



and



## **Department**

**Ubiquitous and Distributed Systems - Cloud & IoT engineer**

---

# **Semantic Analysis and Partial Analysis of PDF Documents: Enhancing Document Understanding and Extraction**

---

July 1, 2023 – August 30, 2023

## **Internship report**

**Author:**

AFERHANE Jamila

**Supervised by:**

M. El AICHI Mohamed Mohyi Eddine

# Contents

Contents.....	2
Table of Figures: .....	7
1 Introduction:.....	10
1.1 Background: .....	10
1.2 Objectives:.....	10
1.3 Scope: .....	11
1.4 Motivation: .....	12
2 Company:.....	13
2.1 Overview: .....	13
2.2 Introduction: .....	13
2.3 Industry and Services:.....	14
2.4 Vision: .....	14
2.5 Collaborations: .....	15
3 Literature Review:.....	15
3.1 Natural Language Processing (NLP): .....	15

3.2 Semantic Analysis:	16
3.2.1 Definition:	16
3.2.2 Mechanism of working:	17
3.2.3 Critical elements of semantic analysis:	18
3.2.4 Machine learning algorithm-based automated semantic analysis:	19
3.2.5 Semantic analysis techniques:	20
3.3 Document Summarization Using Latent Semantic Indexing:	22
3.3.1 Document summarization:	22
3.3.2 Latent Semantic Indexing:	22
3.3.3 Singular Value Decomposition (SVD):	23
3.3.4 Summarization using Latent Semantic Analysis	25
3.4 Keyword extraction:	25
3.4.1 Definition:	25
3.4.2 Simple Statistical Approaches for Keyword extraction:	26
4 Methodology:	28
4.1 Conception:	28

4.1.1 User case Diagram: .....	28
4.1.2 Class Diagram: .....	29
4.1.3 Planification: Work Breakdown Structure (WBS) .....	29
4.2 Website Architecture: .....	30
4.2.1 Architectural style: .....	30
4.2.2 Client-Side Components: .....	31
4.2.3 Server-Side Components: .....	31
4.2.4 Data Management: .....	31
4.2.5 Communication: .....	32
4.3 Algorithms: .....	32
4.3.1 Title extraction algorithm: .....	32
4.3.2 Keyword Extraction Algorithms: .....	33
4.3.3 Text summarization algorithms: .....	34
4.3.4 Bridges: Linking Algorithms to the Server: .....	36
4.4 PDF Document Processing: .....	37
4.4.1 Conversion Algorithm: .....	38

4.4.2 Text Preprocessing:.....	38
4.5 Tools:.....	38
4.5.1 Programming Languages: .....	39
4.5.2 Libraries and Frameworks: .....	41
4.5.3 Development environment:.....	44
5 Front-end development: .....	45
5.1 Logo: .....	45
5.1.1 Design: .....	45
5.1.2 Significance:.....	46
5.2 Theme:.....	46
5.2.1 Color scheme: .....	46
5.2.2 Typography: .....	47
5.2.3 Visual elements: .....	47
5.3 React components: .....	49
5.4 User interfaces:.....	54
5.4.1 Types of analysis interface: .....	55

5.4.2 Semantic Analysis interface:.....	57
5.4.3 Partial Analysis Interface:.....	60
5.5 Design principles: .....	62
5.5.1 Responsiveness: .....	62
5.5.2 Consistency: .....	64
5.5.3 Usability: .....	65
5.6 Pages: .....	65
5.6.1 Home page: .....	65
5.6.2 Semantic analysis page: .....	67
5.6.3 Partial analysis page:.....	67
6 Backend development: .....	68
6.1 Architecture:.....	68
6.2 API Endpoints: .....	69
7 Conclusion: .....	70

## Table of Figures:

Figure 1: How does semantic analysis work.....	18
Figure 2: Singular Value Decomposition .....	23
Figure 3: Use Case Diagram .....	28
Figure 4: Class Diagram .....	29
Figure 5: System architecture .....	31
Figure 6: Preprocessing for summarization .....	35
Figure 7: Document Term Matrix .....	35
Figure 8: LSA model schema.....	36
Figure 9: Project logo.....	45
Figure 10: Purplish pink color .....	46
Figure 11: Daisy Bush color .....	46
Figure 12: ArtClick Sky Blue color .....	47
Figure 13: primary action button example.....	48
Figure 14: Analyze button.....	48
Figure 15: Upload button .....	48

Figure 16: Project components .....	49
Figure 17: Header.....	50
Figure 18: Footer.....	50
Figure 19: Hero component .....	51
Figure 20: Analyzing types in the main page .....	51
Figure 21: Analyzing types when hovering on Semantic Analysis .....	52
Figure 22: Value component.....	52
Figure 23: Mechanism component.....	53
Figure 24: Semantic Analyzer.....	53
Figure 25: Partial Analyzer .....	54
Figure 26: Home interface .....	55
Figure 27: Semantic Analysis card .....	56
Figure 28: Semantic Analysis card while hovering .....	56
Figure 29: Partial Analysis card.....	57
Figure 30: Pratial Analysis card while hovering.....	57
Figure 31: Uploading a pdf document .....	58



Figure 32: Pdf analysis.....	59
Figure 33: Downloaded analysis.....	60
Figure 34: Partial analysis interface.....	61
Figure 35: Header for big screens.....	62
Figure 36: Header for medium screens.....	63
Figure 37: Header for screens less than 768 px.....	63
Figure 38: Analysis type section for big and medium screens.....	64
Figure 39: analysis type section for small screens.....	64
Figure 40: Used images in the project.....	65
Figure 41: Home page.....	66
Figure 42: Semantic analysis page.....	67
Figure 43: Partial analysis page.....	68

# 1 Introduction:

## 1.1 Background:

Semantic analysis, text summarization, and keyword extraction are essential natural language processing (NLP) techniques that play a crucial role in information retrieval, content understanding, and knowledge extraction from textual data. In today's digital age, the exponential growth of textual information, particularly in the form of PDF documents, necessitates efficient methods to process and analyze this vast amount of data.

The need for automating the analysis of PDF documents arises from the challenges posed by the overwhelming volume of information. Manual analysis of lengthy documents can be time-consuming and error-prone, making it difficult for users to efficiently extract key insights and relevant information. Automated semantic analysis, summarization, and keyword extraction present an opportunity to tackle these challenges and empower users with meaningful and concise information from PDF documents.

## 1.2 Objectives:

The objective of this internship project is to **develop a comprehensive system** that applies **semantic analysis**, text summarization, and keyword extraction techniques to **PDF documents**. Additionally, the system aims to provide **partial analysis** capabilities, enabling users to focus on specific sections of interest within the documents. The goal is **to create a user-friendly website** that allows users to **upload PDF files** and choose between full semantic analysis or partial analysis for targeted insights.

The specific objectives of the project are as follows:

### Semantic Analysis:

Develop algorithms and methods for generating summaries, keywords, and title of PDF documents.

### **Partial Analysis:**

Design a user-friendly interface that allows users to specify and target particular sections of interest within the PDF documents. And develop algorithms to perform partial analysis on the selected sections to provide detailed insights.

### **Web-Based System:**

Create a responsive and intuitive website that facilitates user interaction and document upload.

## **1.3 Scope:**

The scope of this internship project encompasses the development of a web-based system that applies semantic analysis and partial analysis s to PDF documents. However, it is important to note that the project has certain limitations and focuses on specific aspects:

### **Supported Document Format:**

The system will primarily support **PDF documents** for analysis. While PDF is a widely used and standardized format, other document formats, such as Microsoft Word or plain text, will not be included in the scope of this project.

### **Language Support:**

The system will initially focus on processing and analyzing documents written in **English**. Extending the language support to other languages is beyond the current scope and would require additional language-specific NLP components.

### **Semantic Analysis Depth:**

The semantic analysis component will provide **summaries**, **keywords**, and **titles** for the input PDF documents. However, the depth of the analysis will be limited to what can be reasonably achieved within the internship duration.

### **Partial Analysis Depth:**

The partial analysis functionality will allow users to analyze specific sections of interest within the PDF documents. However, the system **will not provide a fine-grained analysis of each subsection or paragraph**. The focus is on enabling users to **gain insights into key sections** efficiently.

### **Real-Time Analysis:**

The system will process and analyze uploaded PDF documents in **real time**. However, the processing time will depend on the document's size and complexity. Large or complex documents may require more processing time.

### **External Data Sources:**

The system will not integrate with external data sources, such as web scraping or database connectivity, for gathering additional information about the documents. The **analysis will solely rely on the content present in the uploaded PDF files**.

The scope defined above allows for a focused and achievable project outcome within the internship timeframe. Future iterations and enhancements can be explored to expand the system's capabilities and address additional challenges. The primary goal is to create a functional and effective web-based system that provides valuable insights from PDF documents, contributing to the field of natural language processing and information retrieval.

## **1.4 Motivation:**

The motivation behind undertaking this internship project stems from the growing need to tackle the challenges posed by the vast and ever-expanding volume of textual information in today's digital world. With the exponential increase in digital content, especially in the form of PDF documents, users face difficulties in efficiently processing, comprehending, and extracting valuable insights from this abundance of information. Manual analysis of lengthy documents is time-consuming and may lead to oversight or misinterpretation of critical information, hindering users from making informed decisions.

Automated semantic analysis and partial analysis offer a compelling solution to address these challenges. By developing a system that can automatically process and analyze PDF documents, we aim to empower users with concise, relevant, and meaningful information, enabling them to quickly understand the content and identify crucial insights.

## 2 Company:

### 2.1 Overview:

- **Name:** Intelcom
- **Location :** 9, Zone Industrielle Attasnia -Al Massira Temara 12000
- **Domain:** Information and Communication Technology (ICT)
- **Specialization:** Leading global integrator of secure information system infrastructures
- **Services:** Consulting, IT solutions
- **Values:** Innovation, Talent, Integrity, and Team spirit

### 2.2 Introduction:

**INTELCOM**, a subsidiary of the Spanish multinational SATEC GROUP, is a leading global integrator of secure information system infrastructures, offering a comprehensive range of high-value-added solutions and services.

Throughout its journey, INTELCOM has made significant contributions to the development of advanced telecommunication technologies and information systems. With a pivotal role in providing cutting-edge solutions, the company has been instrumental in empowering public and private organizations, state administrations, and national operators.

They have expanded their portfolio to encompass a diverse range of innovative IT services and domains. These include IT consulting, Business Process Outsourcing (BPO), and embracing the transformative power of Industry 4.0.

## 2.3 Industry and Services:

**INTELCOM**, as a prominent subsidiary of the Spanish multinational SATEC GROUP, excels in the role of a leading global integrator, focusing on secure information system infrastructures. The company boasts an extensive portfolio of high-value-added solutions and services, making it a preferred choice for clients seeking digital transformation.

### Consulting:

INTELCOM offers expert technology consulting services, guiding clients on their digital transformation journey. Through strategic insights and recommendations, the company assists businesses in enhancing their competitiveness, flexibility, and efficiency within their respective industries.

### Solutions:

INTELCOM provides a diverse portfolio of innovative and secure solutions tailored to meet the specific needs of various industries. These solutions encompass areas such as networks and collaboration, security, DevOps & IT infrastructure, datacenter & cloud, and supervision & OSS (Operations Support Systems).

### Services:

Committed to delivering exemplary support to its clients, INTELCOM offers a wide range of professional services. These include multi-client ITO (Information Technology Outsourcing) services, customized ITO services, and customized BPO (Business Process Outsourcing) services. The company tailors these services to meet the unique and personalized requirements of its diverse clientele.

## 2.4 Vision:

INTELCOM envisions becoming a leading force in the ICT sector, creating value and fostering growth through innovative solutions and services. The company aims to contribute to clients' evolution,

efficiency, and productivity while maintaining a culture that values talent, integrity, and teamwork. With a focus on innovation, INTELCOM aims to stay at the forefront of the industry and make a positive impact on the community and digital world.

## 2.5 Collaborations:

INTELCOM establishes strong collaborations with renowned technology partners, recognized as industry leaders in their respective domains. This strategic approach enables the company to stay at the forefront of technological advancements, ensuring the ability to consistently offer clients the most cutting-edge and superior solutions.

Some of their collaborations are CISCO, DELLEMC, Microsoft, McAfee, ORACLE, ...

## 3 Literature Review:

### 3.1 Natural Language Processing (NLP):

**Natural Language Processing (NLP)** is a field of Artificial Intelligence that **makes human language intelligible to machines**. NLP combines the power of **language** and **computer science** to study the rules and structure of language and create intelligent systems capable of understanding, analyzing, and extracting meaning from text and speech.

NLP is used to **understand the structure and meaning of human language** by analyzing different aspects like syntax, semantics, pragmatics, and morphology. Then, **computer science transforms this linguistic knowledge into rule-based, machine learning algorithms** that can solve specific problems and perform desired tasks.

#### How does NLP work?

Using **text vectorization**, NLP tools transform text into something a machine can understand, then machine learning algorithms are fed training data and expected outputs (tags) to train machines to make associations between a particular input and its corresponding output. Machines then use

statistical analysis methods to build their own “knowledge bank” and discern which features best represent the texts, before making predictions for unseen data

### **Common NLP Tasks & Techniques:**

Many natural language processing tasks involve **syntactic and semantic analysis**, used to break down human language into machine-readable chunks.

**Syntactic analysis**, also known as parsing or syntax analysis, identifies the syntactic structure of a text and the dependency relationships between words, represented on a diagram called a parse tree.

**Semantic analysis** focuses on identifying the meaning of language. However, since language is polysemic and ambiguous, semantics is considered one of the most challenging areas in NLP.

In the upcoming section, we delve into Semantic Analysis, a fundamental cornerstone of our project.

## **3.2 Semantic Analysis:**

### **3.2.1 Definition:**

**Semantic analysis** refers to **a process of understanding natural language** (text) by extracting insightful information such as context, emotions, and sentiments from unstructured data. It gives computers and systems the ability to understand, interpret, and derive meanings from sentences, paragraphs, reports, registers, files, or any document of a similar kind.

**Semantic analysis analyzes the grammatical format of sentences**, including the arrangement of words, phrases, and clauses, to determine relationships between independent terms in a specific context. This is a crucial task of natural language processing (NLP) systems. It is also a key component of several machine learning tools available today, such as search engines, chatbots, and text analysis software.

The semantic analysis process begins by studying and analyzing the dictionary definitions and meanings of individual words also referred to as lexical semantics. Following this, the relationship between words in a sentence is examined to provide a clear understanding of the context.



When fueled by **natural language processing** and **machine learning**, systems of semantic analysis tend to achieve human-level accuracy. Several companies rely heavily on semantic analysis-driven tools that automatically draw valuable data from unstructured data such as emails, client reports, and customer reviews.







### 3.2.2 Mechanism of working:

The semantic analysis method begins with a language-independent step of analyzing the set of words in the text to understand their meanings. This step is termed 'lexical semantics' and refers to fetching the dictionary definition for the words in the text. Subsequently, words or elements are parsed. Each element is designated a grammatical role, and the whole structure is processed to cut down on any confusion caused by ambiguous words having multiple meanings.

Upon parsing, the analysis then proceeds to the interpretation step, which is critical for artificial intelligence algorithms. For example, the word 'Blackberry' could refer to a fruit, a company, or its products, along with several other meanings. Moreover, context is equally important while processing the language, as it takes into account the environment of the sentence and then attributes the correct meaning to it.

For example, 'Blackberry is known for its sweet taste' may directly refer to the fruit, but 'I got a blackberry' may refer to a fruit or a Blackberry product. As such, context is vital in semantic analysis and requires additional information to assign a correct meaning to the whole sentence or language.

Technically, semantic analysis involves:

-  Data processing.
-  Defining features, parameters, and characteristics of processed data
-  Data representation
-  Defining grammar for data analysis
-  Assessing semantic layers of processed data
-  Performing semantic analysis based on the linguistic formalism

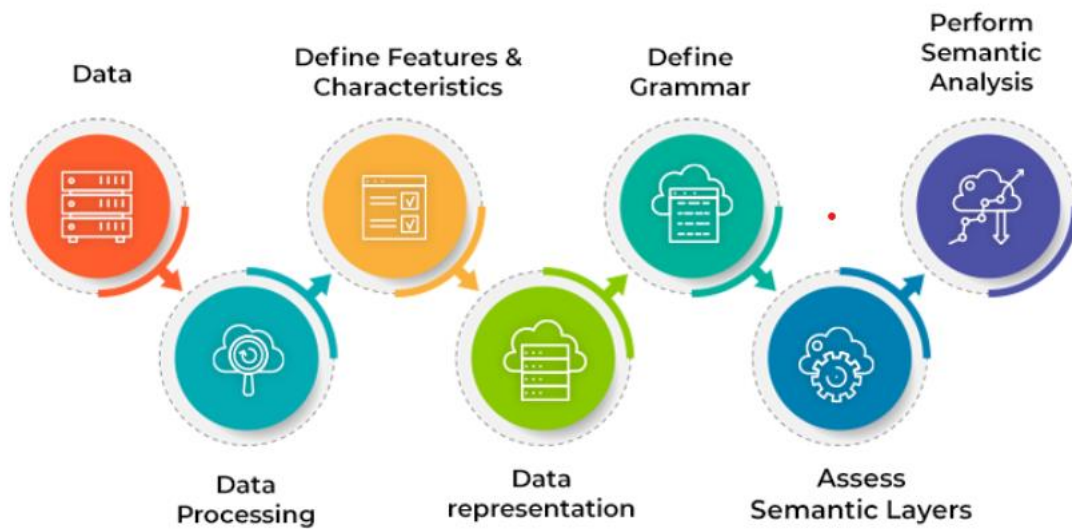


Figure 1: How does semantic analysis work

### 3.2.3 Critical elements of semantic analysis:

The critical elements of semantic analysis are fundamental to processing the natural language:

- **Hyponyms:** This refers to a specific lexical entity having a relationship with a more generic verbal entity called hypernym. For example, red, blue, and green are all hyponyms of color, their hypernym.
- **Meronymy:** Refers to the arrangement of words and text that denote a minor component of something. For example, mango is a meronymy of a mango tree.
- **Polysemy:** It refers to a word having more than one meaning. However, it is represented under one entry. For example, the term ‘dish’ is a noun. In the sentence, ‘arrange the dishes on the shelf,’ the word dishes refer to a kind of plate.
- **Synonyms:** This refers to similar-meaning words. For example, abstract (noun) has a synonyms summary–synopsis.
- **Antonyms:** This refers to words with opposite meanings. For example, cold has the antonyms warm and hot.
- **Homonyms:** This refers to words with the same spelling and pronunciation, but reveal a different meaning altogether. For example, bark (tree) and bark (dog).

Apart from these vital elements, the semantic analysis also uses **semiotics** and collocations to understand and interpret language. Semiotics refers to what the word means and also the meaning it

evokes or communicates. For example, ‘tea’ refers to a hot beverage, while it also evokes refreshment, alertness, and many other associations. On the other hand, **collocations** are two or more words that often go together. For example, fast food, dark chocolate, etc.

### 3.2.4 Machine learning algorithm-based automated semantic analysis:

One can train machines to make near-accurate predictions by providing text samples as input to semantically-enhanced **ML algorithms**. Such estimations are based on previous observations or data patterns. Machine learning-based semantic analysis involves sub-tasks such as relationship extraction and word sense disambiguation.

Let’s understand each one in further detail:

- **Word sense disambiguation**

In semantic analysis, word sense disambiguation refers to an automated process of determining the sense or meaning of the word in a given context. As natural language consists of words with several meanings (polysemic), the objective here is to recognize the correct meaning based on its use.

For example, ‘Raspberry Pi’ can refer to a fruit, a single-board computer, or even a company (UK-based foundation). Hence, it is critical to identify which meaning suits the word depending on its usage.

- **Relationship extraction**

Relationship extraction is a procedure used to determine the semantic relationship between words in a text. In semantic analysis, relationships include various entities, such as an individual’s name, place, company, designation, etc. Moreover, semantic categories such as, ‘is the chairman of,’ ‘main branch located a’’, ‘stays at,’ and others connect the above entities.

Let’s consider a phrase as an example. ‘Elon Musk is one of the co-founders of Tesla, which is based in Austin, Texas.’

This phrase illustrates two different relationships.

Elon Musk is the co-founder of Tesla

[Person]

[Company]

Tesla is based in Austin, Texas

[Company]                      [Place]

### 3.2.5 Semantic analysis techniques:

The semantic analysis uses two distinct techniques to obtain information from text or corpus of data.

The first technique refers to text classification, while the second relates to text extractor.

#### 1. Semantic classification

Semantic classification implies text classification wherein predefined categories are assigned to the text for faster task completion. Following are the various types of text classification covered under semantic analysis:

- **Topic classification:** This classifies text into preset categories on the basis of the content type. For example, customer support teams in a company may intend to classify the tickets raised by customers at the help desk into separate categories so that the concerned teams can address them. In this scenario, ML-based semantic analysis tools may recognize tickets based on their content and classify them under a ‘payment concern’ or ‘delayed delivery’ category.
- **Sentiment analysis:** Today, sentiment analysis is used by several social media platforms such as Twitter, Facebook, Instagram, and others to detect positive, negative, or neutral emotions hidden in text (posts, stories). These sentiments, in a way, denote urgency and may raise ‘call to action’ alarms for respective platforms. Sentiment analysis helps brands identify dissatisfied customers or users in real-time and gets a hint on what customers feel about the brand as a whole.
- **Intent classification:** Intent classification refers to the classification of text based on customers’ intentions in the context of what they intend to do next. You can use it to tag customers as ‘interested’ or ‘not Interested’ to effectively reach out to those customers who may intend to buy a product or show an inclination toward buying it.

#### 2. Semantic extraction

Semantic extraction refers to extracting or pulling out specific data from the text. Extraction types include:

- **Keyword extraction:** This technique helps identify relevant terms and expressions in the text and gives deep insights when combined with the above classification techniques. For example, one can analyze keywords in multiple tweets that have been labeled as positive or negative and then detect or extract words from those tweets that have been mentioned the maximum number of times. One can later use the extracted terms for automatic tweet classification based on the word type used in the tweets.
- **Entity extraction:** As discussed in the earlier example, this technique is used to identify and extract entities in text, such as names of individuals, organizations, places, and others. This method is typically helpful for customer support teams who intend to extract relevant information from customer support tickets automatically, including customer name, phone number, query category, shipping details, etc.

Semantic analysis techniques and tools allow automated text classification of tickets, freeing the concerned staff from mundane and repetitive tasks. In the larger context, this enables agents to focus on the prioritization of urgent matters and deal with them on an immediate basis. It also shortens response time considerably, which keeps customers satisfied and happy.

Moreover, granular insights derived from the text allow teams to identify the areas with loopholes and work on their improvement on priority. By using semantic analysis tools, concerned business stakeholders can improve decision-making and customer experience.

Incorporating semantic analysis into our project introduces a transformative dimension to the way we interact with and extract insights from textual data. By leveraging semantic analysis techniques, we empower our system to decipher the intricate layers of meaning concealed within text. This technology enables us to automatically categorize and classify text based on topics, providing users with a streamlined approach to understand the essence of the text they analyze. Moreover, semantic extraction techniques, such as keyword and entity extraction, equip our system with the ability to identify and extract specific information, enhancing the depth of analysis.

## 3.3 Document Summarization Using Latent Semantic Indexing:

### 3.3.1 Document summarization:

Document summarization is a method of getting the gist or summary of the document from the text. There are 2 methods of summarization. Abstractive summarization and Extractive summarization.

- **Extractive summarization** is basically creating a summary based on strictly what you get in the text. It can be compared to copying down the main points of a text without any modification to those points and rearranging the order of that points and the grammar to make more sense out of the summary (this is the approach used in the project).
- **Abstractive summarization techniques** tend to mimic the process of ‘paraphrasing’ from a text than just simply summarizing it. Texts summarized using this technique looks more human-like and produce more condensed summaries. These techniques are much harder to implement than extractive summarization techniques in general.

Latent Semantic Indexing or Latent semantic analysis is one of the extractive summarization methods.

### 3.3.2 Latent Semantic Indexing:

Latent semantic indexing is used to analyze the relationship between a set of **documents** and **terms** contained in the document. This uses a mathematical concept called **Singular Value Decomposition (SVD)** to compute set of matrices which give the similarity between the documents.

Latent Semantic Analysis uses the mathematical technique of Singular Value Decomposition (SVD) to identify the patterns of relationships between the terms and concepts. This is based on the principle that the words which occur in same contexts tend to have similar meanings.

### 3.3.3 Singular Value Decomposition (SVD):

Singular Value Decomposition is one of the dimensionality reduction techniques. SVD is the factorization of a matrix into 3 matrices. So if we have a matrix A it is represented by

$$A = U\Sigma V^T$$

A is a matrix of dimension  $m \times n$ . U is an orthogonal matrix of dimension  $m \times m$ .  $\Sigma$  is a diagonal matrix of dimension  $m \times n$  and V is an orthogonal matrix of dimension  $n \times n$ .

U is referred to a left singular vector,  $\Sigma$  is a singular value or eigen values, V is a right singular vector.

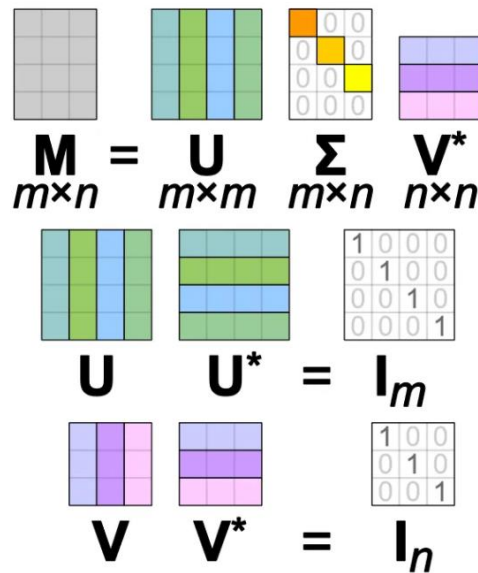


Figure 2: Singular Value Decomposition

(The following example is taken from the book Information Retrieval, Algorithms and Heuristics by David A Grossman and Ophir Frieder)

Consider the following 3 documents

d1: Shipment of gold damaged in a fire.

d2: Delivery of silver arrived in a silver truck.

d3: Shipment of gold arrived in a truck.

Now we create term document matrix

Terms ↓	d1 ↓	d2 ↓	d3 ↓
a	1	1	1
arrived	0	1	1
damaged	1	0	0
delivery	0	1	0
fire	1	0	0
gold	1	0	1
in	1	1	1
of	1	1	1
shipment	1	0	1
silver	0	2	0
truck	0	1	1

$A =$

If we perform the SVD on this matrix we get the following components as:

$$A = U\Sigma V^T$$

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix} \quad V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

Consider the 2 want to classify these documents into 2 topics



The columns of the  $U$  give the weights that would match each of the words belonging to the topic. For example, the word silver has more weightage to the second topic and the word truck has more weightage to 1st topic

In the matrix  $S$  or  $\Sigma$  the 1st diagonal entry gives weightage of the 1st topic and the 2nd diagonal entry gives weightage of the 2nd topic

The columns of matrix  $VT$  gives weightage to the documents belonging to each of the topics. Like the first 2 documents belong more to the 2nd topic and 3rd document belongs to 1st topic.

LSI basically helps in classifying the documents belonging to different topics. LSI is also used in search engines. As matrix  $U$  gives the relevance of the words to each of the topics and we also have the relevance of the documents belonging to each of the topics we can estimate the relevance of the word to each of the documents. In this way, we can get the weightage of the query word belonging to each of the documents.

### 3.3.4 Summarization using Latent Semantic Analysis

The LSI gives the weightage of the documents belonging to different topics. Summarization is done by selecting the top  $N$  documents from each topic depending on the weightage.

This gives a summarization where we get the higher weighted documents from each of the topics for the summary.

## 3.4 Keyword extraction:

### 3.4.1 Definition:

**Keyword extraction** (also known as *keyword detection* or *keyword analysis*) is a text analysis technique that automatically **extracts the most used and most important words and expressions from a text**. It helps summarize the content of texts and recognize the main topics discussed.

Keyword extraction uses machine learning artificial intelligence (AI) with NLP to break down human language so that it can be understood and analyzed by machines. It's used to find keywords from all

manner of text: regular documents and business reports, social media comments, online forums and reviews, news reports, and more.

### 3.4.2 Simple Statistical Approaches for Keyword extraction:

Using statistics is one of the simplest methods for identifying the main keywords and key phrases within a text.

There are different types of statistical approaches, including word frequency, word collocations and co-occurrences, and TF-IDF (Term Frequency-Inverse Document Frequency).

These approaches don't require training data to extract the most important keywords in a text. However, because they only rely on statistics, they may overlook relevant words or phrases that are mentioned once but should still be considered relevant. Let's take a look at some of these approaches in detail:

#### **Word Frequency:**

Word frequency consists of **listing the words and phrases that repeat the most within a text**. This can be useful for a myriad of purposes, from identifying recurrent terms in a set of product reviews, to finding out what are the most common issues in customer support interactions.

However, word frequency approaches consider documents as a mere '**bag of words**', leaving aside crucial aspects related to the meaning, structure, grammar, and sequence of words. Synonyms, for example, can't be detected by this keyword extraction method, dismissing very valuable information.

#### **Word Collocations and Co-occurrences:**

Also known as N-gram statistics, word collocations and co-occurrences **help understand the semantic structure of a text and count separate words as one**.

Collocations are words that frequently go together. The most common types of collocations are bi-grams (two terms that appear adjacently, like ‘customer service’, ‘video calls’ or ‘email notification’) and tri-grams (a group of three words, like ‘easy to use’ or ‘social media channels’).

Co-occurrences, on the other hand, refer to words that tend to co-occur in the same corpus. They don’t necessarily have to be adjacent, but they do have a semantic proximity.

### **TF-IDF (The one used in our project)**

TF-IDF stands for *term frequency–inverse document frequency*, a formula that measures how important a word is to a document in a collection of documents.

This metric calculates the number of times a word appears in a text (*term frequency*) and compares it with the *inverse document frequency* (how rare or common that word is in the entire data set).

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

$$IDF = \log \left( \frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}} \right)$$

Multiplying these two quantities provides the TF-IDF score of a word in a document. The higher the score is, the more relevant the word is to the document.

$$TF - IDF = TF * IDF$$

TD-IDF algorithms have several applications in machine learning. In fact, search engines use variations of TF-IDF algorithms to rank articles based on their relevance to a certain search query.

When it comes to keyword extraction, this metric can help you identify the most relevant words in a document (the ones with the higher scores) and consider them as *keywords*. This can be particularly useful for tasks like tagging customer support tickets or analyzing customer feedback.

In many of these cases, the words that appear more frequently in a group of documents are not necessarily the most relevant. Likewise, a word that appears in a single text but doesn't appear in the remaining documents may be very important to understand the content of that text.

## 4 Methodology:

### 4.1 Conception:

#### 4.1.1 User case Diagram:

A **Use Case Diagram** is a type of visual **representation** used in software engineering and system design to illustrate the **interactions between users** (actors) and **a system**.

In the context of our project, the system we refer to is the "**Analyzio**" system (*Analyzio* is the name we gave to our website), which encompasses both the website and the underlying algorithms for semantic and partial analysis. As for the actors, they exclusively represent the website's users. Here is our comprehensive use case diagram, showcasing the primary interactions within our project:

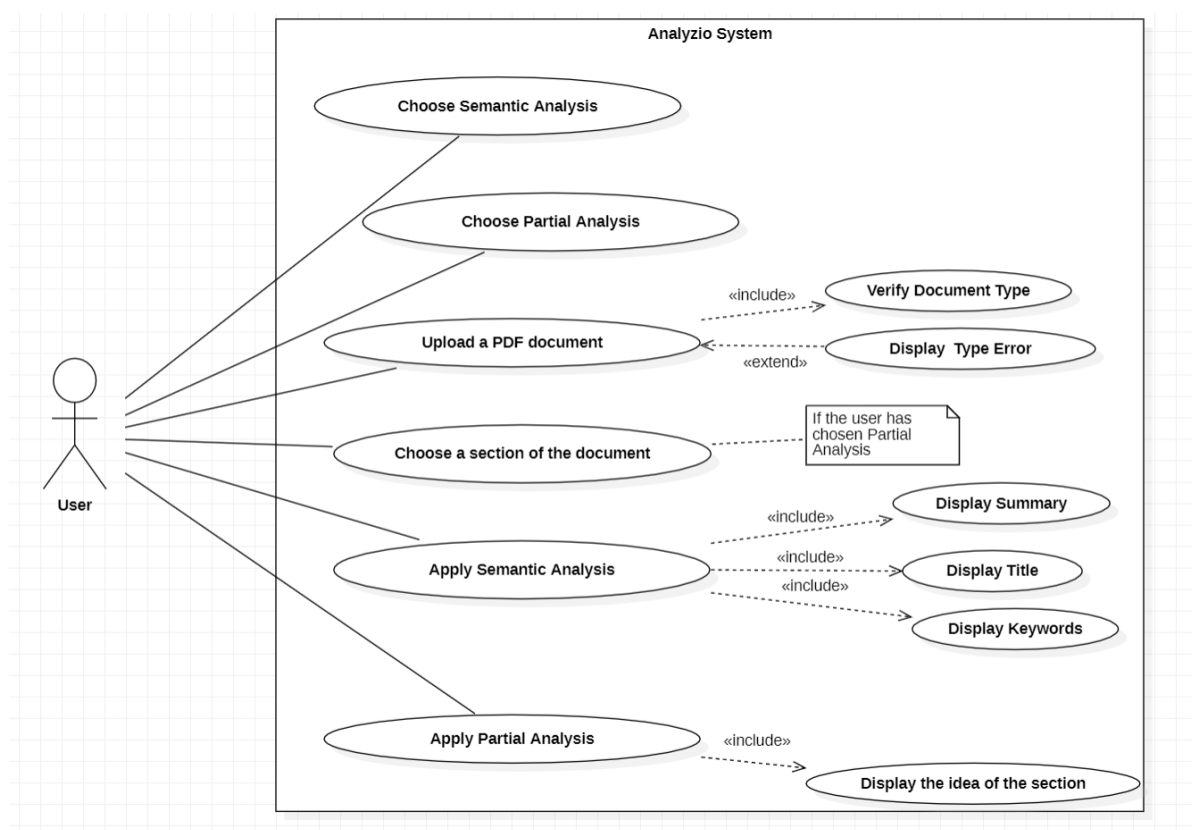


Figure 3: Use Case Diagram

### 4.1.2 Class Diagram:

A Class Diagram is a type of **static structure diagram** that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

In the context of our project, we show the class diagram representing the website, which comprises one Semantic Analyzer and one Partial Analyzer. These analyzers generate the document analysis.

Notably, the user in this project is an external beneficiary of our services without any identification.

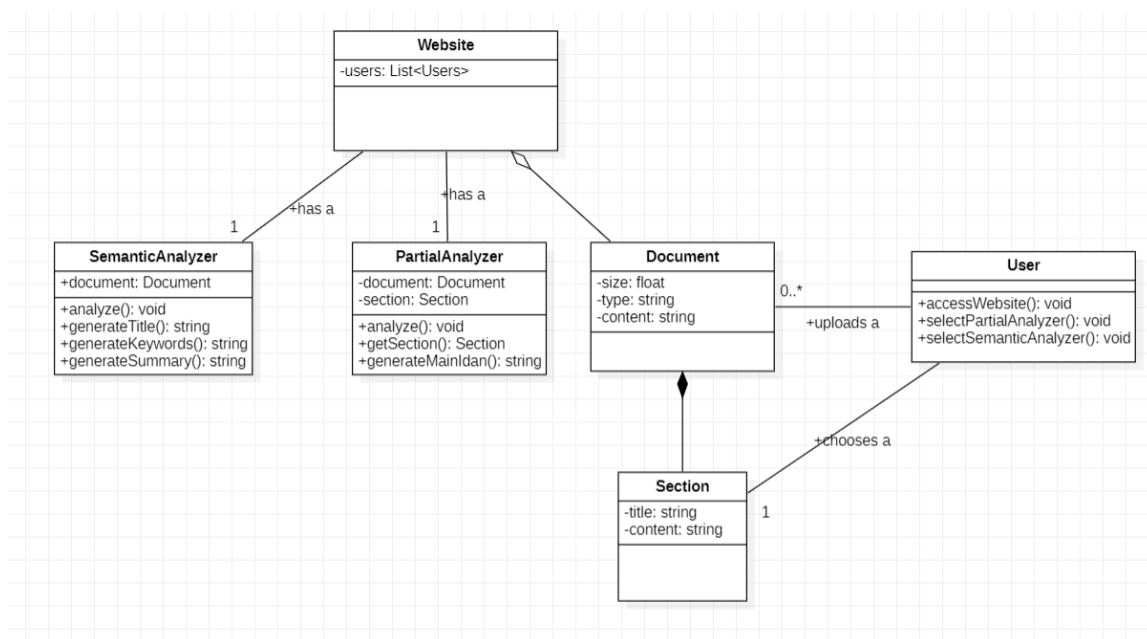


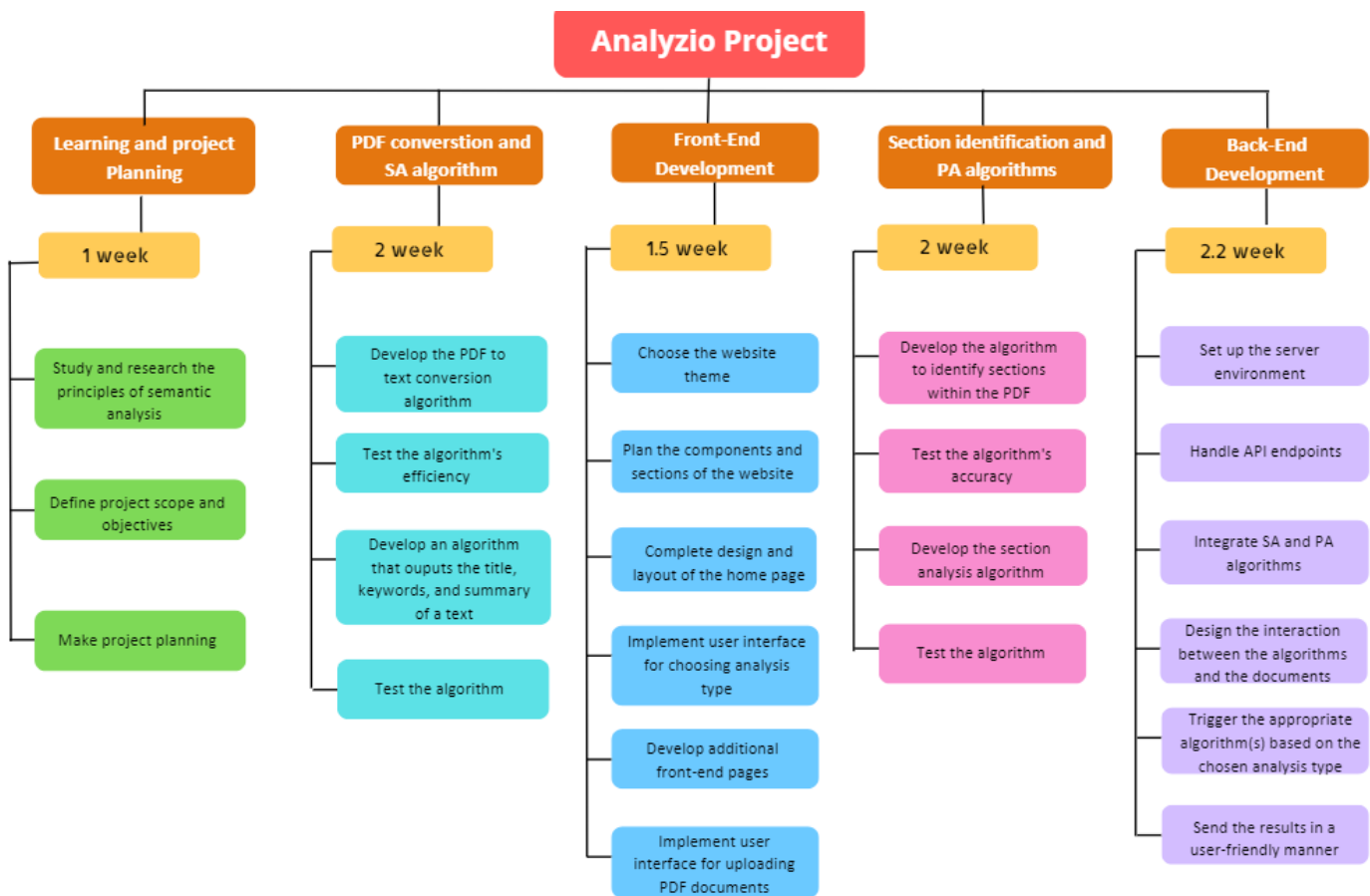
Figure 4: Class Diagram

### 4.1.3 Planification: Work Breakdown Structure (WBS)

A **WBS** breaks down the project into smaller, manageable elements, making it easier to plan, schedule, execute, and monitor the project's progress.

Here is the WBS of our project; the Analyzio system:

Table 1: The work breakdown structure of our project



## 4.2 Website Architecture:

The system architecture of *Analyzio* has been carefully designed to provide efficient and scalable semantic analysis, keyword extraction, and text summarization services to users. This section provides an overview of the architectural components and their interactions.

### 4.2.1 Architectural style:

Our website is built using a **client-server architectural style**. This style separates the client (frontend) and server (backend) components, enabling efficient communication and scalability.

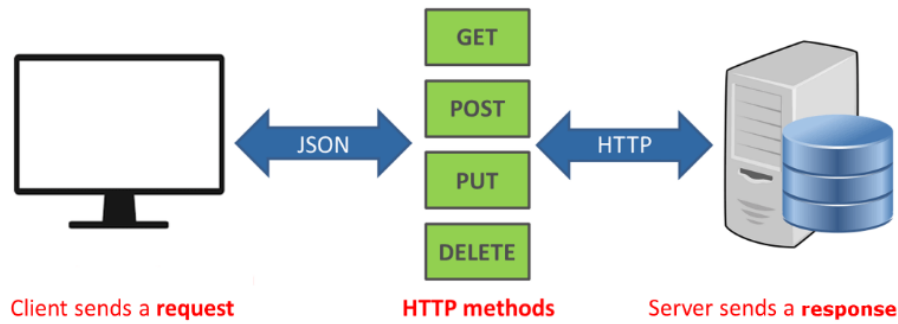


Figure 5: System architecture

#### 4.2.2 Client-Side Components:

**Frontend (React):** The client side of Analyzio is developed using the React library, which enables the creation of a dynamic and responsive user interface. React components are responsible for rendering the web pages and facilitating user interactions.

**User Interface (UI):** The UI includes components for uploading PDF documents, selecting analysis preferences (semantic analysis or partial analysis), displaying analysis results, and interacting with the backend via API requests.

#### 4.2.3 Server-Side Components:

**Backend (Node.js):** The server-side of Analyzio is built on Node.js, a powerful JavaScript runtime environment known for its non-blocking I/O model. Node.js allows the system to handle concurrent user requests efficiently.

**Express.js (Web Application Framework):** Express.js, a web application framework for Node.js, is used to set up routing, define API endpoints, and manage incoming requests and responses.

**APIs (Application Programming Interfaces):** Analyzio's backend exposes a set of RESTful APIs that the frontend uses to communicate with the server. These APIs enable functionalities such as uploading PDF documents, initiating analysis tasks, and retrieving analysis results.

#### 4.2.4 Data Management:

**Document Storage:** *Analyzio* employs a **file-based storage approach** for uploaded documents. When a user uploads a PDF document through the frontend interface, the document is transmitted

to the backend server, where it is stored in a dedicated folder named "uploads." This file-based storage system allows for straightforward document retrieval and analysis without the need for a traditional database.

By opting for file-based document storage, *Analyzio* streamlines the process of document handling and analysis.

#### 4.2.5 Communication:

Our architecture relies on **RESTful APIs (Application Programming Interfaces)** for communication between the frontend and backend. When a user interacts with the frontend, it sends HTTP requests to specific API endpoints on the backend. The backend processes these requests, interacts with the database as needed, and returns data to the frontend.

#### 4.3.6 Purpose:

The chosen website architecture aims to provide an efficient, responsive, and scalable platform for users to access and interact with our system. The separation of frontend and backend components allows for easier development, maintenance, and future enhancements. RESTful APIs ensure seamless communication. Overall, our architecture aligns with our project goals of delivering a high-quality user experience and robust functionality.

### 4.3 Algorithms:

#### 4.3.1 Title extraction algorithm:

(The code of this part is in the python file Server>Algo>Title.py’

Our title extraction algorithm is designed to automatically retrieve the title from a PDF document name. The process involves the following steps:

**Input PDF Document:** The algorithm takes as input a PDF document for which the title needs to be extracted.



**File Name Parsing:** To streamline the title extraction, we begin by parsing the file name of the PDF document. The file name typically contains information about the document, including its title.

**Title Identification:** The parsed file name is then processed to isolate the title. We achieve this by splitting the file name based on specific delimiters, such as slashes or periods, to extract the relevant title information.

**Title Extraction:** The extracted title is then returned as the output of the algorithm, providing users with a clear and concise representation of the document's subject.

### 4.3.2 Keyword Extraction Algorithms:

Keyword extraction is a pivotal component of our project, aimed at identifying and isolating the most important terms or phrases within a given document. These extracted keywords serve as valuable insights, enabling users to grasp the core themes and content of the text efficiently.

The algorithms below are in the python scripts '*Function.py*' and '*Keywords.py*' in *Server>Algos>SemanticAnalysis* folder of our code.

**Algorithm Description:** Our chosen method for keyword extraction utilizes a combination of techniques from the scikit-learn library. Below is an overview of the steps involved in the keyword extraction process:

**Text Preprocessing:** ( *preprocess\_data\_kw()* function) Before extracting keywords, we preprocess the text data. This involves **tokenization**, which breaks the text into individual words, and **the removal of common stop words** to focus on content-carrying terms.

**Term Frequency-Inverse Document Frequency (TF-IDF):** (*extract\_keywords()* function) We employ the TF-IDF method, a powerful technique for evaluating the importance of terms within a document. TF-IDF stands for Term Frequency-Inverse Document Frequency and is widely used in natural language processing.

**Vectorization:** (*extract\_keywords()* function) To compute TF-IDF scores, we first create a term-document matrix using the *CountVectorizer* from scikit-learn. This matrix represents the frequency of each term in the document.

**TF-IDF Transformation:** (*extract\_keywords()* function) The term-document matrix is then transformed using the *TfidfTransformer*, yielding TF-IDF scores for each term in the document.

**Keyword Selection:** (*extract\_keywords()* function) Finally, we select the top keywords based on their TF-IDF scores. The keywords with the highest scores are considered the most important and are extracted as the final result.

### 4.3.3 Text summarization algorithms:

Text summarization is a crucial component of our project, enabling the extraction of concise and meaningful summaries from large documents. This section outlines the algorithms and methods we employ for text summarization, a process essential for distilling key information from lengthy texts.

The steps explained below are in the python scripts '*Function.py*' and '*Summary.py*' in *Server>Algos>SemanticAnalysis* folder of our code.

**Latent Semantic Analysis (LSA):** Our chosen approach for text summarization hinges on Latent Semantic Analysis (LSA). LSA is a natural language processing technique renowned for its ability to uncover latent semantic relationships within extensive text corpora. By applying LSA, we aim to reveal the underlying structure and context of the text, which facilitates the extraction of coherent and informative summaries.

**Preprocessing for Summarization:** (The function *preprocess\_data()* in *Function.py* file) Before subjecting the text to LSA, we perform essential preprocessing steps to enhance the quality of the summarization. These steps include **tokenization**, which breaks the text into individual words or tokens, **removal of common stop words**, and **stemming**, which reduces words to their root forms. This preprocessing ensures that the text is prepared for meaningful analysis.

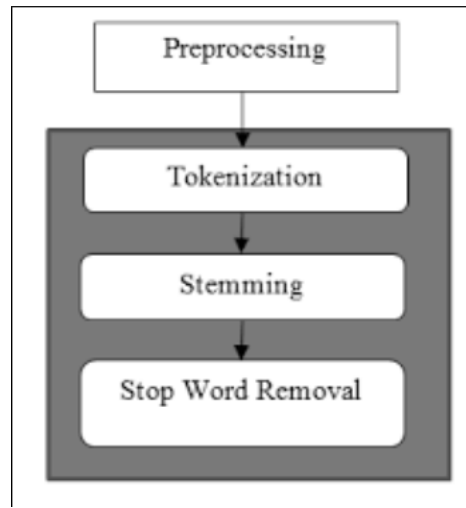


Figure 6: Preprocessing for summarization

**Corpus Preparation:** (*prepare\_corpus()* in *Function.py*) The first step in our text summarization process involves preparing the text corpus. To achieve this, we create a **term dictionary that maps words to numerical identifiers**. Additionally, we convert the list of documents into a Document Term Matrix (DTM), a crucial data structure for LSA.

	words1	words2	words3	words4	words5
doc1	0	0	1	0	0
doc2	2	0	1	1	0
doc3	0	0	1	1	0
doc4	0	0	1	1	1

Figure 7: Document Term Matrix

**LSA Model Creation:** (*create\_gensim\_lsa\_model()* function in *Function.py*) With the term dictionary and DTM in place, we proceed to create the LSA model. This model captures the latent semantic structure of the text by analyzing the relationships between terms. The number of topics to be extracted from the text is specified during this step.

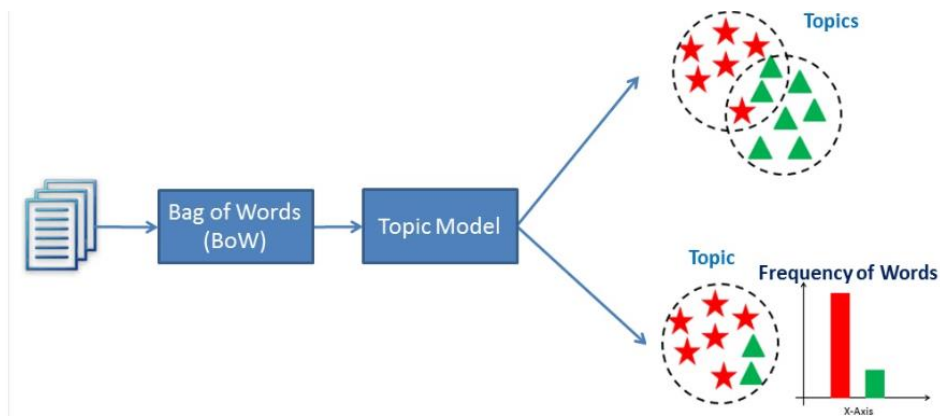


Figure 8: LSA model schema

**Sorting Vectors:** (*sort\_vectors()* function) To identify the most significant sentences for the summary, we sort the document vectors. This sorting is done in descending order of weightage, ensuring that the most contextually relevant sentences are given priority.

**Selecting Top Sentences:** (*selectTopSent()* function) The selection of sentences for inclusion in the summary is a critical step. We utilize a specialized function, *selectTopSent*, to determine the top sentences. This function considers various factors, including the importance of sentences within the latent semantic space.

**Summary Generation:** (*summary()* function) Finally, the selected top sentences are used to construct the summary. These sentences are carefully chosen to encapsulate the essence of the document concisely. The result is a coherent and meaningful summary that distills the key information from the original text.

#### 4.3.4 Bridges: Linking Algorithms to the Server:

In our Analyzio project, the effective integration of algorithms with the server is vital to provide users with seamless and efficient document analysis. To achieve this integration, we've developed bridge scripts that act as intermediaries, enabling communication between the server and the Python-based algorithms responsible for title extraction, keyword extraction, and text summarization.

All bridges are in Server>Algos>SemanticAnalysis.

### **Title Extraction Bridge:** (titleBridge.py)

The first bridge script is responsible for extracting document titles from PDF files. When a user uploads a PDF document, the server triggers this bridge script, which, in turn, invokes the Title Extraction Algorithm. The algorithm identifies and returns the document's title. If successful, this title becomes a key component in organizing and presenting the document's content to the user.

### **Keyword Extraction Bridge:** (keywordBridge.py)

Our Keyword Extraction Bridge script connects the server to the Python-based Keyword Extraction Algorithm. This algorithm plays a crucial role in identifying and extracting significant keywords from uploaded PDF documents. When a user requests keyword extraction, the server utilizes this bridge script to invoke the algorithm. The extracted keywords provide valuable insights into the document's content and aid in categorizing and summarizing the material.

### **Summarization Extraction Bridge:** (bridge.py)

Text summarization is a fundamental feature of Analyzio. To provide users with concise document summaries, we employ the Summarization Extraction Bridge script. When a user selects the summarization option, the server triggers this script, which communicates with the Python-based Summarization Extraction Algorithm. This algorithm processes the document and generates a condensed summary, allowing users to quickly grasp the document's key points.

## **4.4 PDF Document Processing:**

In the context of the *Analyzio* project, one of the fundamental tasks is to convert PDF documents into text format. This is a crucial step as it enables the subsequent analysis of the document's content. The PDF document processing is achieved through a custom Python script, which utilizes the **PyPDF2 library**. The PDF document processing phase consists of two key components:

### 4.4.1 Conversion Algorithm:

The PDF-to-text conversion algorithm (*TextConverter.py* file) is responsible for extracting the textual content from PDF files. This algorithm is instrumental in preparing the input for various text analysis processes. It operates as follows:

- **Identification of Document Sections:** The algorithm can identify specific sections within the PDF document. For instance, it can detect the presence of sections like 'Abstract,' 'Introduction,' 'Conclusion,' or 'References' based on predefined keywords. This feature is useful for segmenting the document effectively.
- **Text Extraction:** Once the document sections are identified, the algorithm extracts the text content within these sections. It carefully excludes any content before the 'Abstract' section and after the 'Conclusion' or 'References' sections.

### 4.4.2 Text Preprocessing:

Text preprocessing (*load\_data()* and *preprocess\_data()* functions in *Function.py*) is an essential aspect of PDF document processing, as it ensures that the extracted text data is clean and ready for subsequent analysis. This process includes:

- **Tokenization:** The extracted text is tokenized, meaning it is divided into individual words or tokens. This step is crucial for further analysis.
- **Stop words Removal:** Common English stop words (e.g., 'and,' 'the,' 'in') are removed from the text to focus on significant content.
- **Stemming:** Words are stemmed to their root form, reducing variations and ensuring consistent word forms. For instance, "running" and "ran" would both be reduced to "run."

## 4.5 Tools:

This section provides an overview of the programming languages, development environment, and version control tools employed in the project, highlighting their roles and contributions to the overall development process.

### 4.5.1 Programming Languages:

Programming languages are the foundation of any software project, determining the system's architecture, functionality, and capabilities. In this project, we leveraged a combination of languages to craft a powerful and versatile system.

#### Python: The Engine Behind Advanced Algorithms

- **Definition:** Python is a versatile, high-level programming language known for its simplicity and readability. It's widely used for web development, data analysis, artificial intelligence, and more.
- **Role in the project:**
  - **Algorithm Implementation:** Python played a pivotal role in implementing advanced algorithms for semantic analysis, partial analysis, keyword extraction, and text summarization.
  - **PDF Document Processing:** Python was used to convert PDF documents into text, enabling further analysis.
  - **Natural Language Processing (NLP):** Python's NLP libraries facilitated tasks such as text tokenization, stopwords removal, and stemming, enhancing the accuracy of our system.
- **Advantages:**
  - **Robust Ecosystem:** Python boasts an extensive ecosystem of libraries and frameworks, accelerating the development process.
  - **NLP Prowess:** Python is renowned for its natural language processing (NLP) capabilities, making it ideal for semantic analysis and text processing.
  - **Community Support:** The Python community is vibrant and collaborative, offering a wealth of resources, documentation, and support.

#### JavaScript (JS) and JSX: Powering Dynamic Frontend and Backend

JavaScript (JS) and its extension, JSX, are versatile languages vital to modern web development. In our project, we strategically employed these languages in both frontend and backend aspects to ensure a comprehensive and efficient system.

## **JavaScript (JS):**

- **Definition:** JavaScript is a core web development language renowned for adding interactivity and dynamic behavior to web pages. It executes directly in web browsers, enhancing frontend responsiveness.
- **Role in the project:** in the backend, JavaScript handled various operations, including data processing, server-side scripting, and routing. This versatility allowed for efficient server management.
- **Advantages:**
  - **Client-Side Execution:** JS operates within users' browsers, reducing server load and improving frontend performance.
  - **Cross-Browser Compatibility:** It enjoys widespread support among major web browsers, ensuring a consistent user experience.
  - **Rich Ecosystem:** JavaScript boasts an extensive library of frameworks and libraries, simplifying frontend and backend development tasks.

## **JSX ( JavaScript Extension):**

- **Definition:** JSX is a JavaScript extension mainly used in conjunction with React, a popular JavaScript library. It enables the creation of dynamic and reusable user interface components.
- **Role in the project: Dynamic User Interfaces:** JSX, within React components, played a pivotal role in constructing dynamic and reusable user interface elements. This streamlined the frontend development process.
- **Advantages:**
  - **React Integration:** JSX seamlessly integrates with React, simplifying the creation of interactive user interfaces within this library.
  - **Component Reusability:** JSX encourages the development of reusable UI components, saving time and effort when building complex frontend elements.



## 4.5.2 Libraries and Frameworks:

Our project harnessed a range of powerful libraries and frameworks, each serving specific purposes to optimize functionality and enhance overall efficiency. These libraries played a crucial role in various aspects of the project, from frontend development to advanced natural language processing (NLP) algorithms.

### 1. React:

**Definition:** React is a JavaScript library that facilitates the creation of dynamic user interfaces for web applications. It allows developers to build reusable UI components that efficiently update and render in response to user interactions.

**Advantages:** React simplifies complex UI development, enhances code maintainability, and improves application performance through its Virtual DOM. Its extensive ecosystem of community-contributed libraries and components streamlines frontend development.

**Purpose:** React was instrumental in our frontend development, enabling us to build interactive and responsive user interfaces. By breaking down the UI into reusable components, React enhanced the user experience and ensured efficient rendering of dynamic content.

### 2. PyPDF2:

**Definition:** PyPDF2 is a Python library designed for working with PDF documents. It provides functions to extract text, merge, split, and manipulate PDF files programmatically.

**Advantages:** PyPDF2 simplifies PDF handling in Python, allowing developers to extract text from PDFs for further analysis. Its functionality extends to tasks like watermarking, merging, and splitting PDFs.

**Purpose:** PyPDF2 played a pivotal role in our project by converting PDF documents into text, a fundamental step in preparing documents for semantic analysis and keyword extraction.

### 3. NLTK (Natural Language Toolkit):

**Definition:** NLTK is a comprehensive Python library for natural language processing (NLP) and text analysis. It offers tools for tasks such as tokenization, stemming, and part-of-speech tagging.

**Advantages:** NLTK provides a rich set of linguistic data and algorithms, making it a valuable resource for NLP projects. Its user-friendly interface simplifies complex text processing tasks.

**Purpose:** NLTK was essential in our semantic analysis and keyword extraction processes. It enabled us to preprocess text data, including removing stopwords and stemming, to prepare it for analysis.

### 4. Gensim:

**Definition:** Gensim is a Python library tailored for topic modeling and document similarity analysis. It is often used for building topic models like Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA).

**Advantages:** Gensim is optimized for large text corpora and provides efficient implementations of topic modeling algorithms. It is known for its scalability and ease of use in developing topic-based applications.

**Purpose:** In our project, Gensim facilitated the implementation of LSA models for text summarization and document analysis. It played a vital role in extracting meaningful information from the textual data.

### 5. CountVectorizer and TfidfTransformer (from Scikit-Learn):

**Definition:** CountVectorizer and TfidfTransformer are components of Scikit-Learn, a popular Python library for machine learning and data analysis. CountVectorizer converts text data into numerical form, while TfidfTransformer calculates TF-IDF scores.

**Advantages:** Scikit-Learn provides a wide range of machine learning tools for various data analysis tasks. CountVectorizer and TfidfTransformer are essential for text data preprocessing and feature engineering.

**Purpose:** In our project, these Scikit-Learn components were utilized to convert text data into a numerical format and calculate TF-IDF scores, which are vital for keyword extraction and text analysis.

## 6. Node.js ( Backend):

**Definition:** Node.js is a runtime environment that allows executing JavaScript code on the server-side. It is known for its event-driven, non-blocking I/O model, making it efficient for building scalable network applications

**Advantages:** Node.js offers high performance, scalability, and a vast ecosystem of libraries and packages. Its single programming language (JavaScript) for both frontend and backend simplify development.

**Purpose:** While not a library itself, Node.js served as our backend runtime environment. It enabled us to handle server-side logic efficiently, ensuring seamless communication between the frontend and backend components of our system.

## 7. Express.js (Backend):

**Definition:** Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It simplifies the creation of server-side applications and APIs.

**Advantages:** Express offers a lightweight, fast, and unopinionated framework for building web servers and APIs. It provides tools for routing, middleware, and handling HTTP requests and responses efficiently.

**Purpose:** In our project, Express served as the backend framework, enabling us to create a robust server-side application. It handled routing, middleware, and communication between the frontend and the server, ensuring smooth data flow and interaction.

#### 8. Multer:

**Definition:** Multer is a middleware for handling multipart/form-data, commonly used for file uploads in web applications.

**Advantages:** Multer simplifies the process of handling file uploads, including file storage and naming. It also integrates easily into Express.js applications as middleware.

**Purpose:** Multer is used to manage file uploads from the frontend, specifically for PDF documents.

#### 9. CORS:

**Definition:** CORS is a security feature implemented by web browsers that allows or restricts web applications running at one origin (domain) to make requests for resources from another origin.

**Purpose:** The 'cors' middleware is used to handle Cross-Origin Resource Sharing, allowing your server to respond to requests from different origins when interacting with your frontend hosted on a different domain.

### 4.5.3 Development environment:

**Visual Studio Code (VS Code):** Visual Studio Code served as the primary code editor for our project. It was used extensively for writing and editing code across various parts of the project, including the frontend, backend, and algorithm implementation. Its feature-rich environment, including extensions for React, Node.js, and Python, made it an excellent choice for code development and debugging.

**Google Colab and Jupyter Notebook:** Google Colab and Jupyter Notebook were predominantly employed for the development and testing of our natural language processing (NLP) and semantic analysis algorithms. These interactive environments allowed us to write Python code for tasks such as text preprocessing, training machine learning models, and evaluating algorithm performance. Colab's cloud-based execution capabilities were especially valuable for resource-intensive NLP tasks.

**GitHub:** GitHub served as the central version control repository for our project. It was used for hosting and managing the entire source codebase, including both frontend and backend components. GitHub enabled collaborative development by allowing team members to clone, commit changes, and create pull requests. It also served as a platform for issue tracking and code review, ensuring that project development progressed smoothly and efficiently.

## 5 Front-end development:

Front-end development is a critical aspect of our project, responsible for creating an engaging and user-friendly interface for our web application. In this section, we will discuss the technologies, components, design principles and more about our front-end development.

### 5.1 Logo:

#### 5.1.1 Design:

The logo for the project represented as '[Analyzio]' in blue, has been meticulously designed to reflect the essence of the application. Its design comprises a clean and modern typeface with a vibrant blue color, signifying **professionalism** and **trustworthiness**. The choice of blue is intentional, as it is often **associated with reliability and intelligence**, aligning with the project's core goal of providing intelligent document analysis.

The logo consists of the word "Analyzio" in a blue, sans-serif font, enclosed within blue square brackets. The brackets are slightly larger than the text, creating a frame around the word.

*Figure 9: Project logo*

### 5.1.2 Significance:

The logo serves as a visual identifier of the project and plays a crucial role in creating a memorable brand image. It encapsulates the project's commitment to **delivering accurate and insightful document analysis**. The brackets around 'Analyzio' emphasize **a sense of completeness and thoroughness**, conveying that the system comprehensively analyzes documents.

## 5.2 Theme:

### 5.2.1 Color scheme:

The project's theme is built upon a carefully curated color scheme that plays a pivotal role in defining its visual identity. The primary colors used in the project are :

- **Purplish Pink:** This warm and inviting color is strategically incorporated to evoke creativity and curiosity. It highlights key elements and calls attention to important features.



*Figure 10: Purplish pink color*

- **Daisy Bush:** A vibrant and energetic hue, Daisy Bush adds dynamism to the project's visuals. It infuses energy and excitement into the user experience.



*Figure 11: Daisy Bush color*

- **Artyclick Sky Blue:** This calm and trustworthy shade of blue is prominently featured throughout the project. It instills a sense of reliability and professionalism, aligning with the project's goal of providing accurate document analysis.



*Figure 12: ArtClick Sky Blue color*

### 5.2.2 Typography:

Typography plays a pivotal role in defining the project's visual identity. The project employs the "Poppins" font family, a sans-serif typeface known for its modern, clean, and highly readable characteristics. This choice of typography offers several advantages:

- **Readability:** "Poppins" is carefully selected for its exceptional readability, making it suitable for presenting textual content to users. Its clear and consistent letterforms ensure that users can easily consume information.
- **Professionalism:** The sans-serif style of "Poppins" conveys a sense of professionalism and modernity. This aligns with the project's goal of providing a reliable and contemporary user experience.
- **Versatility:** "Poppins" offers a wide range of font weights and styles, allowing for versatility in design. It enables the differentiation of headings, subheadings, and body text, enhancing the hierarchy and visual appeal of the project.
- **Cross-Platform Compatibility:** The use of "Poppins" as a web font ensures cross-platform compatibility. Whether users access the project on a desktop, tablet, or mobile device, the typography remains consistent and legible.
- **Aesthetics:** Beyond its readability and functionality, "Poppins" adds a touch of elegance to the project's design. Its well-crafted letterforms contribute to a polished and visually pleasing interface.

### 5.2.3 Visual elements:

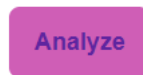
In the project's user interface design, visual elements are essential to communicate functionality, enhance user interaction, and maintain a cohesive look and feel.

- **Primary Action Button:** This button, adorned with a vibrant purplish-pink color with the sky-blue color in the background, serves as the focal point for primary user actions such as initiating analysis or proceeding with key interactions. Its distinct color draws attention, ensuring users can easily identify and engage with core functions.



*Figure 13: primary action button example*

- **Downloading/ Analyzing Button:** For downloading and analyzing documents, a contrasting purplish pink color written in the daisy bush color is employed. This provides a clear visual distinction from the primary action button, aiding users in distinguishing between different types of interactions.



*Figure 14: Analyze button*

- **Upload Button:** To encourage file uploads, a prominent sky-blue button is used. Its bright and inviting color scheme invites users to initiate the upload process, making it a user-friendly and accessible feature.



*Figure 15: Upload button*

These visual elements, although simple in design, play a vital role in guiding users through the project's interface. By using distinct colors and design conventions for each button type, users can quickly grasp their respective functions and engage with the platform efficiently. The minimalist approach to visual elements ensures a clean and user-centric design, allowing users to focus on the core functionalities of the project.



## 5.3 React components:

In this section, we'll introduce and describe the key React components utilized in the project. These components play a vital role in shaping the user interface and enabling various functionalities.

All of these components are stored in the project's '*components*' folder. Each component consists of two files: one containing **the JSX (HTML+JS) code**, and the other containing the corresponding **stylesheet (CSS)**."

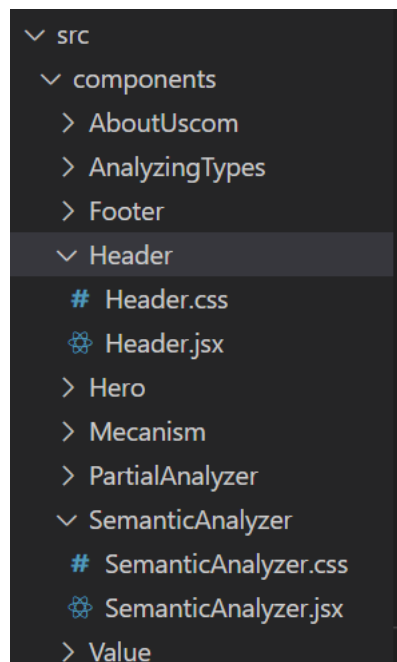


Figure 16: Project components

### 1. Header Component:

- **Description:** The Header component is responsible for rendering the project's navigation menu at the top of the webpage.
- **Functionality:** It allows users to navigate between different sections and pages of the project.
- **Usage:** Found at the top of each page, providing easy access to various sections of the project.

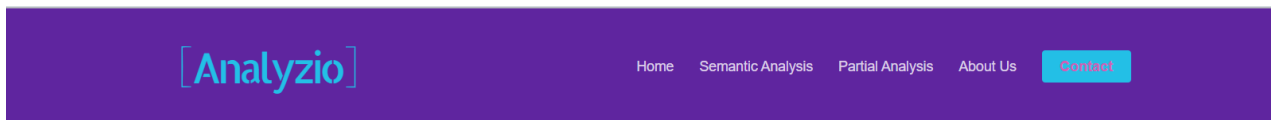


Figure 17: Header

## 2. Footer Component:

- **Description:** The Footer component represents the footer section of the project, providing essential contact information and a brief description.
- **Functionality:** It offers users quick access to contact details and serves as a branding element.
- **Usage:** Located at the bottom of each page, offering contact information and reinforcing branding.



Figure 18: Footer

## 3. Hero Component:

The Hero component serves as the project's main introduction, capturing the user's attention and providing an overview of the project's core features and benefits.

- **Description:** The Hero component is designed to make a compelling visual and textual introduction to the project.
- **Functionality:** It introduces users to the project's purpose, emphasizing the efficiency of PDF analysis, and encouraging them to explore further.
- **Usage:** Located prominently at the top of the main page, it sets the tone for the entire user experience

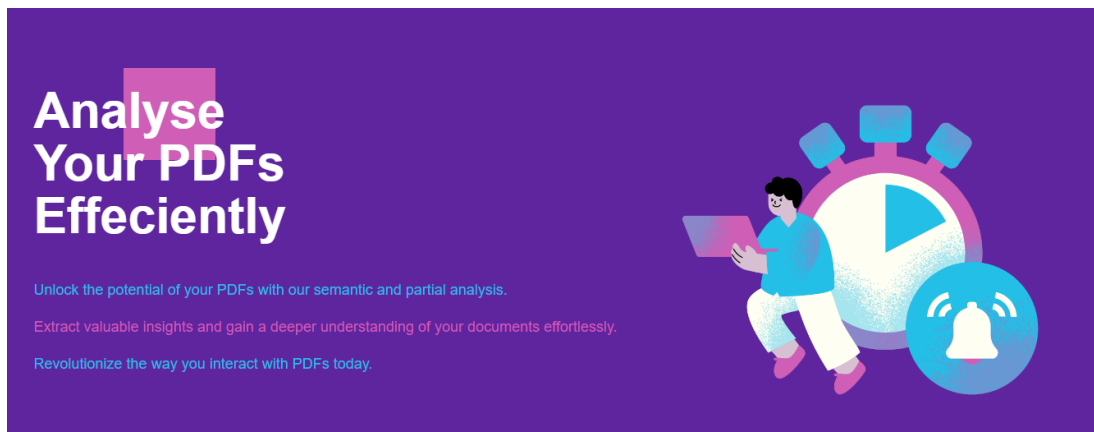


Figure 19: Hero component

#### 4. Analyzing Types Component:

- **Description:** The Analyzing Types component is responsible for displaying document analysis options.
- **Functionality:** It enables users to choose between two types of document analyses: Semantic Analysis and Partial Analysis.
- **Usage:** Typically found on the project's main page, helping users select the analysis type they require.



Figure 20: Analyzing types in the main page

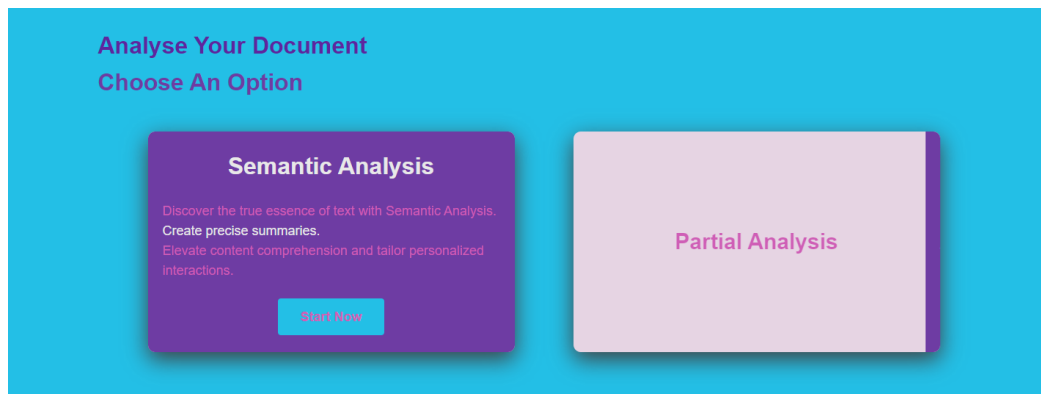


Figure 21: Analyzing types when hovering on Semantic Analysis

## 5. Value Component:

- **Description:** The Value component showcases the project's core values and goals.
- **Functionality:** It presents the value proposition and key benefits offered by the project.
- **Usage:** Found on the main page, providing insight into the project's value to users.

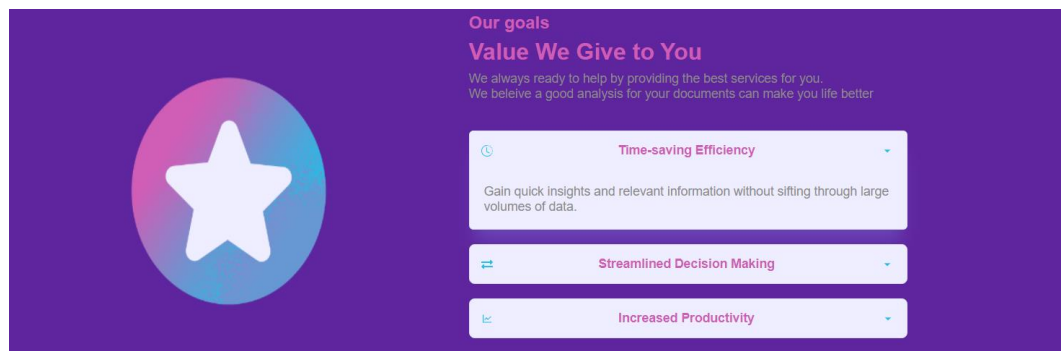


Figure 22: Value component

## 6. Mechanism Component:

- **Description:** The Mechanism component explains how the project works in a straightforward manner.
- **Functionality:** It describes the process users need to follow to utilize the project effectively.
- **Usage:** Commonly found on the main page, guiding users through the project's functionality.

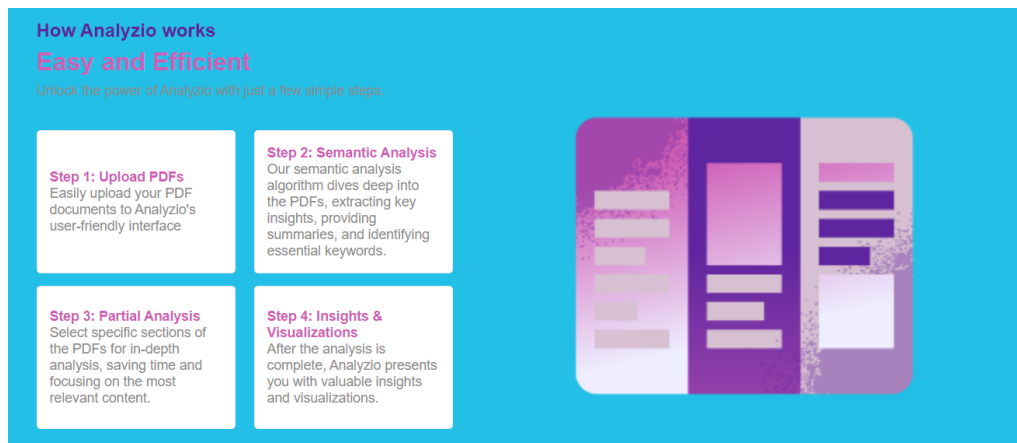


Figure 23: Mechanism component

## 7. Semantic Analyzer Component:

- **Description:** The Semantic Analyzer component represents the user interface for semantic document analysis.
- **Functionality:** It allows users to upload PDF documents, initiate analysis, and retrieve results.
- **Usage:** Accessed through the project's navigation menu, this component facilitates semantic analysis.

**Semantic Analyzer**  
 Analyze your documents easily and rapidly

Semantic Analyzer [English] [French] Summary Length [Slider]

Upload PDF [Analyze] Download

Title:  
 Keywords:  
 Summary:

Figure 24: Semantic Analyzer

## 8. Partial Analyzer Component:

- **Description:** The Partial Analyzer component serves as the user interface for partial document analysis.
- **Functionality:** It enables users to upload PDFs, select specific sections for analysis, and retrieve relevant insights.
- **Usage:** Accessed through the project's navigation menu, this component supports partial analysis requirements.

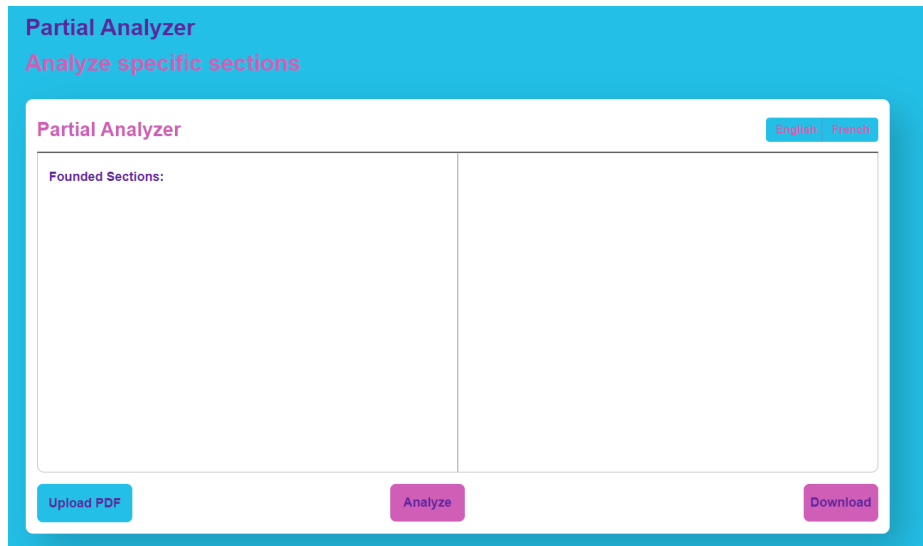


Figure 25: Partial Analyzer

These React components collectively shape the user interface and functionality of the project. They are carefully designed and implemented to provide a seamless and user-friendly experience for individuals seeking document analysis services.

## 5.4 User interfaces:

In the context of our web application, user interfaces are pivotal elements that enable users to interact with our system efficiently and intuitively. This section provides an overview of the various interfaces available, focusing on their functionality and design principles.

Our application comprises several distinct user interfaces, each serving a specific purpose. Here's an overview of these interfaces:

### 5.4.1 Types of analysis interface:

This interface provides users with two options for document analysis: Semantic Analysis and Partial Analysis.



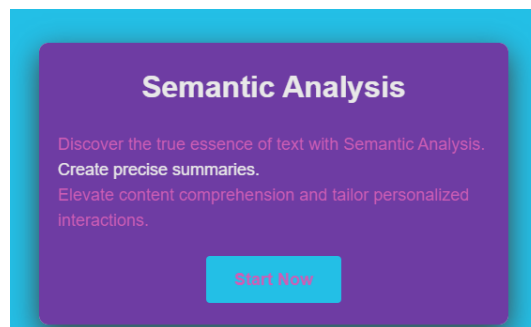
Figure 26: Home interface

– **Semantic Analysis card:**

When a user hovers over the Semantic Analysis card, it elegantly transforms into a vibrant purple card. This card serves as an informative gateway to Semantic Analysis. Prominently featured on the card is a "Start Now" button, inviting users to initiate a Semantic Analysis. A simple click on this button seamlessly redirects users to the dedicated Semantic Analysis page, where they can delve into comprehensive document analysis.



*Figure 27: Semantic Analysis card*



*Figure 28: Semantic Analysis card while hovering*

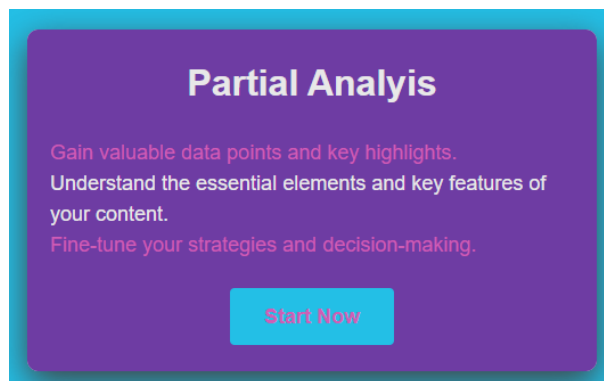
– **Partial Analysis Card:**

Much like the Semantic Analysis card, the Partial Analysis card also metamorphoses into an engaging purple card upon user interaction. This card succinctly elucidates the concept of Partial Analysis, highlighting its significance and advantages. It too boasts a "Start Now" button, beckoning users to embark on their Partial Analysis journey. Clicking this button efficiently guides users to the dedicated Partial Analysis page, where they can explore and select specific sections of their documents for in-depth scrutiny.





*Figure 29: Partial Analysis card*



*Figure 30: Partial Analysis card while hovering*

### 5.4.2 Semantic Analysis interface:

This interface includes the following components:

- **PDF Content Display:** Upon uploading a PDF document via the "Upload PDF" button, the content of the document is displayed in a dedicated area within the interface. This component allows users to review the document's content before analysis.

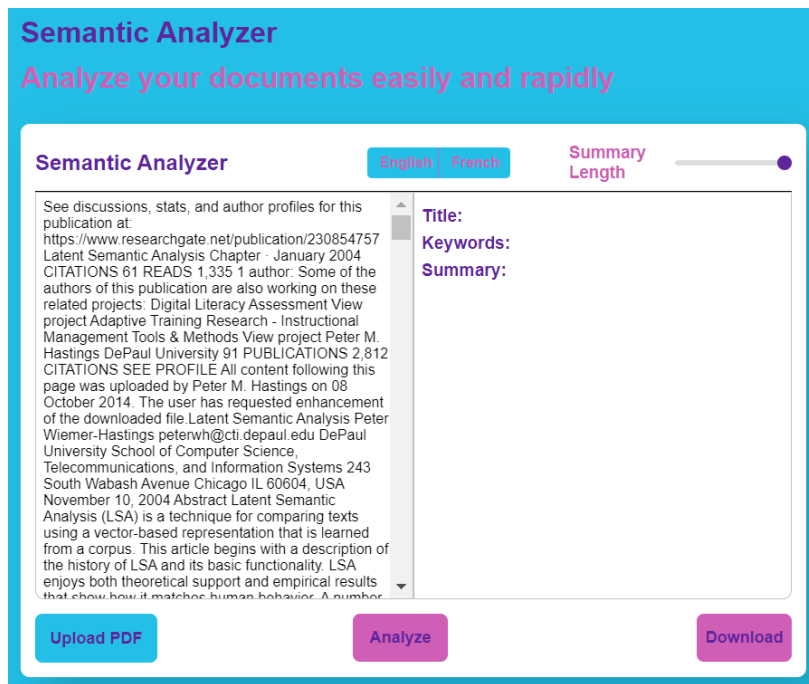


Figure 31: Uploading a pdf document

- **Language Options:** The interface offers language options for users to select the desired language for their analysis. While both English and French options are available, only English is currently functional. This feature enhances accessibility for a broader user base.
- **Analysis Process:** After uploading a PDF, users can initiate the analysis process by clicking the "Analyze" button. Behind the scenes, our system employs advanced algorithms to perform semantic analysis on the document.
- **Title, Keywords, and Summary Display:** Once the analysis is complete, the interface presents the following key information:

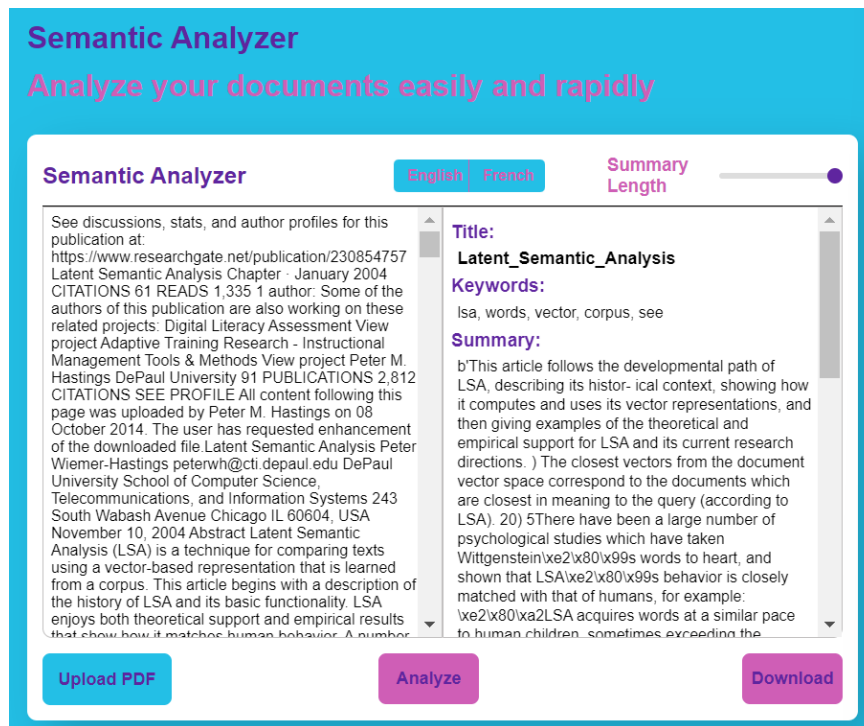


Figure 32: Pdf analysis

- **Title:** The extracted title of the document, providing users with a concise representation of the document's subject.
  - **Keywords:** A list of essential keywords identified within the document. These keywords offer insights into the document's main topics.
  - **Summary:** A summarized version of the document's content. This summary offers users a quick overview of the document's main points and insights.
- **Download Analysis:** To facilitate further use and reference, users can download the analysis results (title, keywords, and summary) as a text file. This feature allows users to save and share the document's key insights conveniently.

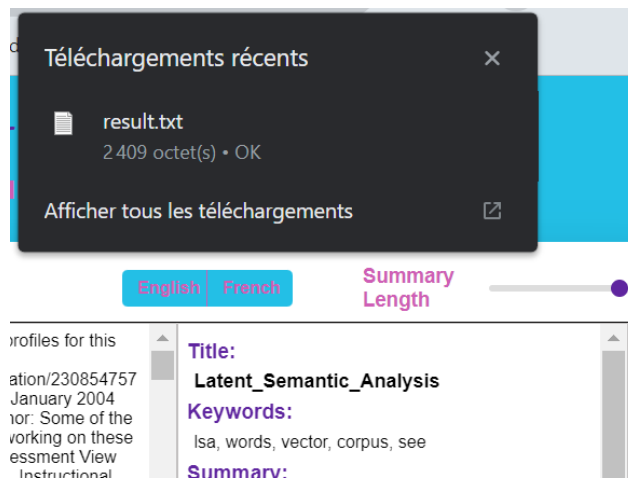


Figure 33: Downloaded analysis

### 5.4.3 Partial Analysis Interface:

The Partial Analysis interface offers users the ability to **perform focused analysis on specific sections of their uploaded PDF documents**. The interface is designed to provide users with a straightforward method for selecting and analyzing specific document sections to extract the main ideas. The interface has the following components:

- **Upload and Document Selection:** Users begin by uploading a PDF document using the "Upload PDF" button. Once uploaded, the interface scans the document to identify and list the document's titles. Users can then select the section they wish to analyze from the list of identified titles.

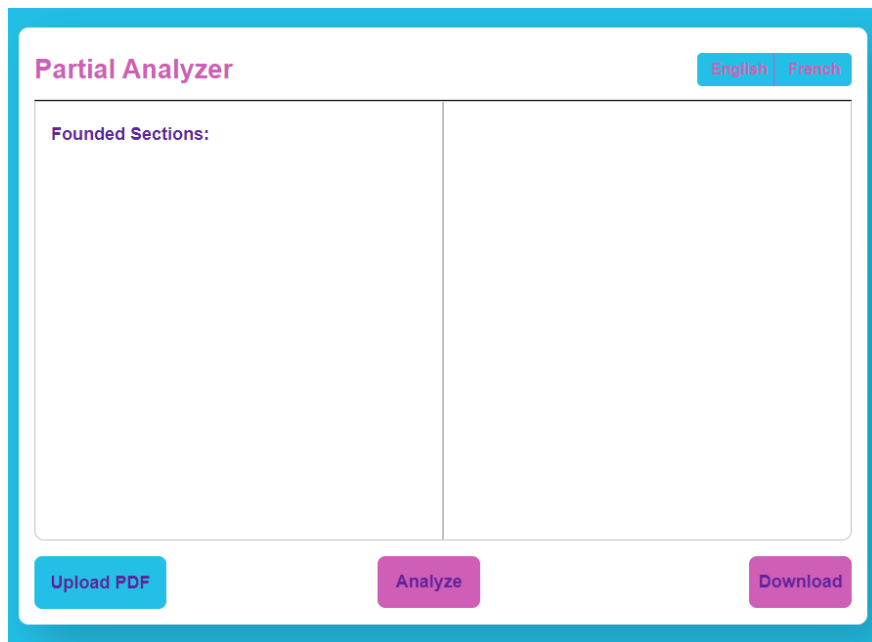


Figure 34: Partial analysis interface

- **Titles Display:** In the first block of the interface, the titles found within the uploaded PDF document are displayed. This component provides users with an overview of the document's structure, helping them identify the sections they want to explore further.
- **Main Idea Display:** In the left block of the interface, users can expect the main ideas or content of the selected section to be displayed. This component is designed to show users a concise representation of the content within the chosen section.
- **Language Options:** Similar to the Semantic Analysis interface, users can select their preferred language (English or French) for analysis. However, it's important to note that currently, only English language analysis is implemented.
- **Analysis Process:** Upon selecting a section for analysis, users will have the option to initiate the analysis process. While the logic for analyzing specific sections is not yet implemented, the interface's structure and user flow have been prepared for this functionality.

The Partial Analysis interface represents a valuable feature that allows users to delve into the main ideas of specific sections within PDF documents. While the logic for section analysis is currently in development, the interface is poised to provide a seamless and focused user experience. Users can expect efficient section selection and analysis, further enhancing the overall utility of the application.

## 5.5 Design principles:

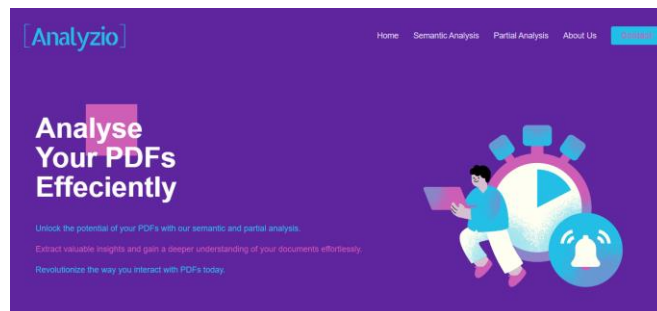
In the development of our project, we adhered to a set of fundamental design principles to ensure an exceptional user experience, accessibility, and overall efficiency. These principles underpin our project's success and user satisfaction.

### 5.5.1 Responsiveness:

Our project is designed to be responsive, adapting seamlessly to various screen sizes and devices. Responsive design principles guarantee that users experience a consistent and visually appealing interface, regardless of whether they're using a desktop, tablet, or mobile device.

The following images show some parts of our project for different endpoints.

#### The head of the home page:



*Figure 35: Header for big screens*

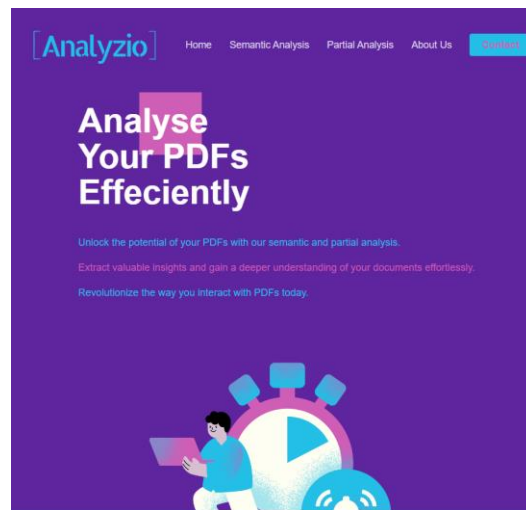


Figure 36: Header for medium screens

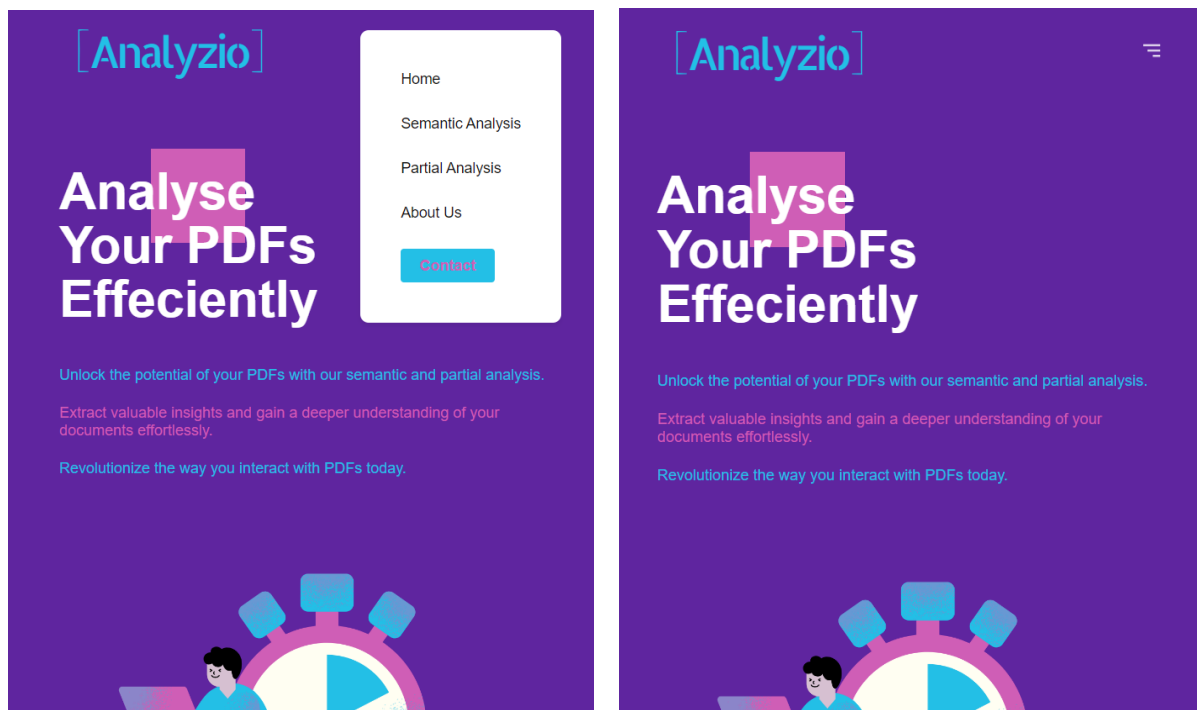


Figure 37: Header for screens less than 768 px

## The type of analysis section:



Figure 38: Analysis type section for big and medium screens

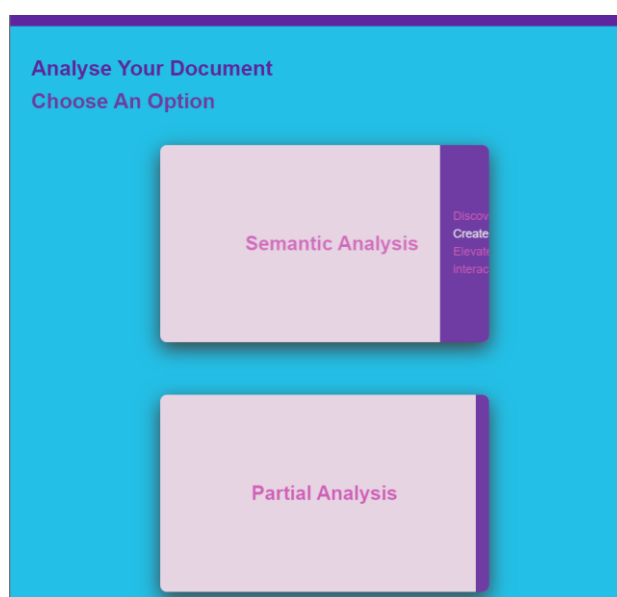
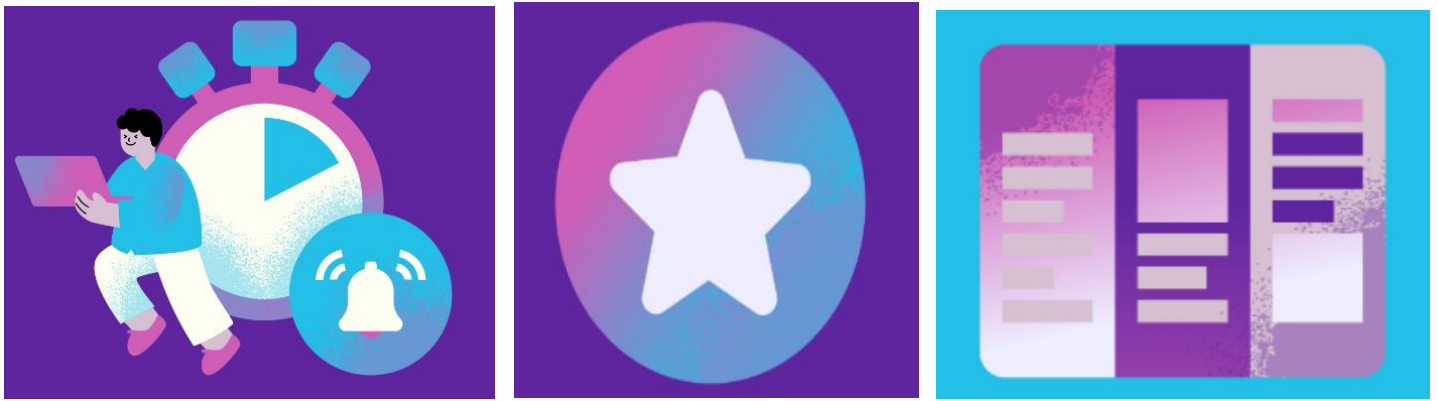


Figure 39: analysis type section for small screens

### 5.5.2 Consistency:

Consistency is a core design principle in our project. It's evident in our choice of color schemes, typography, and component design. Consistency creates a cohesive and professional appearance, making it easier for users to navigate and understand our application. For example, the images used in the project are very coherent with the chosen theme.





*Figure 40: Used images in the project*

### **5.5.3 Usability:**

Usability is paramount in our project's design. We've focused on creating user interfaces that are intuitive and user-friendly. This involves clear labeling, straightforward navigation, and logical interactions, ensuring that users can effortlessly achieve their goals. For instance, to access the Semantic Analysis page, there are two intuitive pathways: users can directly click on the Semantic Analysis option in the header or choose the corresponding card in the Analyzing Types section. This dual accessibility exemplifies our commitment to a user-friendly experience.

The usability extends to even the smallest details. For instance, the Contact button in the menu intuitively guides users to the Header section, where they can easily find the means to get in touch with us.

## **5.6 Pages:**

### **5.6.1 Home page:**

The Home Page serves as the gateway to your project. It integrates various essential components to provide a seamless user experience. Starting with the Header, users can easily navigate through the project's sections and access key functionalities. The Hero component takes center stage, presenting a visually captivating introduction to the project. It conveys the project's primary objective: efficient PDF analysis. The Analyzing Types section on the Home Page offers users the choice between two analysis methods: Semantic Analysis and Partial Analysis. This interactive feature guides users to their preferred analysis type. Value and Mechanism components furnish users with valuable information

about the project's goals and operational processes, instilling trust and understanding. The Footer elegantly wraps up the Home Page, providing essential contact details and reinforcing the project's vision.

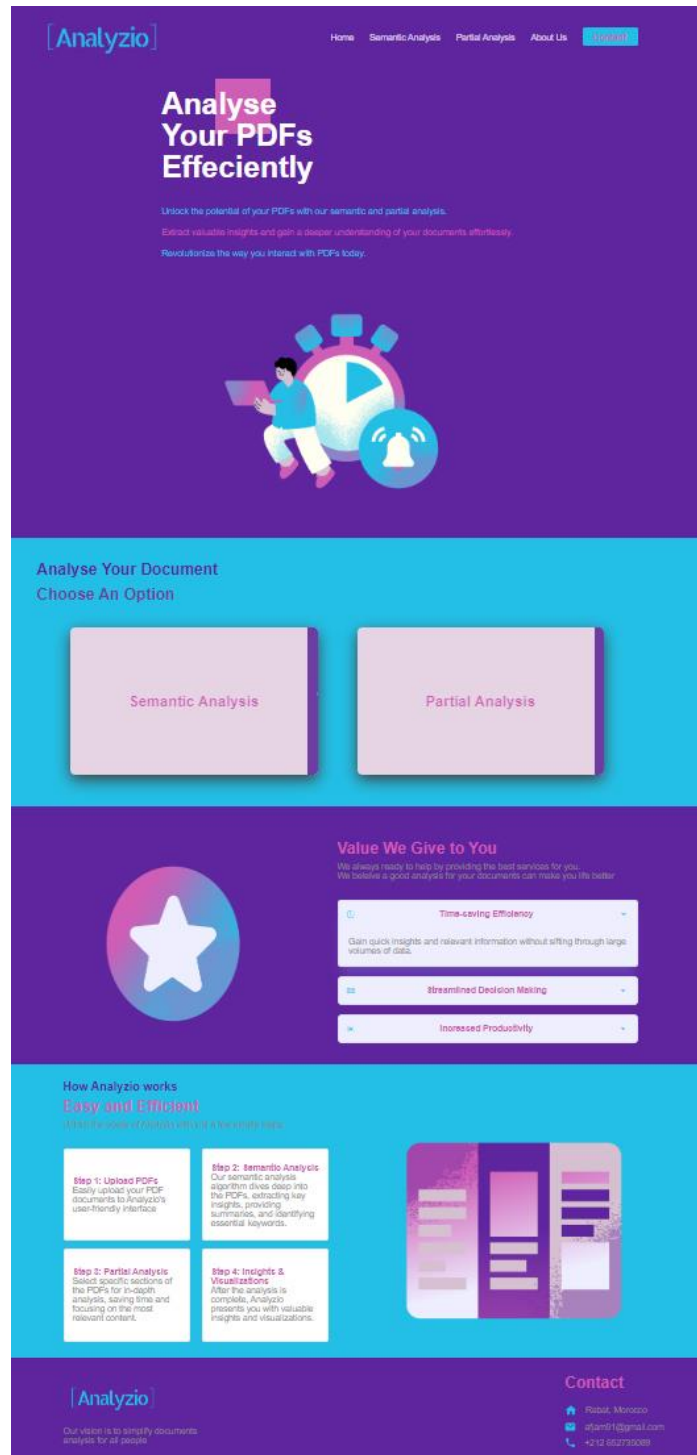


Figure 41: Home page

## 5.6.2 Semantic analysis page:

The Semantic Analysis Page is the hub for performing in-depth analysis of PDF documents. It features a versatile Semantic Analyzer component that empowers users to upload PDFs, extract valuable insights, and access results in the form of titles, keywords, and summaries. Language options are available, with English currently functional. The Header and Footer components provide consistent navigation and contact information, ensuring a seamless user experience.

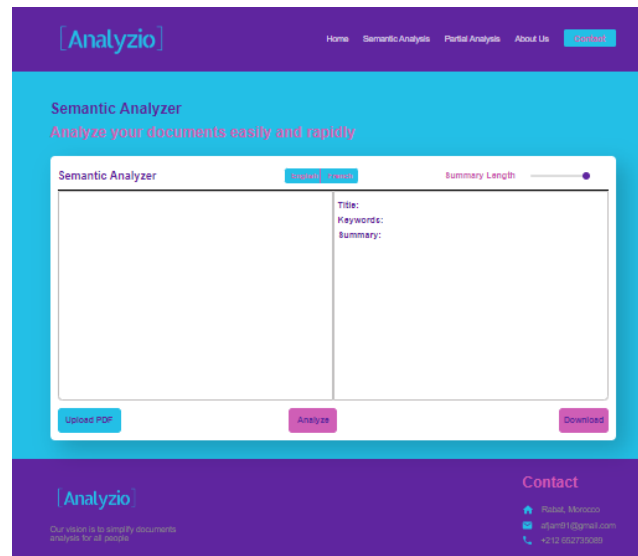


Figure 42: Semantic analysis page

## 5.6.3 Partial analysis page:

The Partial Analysis Page facilitates in-depth analysis of specific sections within PDF documents. Users can upload PDFs and view the found titles, with the ability to choose a section for further analysis, although this feature is not yet implemented. The **Header** and **Footer** components maintain consistent navigation and contact information.

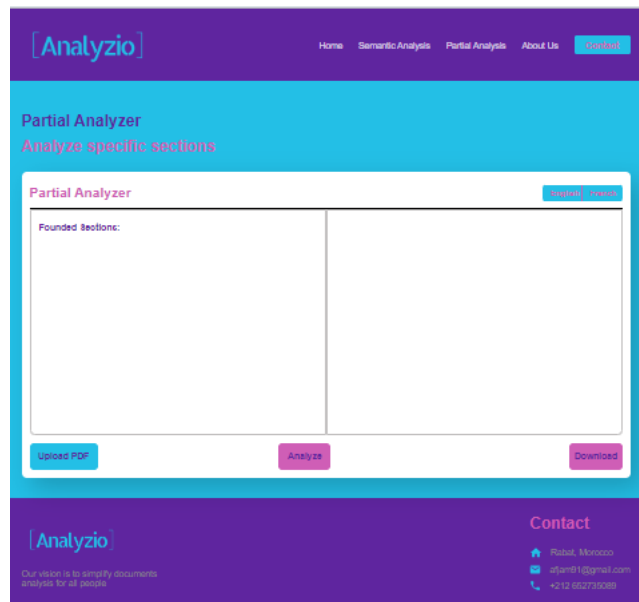


Figure 43: Partial analysis page

## 6 Backend development:

In the backend development phase of our project, we focused on building the server-side components that handle data processing, and the core functionalities of our application. This section provides an overview of the key aspects of the backend development process.

### 6.1 Architecture:

For this project, we opted for a **Monolithic Architecture** ( **all the backend is composed in one piece**), a design pattern well-suited to the project's simplicity and straightforward requirements.

#### Key Characteristics of Monolithic Architecture:

- **Single Codebase:** In a monolithic architecture, all components of the application, including the server, database, and user interface, are tightly integrated within a single codebase. This cohesive structure simplifies development, as there's no need to manage separate codebases for different components.
- **Ease of Development:** The monolithic architecture's primary advantage lies in its simplicity. With all components closely interconnected, developers can work on the entire application

without the complexities of managing microservices or separate modules. This streamlined development process accelerates feature implementation and reduces development time.

- **Deployment Simplicity:** Deploying a monolithic application is straightforward. Since it consists of a single codebase, deployment involves fewer configuration steps compared to more complex architectures. This simplicity results in quicker and more reliable deployments.
- **Inter-Component Communication:** In a monolithic architecture, communication between different parts of the application is seamless and efficient. Functions and procedures can be called directly, simplifying data transfer and reducing latency.

## 6.2 API Endpoints:

In our project, API endpoints has been meticulously crafted to manage the flow of data between the frontend and backend components. These endpoints are crucial for various operations, including the extraction of titles, keywords, and summaries from uploaded PDFs. Here, we present an overview of the key endpoint.

**POST /analyze:** This endpoint handles the analysis of uploaded PDF documents. Upon receiving an upload request, it initiates several processes to extract valuable information.

The */analyze* endpoint is a crucial part of our application, responsible for conducting in-depth analysis on uploaded PDF documents. It follows these steps:

- Upon receiving an upload request, it stores the uploaded PDF in the 'uploads/' directory, retaining the original filename.
- A Python script (*titleBridge.py*) is executed to extract the title of the PDF document. The result is sent back to the endpoint.
- A separate Python script (*KeywordsBridge.py*) is invoked to retrieve keywords from the PDF content, which are subsequently sent to the endpoint.

- Finally, the endpoint initiates another Python script (bridge.py) to generate a comprehensive summary of the uploaded PDF.

This endpoint exemplifies the efficient interaction between our frontend and backend components, ensuring that users can gain valuable insights from their documents effortlessly.

## 7 Conclusion:

In the ever-evolving landscape of document analysis and information extraction, our project stands as a testament to innovation and efficiency. With a keen focus on providing users with powerful yet intuitive tools, we embarked on a journey to revolutionize the way individuals interact with PDF documents. This report serves as a comprehensive overview of our project, highlighting key aspects that have contributed to its success.

From the inception of our project, we set out to address a fundamental need: the ability to quickly and comprehensively analyze PDF documents. The creation of semantic analysis and partial analysis functionalities became our guiding light, offering users the flexibility to choose the depth and scope of their document examinations.

The user interfaces we meticulously designed exemplify our commitment to usability and accessibility. Navigating through the project's various components, users can effortlessly upload PDFs, extract titles, keywords, and summaries, and explore valuable insights—all while enjoying a visually appealing and responsive interface.

Our development approach, featuring a monolithic architecture, file-based storage, and well-structured API endpoints, has paved the way for seamless integration and robust functionality. Users can interact with our project confidently, knowing that their documents are processed with precision and speed.

The design principles we adhered to—consistency, usability, and responsiveness—have underpinned every decision, ensuring a cohesive and user-friendly experience. These principles have not only shaped our project but have set a standard for future endeavors in document analysis.

In conclusion, our project has redefined the boundaries of document analysis. By combining cutting-edge technology, a user-centric approach, and a commitment to quality, we've empowered users to unlock the potential of their PDFs. From extracting titles and keywords to generating insightful summaries, our project exemplifies the power of technology harnessed for simplicity and accessibility.

As we look to the future, our project stands as a testament to what can be achieved with dedication and innovation. We are excited about the possibilities it opens up and the impact it can have in various domains, from research and academia to business and beyond.

We extend our gratitude to all those who supported us on this journey, and we look forward to continued growth and innovation in the world of document analysis.

## **8 Code**

You find the code of our project in the following repository:

[https://github.com/JamilaAferhane/Analyzio\\_website.git](https://github.com/JamilaAferhane/Analyzio_website.git)