Feasibility Report on

AI-POWERED DOCUMENT SUMMARISER

By

Anvit Gupta
22114009
anvit_g@cs.iitr.ac.in

Krishna Aggarwal
22114048
krishna_a@cs.iitr.ac.in

Sarvasva Gupta
22114086
sarvasva_g@cs.iitr.ac.in

Utkarsh Lohiya
22114103
utkarsh_l@cs.iitr.ac.in

Vineet Kumar
22114107
vineet_k@cs.iitr.ac.in

Executive Summary

Project Overview:
The project is aimed at creating a AI-based web application which enables users to enter lengthy documents and receive and concise summary of the content.

Key Features:
Document input: The application allows users to paste or input text documents of varying lengths in any language.
Language Translation: If language is non-English, then converting it into English for translation.
Auto summarization: NLP algorithms will process the text and then automatically generate the summary. And translate the summary into input language.
Interactive Output: Separate output page will show the summary as well as input document. User can copy the generated summary to clipboard.
Number of words and Characters in input text by the user and in the summary generated by the application. This feature will help to get the required summary in the pre-defined word limit for the user.
Error Handling: Implement error handling to gracefully manage scenarios where users input invalid documents, unsupported format or if any processing issue encountered.

Optional Features:
Multiple document input and output: User can choose how many summaries he wanted and then he will decide the best of all. Also, allowing multiple files input.
Document Format Handling: Support various document formats, including plain text and common document formats like PDF and DOCX.
Feedback system: User may provide feedback on the output summary generated.
Extractive and abstractive summarization

Preliminary Requirements Analysis

1. Introduction: The AI Document Summarizer is a software application designed to automatically generate concise and coherent summaries of long and complex documents. This preliminary requirement analysis outlines the high-level functional and non-functional requirements for the development of the AI Document Summarizer.
2. Functional Requirements:

- The system should accept various document formats, including PDF, DOCX etc.
- It should support multiple languages for document content.
- The system should employ natural language processing (NLP) techniques to analyze and understand the content of the input documents.
- It should extract key sentences and paragraphs that represent the most important information.
- The AI should be able to identify and summarize content from various domains, such as news articles, research papers, and legal documents.

- Users should have the option to choose the length and style of the summary (e.g., short, medium, long).
- It should maintain the original context and flow of the document.
- Users should be able to upload documents and access summarized content effortlessly.

3. Non-Functional Requirements:

- The system should generate summaries quickly, with response times typically less than a few seconds.
- The user interface should be intuitive, and users should require minimal training to use the system effectively.
- It should have provisions for customer support and issue resolution by feedback

Introduction

Project Overview: The "AI-Powered Document Summarizer" project aims to harness the power of artificial intelligence and natural language processing (NLP) to simplify the comprehension of lengthy documents. This web application empowers users to submit extensive textual documents and receive concise and coherent summaries, thereby streamlining the process of information consumption.

Purpose, Scope, and Objectives: This feasibility study assesses the viability of creating an AI document summarization system, exploring project scope, features, and goals.

Problem Description: In today's information-driven landscape, individuals and organizations grapple with the overwhelming influx of textual content, spanning research papers, articles, legal documents, and reports. This deluge of data presents a formidable challenge—how to discern and extract vital insights from extensive documents swiftly. The "AI-Powered Document Summarizer" seeks to address this challenge by offering an automated solution for distilling essential information into coherent, easily digestible summaries.

Goals and Objectives

- **Goal:** Simplify understanding of lengthy documents and provide quick access to important information.
- **Objectives:**
  - **Efficiency:** Save users time by summarising documents automatically.
  - **Clarity:** Present information clearly.
  - **Customisation:** Allow users to tailor summaries.
- **Problem It Aims to Solve:** The project addresses the challenge of extracting key insights from extensive content benefiting individuals and organisations.

Technical Requirements and Feasibility

Technical Requirements:

1. NLP Processing and Algorithms:

- Implement Natural Language Processing (NLP) algorithms for document summarization.
- Utilize NLP libraries such as NLTK or spaCy to process and analyze text efficiently.

2. **Web Application Development:**
   - Backend development using Python, with Flask as the web framework.
   - Frontend development using React-JS for a dynamic and responsive user interface.
3. **Real-time Processing:**
   - Ensure real-time summarization for a seamless user experience.
   - Optimize algorithms and processes to generate summaries quickly.
4. **File Handling and Document Parsing:**
   - Implement file handling libraries to support various document formats, including plain text, PDF, and DOCX.
   - Develop mechanisms for parsing and extracting content from these formats.
5. **User Interface (UI) and User Experience (UX):**
   - Design a user-friendly interface using React-JS and CSS.
   - Utilize design tools like Figma or Freeform to create an intuitive and appealing UI/UX.
6. **Asynchronous HTTP Requests:**
   - Use Axios library in React for handling asynchronous HTTP requests between the frontend and backend.
7. **Error Handling and Feedback Mechanism:**
   - Implement error handling to gracefully manage invalid input documents or unsupported formats.
   - Provide a feedback system for users to comment on the generated summaries.

Feasibility Assessment:

1. **Technical Feasibility:**
   - The technologies required for the project, including Python, NLP libraries, React-JS, and file handling libraries, are well-established and readily available.
   - The development team possesses the necessary skills and experience to implement the required technology stack.
2. **Resource Availability:**
   - Adequate computing resources and infrastructure are available to host and run the web application.
   - The team has access to necessary development tools, version control systems, and design software.
3. **Budget and Costs:**
   - The project can be implemented within a reasonable budget, considering the availability of open-source tools and frameworks.
4. **Timeframe:**
   - The project can be completed within the allocated timeframe, given the scope and complexity of the requirements.
5. **Risk Analysis and Fallback Plan:**
   - Identified risks, such as potential technical challenges or resource constraints, have been assessed and accounted for.
   - A fallback plan is in place to mitigate risks and ensure project continuity.

Risk Analysis

1. **Technical Challenges:**

- Risk: Running into technical difficulties with NLP algorithms, AL model training, or integration with React JS.
- Fallback Plan: Seeking online resources, help from professors, and allocating time for research and troubleshooting.

2. **Algorithm Performance:**
   - Risk: The AI might not produce high-quality summaries (i.e., incomplete summaries, long summaries, etc.)
   - Fallback Plan: Keep trying different things, like exploring different NLP models, introducing length constraints in summarization algorithms, etc., until you get better results.

3. **Testing and Quality Assurance:**
   - Risk: The app might have many bugs or not work well.
   - Fallback Plan: Testing the app in phase containments (i.e., incremental testing)

4. **Documentation and Knowledge Transfer:**
   - Risk: Lack of documentation can make it difficult for team members to maintain or understand the project.
   - Fallback Plan: Document the project thoroughly, including code, algorithms, and deployment procedures.

5. **User Interface (UI) and User Experience (UX) :**
   - Risk: The front-end design may not meet user expectations or usability standards.
   - Fallback Plan: Using user interface design apps like Figma to properly design the web application and seek feedback from users and team members.

6. **Team Coordination:**
   - Risk: Challenges in team coordination, conflicting schedules, or communication issues.
   - Fallback Plan: Establish clear communication channels and regular team meetings through MS Teams and WhatsApp groups.

Outline plan and Project Activities

During the development of a large and complex software development, various models and stages are to be completed to achieve the software which satisfies all the functional requirements stated by the customer as well as non- functional requirements of the product application. Instead of both these requirements, developers must take care of the future enhancements in the functionalities, so that its scope in the market increases in the future.
Various activities involved in this project is outlined below:

1. **Requirements analysis and key features**: Functional and non-functional requirements that the end application before production must satisfy are gathered and analysed in this stage.
   This process can be divided into two parts:

   a. **Requirement Gathering**: It involves the gathering or collection of requirements from the user or customer. Different groups of customers and users are targeted to include all the requirements in this stage.

b. **Requirement Analysis and specification**: The gathered requirements are analysed and well documented to avoid future conflicts between developer/company and customer.

   The output of this stage will include all the requirements by the user (functional) and required by the device to function correctly (non-functional) into the one document called the SRS document.

2. **Prototype Design stage**: Prototype or wireframe can be any mock-up or demo of what a website will look like when it goes live. This stage involves how the website will look like and how it will behave. Prototype designing will be done on Figma.

3. **Detailed design**: This involves the preparation of the whole design and the user interface of the application. Designing will be done in the shared documents on Figma.

4. **Implementation Stage**: Converting the prepared design model in the previous stage to the actual software application through programming. This involves the use of React-js or Node-js, HTML, CSS and other support languages.

5. **Testing**: This stage is aimed at checking and validating that the software reached the non-functional and some functional requirements specified earlier in the requirement stage. This stage also aims at testing the software for any kind of bugs or errors and then fixing it immediately.
   Although testing is performed at every stage so that bugs and errors occurred in one stage will not pass to the next stage, thereby decreasing risk associated with the development process.

6. **Extensions and Additions:** This stage is aimed at improving the existing functionalities and/or adding new functionalities to the system. This helps in increasing the value of the application in the market with the other available document summarisers. Adding new features and improving the existing ones is the very important part of SDLC.

7. **Final testing:** This stage involves final testing of the project including functional and non-functional requirements before the application is set into the production stage. Functional requirements specified by the user at the time of requirements gathering stage, must be satisfied in this testing, bugs and error fixing, non-functional requirements like performance, scalability, flexibility, reliability etc. are all tested to meet the end user satisfaction.

Conclusion

- The proposed AI-Powered Document Summarizer project holds significant promise in addressing the growing need for efficient summarization of lengthy documents. By leveraging cutting-edge NLP algorithms and technologies, the application aims to simplify information consumption and enhance productivity for users dealing with extensive textual content.
- The technical feasibility assessment demonstrates that the project is technically viable and within reach. With the right mix of NLP algorithms, web development frameworks, and

effective error handling mechanisms, the application can efficiently process diverse document formats and provide accurate summaries in real-time.

– Moreover, the availability of essential resources, a skilled development team, and the flexibility of open-source technologies contribute to the project's feasibility. The project aligns with our company's mission to innovate and leverage technology for enhancing user experiences, making it a compelling venture to pursue.
– In conclusion, this feasibility report affirms that the AI-Powered Document Summarizer project is both technically and practically feasible, setting the stage for its successful development and eventual deployment.

Contribution

- Anvit Gupta - Sections deciding and dividing the task among members, Title page, outline plan and activities.

- Krishna Aggarwal - Executive Summary, preliminary requirements analysis

- Vineet Kumar – Introduction

- Utkarsh Lohiya - Technical Requirements and Feasibility, Conclusion

- Sarvasva Gupta – Risk Analysis