotton yarn measure, used to gauge the thickness of threads or yarns. Additionally, it mentions screw threads, which are helical ridges on cylindrical fasteners. It also defines thread as an individual strand of spider silk.

Arts and Entertainment

In this section, the article discusses films titled "Thread" or "Threads", including a 2016 Greek film, a 1932 film directed by G.B. Samuelson, a 1984 apocalyptic nuclear war drama film, and an animated short film from 2017. The article also mentions a poem by Patti Smith, a Stargate SG-1 episode titled "Thread", and a lethal spore in the Dragonriders of Pern universe.

Figure 6.3: Output No: 3


Terms of Use, Privacy Policy, and Trademark Information

The text includes a link to the Terms of Use and Privacy Policy, outlining the rules and guidelines for using Wikipedia. It also mentions that Wikipedia is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization. This emphasizes the importance of respecting intellectual property rights.

Related Links and Navigation

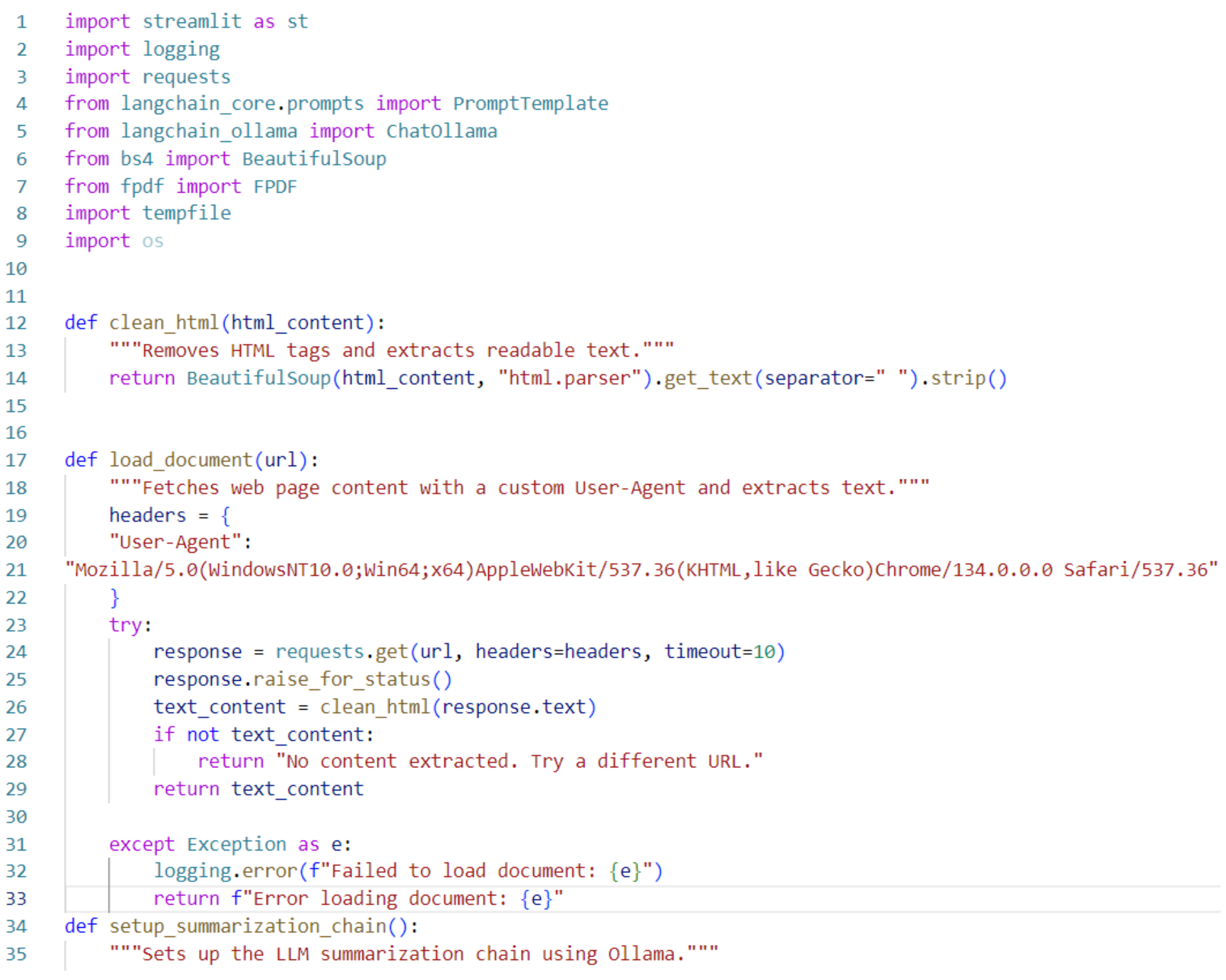
Throughout the page, there are links to other related topics, such as the table of contents, which allows users to navigate the page more easily. The text also mentions 14 languages, suggesting that Wikipedia may offer multilingual support for users from diverse linguistic backgrounds.

In conclusion, this summary provides a detailed overview of the disambiguation pages and related information on Wikipedia. It highlights the importance of clear categorization, licensing information, and navigation tools to ensure a smooth user experience.

Figure 6.4: Output No: 4

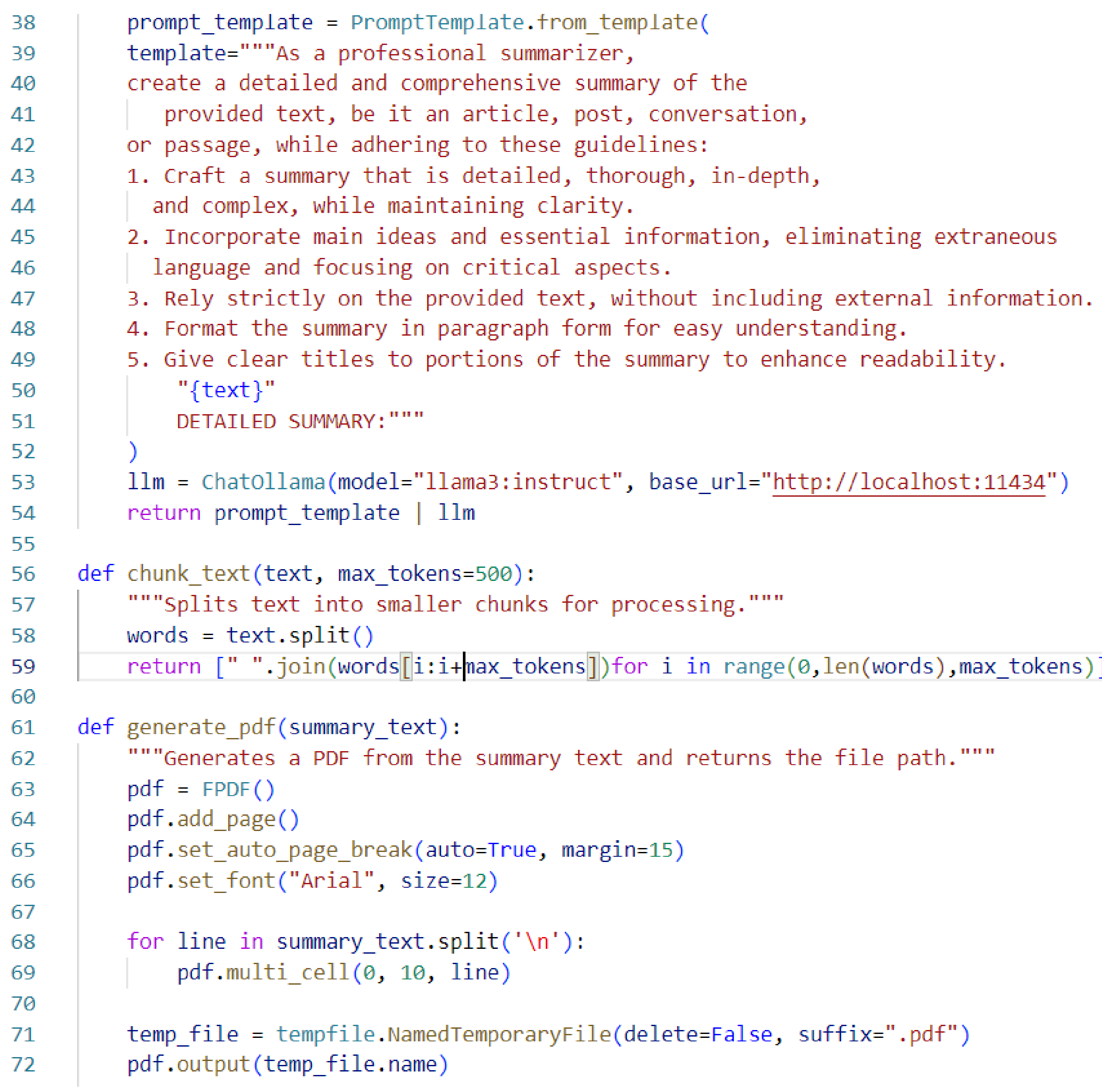
6.3 Source Code

This section presents the visual representation of the source code , showcasing different stages of the implementation.



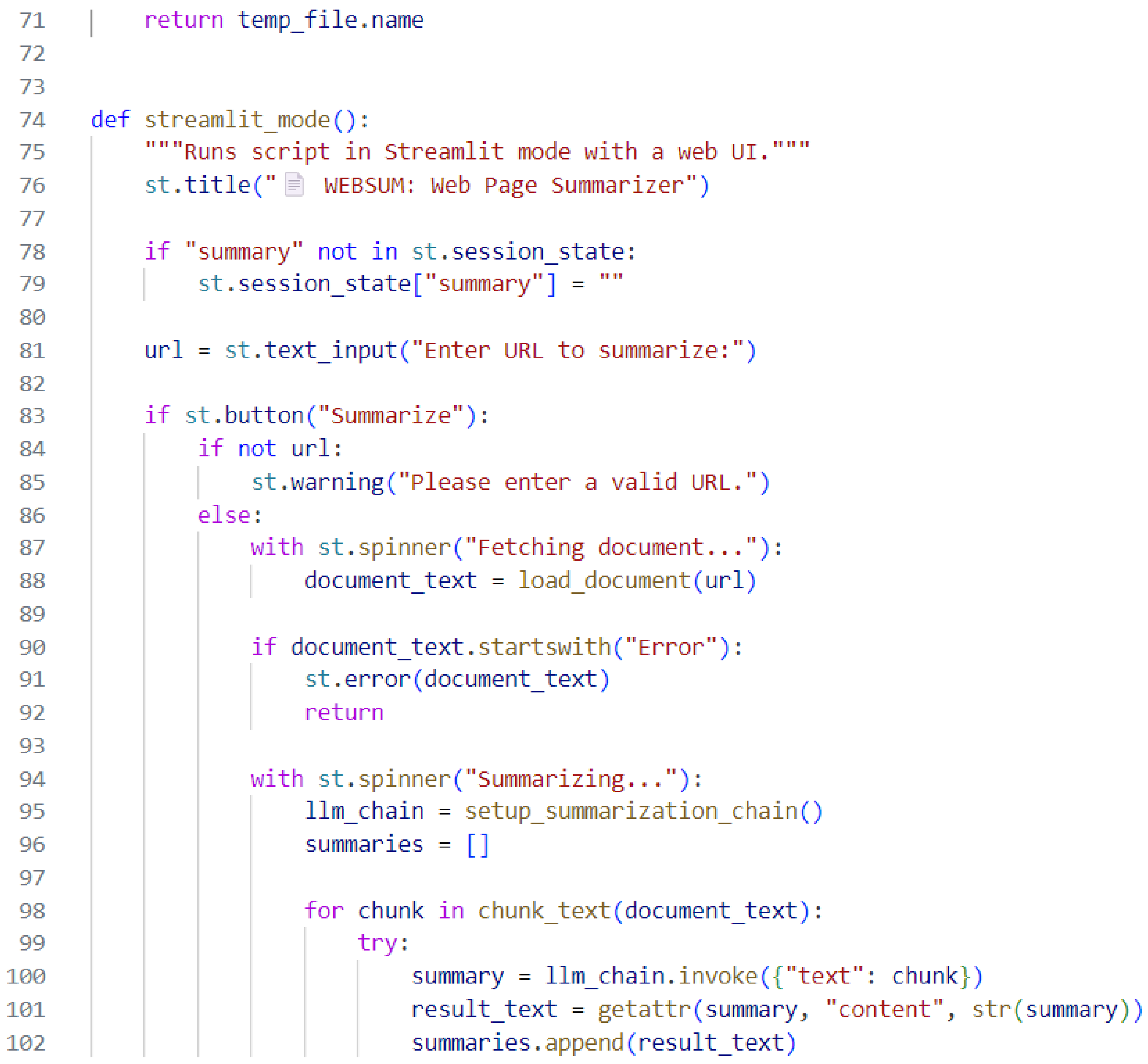
```python
import streamlit as st
import logging
import requests
from langchain_core.prompts import PromptTemplate
from langchain_ollama import ChatOllama
from bs4 import BeautifulSoup
from fpdf import FPDF
import tempfile
import os


def clean_html(html_content):
    """Removes HTML tags and extracts readable text."""
    return BeautifulSoup(html_content, "html.parser").get_text(separator=" ").strip()


def load_document(url):
    """Fetches web page content with a custom User-Agent and extracts text."""
    headers = {
    "User-Agent":
"Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,like Gecko)Chrome/134.0.0.0 Safari/537.36"
    }
    try:
        response = requests.get(url, headers=headers, timeout=10)
        response.raise_for_status()
        text_content = clean_html(response.text)
        if not text_content:
            return "No content extracted. Try a different URL."
        return text_content

    except Exception as e:
        logging.error(f"Failed to load document: {e}")
        return f"Error loading document: {e}"
def setup_summarization_chain():
    """Sets up the LLM summarization chain using Ollama."""
```

Figure 6.5: Source Code No: 1



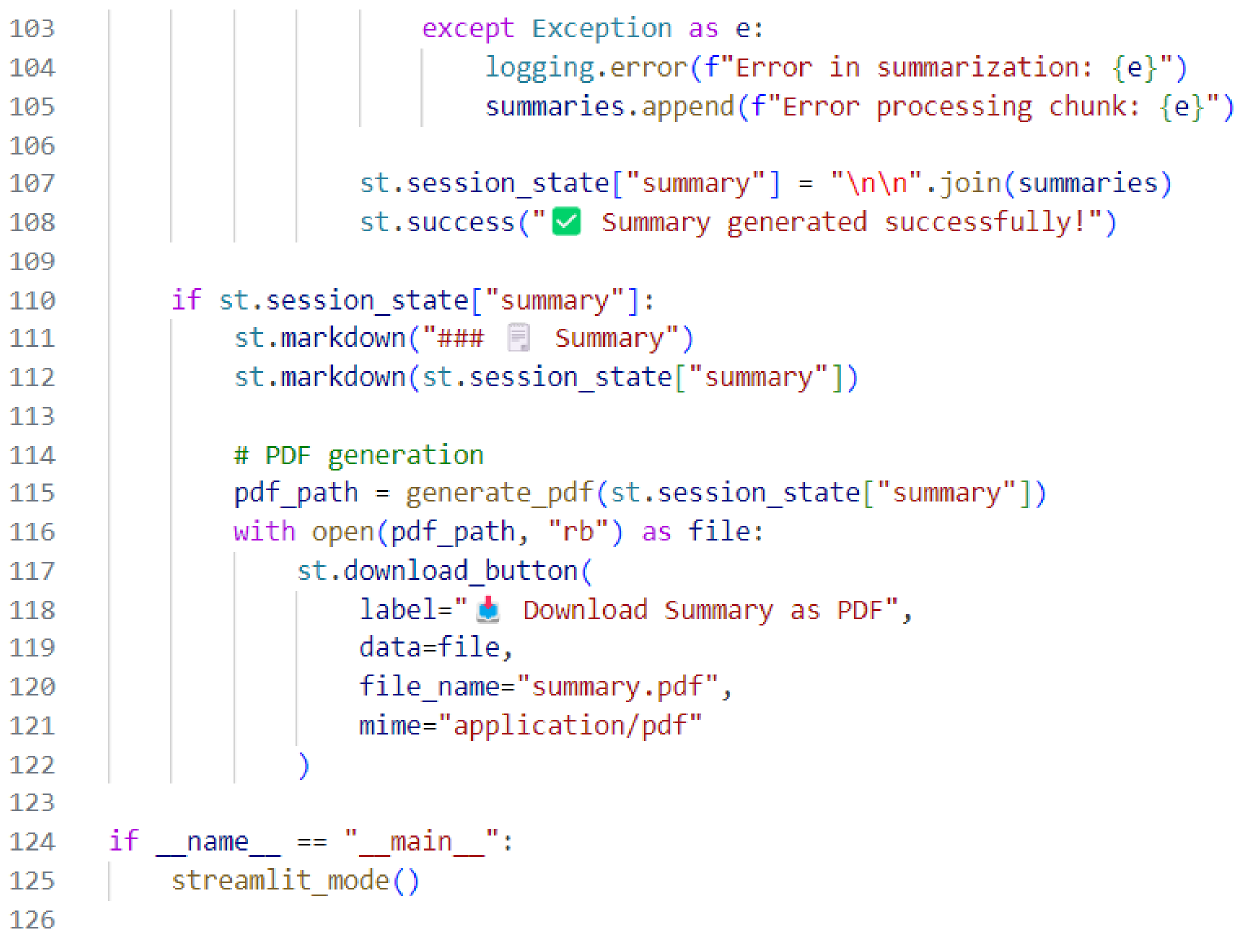
```
    prompt_template = PromptTemplate.from_template(
    template="""As a professional summarizer,
    create a detailed and comprehensive summary of the
       provided text, be it an article, post, conversation,
    or passage, while adhering to these guidelines:
    1. Craft a summary that is detailed, thorough, in-depth,
      and complex, while maintaining clarity.
    2. Incorporate main ideas and essential information, eliminating extraneous
      language and focusing on critical aspects.
    3. Rely strictly on the provided text, without including external information.
    4. Format the summary in paragraph form for easy understanding.
    5. Give clear titles to portions of the summary to enhance readability.
        "{text}"
        DETAILED SUMMARY:"""
    )
    llm = ChatOllama(model="llama3:instruct", base_url="http://localhost:11434")
    return prompt_template | llm

def chunk_text(text, max_tokens=500):
    """Splits text into smaller chunks for processing."""
    words = text.split()
    return [" ".join(words[i:i+max_tokens])for i in range(0,len(words),max_tokens)]

def generate_pdf(summary_text):
    """Generates a PDF from the summary text and returns the file path."""
    pdf = FPDF()
    pdf.add_page()
    pdf.set_auto_page_break(auto=True, margin=15)
    pdf.set_font("Arial", size=12)

    for line in summary_text.split('\n'):
        pdf.multi_cell(0, 10, line)

    temp_file = tempfile.NamedTemporaryFile(delete=False, suffix=".pdf")
    pdf.output(temp_file.name)
```

Figure 6.6: Source Code No: 2



```python
    return temp_file.name


def streamlit_mode():
    """Runs script in Streamlit mode with a web UI."""
    st.title("📄 WEBSUM: Web Page Summarizer")

    if "summary" not in st.session_state:
        st.session_state["summary"] = ""

    url = st.text_input("Enter URL to summarize:")

    if st.button("Summarize"):
        if not url:
            st.warning("Please enter a valid URL.")
        else:
            with st.spinner("Fetching document..."):
                document_text = load_document(url)

            if document_text.startswith("Error"):
                st.error(document_text)
                return

            with st.spinner("Summarizing..."):
                llm_chain = setup_summarization_chain()
                summaries = []

                for chunk in chunk_text(document_text):
                    try:
                        summary = llm_chain.invoke({"text": chunk})
                        result_text = getattr(summary, "content", str(summary))
                        summaries.append(result_text)
```

Figure 6.7: Source Code No: 3



```
                    except Exception as e:
                        logging.error(f"Error in summarization: {e}")
                        summaries.append(f"Error processing chunk: {e}")

                st.session_state["summary"] = "\n\n".join(summaries)
                st.success("✅ Summary generated successfully!")

    if st.session_state["summary"]:
        st.markdown("### 📄 Summary")
        st.markdown(st.session_state["summary"])

        # PDF generation
        pdf_path = generate_pdf(st.session_state["summary"])
        with open(pdf_path, "rb") as file:
            st.download_button(
                label="📥 Download Summary as PDF",
                data=file,
                file_name="summary.pdf",
                mime="application/pdf"
            )

if __name__ == "__main__":
    streamlit_mode()
```

Figure 6.8: Source Code No: 4

Chapter 7

Result and Discussion

7.1 Results

This section presents the results achieved after the successful development and execution of the project titled **“WEBSUM – Summary Generator”**. The outcomes are discussed based on three key aspects: Model Performance, LLM Integration, and User Interface (UI) Implementation.

7.1.1 Model Performance

In this project, the summarization capability was powered by the LLaMA 3: Instruct model, running locally on Windows 11 through the Ollama framework. The model was invoked using the `ChatOllama` module provided by LangChain, which facilitated interaction with the LLM in an efficient and modular manner.

The summarization model showed excellent performance in generating detailed and human-readable summaries for static web pages. The summarization prompt was designed using LangChain’s `PromptTemplate`, with strict instructions for comprehensive summarization including clear titles, important points, and structured formatting.

To handle large content from web pages, the input text was divided into smaller chunks of 500 tokens each. This chunking approach maintained the summarization quality without exceeding model input limitations. The model showed good capability in capturing the essence of long articles while discarding unnecessary information.

However, the model exhibited limitations in handling dynamic web content that required JavaScript rendering. Moreover, the summarization speed was dependent on local system configuration as the model was executed locally without GPU acceleration.

7.1.2 LLM Integration

The project achieved effective integration of the Large Language Model (LLaMA 3) using the LangChain framework in a modular architecture. The LangChain `PromptTemplate` was utilized to create customized prompts to guide the model for detailed summarization tasks.

The model execution was facilitated using Ollama running on Windows 11, allowing local LLM execution without relying on external cloud APIs. This provided a privacy-preserving solution where user data remained secure on the local machine.

Additionally, LangChain's capabilities like prompt chaining and chunk-wise summarization were used efficiently to overcome the token limitations of the model. The integration of model invocation within the Streamlit workflow ensured a smooth backend operation for real-time summarization requests from users.

7.1.3 User Interface (UI)

The project included the development of a simple and user-friendly interface using the Streamlit library, suitable for Windows 11 execution through PowerShell. The interface allowed users to enter a URL, fetch the web page content, and generate a summary at the click of a button.

Streamlit's clean design ensured easy navigation and readability of the summary. The generated summary was displayed in a structured format with section titles and paragraphs for better understanding.

Further, an additional feature of downloading the generated summary as a PDF was incorporated using the FPDF Python library. This enhanced the usability of the project, allowing users to retain a portable document of the summarized content.

7.1.4 Highlights

The project demonstrated significant results through its architecture and implementation. The following are the key highlights of the outcomes:

- Successful summarization of static and structured web content using LLaMA 3 model through Ollama locally.
- Privacy-preserving LLM execution without external API calls ensuring data security.
- Modular and optimized LLM integration using LangChain's advanced utilities like `PromptTemplate` and chunk-wise processing.

- Development of an interactive UI using Streamlit for user-friendly summarization workflow.
- Additional feature of summary download in PDF format enhancing usability.
- Execution environment optimized for Windows 11 with PowerShell for local deployment.

7.2 Discussions

This project is a web-based application developed using Python, Streamlit, and LangChain-Ollama integration. The primary objective is to allow users to input any web page URL and get a detailed, well-structured summary of the content. It fetches the web content, cleans the HTML, splits large content into manageable chunks, summarizes each part using LLM, and provides an option to download the final summary as a PDF.

7.2.1 Strength of Implementation

The implementation of the WEBSUM project demonstrates a significant strength in terms of modularity, usability, and performance. The use of Streamlit for building the user interface provides an interactive and visually appealing platform for users to interact with the application effortlessly on Windows 11 systems.

One of the key strengths is the integration of the locally hosted Ollama Large Language Model (LLM) for summarization, which ensures data privacy, eliminates the need for internet-based APIs, and provides faster response time for generating summaries. The clean separation of functions such as loading the web document, cleaning the HTML content, chunking large text, invoking the LLM summarizer, and generating a PDF makes the project well-organized and maintainable. The project also ensures that summaries are well-structured and detailed, with an additional feature of providing downloadable summaries in PDF format, which adds professional utility to the tool.

7.2.2 Limitations and Challenges

Despite its strengths, the WEBSUM project faces certain limitations and technical challenges. One of the primary challenges is its inability to handle dynamic websites that load content using JavaScript or AJAX calls, as the current implementation relies on simple HTTP requests and BeautifulSoup for content extraction, which works effectively only on static web pages.

The summarization output quality heavily depends on the capability of the locally installed Ollama model, and certain complex web documents might result in context loss due to the word-based chunking approach rather than token-aware splitting. Additionally, the dependency on a locally hosted LLM server restricts the deployment of this solution to local machines only, limiting its usability in distributed or online environments. Another limitation is the lack of multilingual summarization support, which confines its usage primarily to English content.

7.2.3 Potential Improvements

To address the existing challenges and enhance the functionality of the project, several potential improvements can be considered. The content extraction mechanism can be upgraded by integrating dynamic content handling tools such as Selenium or Playwright, which would allow fetching content from JavaScript-based websites. Implementing token-based text chunking rather than word count will significantly improve the summarization accuracy and preserve the context in larger documents.

To expand usability, the application can be adapted for cloud deployment, enabling remote access and scalability. Furthermore, enhancing the user interface with advanced customization features like summary length selection, summary style options, or language detection and translation support would make the tool more versatile.

Incorporating additional features such as keyword extraction, sentiment analysis, and visual analytics like word clouds will further enrich the user experience and make the output more insightful and engaging.

7.3 Future Scope

There is a vast scope to enhance the project further in multiple dimensions:

- Integration of JavaScript-rendered dynamic content extraction using advanced libraries like Selenium or Playwright.
- Extending summarization capability to multiple languages using multilingual LLMs.
- Summarization support for multiple document formats like PDF, DOCX, TXT directly.
- Enhancing model performance through local GPU acceleration or optimizing the chunk processing logic.
- Extracting the summaries of images in the website.

Chapter 8

Conclusion

The development of this **offline Web Summarizer**, powered by **LLaMA, Ollama, LangChain, and Streamlit**, addresses the pressing need for a **secure, efficient, and cost-effective** text summarization solution. In an era of unprecedented digital content growth, traditional summarization tools often rely on cloud-based models, raising concerns about **data privacy, API costs, and internet dependency**. This project presents an innovative alternative by enabling **local execution of large language models**, ensuring that users can generate summaries **without compromising security or incurring financial burdens**.

By leveraging **Ollama**, the system achieves **high-speed inference** without requiring external cloud processing, making it suitable for **privacy-sensitive applications** in research, healthcare, finance, and law. Additionally, **LangChain** enhances the summarization process by **integrating memory and retrieval mechanisms**, enabling **context-aware and structured text generation**. The inclusion of **Streamlit** further improves usability, offering an intuitive **web-based interface** that makes the tool accessible to users of all technical backgrounds.

Beyond individual users, this Web Summarizer has significant implications for **organizations, researchers, and businesses** that require **automated, domain-specific summarization** without the limitations of cloud-based services. By eliminating **API costs**, the system ensures affordability, making **LLM-driven summarization accessible to a wider audience**. Furthermore, it enhances **productivity and decision-making** by enabling users to extract key insights from large volumes of text in **real time**.

This project also lays the groundwork for future enhancements, including:

- **Multilingual summarization** for greater accessibility.
- **Real-time speech-to-text summarization** for processing audio content.
- **Domain-specific fine-tuning** to improve accuracy in specialized fields such as medicine, finance, and law.

Ultimately, this Web Summarizer stands as a **versatile, scalable, and privacy-preserving** solution, demonstrating the potential of **local LLM execution** for handling vast amounts of textual data efficiently. As advancements in **natural language processing (NLP) and large language models (LLMs)** continue, this project lays a **strong foundation** for future innovations in **AI-driven summarization**, bridging the gap between **accessibility, security, and performance** in modern information processing.

....

References

[1] Daniel Jurafsky, James H. Martin: *Speech and Language Processing* (3rd Edition), Pearson, 2023.

[2] Hobson Lane, Hannes Hapke, Cole Howard: *Natural Language Processing in Action* (2nd Edition), Manning Publications, 2021.

[3] Uday Kamath,Kevin Keenan,Garrett Somers,Sarah Sorenson: *Large Language Models: A Deep Dive: Bridging Theory and Practice*, Technical Report, 2023.

[4] Tong Xiao, Jingbo Zhu: *Foundations of Large Language Models*, Springer, 2023.