
UNIVERSITY OF BIRMINGHAM

# An all-encompassing Web Application for Text Summarisation and Evaluation

Rahul Sagar Gheewala – 2066546

Supervisor: Dr. Venelin Kovatchev

Inspector: Dr. Jacqueline Chetty

M.Sc Computer Science with a Year in Industry

M60 Final Year Project

Word count: 16,668

April 2024

**Abstract**

The proliferation of digital information has led to an overwhelming surge in the information users can process and consume. Automatic Text Summarisation (ATS) serves as a solution that helps users efficiently navigate through this vast amount of information. However, recognising that most end-users of ATS are non-technical, little has been done to democratise this technology and leverage these users to improve the process. This project sets itself apart by providing a user-friendly platform that broadens accessibility and encourages collaborative improvement among technical and non-technical users to advance the field. Our platform offers a range of functionalities, including comprehensive text summarisation, evaluation and testing using state-of-the-art models and metrics. Notably, users can train models with their data, conduct visual analyses of summary construction, and contribute to advancing the field by providing data. Additionally, we offer a unique, user-inspired feature that allows for creating web-enhanced summaries that integrate web data.

To ensure that our users' needs are considered and at the forefront of our development, we follow an Agile software development workflow using research-backed Human-Computer Interaction (HCI) methodologies. This approach has allowed us to optimally use our resources and gain comprehensive user feedback, which has been instrumental and pivotal in shaping our application. For each of our iterations, user testing was a significant component that guided our implementation choices and helped us develop an intuitive and user-centric application. Through thorough analysis and evaluation, we believe we can contribute significantly to ATS and educational technology with future work.

**Acknowledgements**

I am profoundly grateful to my supervisor, Dr. Venelin Kovatchev, whose support and guidance have been exceptional. Throughout the project, Dr. Kovatchev provided me with invaluable insights and suggestions, consistently going above and beyond to ensure its success. Alongside Dr. Kovatchev, I would also like to extend my heartfelt thanks to Dr. Jacqueline Chetty. Their belief in me has not gone unnoticed and has meant much to me.

I want to thank my parents, grandmother, and brothers, Rohan and Amar, who are the pillars of my strength. I am deeply grateful for their endless love and support. Lastly, I dedicate this paper to my grandfather, Mukund Hansraj Gheewala, who sadly passed during the tenure of my studies. My grandfather was a studious man who strongly believed in the power of education; he continues to motivate me to strive for academic excellence. Though he is no longer with us, I know he would be proud of my achievements.

# Contents

# Chapter 1: Introduction

## 1.1 Background

*Natural Language Processing* (NLP) is an interdisciplinary research field representing the dynamic intersection of Artificial Intelligence (AI) and the study of linguistics. At its core, NLP focuses on allowing machines to understand, interpret, and generate natural language that is both meaningful and useful to humans. The underlying process of NLP can be briefly described as the bidirectional translation between structured and unstructured data [1]. In this high-level overview, 'unstructured' refers to data of a qualitative nature, which could be in the form of text or speech and could have been generated by a human or computer. Typically, this data type is relatively complex and has no predefined structure. Conversely, 'structured' data, as the name suggests, has some form of structure that can be tabularised and quickly processed by a machine. The direction of going from unstructured to structured data is referred to as *Natural Language Understanding* (NLU), and the alternative is *Natural Language Generation* (NLG) [2].

NLU is used to perform tasks such as syntactic and semantic data analysis tasks to derive underlying meaning, allowing a deeper understanding of context. NLG, on the other hand, leverages AI to synthesise coherent, contextually relevant natural language, a process that aims to capture the underlying essence of the information [2]. Through these techniques, computers are now equipped with remarkable proficiency to generate natural language, which has led to significant developments in various fields. With advancements in computational linguistics along with statistical, machine learning, and deep learning models, the scope of what NLP offers continues to grow. Some prevalent examples include chatbots and virtual assistants to aid with support, machine translation enabling language translation without human intervention, auto-correction to enhance written fluency and accuracy, and products such as Amazon's voice-controlled system, Alexa.

When discussing the usability of NLP applications, it is essential to consider *Human-Computer Interaction* (HCI) as an integral component of the user experience. HCI is a multidisciplinary field that studies the theory, design, and implementation of interfaces used by computers and technology systems. Research within this field attempts to facilitate and improve these interactions between machines and humans, using methodologies to evaluate a given application's usability and user satisfaction [3]. One method is user testing, where feedback is collected on how participants interact with some software; researchers then try to find patterns and previously undiscovered issues. Using this feedback, researchers can identify problems in usability, user preferences, and areas that can be improved through the interface's design and functionality. Another method is *user-centred design* (UCD), in which users are actively involved through the application's design and development process. This methodology encourages the creation of technology that closely aligns with the user's requirements and expectations, prioritising user satisfaction. HCI has contributed significantly to making technology accessible to the broadest possible audience through intuitive and easy-to-comprehend design choices that users can effortlessly pick up.

In this study, we will provide a platform for NLP technologies using HCI methodologies. We will follow a software development process to design and implement a web application catering to a non-technical audience. The selected domain of NLP technique will be Text Summarisation, a process that involves condensing user-provided text into a shortened version. The goal is to produce a concise and coherent summary that retains essential information and ideas while reducing the time required for consumption.

## 1.2 Problem Outline

**Motivations for a Text Summarisation Project**

The project has been motivated by the proliferation of digital information in recent years, which has led to an overwhelming surge in the amount of textual information users can process and consume. Some reasons responsible for this explosion include the recent success of generative AI, which can produce contextually relevant textual content within seconds, a paradigm shift in which people can share textual information more efficiently than ever before, and better-optimised search engines, which have accelerated accessibility to relevant information and have grown the demand for information.

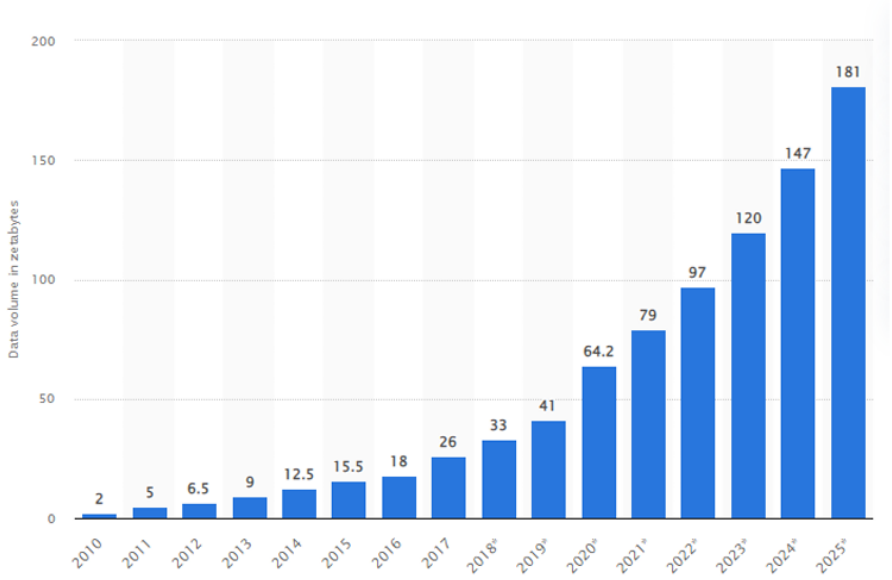



*Figure 1 Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025 [4]*

While significant advancements, it is also essential to consider the implications of such a rapid expansion of textual information. With vast information available, users often need help finding the most relevant information from various documents. Such volume has also made filtering out noise and identifying reliable vital points increasingly challenging. To further compound matters, according to Dr Gloria Mark, the average user's attention span on screen was 2.5 minutes back in 2004; today, it is only a mere 47 seconds [5]. With platforms like TikTok, Instagram, and Facebook continuing to set new standards for brevity and engagement, our attention spans will probably decrease as time progresses. This trend further intensifies the need to make critical textual content points more readily available, reducing consumption time.

One method to solve this problem is to use the expertise of humans to generate accompanying summaries. Humans can offer unparalleled contextual understanding and quality control when producing summaries. Humans can effectively identify nuances and subtleties, enabling them to form reliable distinctions between trivial and vital sections of text. Once the summariser has grasped the text's central concept, they can produce a summary that can adapt to the audience's needs by varying the detail, tone, and style. When generating human summaries, it is not just a function of the user input but also the reader's mental state when generating the summary. It is possible that the reader may already be an expert in the domain and may have pre-existing knowledge of topics and a general idea of what they would like to understand from the text [6]. Though prior learning can be ideal when tailoring summaries, this also means there is significant variability in quality and consistency, which adds the potential for bias. Further drawback factors include constraints such as

the time commitment and cost of involving humans. Human summaries may serve as a temporary measure for these reasons but are not scalable enough to meet current demand.

A more feasible and scalable solution is *Automatic Text Summarisation* (ATS), which uses algorithms to identify the critical points in large volumes of text to generate a summary. The time taken to perform ATS is nominal compared to human-generated summaries, meaning that ATS can create multiple summaries within the timeframe it takes to produce one manual summary. Additionally, due to the algorithmic nature of ATS, the summaries produced are consistent and exclude elements of subjectivity and bias present in human summaries. However, this is only true given the assumption that measures were taken to avoid bias when the ATS system was developed. Such measures include using a representative training set that equally values all perspectives, algorithms to find and minimise bias, and continuous monitoring to cater to bias that may occur during the system's lifetime. ATS has become widely used across various domains and has revolutionised how users obtain information. ATS has the potential to be used in virtually any domain in which there is a substantial amount of textual data and can offer, to at least some degree, ease of consumption.

**Motivations for Our Project**

Despite ATS's advancements, there is potential for improvement in user control, transparency, adaptability, and reachability for non-technical users. Although most ATS end-users are non-technical, the underlying technology is mainly inaccessible to them. Recognising that text summarisation aims to meet the end user's needs; their importance should not be neglected. Instead, their involvement can and should be leveraged to help improve the end-user experience, gain user trust, and advance the field.

**Advancement of Text Summarisation**: One of the main challenges for improving text summarisation is acquiring training data. Unlike *Large Language Models* (LLMs), which benefit from large volumes of training data, the training data required for text summarisation is not readily available. While methods such as data augmentation have been explored [7], annotated human-generated summaries are much more preferred due to their ability to capture nuances and subtleties. However, producing such is costly and time-consuming, which must be addressed. By motivating non-technical users to help mass-produce these datasets, costs can be reduced, accelerating the advancement of the field. Many non-technical users can also offer domain-specific knowledge; their input could help encourage models to accurately capture critical aspects of a domain. There is also potential to assist with feature engineering, in which users can identify relevant features or characteristics of good summaries that can be noted and used to improve the model architecture or training process. This would provide great strides for domain adaptive text summarisation. In addition to providing training data, such users can give feedback on datasets and the usefulness of summaries generated by text summarisation models. This would allow experts to better understand and identify improvement areas, helping refine their processes and produce higher-quality summaries for the user. This can also be extended to users being able to edit summaries manually to improve clarity and understanding, which would again provide higher-quality training data and improve performance for the machine learning algorithms in use.

**Transparency:** Another challenge of ATS is providing transparency to its users, which is necessary to build trust and understanding. Since text summarisation removes text sections, there is the potential for misrepresentation, which can carry severe implications. For example, if a model favours removing specific topics or sentences, it could significantly change the overall sentiment and meaning of the original text. This can be dangerous in domains such as journalism, where precautions must be taken to portray truth and maintain integrity. For documents of a shorter length, a user can verify if the main points have been covered by reading the main body and making an assessment. However, this

may not be possible for longer documents, and if they were to, it would contradict the purpose of ATS. It is necessary to provide model insights on generating summaries, including the algorithms and criteria for formulating a summary. This could further be extended to allow users to perform testing and evaluation using their data to help understand the reliability and accuracy of the summaries generated.

**Customisability**: Another area for improvement is user control and adaptability. Due to inaccessibility, non-technical users are restricted to the summary version they receive. If provided with a medium, users could choose a model and adjust parameters to customise a summary to meet their requirements. This would avoid using a 'one-size-fits-all' approach, which may result in summaries that are not tailored towards specific styles and may not meet the complete requirements of the user. Different text summarisation models offer unique properties that appeal to individual users; these include type of summarisation, stylisation, content selection, language fluency, and coherence. Some models may favour longer but more in-depth outputs, which may be helpful in applications to grasp the underlying concepts of a text fully. For others, this may be too comprehensive for their needs, and they may prefer summaries with greater brevity. There is a need to empower users and allow them to have more personalised and relevant summaries.

To begin addressing these challenges, there is an unprecedented need for a solution that makes ATS as accessible as possible to non-technical users. Currently, exploring ATS techniques is far too challenging for most of its users. These users are restricted by the knowledge and technical expertise required to leverage text summarisation models effectively and perform testing with evaluation. Achieving this independently would involve implementing complex code and sourcing the most suitable models, metrics, and datasets, an unrealistic expectation.

## 1.3 Project Aims

For this project, we will attempt to address the challenges discussed in section 1.2 by developing a user-centric web application for text summarisation, testing, and evaluation. We will employ design considerations to cater to non-technical users who may not have programming experience or prior NLP knowledge. Throughout the project, we aim to tailor our approach following user feedback and preferences to maximise user satisfaction. Our contributions will not be offered by optimising models and producing higher-quality text summarisation results but by enhancing the practical utility of NLP technologies. We hope to promote broader engagement and understanding of state-of-the-art tools across diverse user communities, ultimately driving democratisation and advancement.

This project encompasses several specific aims, each contributing to the overall goal of providing an accessible and usable platform for our users. The following provisional aims delineate the general focuses of our project:

- **User-Friendly Interface**: We aim to create a user-friendly interface that provides a seamless and intuitive experience to explore text summarisation techniques. To achieve this, we will prioritise simplicity and accessibility without compromising technical functionality. We will leverage interface design principles and input validation methods to mitigate or resolve potential errors when using text summarisation technology. This will protect our users from the disappointment of encountering confusing or overwhelming errors. Furthermore, we will integrate HCI design principles to achieve minimalism and help reduce a user's cognitive load.
- **Customisation and Experimentation**: We aim to provide a platform for our users to experiment with different state-of-the-art text summarisation models, in which they can fully adjust diverse parameter options to explore each model's capabilities. In addition to being a

valuable summarisation tool, users will be encouraged to experiment and observe how a model's behaviour can change depending on the setting. This can further be compared with different model outputs, helping users gain insights into how various parameters and model architectures can affect the quality and styles of summaries.

- **Usability with Own Data**: We aim to allow users to use their data when applicable, for example, training a model or testing. When provided with biased data or that of specific domains, users can explore a model's behavioural effects with user-inputted data. This may be useful to understand how data quality can impact the summarisation process and how it can be used maliciously or intentionally to produce summaries directly relevant to a domain or user's interests.
- **Output and Score Analysis**: A significant component of our application will be to allow our users to test and evaluate text summarisation models. We will make state-of-the-art evaluation metrics available for users to measure a model's effectiveness quantitatively. We will take extra consideration to ensure fairness between different models to enable users to make an informed decision if they choose to make a preference. Additionally, we will allow users to visually compare an original text and its summary to provide insights into how the summary was constructed. We will also provide functionality to allow users to rate summaries and provide their data to provide insights to developers and advance the field.
- **Enhanced Summarisation**: As part of our goal to develop a user-centric application, we will prioritise obtaining information from our users to improve our application. We will also engage with our users to try to understand the areas where current text summarisation techniques could be further improved from a user's perspective. Using this, we will devise a novel approach to address this challenge, enabling them to derive better value and insights from the summarisation process.

This dissertation will address these aims and attempt to democratise access to advanced text summarisation technology. By empowering non-technical users with advanced NLP technology, we hope to broaden accessibility and achieve collaborative improvement among technical and non-technical users.

# Chapter 2: Literature Review

While our project is primarily HCI-based, we require a robust and comprehensive understanding of NLP text summarisation techniques to provide the context for our proposed solution. This chapter delves into our extensive research on an existing solution, state-of-the-art text summarisation models, evaluation metrics, datasets and HCI methodologies.

## 2.1 Related Work

After an extensive literature review, it was intriguing to discover only one paper directly addressing the goal of broadening accessibility to text summarisation techniques. This observation led me to consider several possibilities: either my choice of keywords and phrasing were suboptimal in my search queries, perhaps I had delved into a domain in which very little progress had been made, or my perceived problem might not have had much significance to the academic community. To explore this further, I meticulously examined the 'Related Work' section of the paper I found to help me find relevant work and motivations. It became apparent that there was indeed minimal related research, and this section only consisted of broad topics. I realised it was not my search queries or a lack of relevance but a scarcity of directly related research.

We refer to SummerTime [8], which proposes developing a toolkit for NLP non-experts to assess text summarisation models effectively. This toolkit facilitates experimentation by providing access to models, datasets, and evaluation metrics to enhance research capabilities in summarisation-related tasks. The toolkit integrates commonly used libraries and APIs used by NLP experts to offer insights into the behaviour of models, empowering users to make informed decisions when they choose a text summarisation model for their specific requirements.

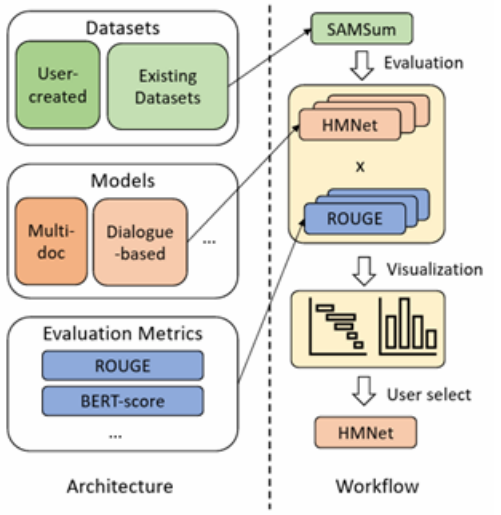



*Figure 2 SummerTime Architecture and Workflow [8]*

As shown in Figure 2, the tool allows the user to choose between a user-created or existing dataset, which is then processed through a summarisation pipeline. Through testing various models in conjunction with evaluation metrics, the tool provides visualisations to facilitate comparative analysis.

The toolkit also provides functionality to streamline choosing an optimal model by providing a workflow that automates model selection. Users can, again, load a custom or existing dataset and provide attributes such as the type of summarisation task. SummerTime then produces candidate solutions by combining different pipeline modules with summarisation models. To efficiently identify the top-performing models, successive halving is used, which starts by evaluating a smaller sample of models and then progressively eliminates models that have been outperformed. Across this duration, the sample is then iteratively increased to minimise the variance and ensure robustness. The final results include a set of models that excel in at least one of the evaluation metrics, which facilitates the user's decision-making process in choosing the best model.

Overall, SummerTime is a promising solution that assists NLP non-experts with text summarisation-related tasks and experimentation by only needing to change several lines of code. The toolkit effectively integrates various models, datasets, and evaluation metrics for text summarisation-related tasks such as single-document, multi-document, and dialogue summarisation, making it easier to perform such tasks. We also recognise the value SummerTime offers through its automatic model

selector and visualisation tools to compare model performances. Lastly, we take note of the toolkit's efforts to help users by making state-of-the-art summarisation methods more accessible to non-expert users and finding the most suitable models. However, our project aspires to take inspiration from this work and develop it further.

Similar motivations exist when comparing our project to the referenced tool, but significant differences exist in purposes and target audiences. While the tool also aims to enhance accessibility to text summarisation technology, it targets NLP non-experts presumed to have a technical background, inadvertently excluding non-technical users. The tool relies on the user's ability to modify code in its notebook format, which lacks a user-friendly graphical user interface. For our users, as evidenced in our research, it was intimidating and off-putting, even if it just meant changing a few lines of code. Furthermore, while the tool can offer side-by-side comparisons, the main feature of model selection uses an end-to-end approach, which limits a user's flexibility in customising the model's output behaviour. This hinders users' ability to provide preferences and experiment critically; our focus will be to provide our users with all the tools necessary to make their personal decisions. Another difference will be our focus on models. SummerTime compares different summarisation methods, such as single-document, multi-document, and dialogue-summarisation, but we will focus on comparing different models of standard summarisation. We recognise that users may have interests in these subcategories; however, as we provide a platform to all abilities, we will use these methods to give the ideal elevation to those types of summarisation.

Our project will take inspiration from SummerTime and introduce new features to improve a user's overall experience. One of our most notable improvements will be implementing an intuitive and user-friendly interface, which will keep simplicity and accessibility in mind and will not require a set-up process or technical expertise. As part of our educational approach, we will also add elements such as tooltips and dedicated sections to explain underlying concepts and understand how to leverage our tool effectively. This will develop from the weak descriptions provided by SummerTime, which cannot offer complete insights into each model. We will also offer more precise insights into evaluation metrics by providing alternate visualisations instead of scatter graphs and radar plots. Though these can be effective, users may struggle to make simple comparisons side-by-side. We will also provide descriptions of each metric and what it is testing for, requiring no assumptions from a user.

We will also introduce additional features, including a summariser tool that allows users to select and experiment with different text summarisation models and change parameters, a model training section which will help in customising the training of models, an output analysis component for visually comparing model outputs and allowing users to provide input, a score analysis tool for evaluating the model effectiveness, user-enhanced text summarisation from which development will be guided by user preferences, and a FAQ section that responds to some common questions regarding the application and NLP. These additional functionalities improve the toolkit's usefulness by being more user-centric and giving users greater control and insights into text summarisation.

In summary, our project will incorporate some critical features of SummerTime and introduce several new features that enhance the user experience. This makes our project a significant extension and improvement of the SummerTime toolkit.

## 2.2 Overview of Text Summarisation

Text summarisation is a pivotal task used within NLP. It involves transforming extensive textual information into a more concise version, preserving essential information and the overall meaning. This section will provide a comprehensive overview of text summarisation techniques.

As Figure 3 shows, three main categories of text summarisation approaches exist. These categories revolve around defining the summary goal based on the type of input text, the chosen summarisation style, and the desired output format. This project will focus on text summarisation based on the output type, determining whether a summary should be Extractive or Abstractive.

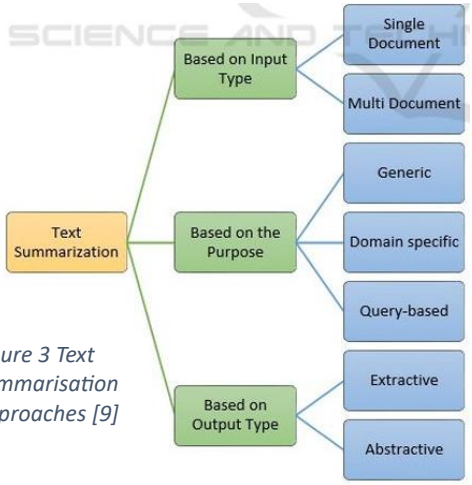



*Figure 3 Text Summarisation Approaches [9]*

### 2.2.1 Extractive Summarisation

In Extractive Summarisation, significant phrases and sentences are directly extracted from the source text to produce a summary. This type of summarisation does not require rewriting or rephrasing; it only requires existing context to construct the summary. Some common approaches to extractive summarisation include:

- **Sentence Scoring**: In this approach, a representation of the original text is first created. Then, an algorithm iterates through each sentence to determine which sentence is most important to the document or how well it can collate information across multiple topics. The most significant sentences representing the whole text are selected depending on their scores. [10]
- **Graph-based Algorithms**: With this approach, the source text is represented as a graph, in which the sentences are nodes, and the edges show how each sentence relates to the others. LexRank is an algorithm which uses this approach, where graph-based centrality is used to score sentences to identify which to use in the constructed summary. [11]
- **Machine Learning Approaches**: In this approach, supervised and unsupervised machine learning models can be trained to predict the significance of summarisation sentences. This uses labelled or unlabelled training data. [12]

Extractive summarisation offers benefits such as preserving the original text's context and meaning, as it only uses segments from the source text. This means there is no distortion, and accuracy is maintained in each sentence. Additionally, as new data is not required to be generated, Extractive summarisation is computationally fast and efficient when using large-volume source texts. However, disadvantages include coherence and readability, as removing surrounding sentences may exclude important context, and comprehending the new version without rephrasing may be challenging. Additionally, the overall meaning may change if specific sentences are stripped, leading to incomplete and biased summaries.

### 2.2.2 Abstractive Summarisation

Abstractive summarisation differs from Extractive summarisation in using paraphrasing to achieve its goal. The source text is rewritten and reformulated to resemble a human-generated summary closely, achieved using natural language generation models, which allow for a better contextual understanding and summaries with better coherence and readability. The main approaches of Abstractive Summarisation include:

- **Sequence-to-Sequence models**: With this approach, different variants such as encoder-decoder architecture and Transformer models are used to generate summaries through learning the input text. [13]
- **Attention Mechanisms**: This approach involves models focusing on relevant parts of the input text by calculating attention weights. This method uses the most significant input information when decoding to produce concise and contextually accurate summaries. [14]
- **Reinforcement Learning**: This method rewards models that can produce more contextually relevant and higher quality summaries. This is based on specific criteria, which the model favours. [15]

Although Abstractive summarisation can offer benefits such as creativity and flexibility, it can produce erroneous results with factual inaccuracy; this is more likely for longer, more complex inputs. Additionally, the generation process can be complex and computationally expensive.

To choose between these methods, users should consider the nature of their input text, the quality they would like the summary to be, and the computational resources available for processing.

## 2.3 Summarisation Algorithms

This section will discuss the summarisation algorithms we will use in our application. Each algorithm produces different results because of its unique architecture, how it was trained, and the nature of the tasks it can accomplish.

- **Pegasus** (Pre-training with Extracted Gap-sentences for Abstractive Summarisation): Pegasus is a transformer-based encoder-decoder model developed by Google to perform abstractive summarisation. The model uses a self-supervised pre-training objective called gap-sentences generation (GSG). GSG works by removing or masking essential sentences from the source text, which are generated together as one output sequence using the remaining sentences [16]. Using contextual embeddings and advanced training techniques, the model improves fine-tuning performance without requiring many training examples to achieve state-of-the-art performance. [17]

- **BART** (Bidirectional and Auto-Regressive Transformers): Developed by Facebook AI, BART also performs abstractive text summarisation using a transformer-based encoder-decoder architecture. BART is pre-trained on a denoising autoencoder objective, meaning that it is trained by corrupting text with an arbitrary noise function and then tasked to learn how to recreate the source text. This means it is ideal for recreating clean sentences from noisy ones, so BART is suitable for text summarisation. [18][19]

- **BERT** (Bidirectional Encoder Representations from Transformers): BERT is a pre-trained language model with an encoder-only transformer architecture. Developed by Google, BERT is trained in Masked Language Modelling and Next Sentence Prediction. Though not typically used for summarisation tasks, it can be leveraged to perform Extractive summarisation. [20]

- **T5** (Text-To-Text Transfer Transformer): T5 is an encoder-decoder transformer-based architecture which uses a text-to-text approach. In this format, the input and output are always text strings, which differ from BERT-style models, which typically have output labels or a span of input. T5 comes equipped with a lot of flexibility, making it suitable for Abstractive and Extractive purposes. For our application, we will be using it for Abstractive summarisation [21][22].

- **LexRank**: As previously mentioned, LexRank is a stochastic graph-based algorithm that constructs a text graph representation. As part of this graph, sentence importance scores are computed using graph centrality measures. LexRank serves to be useful for Extractive summarisation [11]

The above algorithms can offer distinct approaches to summarisation, each with its positives and negatives, catering to diverse summarisation requirements and preferences.

## 2.4 Evaluation Metrics in Summarisation

We will also employ Evaluation Metrics to assess our text summarisation models quantitatively for our application. Evaluation metrics serve as an essential feedback loop for researchers and developers to understand how a model can be improved. It also enables comparative analysis amongst different models to understand their limitations and task capabilities. Each metric takes a generated and reference summary as input to provide scores.

- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation): ROUGE is not a standalone metric but a collection of different sub-metrics. ROUGE is tasked with measuring the overlap of n-grams, word sequences and word pairs between the generated and reference summary. Variants of ROUGE include ROUGE-N (quantifies the overlap of N-grams) and ROUGE-L (calculates the Longest Common Sequence between the summary and candidate). [23][9]
- **BLEU** (Bilingual Evaluation Understudy): Though BLEU was initially developed for machine translation evaluation, it can also be used to compare generated and reference summaries. BLEU has a scoring system which ranges from 0 to 1, where 1 represents an exact match between the reference and generated summary. BLEU focuses on measuring the precision of word n-gram sequences and includes a 'brevity penalty', which penalises outputs shorter than the reference summary to prevent the formation of subsets. [24][25]
- **METEOR** (Metric for Evaluation of Translation with Explicit ORdering): METEOR considers both precision and recall of unigram matches between generated summaries and reference summaries, although it provides more importance to precision. METEOR can also leverage a stemmer and synonymy, which it can use to incorporate semantic similarity in its calculations. [26]
- **BERTScore**: BERTScore is a metric that evaluates the quality of generated summaries by comparing them with reference summaries using BERT. BERTScore obtains the semantic aspect of the generated summary using the contextualised embeddings the BERT model creates. Cosine similarity is calculated between the contextualised embeddings of the summary and reference, from which precision, recall and F-scores are calculated. [27][25]

## 2.5 Benchmark Datasets for Summarisation

Although we will aim to provide our users with the functionality to use their data, we have decided to incorporate a benchmark dataset for our application. These datasets are a standardised resource for training, testing, and evaluating our text summarisation models. Our candidate datasets include:

- **CNN/Daily Mail**: This dataset comprises news articles with corresponding bullet-point summaries. It was initially created for reading comprehension and abstractive question-answering purposes, but the recent version is adapted for Extractive and Abstractive summarisation.
- **PubMed**: PubMed contains biomedical literature abstracts for summarising tasks involving healthcare and life sciences. Each question has a long and short answer, which can be used to evaluate summarisation processes.

- **SAMSum**: This dataset consists of messenger-styled conversations annotated with summaries. The summaries are styled to explain what was discussed in the conversation in the third person.

Such datasets can help refine and improve text summarisation models, ultimately contributing to advancements in NLP fields.

## 2.6 HCI Usability

This section will be dedicated to our research on HCI methodologies and approaches to improving user experience. From [28], we learn that 'The requirements for a system, in enough detail for its development, do not arise naturally. Instead, they need to be engineered and have continuing review and revision.' We aim to set out provisional requirements, but we expect adaptations throughout to meet the needs of our users best.

The paper 'HCI Practices for Building Usable Software' emphasises the importance of observing user interactions with the system. This includes not just their interactions with the UI but also how they approach a problem, respond to feedback from the system, and adapt to the system over time. Such observations are instrumental in providing a developer with insights into the user's mental model, which can be used to improve the system's design and usability. Additionally, the paper suggests that usability should be considered from the beginning of the software development process and continuously reviewed and evaluated throughout the software development life cycle. Achieving this involves integrating usability engineering methods and following robust HCI approaches. [29]

In our project, we will employ a crossover of several approaches to optimally use resources, such as time allocation, task management, and resource utilisation. [30] discusses four state-of-the-art HCI approaches and typical standards when developing a project for end-users. These approaches include User-Centred Design (UCD), Usability Engineering (UE), Participatory Design (PD), and Human-Centred Design (HCD). We plan to adopt and incorporate ideas from each in our software development process.

UCD is a design strategy that focuses on putting the end-user at the forefront of each design decision. This is achieved through understanding the user and their needs in each phase of the design process. This approach ensures that by the end of the software development process, the end product is highly usable and accessible to users. On the other hand, HCD is an approach that views the user as a person and attempts to consider human capabilities and limitations. By putting 'real' people at the centre of the design process, the consideration of the user comes above all at every step in the design process. This is a step above UCD, in which the primary focus is on how the user can use a product instead of considering psychological and emotional needs. [30]

UE is a multidisciplinary field incorporating engineering principles and cognitive psychology to make a system more intuitive and user-friendly. This approach emphasises making the UI simple to absorb to ensure users can interact efficiently and enjoy an error-prone experience. PD is a design methodology in which the system's intended users can contribute as co-designers throughout the design process. This more value-centred design approach aims to democratise and make the software accessible. When used conjunctively, these approaches can give developers a comprehensive understanding of a user's needs and expectations, helping them to achieve a user-centric software solution. [30]

In developing our methodology for conducting usage studies, we have taken inspiration from [31], which discusses practical approaches for gathering valuable user feedback. The paper discusses the strategy of walkthrough demonstrations, which differs from traditional methods where a user may

explore a system to identify usability issues. With this approach, the developer showcases the application to users to try and understand impressions and perspectives. This strategic shift gives developers complete control over the workflow, where they openly discuss the system's capabilities and limitations. As a result, the emphasis changes from mere tool usage to an opportunity to gain insights into the value the application can deliver. Rather than just detecting issues, the user is encouraged to propose improvement ideas and appreciate the product's overall value and potential extensions.

Observations are another valuable methodology for understanding how users interact with a system while trying to assess problem solutions. This approach ranges from closed tasks in which one solution exists to open tasks in which users can employ strategies to solve a given or self-formulated problem. Through close observations, a developer may be inspired to produce new ideas or identify areas for improvement in the user experience. Observing can also help identify user patterns when interacting with a system, such as navigation paths or frequently used features which could be extended. It also presents an opportunity to see any workarounds a user may use, highlighting more efficient approaches or the need to remove redundant components which may confuse users.

By following these HCI methodologies and techniques in our approaches, we will strive to comprehensively understand user needs and preferences throughout the software development cycle. We draw inspiration from our literature review to develop an iterative and robust approach, informed by user feedback and observations, to create a user-centric application that achieves our set aims.

# Chapter 3: Methodology

This chapter will explain the reasoning behind our chosen methodologies for this project. We will begin by defining our software development process choice and ways of eliciting user feedback. This will include our selection of user study design, how we plan to collect user feedback effectively, and how we plan to analyse and optimally incorporate this feedback.

## 3.1 Project Management

For our application, we have chosen *Agile Project Management* (APM) as the guiding framework for our software development process. APM consists of managing a project using short sprints, a collection of iterations to deliver incremental improvements to a system. We have opted to use APM because of its central focus on iterative development, flexibility, and frequent collaboration with end-users. This has aligned seamlessly with our goal of developing a user-centric application. Our chosen approach has also allowed us to prioritise adaptability, in which we will effectively respond to continuously changing requirements guided by our users. Each sprint will be designed to integrate feedback from our users effectively throughout the development process.

We decided to adopt APM over traditional methodologies such as Waterfall because we recognised early on that user-centric software systems often require changes to achieve user satisfaction. Waterfall is an approach that uses linear progression in the form of distinct phases, which differs from APM in that there are minimal opportunities to make amendments to the system once the software development process has begun. Using APM has been particularly advantageous when integrating HCI principles, which require close interaction with end-users to ensure a design is intuitive and user-friendly. Through our continuous engagement with users, Agile will help us facilitate the timely incorporation of user feedback and help create an application that aligns closely with user needs and preferences. Furthermore, following an incremental delivery model allows us to prioritise high-value features, meaning that the essential features are developed and validated at the beginning of the software development process. This means we reduce the risk of investing time and resources in areas that may not align with our users' expectations or project objectives, which is particularly common in projects that incorporate complex user interactions. [44]

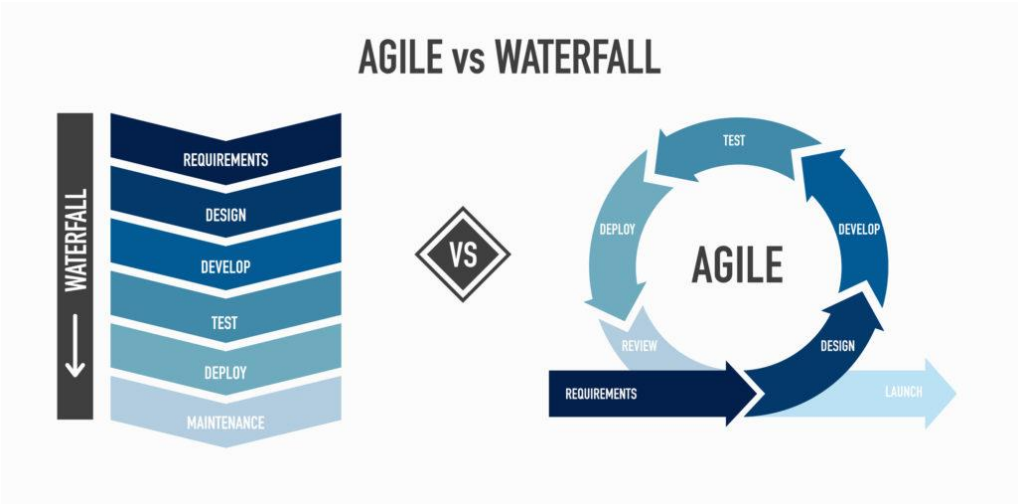



*Figure 4 Diagram comparing Agile with Waterflow [45]*

## 3.2 Design of User Studies

The design of user studies is imperative to validate the HCI principles and NLP techniques we employ in our application. The methodologies we will use for our user studies are structured to collect comprehensive user feedback to ensure we develop an intuitive, user-friendly system that meets our requirements. We will take great care in participant recruitment, study protocols, and experimental conditions.

**Participant Recruitment**

Although our project does not specify any initial requirements for a user, for our testing purposes, we will attempt to recruit people who are experienced with web browsers and can dedicate time to the study. While we will try to recruit a diverse pool of participants, we require a specific technical aptitude to accelerate our refinement process concerning the project deadline. Partly due to availability, many of our participants will be university students who fit this requirement and are familiar with sound web design principles. Amongst these, we will search for a diverse demographic to ensure the feedback we receive reflects a broad range of perspectives and usage scenarios. We will find our participants through online networking and the author's network.

**Study Protocols**

We will design a predefined protocol for each study we conduct to elicit relevant feedback from each user. Our study will try to find a balance between getting feedback on the system UI and the NLP technology as part of our functionality. Each protocol will be designed to outline the tasks we expect our users to perform, the materials we present to our participants, and our methodologies for data collection, such as discussion and observational strategies. Additionally, each protocol will be designed to include structured data that can be used for analysis, with the flexibility to capture spontaneous insights and observations from our users.

**Experimental Conditions**

To avoid influence from other users, we will primarily organise 1:1 sessions for our study. We will adjust experimental conditions to assess different aspects of our software design. Rather than showing all participants one version of our software, we will present different NLP algorithms and UI versions to help us determine which version will work best. By systematically changing our experimental conditions, we will be able to isolate the impact of specific design choices and algorithmic implementations on the overall experience of our users.

## 3.3 Collection of User Feedback

How we collect user feedback will be critical to our research methodology. It will allow us to receive valuable insights into our system's usability, effectiveness, and user satisfaction. We will carefully craft structured activities for our users, which will help us elicit quantitative and qualitative data.

**Data Collection Methods**

We will employ several methods to collect our required data, which we will revise throughout our development process. We will attempt to follow the best practices to gather high-quality multi-modal feedback from our participants. We intend to collect quantitative feedback on metrics such as task completion rates, time taken per task and error rates to help provide us with objective measures of our system's performance. We will also take note of qualitative feedback, which may include participant observations, verbal feedback, and written feedback to help us understand the preferences of our users, as well as the challenges they may encounter and areas for improvement. Ultimately, we want to combine our quantitative and qualitative data to form a comprehensive assessment of the usability and effectiveness of our system.

**Ethical Considerations**
Our ethical considerations will guide user feedback collection to ensure participants' safety, privacy, and well-being. We will make sure to receive consent from all participants, and measures will be taken to protect sensitive information and maintain confidentiality. Participants can withdraw from the study at any time should they wish to do so. Efforts will also be made to minimise any potential risks or discomfort associated with participation.

## 3.4 Analysis and Incorporation of User Feedback

Our analysis and incorporation of user feedback will be integral components of our workflow's iterative aspect and will guide our system's ongoing development and refinement. Our feedback analysis will involve systematically examining all our collected data to determine any patterns or trends and any insights that may relate to our research objectives.

**Data Analysis**
Our data analysis process will involve several steps to extract meaningful insights from the collected feedback. As aforementioned, quantitively, we will assess task completion rates and examine the percentage of completed tasks to understand the system's usability and effectiveness. We will also analyse the time taken to complete each task and the time spent on each section to gain insights into what the user may be interested in or not be intuitive enough to pick up immediately, hence adding a time cost. This will also assist us in identifying potential bottlenecks as well as areas in which we can make improvements to add efficiency. We will also extend our studies to include Likert scales to help understand our users' preferences and provide comparisons. Qualitatively, we will analyse data on participant observations to understand user behaviour, choices, and challenges. Additionally, we will meticulously analyse verbal feedback from discussions to understand our participants' subjective experiences, perceptions, and suggestions. Our comprehensive data analysis of both quantitative and qualitative measures will allow us to gain a deep understanding of our application's strengths and weaknesses. It will be pivotal in guiding our iterative refinements to achieve an accessible and usable application.

**Iterative Refinement**
We will leverage findings from the analysis of user feedback to inform our iterative software development and drive improvement in both the interface design and NLP functionality. We will use a systematic approach to incorporate this feedback into subsequent development cycles, prioritising aspects such as identified usability issues, user preferences, and performance challenges. Our process of iterative refinements will ensure software changes evolve in alignment with our users' needs and expectations, resulting in improved efficiency, effectiveness, and usability.

# Chapter 4: Design and Specification

This chapter will discuss the reasoning behind our technical choices and explain why they were most suitable for this project. We will also detail the project's requirements using the MoSCoW model and discuss the User Interface and Architecture design choices.

## 4.1 Technical Choices

This section will discuss our choice of programming languages to create our full-stack application. We will also discuss the data, evaluation metrics and algorithms we have decided to integrate into our application.

**Backend Programming Language**

Regarding programming languages for NLP, Python and Java are widely regarded as the two best candidates. Both programming languages are equipped with powerful libraries and tools that can assist developers in fully leveraging NLP technologies. However, Python was chosen over Java due to several compelling reasons. Firstly, Python's ease of readability and concise syntax make it ideal for rapid prototyping and iterative development, especially for projects undertaking an Agile methodology. As our application has continuously evolving requirements, Python's ease of use will simplify the coding process and allow us to dedicate more time to implementing improvements. Python also contains many state-of-the-art NLP libraries widely used by the NLP community. This includes the renowned Transformers library, which has proven invaluable to developers on cutting-edge NLP tasks. With such libraries, the programming language will expedite our software development process and help us provide top-of-the-range NLP functionality. Moreover, with text summarisation being a field which is not covered by university modules, we seek a language with strong community support and resources to educate and facilitate efficient development.

**Frontend Programming Languages**

For our project, we have chosen HTML, CSS, and JavaScript as the languages for our front-end technology. This decision was based on their profound influence on modern web development and how they encompass familiar stylisations of UI elements. This is important as our users have grown accustomed to it through years of browsing, and familiarity will support us in making intuitive design choices to create our interface. HTML will allow us to create the essential structure for our web pages, enabling compatibility amongst different browsers and devices. CSS will allow us to take complete control of our stylisation, in which we can create customised and visually appealing design choices for our user interface. JavaScript will work with these technologies to add interactivity and functionality to our UI elements, helping us to add dynamic behaviour that creates an engaging user experience. Additionally, Python is equipped with Flask, an ideal web application framework that allows developers to build web applications quickly using these languages. Frameworks such as React, Angular, and Vue.js were considered as they come with advanced features. However, given the project deadline, we appreciated that the steep learning curves would be a significant trade-off when time could be better spent developing functionality and user studies. Our framework allows us to maintain simplicity whilst also being able to achieve our project goals.

**Summarisation Models, Evaluation Metrics and Dataset**

Our choice of summarisation algorithms and evaluation metrics was strategically decided to provide our users with comprehensive functions while focusing on practicality and effectiveness. While we considered more niche models and metrics, we noticed lower-quality summaries and less informative metrics through our testing. We therefore decided to explore the current state-of-the-art, from which we have selected a diverse set to cater to different user needs and preferences. To avoid overwhelming our users with choices, we included five summarisation algorithms, three of which are used for Abstractive summarisation and two for Extractive summarisation. Abstractive summarisation produces summaries that more closely resemble human summaries, so we believed

they would be more attractive to our users and, therefore, provided a choice of three. Regarding model variants, we selected the base version (e.g. t5-base instead of t5-small and t5-large) to allow us to make fair comparisons amongst each model. Regarding our evaluation metrics, we decided to include the four we discussed in our literature review because they contribute towards covering a spectrum of evaluation criteria. Though ROUGE, BLEU and METEOR are considered the standard metrics, BERTScore was included due to its additional unique properties. Furthermore, integrating these metrics into our application aligns with best practices in the field, allowing our users to understand how researchers can make informed decisions about a model's performance and its optimisation strategies.

Out of the three datasets we discussed, we decided to focus on the CNN/Daily Mail dataset. Although all datasets were suitable, the CNN/Daily Mail dataset stood out because it aligns with a critical aspect of our project. In addition to being a helpful summarisation tool, we also want users to be able to test the integrity of text summarisation models to ensure fair and unbiased summaries are produced. This dataset consists of a comprehensive collection of news articles paired with bullet-pointed summaries, making it ideal to check if the main points have been covered accurately.

## 4.2 Requirements

To create our application, we must first establish functional and non-functional requirements to guide our development effectively. Functional requirements outline specific behaviours and functionalities our application must serve, whereas non-functional requirements focus on quality attributes and operational constraints. To do so, we will use the MoSCoW model, which assigns requirements to one of the following priority levels: Must, Should, Could & Won't. By categorising our requirements, we create flexibility to incorporate user feedback and adjust our priorities for subsequent iterations. This is particularly beneficial to us as we conduct an HCI project, and we can ensure that the core functionalities required for an intuitive and user-friendly interface are prioritised appropriately. Due to the iterative nature of our project, we anticipate numerous changes and adaptions to our requirements, guided by user feedback and preferences. This could be through adjusting priority levels or the introduction of new requirements. Therefore, we will put our requirements in Appendix B, where the final set of requirements will be the most up-to-date upon completion of the study.

## 4.3 User Interface

This section discusses design considerations for our application's UI, recognising its pivotal role as the primary gateway for user interaction. As our main guidelines, this section will discuss ideas from [32] to ensure we follow the best web design practices. Additionally, we will adhere to the *Web Content Accessibility Guidelines* (WCAG) provided by the *Web Accessibility Initiative* (WAI) to ensure our final design is accessible to all users. [33][34]

To achieve optimal design, it is imperative first to understand which areas of the design commonly require improvement. The *World Wide Web Consortium* (W3C) highlights some aspects, including structure, navigation, consistency, feedback, and control. [35]

By improving consistency, we inadvertently address a few other common issues. Improving consistency requires careful consideration of content and layout, which often improves structure and navigability. Consistency allows users to navigate an application and understand the content's structure, effectively reducing the user's cognitive load more efficiently. This is because it prevents users from relearning how to interact with different application parts. The steps to create a consistent interface include defining the consistency, ensuring that the level of consistency is adequate and finally, analysing the application for areas where consistency may not be wanted [36].

As for navigation, we will design clear and intuitive navigation pathways that guide users through the application seamlessly. We will use a standard navigation bar, which will be consistent and easily accessible from each component, allowing our users to transition between features. Where possible, we will utilise descriptive embedded links to accurately direct users to their chosen destinations, enhancing the application's navigation efficiency. Each of our headings will be unique and descriptive, giving users clear markers to locate specific content of their interests. Through unique and descriptive headers, we mitigate the potential for confusion and can accurately transfer our users to their destinations. While not strictly an aspect of navigation, we will take great care in organising our application's information. Each part of our application will be strategically and appropriately located within a page to ensure it is visible to the user before other details [32].

When applicable, we will implement feedback mechanisms to provide users with timely and informative responses to each action. Our course of action will first implement input validation mechanisms to minimise the probability of user error. Feedback will be provided to guide the user if an error cannot be automatically resolved. The methods we will use to achieve this will include visual cues such as animations, tooltips, and status indicators to keep users fully informed of the status of actions. This could be a successful action or an alert where we aim to guide how to proceed. We will adapt the response type to the action frequency to avoid unnecessary cognitive overload. We will provide subtle responses for standard and frequent actions, but the respective reactions will be more substantial for less frequent but perhaps more critical actions [32].

To control our UI, we, as developers, will try to consider our users' needs. Our application will be developed such that if the user requirements change, this can be reflected in the UI. This could include features for more amateur users, such as additional explanations. Still, for more expert users, this could involve providing shortcuts to improve their efficiency. While we plan to provide our users with a range of features, we will always consider their memory load, which should be minimised, and they should not need to maintain a memory of large amounts of information from one page to another. With added control, we aim to ensure each task is organised so that the minimum number of clicks possible to achieve its intended aim is still required. Ultimately, all the necessary information for the end user should be made available to them in the most convenient way possible for their task. [32]

## 4.4 System Architecture

This section will provide a high-level overview of our system's architecture. In our application, the interaction between the front and back end will adopt a client-server model, where the front end (client) sends requests to the back end (server) for processing. To achieve this, we will use Flask, a microweb framework that contains a built-in web server that can be utilised for testing and development. Flask was chosen because of its lightweight and flexible architecture, which is known for allowing for the rapid growth of web applications. Flask also allows simple user interaction, efficient request handling, processing, and dynamic data representation. Figure 5 provides an overview of how Flask works.

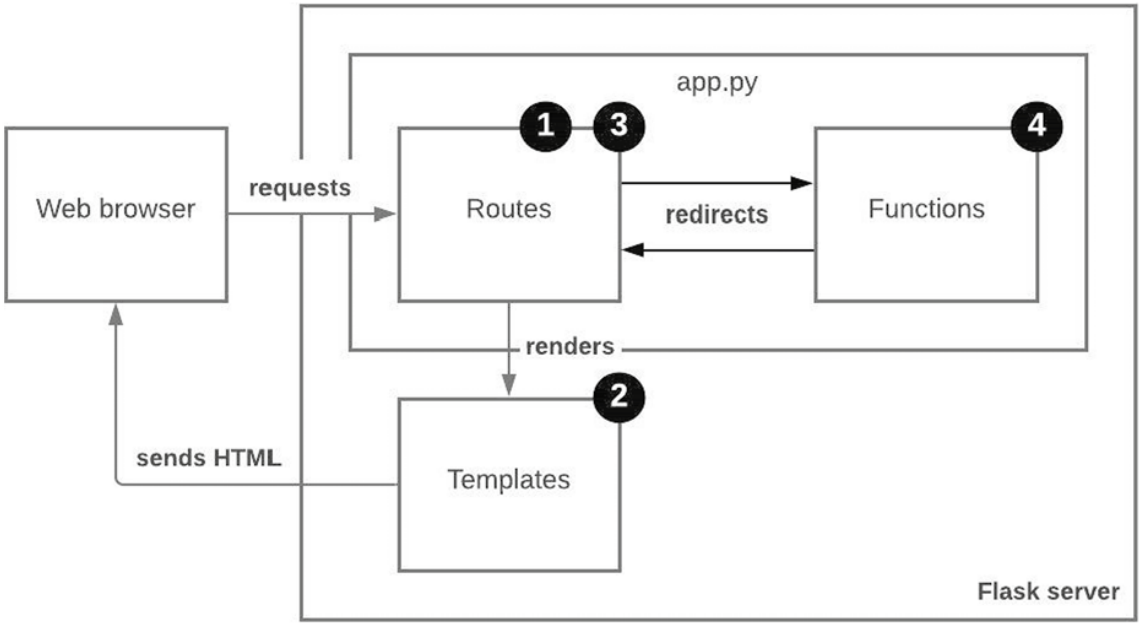



*Figure 5 Overview of Architecture [37]*

**Workflow of Flask Application**

1. **User Interaction**: The process begins when a user interacts with the front end through their web browser, either by inputting text, selecting options, or triggering some action. This then calls a JavaScript function, which performs a quick check to ensure the input is valid. HTTP requests are sent to the flask server if no errors are detected.
2. **Request Handling**: The HTTP requests are received by specific backend API endpoints called routes, handled through App Routing. A route is simply a function in app.py that processes this provided information. The route, hence called function, is decided by the server's routing mechanism, which handles the request based on the URL.
3. **Processing and Response**: Flask executes the Python code, which could run NLP algorithms, process data, or interact with a database. The overall purpose of this function may be to generate summaries or perform analysis, in which an appropriate response is formulated. Once the function has been executed, its result is sent back to the route.
4. **Data Presentation**: Once the backend processing is complete, Flask returns a response to the front end using a template to render an HTML page. The front end then dynamically updates the UI using the templates provided by Flask to present the generated summaries, analysis results, or additional information to the user.

# Chapter 5: Implementation

This section details our implementation procedure, where each component has been isolated, and discusses its specific implementation approach. As each element serves a distinctive purpose and offers unique functionality, each drew from its respective user study and design process. User feedback was collected and incorporated throughout the implementation to refine the application's features, user interface, and performance. Our implementation choices have been largely influenced by prior research from our literature review. Appendix E is a component diagram providing an overview of our application. To see screenshots of each component, see Appendix C. Being an HCI project, we will place less emphasis on our code writing but more on the decision-making behind these implementations.

## 5.1 Home Screen

As the primary entry point for our users, we must provide a comprehensive overview of our application's functionalities and capabilities. A critical analysis of our design choices reveals the strategic approach we used to help improve user engagement, facilitate intuitive navigation, and effectively communicate our key features.


*Figure 6 Colour Palette of Web Application Design*

For the theming of our home screen, which extends throughout our application, we leveraged Bootstrap CSS. Bootstrap is an open-source CSS framework for designing responsive, front-end web applications. It has a wide array of pre-designed components and responsive layout capabilities. By exposing our users to multiple themes and learning their preferences, our final design proved captivating and aesthetically pleasing and could be associated with an educational platform. In earlier iterations, we experimented with designs that included various animations and vibrant colours in an attempt to draw in our users. However, user feedback suggested these to be overwhelming and distracting, leading us to refine our approach to balance visual appeal and usability. Our final design is minimalistic and uses a minimal colour palette. With a minimum contrast ratio of 3.46, our colour choices surpass WGAC's requirements for legible text.

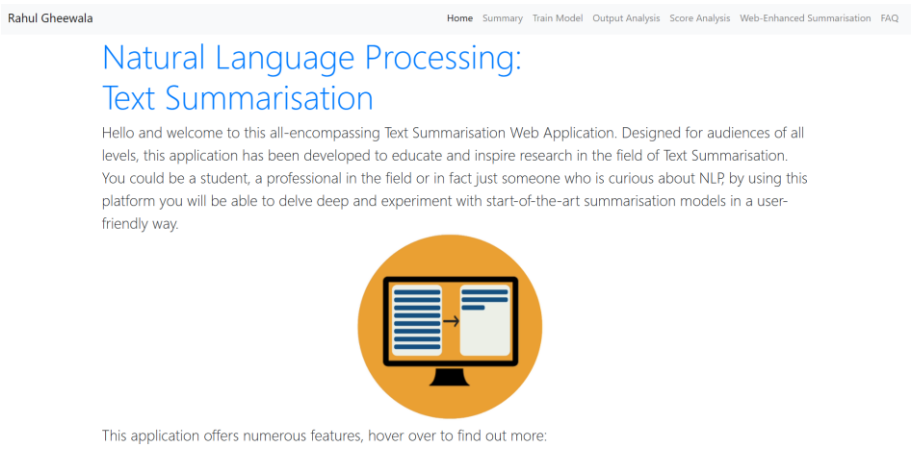



*Figure 7 Top section of Home Screen*

Positioned at the top of the page, we have employed a fixed-top navigation bar as a central hub for accessing each of our application's features. Using fixed positioning, visibility remains consistent throughout the application and offers easy access to essential controls. Each item in the navigation bar is logically ordered, providing intuitive navigation pathways that align with a user's natural progression when using our application. The header section, presented as a jumbotron with a visually appealing background, helps capture users' attention and sets the tone for their interaction with the application. Our choice of a prominent title and a concise introduction effectively communicates the purpose of our application to its users.

The primary purpose of our home screen has been to provide an overview of the application, which is critical in helping users understand the application's purpose and capabilities. Without a grasp of the overall features, users may feel lost or unsure of how to effectively leverage the application. In our first iteration, we presented the features as bullet points, intending to make them easily digestible. However, our observations showed that users were either skipping over or not engaging with them as initially expected. As this was only an observation, we could not concretely conclude that this was the case, as some users might have been fast readers. To test our hypothesis, we used eye-tracking software to help us analyse the visual attention patterns of our users. These results validated our hypothesis that users were skipping over features and not fully engaging. We also noticed a trend that users spent less time reading the 'Training' and 'Analysis' sections, and this was later confirmed by feedback to be because of intimidation of terminology. In our next iteration, we tried using a checkbox prompting our users to confirm they have read all points before proceeding. However, this approach was also ineffective. We realised we would need a new strategy to incentivise our users, such that they would be internally motivated to discover each feature.

Our final solution was to use feature tiles strategically arranged, with each tile serving as a visual cue, highlighting the core functionalities of the application. An icon accompanies each tile, and the feature's name and description are revealed when hovered. We meticulously selected icons that were not immediately inferable, but once a feature was understood, a user could form a visual link. Now, inspired by genuine interest, users are motivated to systematically learn what each icon represents and are no longer intimidated by terminology at first glance. Furthermore, the placement of tiles in a grid layout optimised space utilisation and enhanced the visual appeal of the home screen.


*Figure 8 Eye Tracking performed on Feature Tiles, using GazeRecorder [48]*

| Feature | Average Time using Bullet Points (Group A) | Average Time Using Feature Tiles (Group B) | Percentage Increase (%) |
|---|---|---|---|
| **Summariser** | 9 | 15 | 67 |
| **Model Trainer** | 3 | 12 | 300 |
| **Output Analysis** | 4 | 13 | 225 |
| **Score Analysis** | 3 | 13 | 333 |
| **Web-Enhanced Summarisation** | 9 | 12 | 33 |
| **FAQ** | 4 | 7 | 75 |

*Table showcasing increase in average engagement times (to the nearest second), using groups A and B (N = 30)*

For this experiment, we used GazeRecorder's ability to capture the time a user viewed a screen section. The software requires a robust calibration process, therefore returning timings and spatial precision with high accuracy. After each recording, we drew boxes around regions of interest. Then, the average time spent on each box was calculated for each group of 15. We can see significant improvements in user engagement, with a percentage increase in every feature. To avoid the text being a factor, we carefully restructured the new descriptions to have the same number of words and style of writing. Therefore, we can conclude that feature tiling is the only independent factor, making users spend more time reading each description.

## 5.2 Summariser

This component allows our users to input text and generate customisable summaries by modifying parameters. For this component, including those which follow, we have strategically selected *Graphical User Interface* (GUI) elements to help improve the usability and effectiveness of our software. Here, we have chosen elements such as radio buttons, dropdown menus, input fields, and buttons considering usability principles and user preferences.

By default, the page is preloaded with 'Abstractive', but selecting 'Extractive' loads its respective models and parameters. To choose between 'Abstractive' and 'Extractive' summarisation, we have chosen radio buttons as they offer a clear binary choice, aligning with design simplicity principles. We have adopted a dropdown menu when selecting a model, as it provides a compact but accessible way to choose between several options. They also allow for the optimisation of screen space, letting users make quick decisions without being overwhelmed with clutter. For options such as length and the number of beams, we provide input fields that allow input of numerical values and precise control. We also use sliders to adjust parameters such as the length penalty. They offer an interactive dimension that visually represents the parameter's value, allowing users to change them within a defined range intuitively. Finally, a large, bold 'Generate Summary' button is used as a clear indicator to start the summarisation process. The button has a distinct visual appearance and is positioned atop the output box, making it intuitive and straightforward to understand its purpose.

We have iterated our original design concerning user feedback to include better input validation methods and parameter default values. We also provide more information to describe the influence of each parameter. Instead of flagging invalid values and expecting correction, our code now attempts to perform this automatically to minimise manual intervention. For example, if the maximum length is chosen to be less than the minimum, it is corrected to equal the minimum size. If the number of beams exceeds the threshold of 12 (empirically set), it is set to max. Additionally, radio buttons and sliders provide natural bounds, preventing errors. We also turn off parameter fields unless prior inputs on which they depend have been provided, ensuring an error-free experience. With regards to default values, these were empirically set to offer a good starting point for users, reducing the need for extensive adjustment. We have also included parameter information

in tooltips, easily identifiable in blue, and information is presented when hovered over. Through these amendments, our research proves users have had an improved experience.

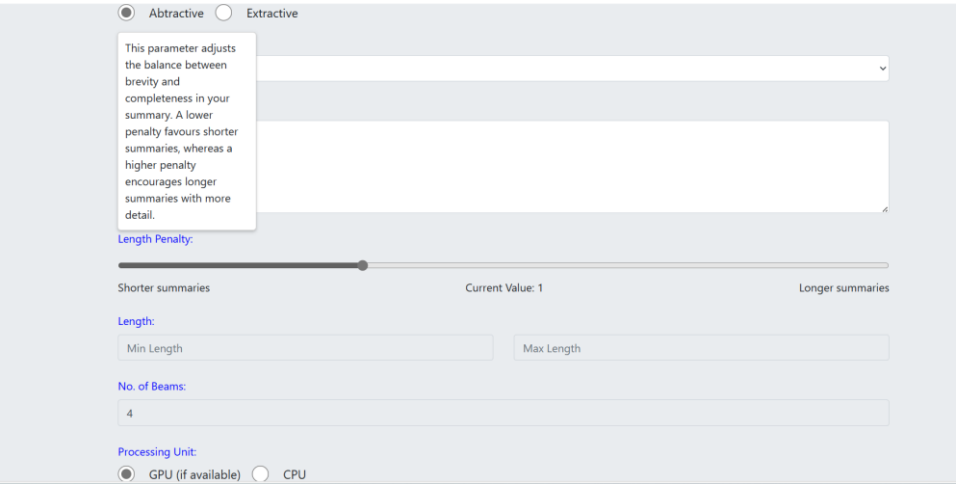



*Figure 9 Screenshot showing Tooltip and GUI elements for Abstractive Option*

Below, for completeness, we provide a view of the Extractive option and its respective parameters.

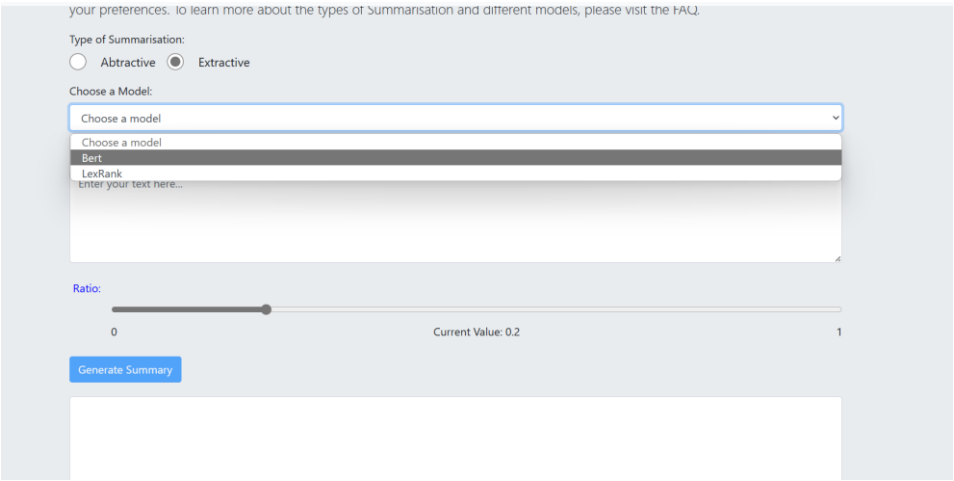



*Figure 10 View of Extractive option*

## 5.3 Model Training

This component provides our users with the functionality to train summarisation models using their datasets, allowing users to observe first-hand the effects of fine-tuning models using biased or domain-specific data. As we were restricted by our users' available time to perform training, we instead took the opportunity to provide walkthrough demonstrations, showcasing the training process and outputs using a model the author had trained as preparation.

We have decided to offer only training capabilities for Abstractive models. Given the substantial time investment required to train models, from our experimentation, we have found that Abstractive models yield more interesting results, making it more worthwhile for the user. We use Hugging Face's Seq2SeqTrainer for model training, part of the Transformer library. We chose this class because it efficiently implements sequence-to-sequence training, streamlining the training process. Seq2SeqTrainer's parameters allow performance optimisation, management of computational resources, and organisation of training data efficiently. Similar to 'Summariser', we have used GUI elements to obtain parameter information from a user accurately.

Once a model is selected, users are prompted to load their training data in the required Excel/CSV format. These formats were employed because they are highly familiar to non-technical users. When loaded, the UI provides feedback on whether the correct file type has been supplied.

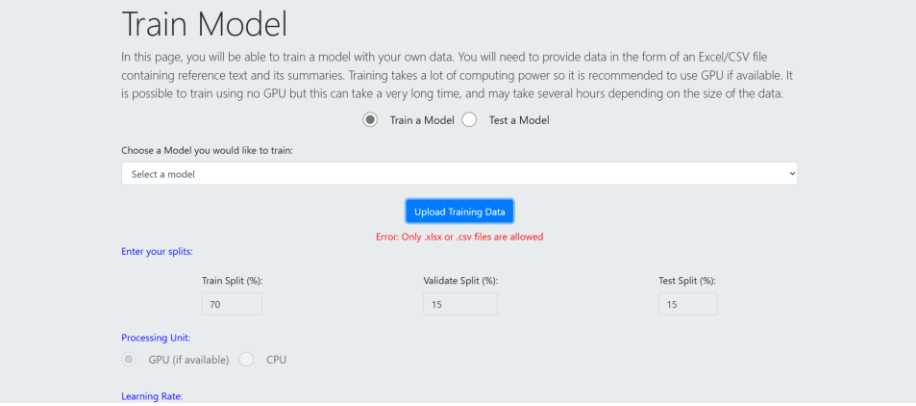



*Figure 11 Case where the incorrect file type*

If a correct file type is provided, the user is prompted to enter the column names for the original and target value columns. If a match exists, the UI offers further feedback, allowing progression.

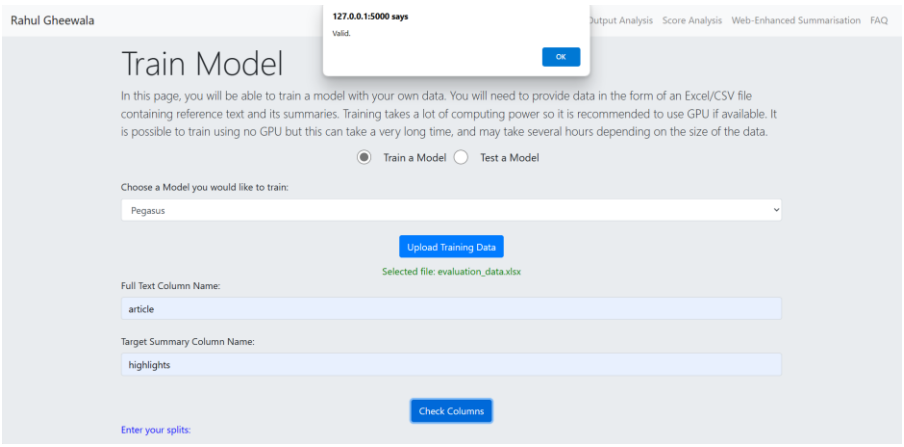



*Figure 12 Case where data column values are recognised.*

At this step, the user's minimum requirement is to enter a model name, enabling the 'Train Model' button. However, users can change parameters and are advised to use information from the tooltips. On clicking 'Train Model,' the UI alerts the user that the training process has begun.

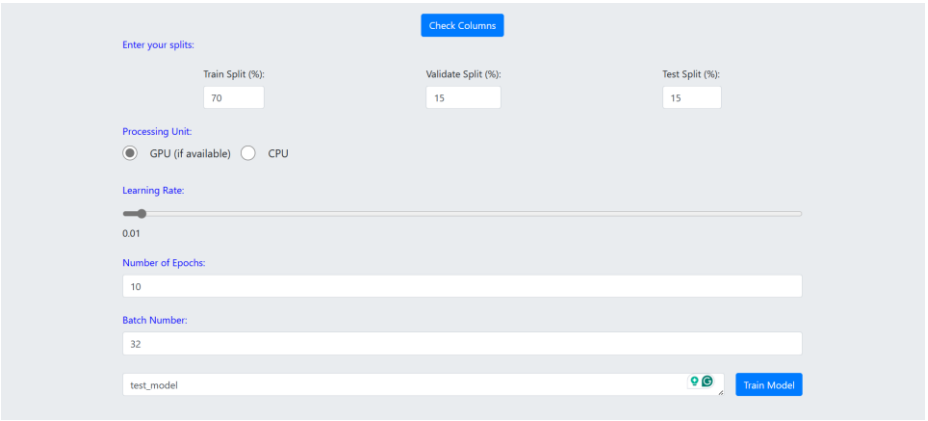



*Figure 13 Final Step for Initiating Training*

Once the training process has been completed, the user is again notified and recommended to change to ‘Test a Model,’ where the new model is available. Overall, the page's layout and structure are intuitive, with a clear separation between options to train a new model and test an existing one.

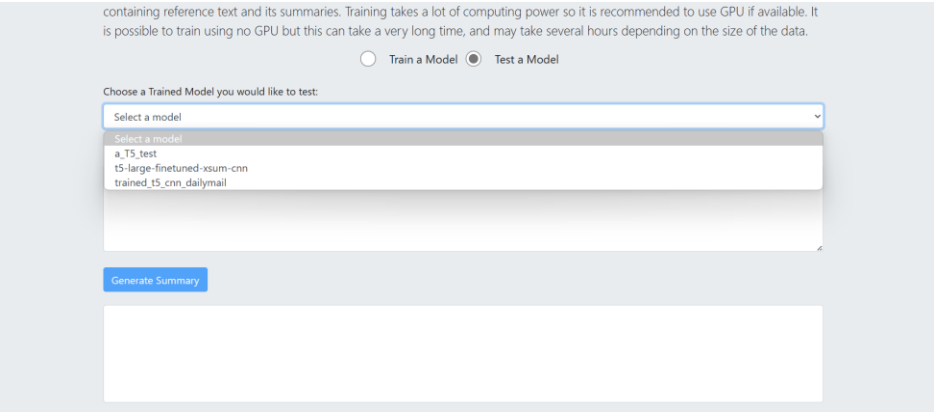



*Figure 14 New models in 'Test a Model'*

Throughout our demonstration, users provided valuable feedback and guided our implementation. Although we offered a trained model, users wanted to see the results of a model trained with the complete dataset for comparative purposes. Therefore, we have included ‘t5-large-finetuned-xsum-cnn’ from Hugging Face as an option (trained using our project’s choice of dataset). A second valuable recommendation was to improve our input validation for training splits, as our current approach was inefficient. Previously, we used a summation check to ensure values summed to 100, requiring cognitive effort from the user. If mistakes were made, this required manual adjustments, adding friction to their workflow. Therefore, we addressed the issue using dynamic adjustments to recalculate the training splits based on user input automatically. When a user enters a percentage for one split, the system adjusts the remaining splits accordingly to ensure they sum up to 100%. Pseudocode of the algorithm is provided on the next page. Although the algorithm produces mathematically correct splits, hence compilation, users are still warned that significant deviation from the proposed default splits may cause poor training performance.

**Algorithm 1** Update Splits

```
1: function updateSplits(changed)
2:     train ← parseInt(document.getElementById('train').value)or0
3:     validate ← parseInt(document.getElementById('validate').value)or0
4:     test ← parseInt(document.getElementById('test').value)or0
5:     if changed = 'train' then
6:         adjustTrain(train, validate, test)
7:     else if changed = 'validate' then
8:         adjustValidate(train, validate, test)
9:     else
10:        adjustTest(train, validate, test)
11:    end if
12: end function
13: function adjustTrain(train, validate, test)
14:    remaining ← 100 − train
15:    if remaining < 0 then
16:        train ← 100
17:        validate ← 0
18:        test ← 0
19:    else
20:        if validate > remaining then
21:            validate ← remaining
22:            test ← 0
23:        else
24:            test ← remaining − validate
25:        end if
26:    end if
27: end function
28: function adjustValidate(train, validate, test)
29:    remaining ← 100 − validate
30:    if remaining < 0 then
31:        validate ← 100
32:        train ← 0
33:        test ← 0
34:    else
35:        if train > remaining then
36:            train ← remaining
37:            test ← 0
38:        else
39:            test ← remaining − train
40:        end if
41:    end if
42: end function
43: function adjustTest(train, validate, test)
44:    remaining ← 100 − test
45:    if remaining < 0 then
46:        test ← 100
47:        train ← 0
48:        validate ← 0
49:    else
50:        if train > remaining then
51:            train ← remaining
52:            validate ← 0
53:        else
54:            validate ← remaining − train
55:        end if
56:    end if
57: end function
```

1

*Figure 15 Training Split Algorithm*

## 5.4 Output Analysis

**Visual Output Analysis**

'Visual Output Analysis' provides visual insights into summarisation, allowing interaction with generated summaries. Using visualisations, users can quickly gain insights and make connections between the relationships of words in a generated summary and their sentence of origin. The interactive nature of visualisations allows deeper exploration and analysis, helping users make more informed assessments of a model's performance. Benefits include:

- **Enhanced Understanding**: Users can directly map a word to its most probable source, which can help clarify any ambiguity with meanings that may be lost in a condensed summary.
- **Fact Checking**: Abstractive models sometimes suffer from hallucinations, content that cannot be directly inferred from a source text. While they could be factual hallucinations that provide helpful background information [38], they may also be factually incorrect, raising concerns.
- **Contextual Relevance**: Seeing a word's origin can provide transparency, helping a user understand if it has been taken out of context or misrepresented during the summarisation process.

This feature's development was more challenging than first anticipated. For Extractive summarisation, it is trivial to map a generated sentence to its corresponding source sentence, as it is essentially a subset. However, it is not for Abstractive summarisation, as sentences are merged during summarisation. For example, a word in a generated sentence may exist in many sentences of the original input. This raises the fundamental question: from which specific sentence did each word originate?

Our first approach used sentence similarity using a window of words. However, this worked poorly due to context constraints and the proximity of target words. Instead, we introduce an algorithm that relies on the proximity of common adjacent target words among the input sentences and the generated summary. Additionally, we rely on well-structured summary outputs where sentences are used, a behaviour we have witnessed with our algorithms. Due to their uniqueness in context, we only consider verbs and nouns as target words. Essentially, we assume that if words are closely related in a summary sentence, then it is more probable that they exist together in the sentence of their origin. We perform summation using the reciprocal of the distances, where closer words add higher scores. Compared to our actual implementation, our pseudocode omits details and is simplified considerably to improve understandability.

**Algorithm 2** Mapping Words from Summary to Original Sentences

**Require:** Summary, Original Sentences (containing only verbs and nouns)

```
% Only consider verbs and nouns in the sentences and summary
Initialize an empty mapping list: mapping_list
for each word w in Summary do
    Initialize an empty scores dictionary: scores
    for each sentence s in Original Sentences do
        if w exists in s then
            Find neighbouring words of w: neighbours
            Initialize distance sum: dist_sum = 0
            for each word n in neighbours do
                Calculate distance of n to w in s: dist
                dist_sum+ = 1/dist
            end for
            Update scores[s] with dist_sum
        end if
    end for
    Find sentence s_max with the highest score in scores
    Add w and s_max to mapping_list
end for
return mapping_list
```

*Figure 16 Algorithm for creating [word: sentence] pairs.*

Based on user feedback, our final implementation is designed with the summarised version on the left and the input on the right. In terms of the frontend implementation, we make each key in mapping_list hoverable, and once hovered, CSS and JavaScript are used to dynamically highlight the word and its corresponding value of the dictionary. Although the inputted text is already above, users have preferred horizontal side-by-side comparisons. Below, 'Birmingham' has been hovered on, and we can see an example of factual hallucination. With many computer science departments in Birmingham, we should not concretely assume the University of Birmingham.

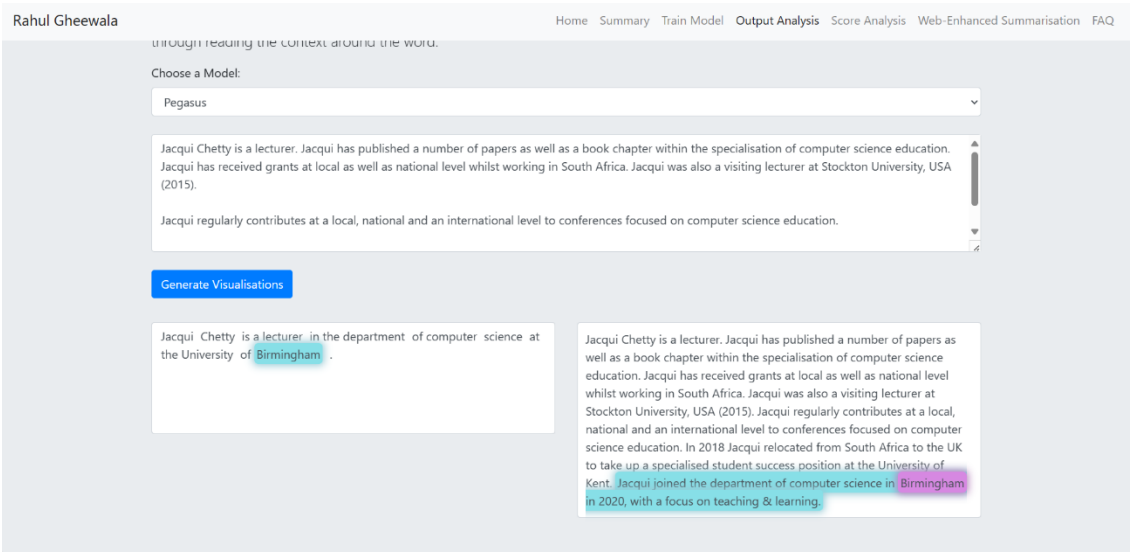



*Figure 17 Visual Output Analysis with Example*

**User Output Analysis**

For this feature, user data is collected as ratings and summaries, which can later be used for insights and training data to help advance the field.

Regarding rating a summary, the page loads a random entry from the CNN dataset and a corresponding summary generated by a random model unknown to the user. This data is stored in an Excel file with 1000 entries, where 200 have been allocated to each model through shuffling. Randomness ensures no prior cognitive bias and data ordering, providing representative ratings. Summaries were also generated in advance instead of on load, providing a more seamless experience for the user. Once a user has read both the input and the summary, they must give a rating for coherence, conciseness, readability, content and overall. This is completed using a star scoring mechanism, where each metric has a tooltip assisting the user. On submission, the data is stored in the file, and the current entries on the screen are replaced. This data can be viewed in 'Score Analysis', which will be discussed in the next section. We chose stars due to their simplicity, ease of understanding, and user familiarity.

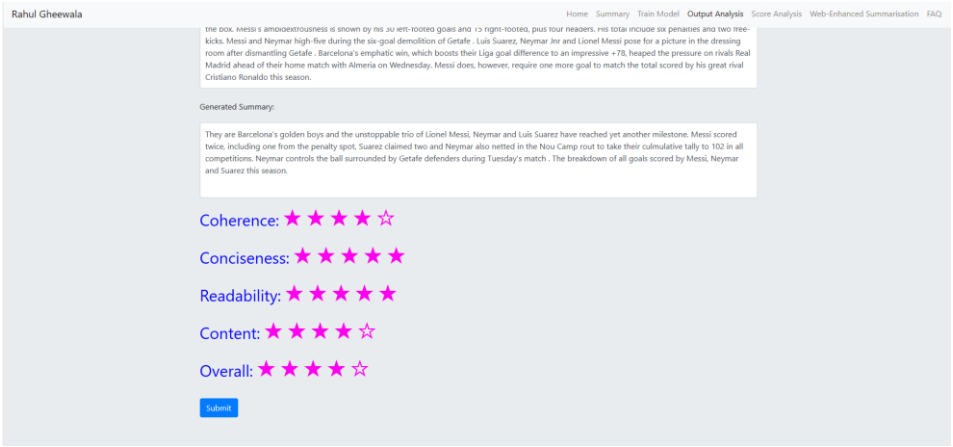



*Figure 18 'Rate a Summary' section.*

'Provide a Summary' similarly takes a user's version of a summary and stores it in an Excel file for later use. We have refrained from guiding because we believe users should be free to define their summaries based on their understanding and interpretation of the content.

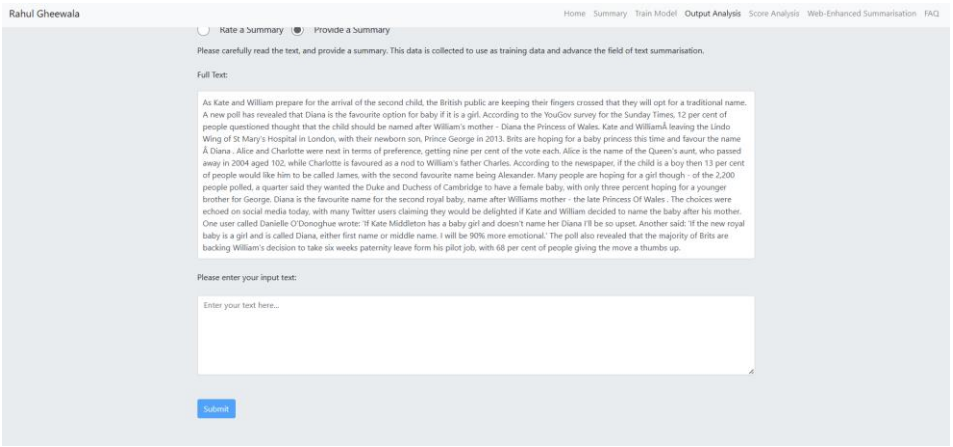



*Figure 19 'Provide a Summary' section*

## 5.5 Score Analysis

In this feature, we introduced state-of-the-art text summarisation evaluation methods along with additional metrics to allow comparison between different model outputs. Checkboxes enable users to select multiple models for comparison, providing flexibility in the evaluation process. If only one is chosen, this model's performance results are displayed.

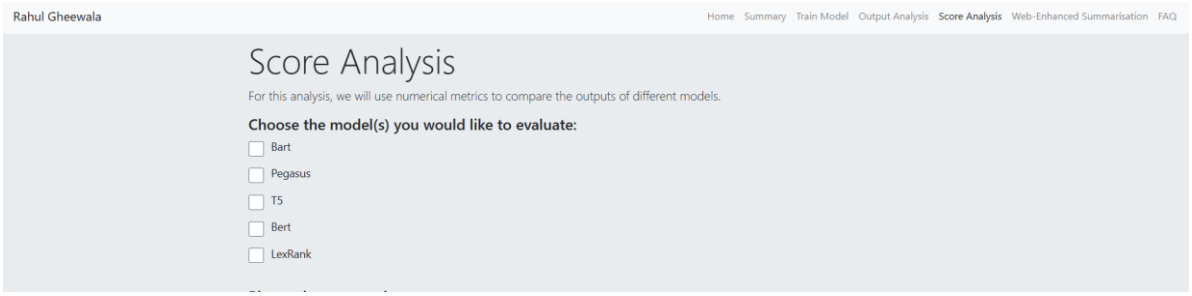



*Figure 20 Choosing Models for Evaluation*

A user can then choose between 'Single', 'Multiple' and 'Domain', representing the type of input the user provides to the system. If 'Single' is selected, a user inputs one summary and its 'expected' version of a summary.

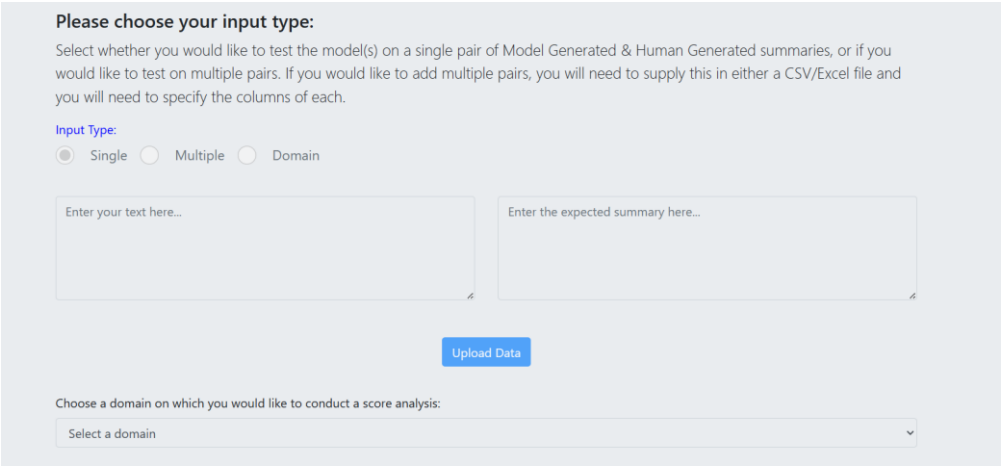



*Figure 21 Single Option*

If 'Multiple' is chosen, a user provides a file with multiple pairs and their respective columns.

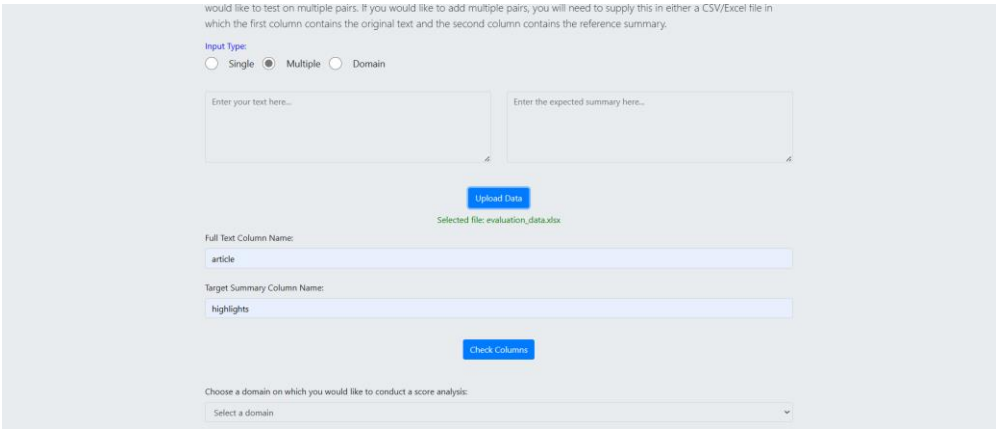



*Figure 22 Multiple Option*

A user can choose ‘Domain’ should they not want to provide their data or would like to test how a model’s performance is influenced using different domains.

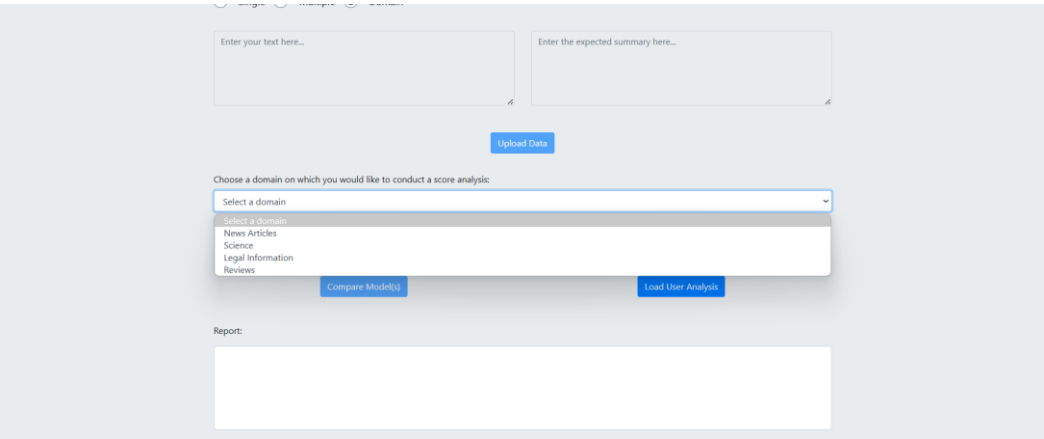



*Figure 23 Domain Option*

Upon choosing model(s) and inputting data, users are presented with the range of options displayed in Figure 24. The first four options have been discussed extensively prior in the report. In response to suggestions from our users, we introduce ‘Semantic’, ‘Similarity’ and ‘Time’. Both ‘Semantic’ and ‘Similarity’ compare inputs with their summary, providing insight into the degree of deviation. A more considerable discrepancy may be inferred as a higher likelihood of misrepresentation in the summary. Furthermore, ‘Time’ calculates the time required to generate the summary, which is helpful to users for whom time efficiency is a deciding factor. Combined, our metrics further help our users make informed decisions.

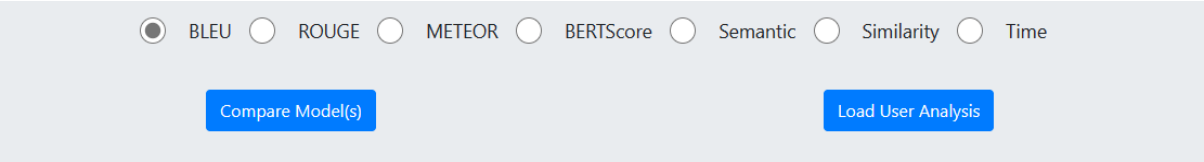



*Figure 24 Metrics for Evaluation*

**Metric Values**

For ‘Single’ inputs, we relay the metric values provided by the evaluation function. However, for ‘Multiple’ values, we calculate the scores for each entry and average these for our final graph representation. The ‘Domain’ option is just an extension of ‘Multiple’ but with five provided entries of domain-specific data. While providing singular inputs allows for faster generation of graphs, users can give a data set to analyse a model’s behaviour more reliably as outliers have less impact. Additionally, while a user could independently generate results for each model, we allow side-by-side views for easy comparison.

| Metric | Source |
|---|---|
| **ROUGE** | ‘evaluate’ library |
| **BLEU** | ‘evaluate’ library |
| **METEOR** | ‘meteor_score’ from nltk.translate |
| **BERTScore** | ‘evaluate’ library |
| **Semantic** | ‘SentimentIntensityAnalyzer’ from nltk.sentiment |
| **Similarity** | ‘similarity’ from Spacy |
| **Time** | ‘time’ library (built-in Python) |

*Table to show the source of Metrics*

**Displaying Results**

In our process, we use the library ‘matplotlib’ to visually represent the results of each metric. We provide each model's sub-metrics (name and label) and give a title depending on the input type. By employing a combination of colours and a bar chart format, we aimed to facilitate straightforward comparisons between different models. Our report provides information to our users to help interpret each result.

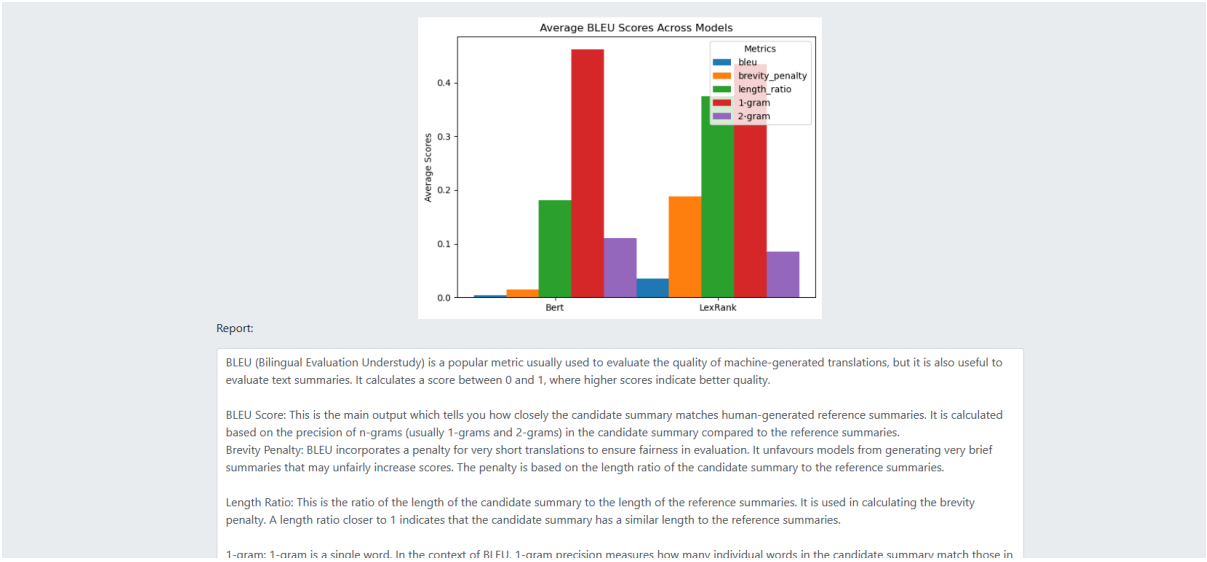



*Figure 25 Evaluation of Bert and LexRank on Science domain*

The ‘Load User Analysis’ is a standalone feature that does not require any user input to be enabled. This button loads a graphical representation of averaged scores of the summary ratings (stored in an Excel file) from ‘Output Analysis’. Rather than solely relying on computed metrics, this graph allows users to learn the consensus amongst fellow users.

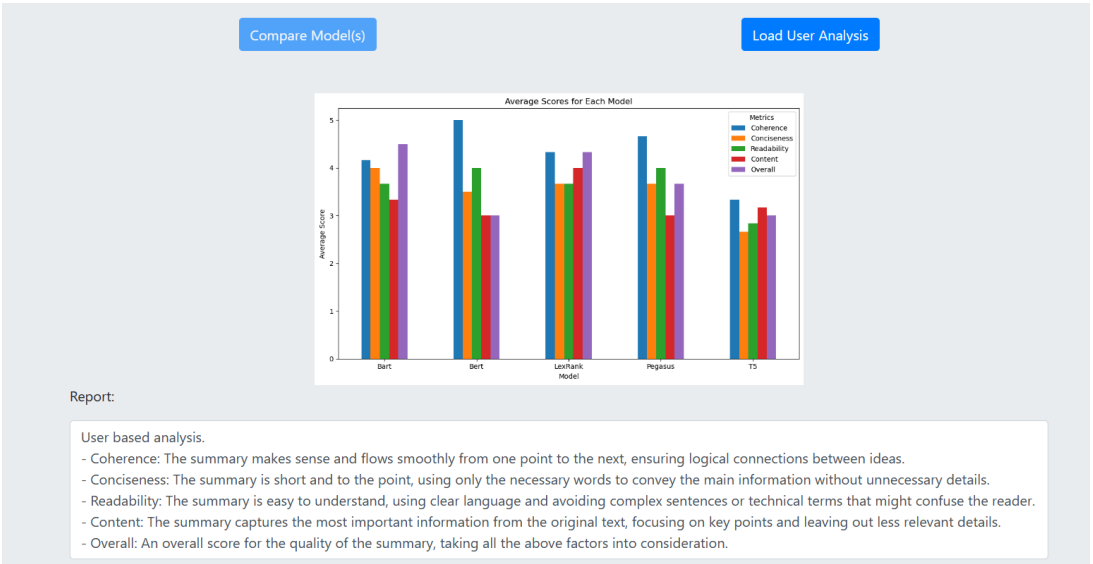



*Figure 26 User Analysis*

## 5.6 Web-Enhanced Summarisation

We have provided our users with predefined features for much of our application and received feedback for iterative improvements. However, this functionality has been wholly recommended and refined by our users' suggestions and requirements from start to finish. From our study Appendix A.2.3, we learned our users were sometimes discouraged from using summaries because of the loss of crucial contextual information. As a summary reduces the amount of textual content, inevitably, data is lost, meaning each word now carries more weight in representing the full context. If a user cannot understand a particular word, it can be detrimental in trying to understand a full context. Naturally, users may browse this item for more information, which increases consumption time. Our feature streamlines this process, helping users stay focused on their summaries. Users may not need supplementary information, but should they, accessibility is provided as additional insights.

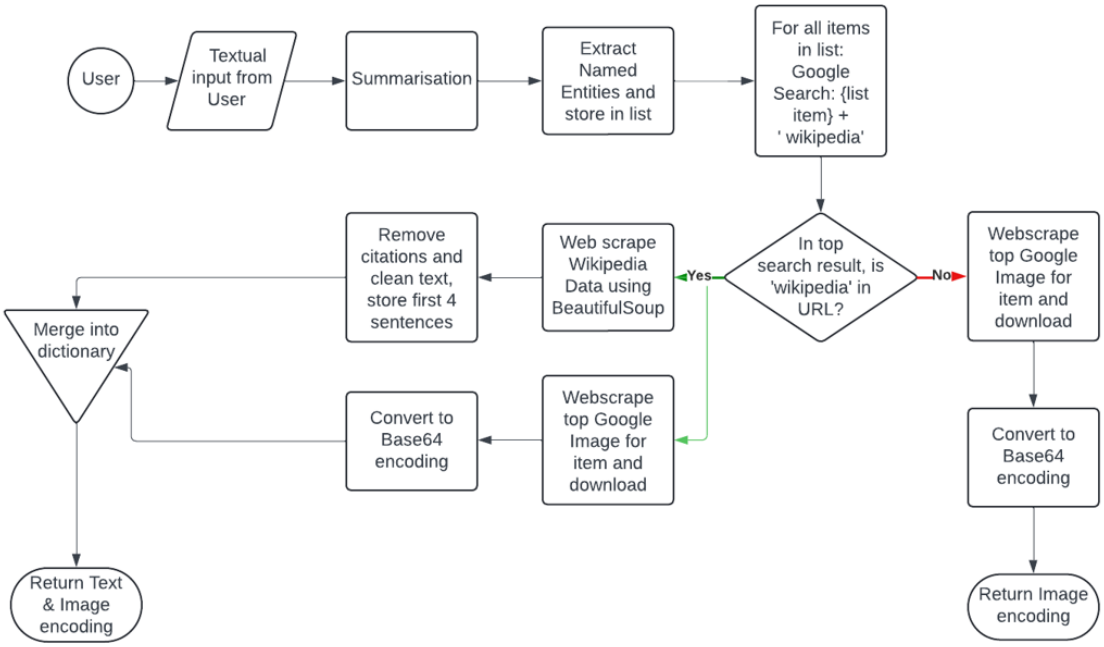



*Figure 27 UML Activity Diagram for Web-Enhanced Summarisation*

Figure 27 shows the process we use to extract our information. We perform web scraping using BeautifulSoup, which attempts to reliably extract data from web pages with varying structures and formatting. However, our experimentation showed that some websites were parsed more accurately than others, making them unreliable. Therefore, we resorted to using only Wikipedia articles, which our testing proved to be consistently accurate due to its processing. Even though an entity in the summary may have a better search result, we favour accuracy over risking poorly formatted representations. We only return a Google image if no Wikipedia page exists for an item.

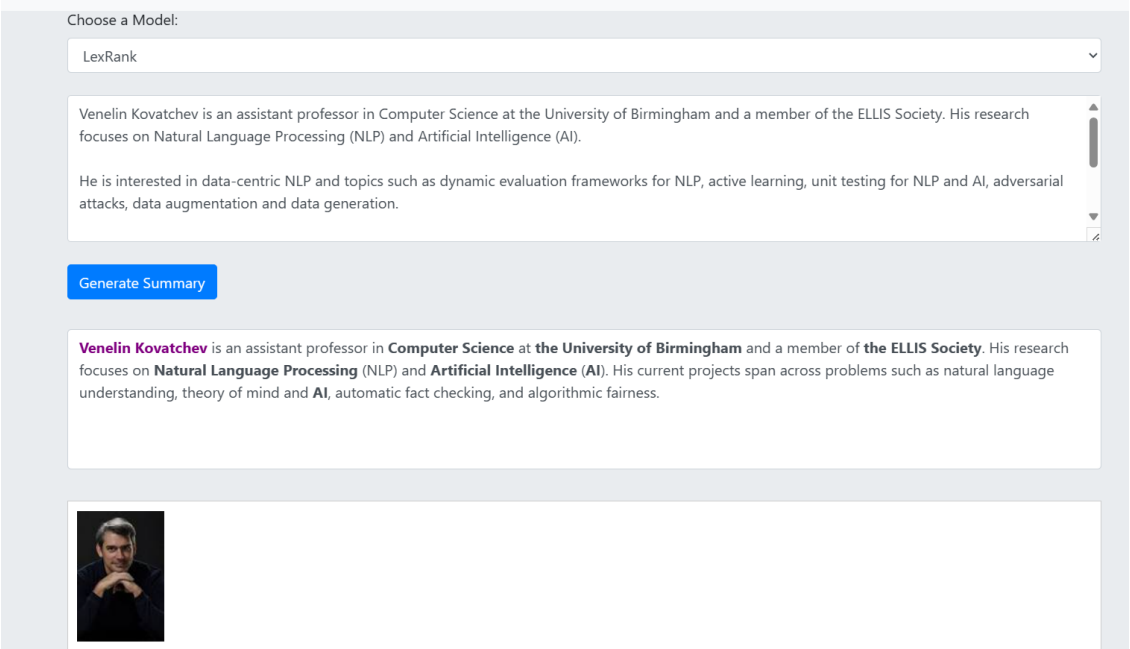



*Figure 28 Input text adapted from vkovatchev.com, showing an example with no Wikipedia article found.*

For entities labelled as 'PERSON', we require at least two tokens (capturing a name and surname) to increase the likelihood of fetching information representing the correct individual.

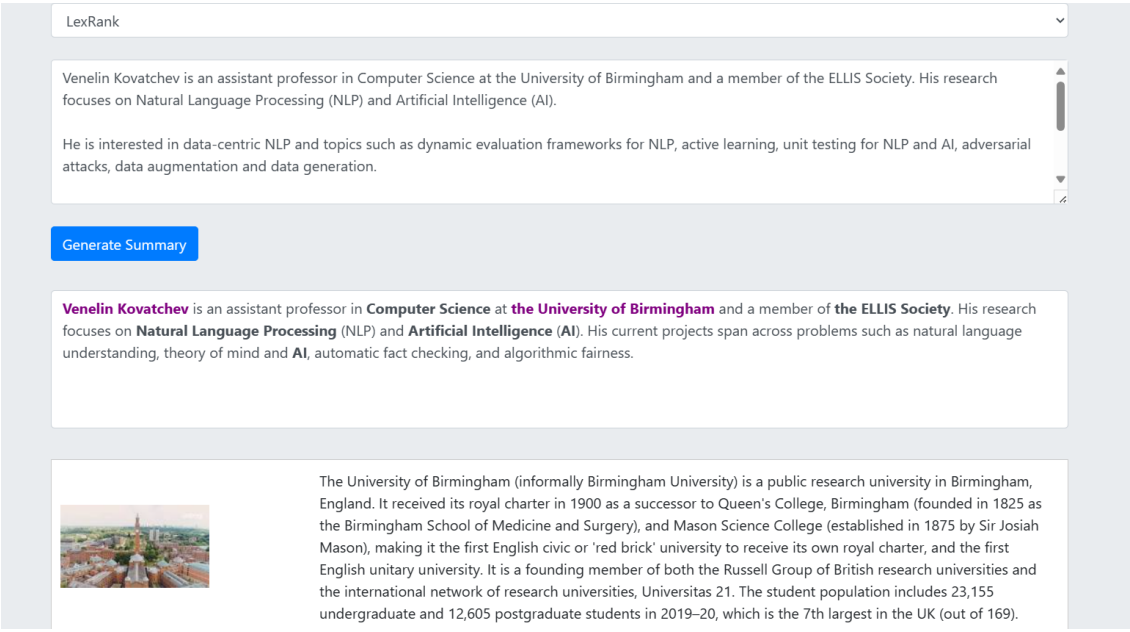



*Figure 29 Example showing a found Wikipedia article*

Users can click on bold items, which are extracted entities, to load the supplementary information. Once clicked, the item is coloured purple. This visual change was requested to help users track what they have visited. Our study found that the top four sentences of any Wikipedia article provide enough contextual information to meet our users' needs.

## 5.7 FAQ

The FAQ section offers users quick answers to common queries we observed during feedback sessions. This component will help solve potential issues for future users of our platform. This feature reflects our proactive approach to addressing users' needs and improving overall usability. By anticipating common questions and concerns, we streamlined user interaction and allowed independence for self-sufficiency.

In our implementation, we employ collapsible accordion-style panels, leveraging animations, to present each question and answer in a structured format. This design choice allows users to easily navigate through the FAQ content, expanding only the section relevant to their queries while keeping the rest condensed for a clutter-free interface.

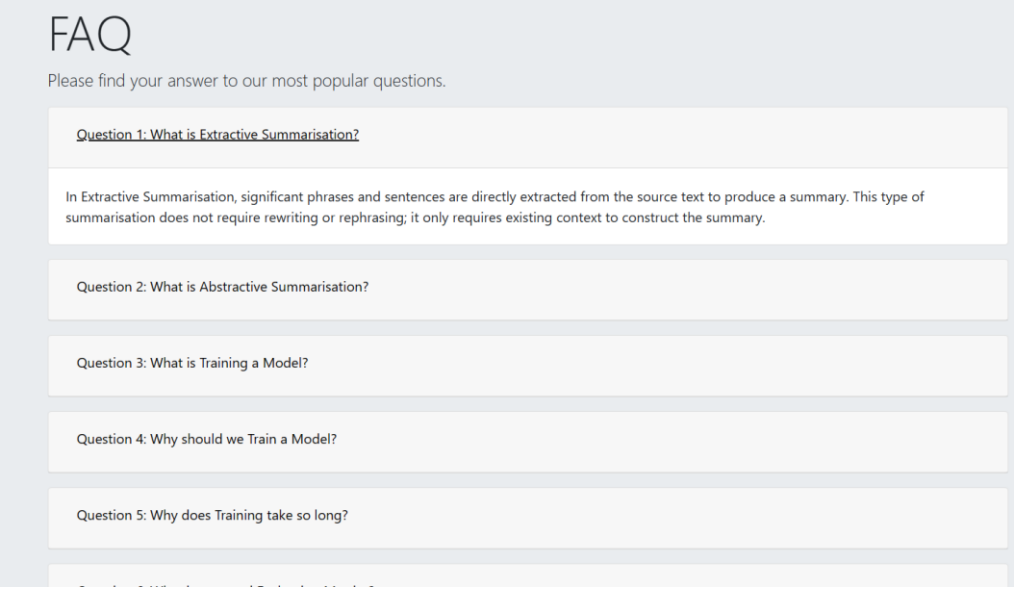



*Figure 30 FAQ Section*

## 5.8 Testing

During our development, testing has played a crucial role in ensuring our system is both useable and reliable. To debug the front end of our application, we used ‘debugger’, a JavaScript tool used by developers to inspect and debug code during runtime. Previously, we used console logging but changed to ‘debugger’ as it offers features such as breakpoints, code navigation and real-time variable inspection, which has been ideal for fast iterative development. Furthermore, ‘debugger’ allows us to skip into functions and step through code one line at a time, helping us to notice bugs more quickly. Using ‘debugger’, we tested our test cases and asserted whether each UI element performed its intended behaviour.

```
412
413             }),
414         })
415         .then(response => response.json())
416         .then(data => {
417             document.getElementById('summaryResult').innerText = data.summary;
418             debugger
419         })
420         .catch((error) => {
421             console.error('Error:', error);
422             document.getElementById('summaryResult').innerText = 'Error generating summary.';
423         });
424     }
425     </script>
426     </body>
427     </html>
```

*Figure 31 Example of using 'debugger'*

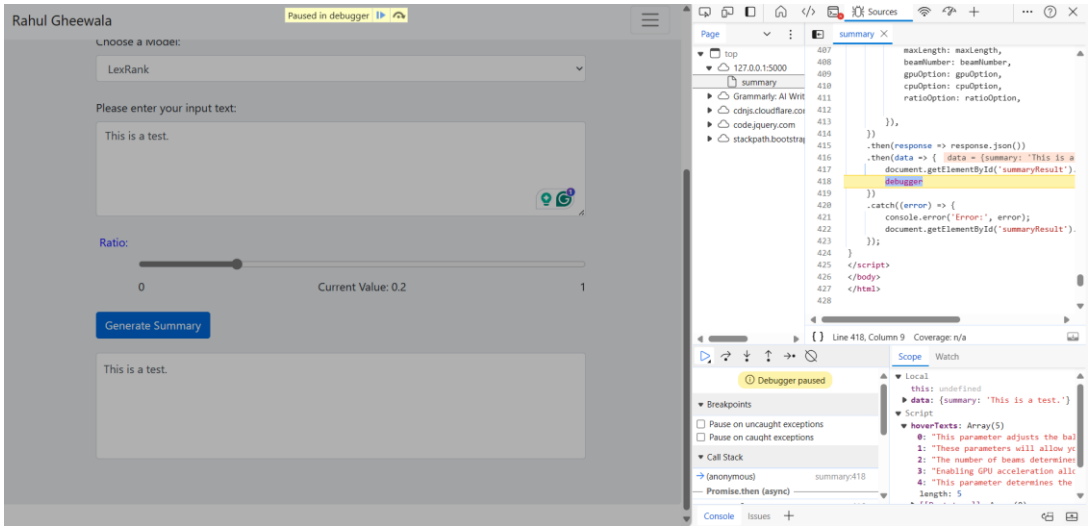



*Figure 32 JavaScript execution paused at a breakpoint, allowing inspection of variable values.*

For debugging HTML and CSS, we benefited from Brackets [49], a code editor for web development. Brackets offer a live preview, meaning every time we made changes, they were instantly viewable on screen. This was highly beneficial for modifying prototypes on the fly in response to immediate user feedback. Not needing to run our large project every change accelerated our development.

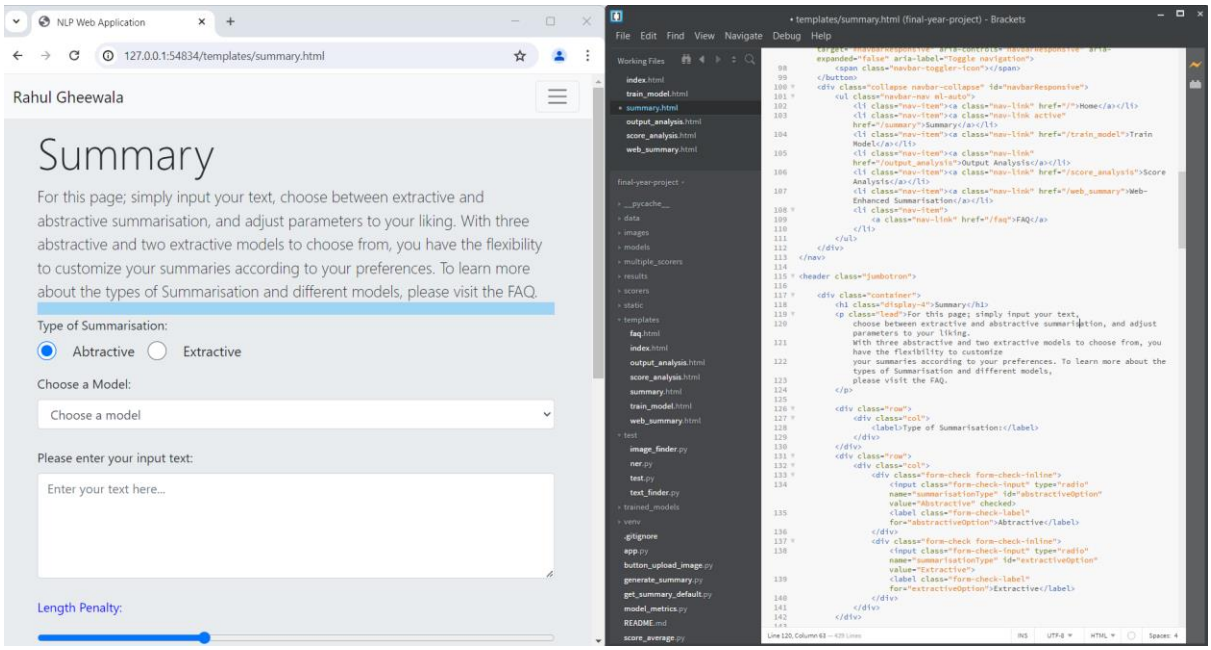



*Figure 33 Screenshot showing development using Brackets.*

As part of our testing, we performed a comprehensive scan for each component using axe DevTool's 'Web Accessibility Testing' Chrome extension [51]. This tool is 'driven by the world's most trusted accessibility testing engine, axe-core'. Using this, we iteratively amended our UI components until the scan no longer displayed errors, ensuring we strictly followed all WCAG. See Figure 34.

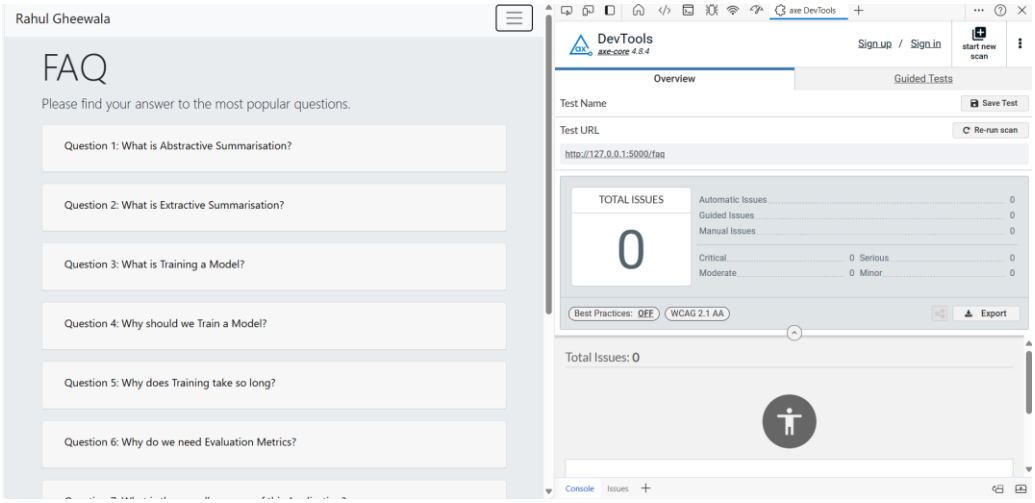



*Figure 34 DevTool's scan on FAQ section*

As for user testing, our project is HCI-based, so our report will focus on System and Acceptance testing.

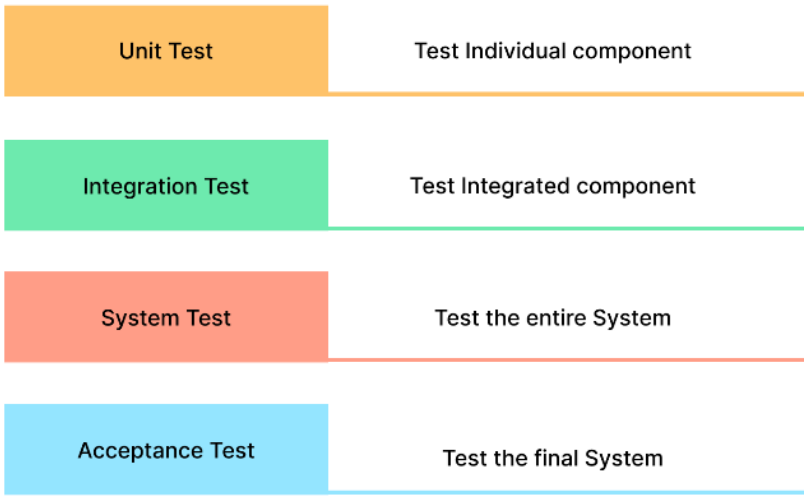



*Figure 35 Software Testing Hierarchy [39]*

System Testing involves checking if a system meets the requirements specification, including functional and non-functional requirements. On the other hand, acceptance testing checks if the system meets the end user's requirements. System Testing was performed using both the developer and testers, and Acceptance Testing took place during user sessions. As a significant aspect of our project, methods for both types will be discussed extensively in the evaluation.

# Chapter 6: Evaluation

A common challenge with evaluating HCI projects is differentiating between evidence and opinion, especially when assessing user experience. While opinions can provide valuable insights, they are subjective and may not always align with the application's usability and effectiveness. Therefore, it has been essential to rely on evidence-based methods for evaluating our application's usability and user experience.

## 6.1 System Functionality

This section evaluates whether our system has provided enough functionality for users to perform tasks and efficiently. We will also assess the user's performance and effectiveness in using the system. To perform our testing, we primarily draw from Appendix A.1 to conclude our users' proficiency with the system. We also use KLM (keystroke-level model) [50], which belongs to the family of GOMS models. The KLM model aims to predict the time it takes an expert user to perform a task on a computer system and is used to estimate the efficiency of user interactions with a system.

To begin with, we provide line graphs representing data from Appendix A.1. The relevant study should be read to better understand the details of the following experiments. To achieve these results, we organised remote and in-person sessions for our user testing phases, allowing interaction with our system. When done remotely, Zoom's 'Remote Connect' option was used to control our PC.

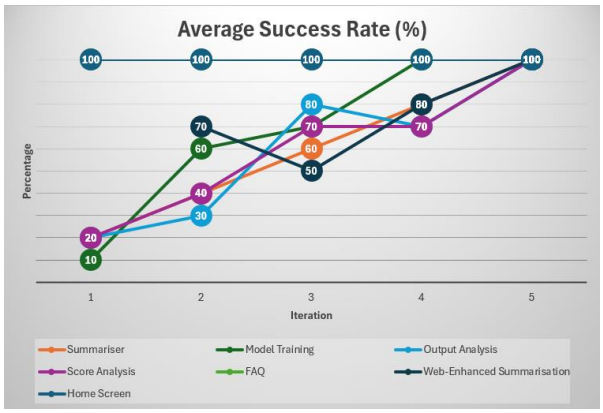



*Figure 36 Average Success Rate*

These are the average percentage of users who completed the task from the study across iterations. Where drops can be seen, this was concluded to be because of the introduction of new implementations, which either introduced new errors or confusion. By the final iteration, it was evident that all users were able to complete tasks for all features, as displayed by a 100% success rate. This suggests the system successfully provided intuitive and usable functionality for all features.


*Figure 37 Errors identified by users*

Errors were identified using our users' judgement, and we summed all those identified throughout a testing iteration. While we agreed with most of these errors, some were false positives. These were highly effective in helping us understand how our system was misinterpreted, allowing us to improve our explanations and overall understanding. By the end of our testing phases, our users had identified 0 errors for all features.

We timed how long it took for each of those tasks that could be completed. To allow comparability, users were discouraged from exploring features and were asked to complete tasks based on their intuition of our interface design. As can be seen, initially, it took substantially longer than the last iteration when our UI was optimal. This was achieved through better descriptions, the employment of UI principles, and user-recommended implementations. We use a constant of 1 for the Home Screen and FAQ as no measurable tasks were involved.

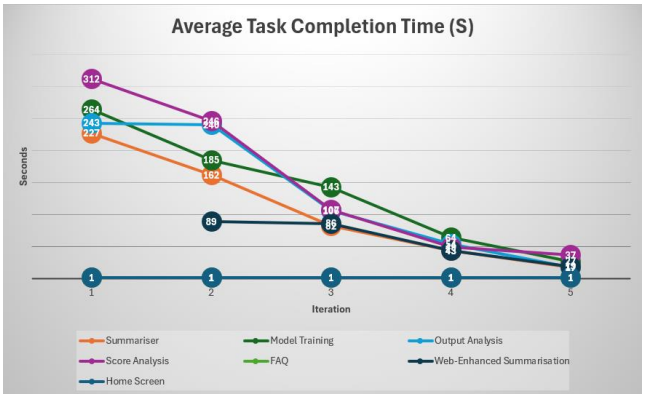



*Figure 38 Average time for task completion*

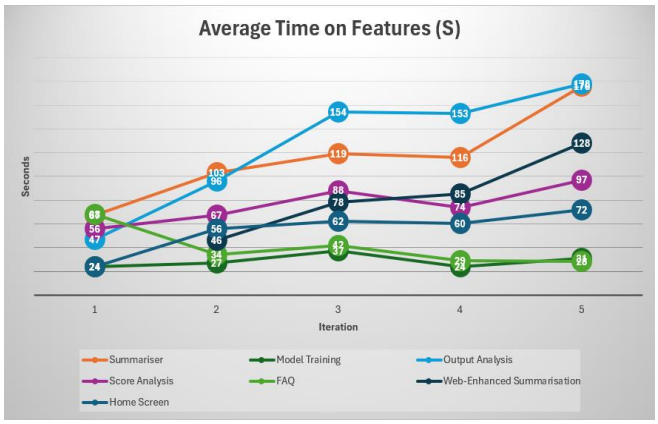



*Figure 39 Average time spent on features*

Recognising that the task completion time is not representative of user interests but mainly of the UI quality, we then allowed our users to explore features at their leisure. This allowed us to discover which features they dedicated most of their time to, giving us insights. It was interesting to see a reduction in the time spent on the FAQ; we infer this was because of better descriptions and accessibility of features, requiring less dependence on the FAQ. Surprisingly, though not a significant difference, 'Summariser' topped the 'Web-Enhanced Summarisation' feature, which we believed would appeal most to our users, as their recommendations inspired it. However, this could be because results modifying parameters may have required more computational time, increasing the time to generate a result and, therefore, time spent on a feature.

While we received good insights into the average time required to complete tasks, we use KLM to quantify this in terms of operations (e.g. number of clicks, points, etc.). KLM calculates the best-case scenario regarding minimal operations needed to complete a task. Each operation is multiplied by the average time typically required for the operation. As the developer, we used our knowledge of the most optimal routes. We manually tallied each operation and entered it into a KLM calculator [40]. Figure 40 represents the substantial reductions in operations between our 2nd and 5th iterations. We started from the 2nd iteration when the Web feature was introduced. Though not reflective of actual usage as it doesn't consider factors such as time required for processing, this has been helpful in helping us quantify our improvements to the UI in making it as minimalistic as possible.


*Figure 40 KLM Calculations Comparison.*

## 6.2 User Feedback

This section evaluates user feedback to assess their experience with our application. Our evaluation methods included qualitative and quantitative data to gain a full spectrum of user insights.

Through observations and walkthroughs, we deeply understood each of our user's mental models, which allowed us to make UCD and HCD decisions. In each of our testing sessions, after collecting data to test system functionality, we immediately proposed design choices during the presence of our users. We used A/B testing to compare different versions of our application and systematically made tweaks, isolating the impact of specific design and algorithmic choices. This direct feedback meant we did not have to rely solely on surveys and questionnaires, which take time to complete, and instead gathered users' insights while fresh in their memories. Subsequently, these recommendations became a part of the input for the following user's A/B testing. Where we saw a clear trend in preferences, these were incorporated into the final design. This iterative approach ensured that our design decisions were grounded in user feedback and preferences, leading to a more user-centred and human-centred design, as discussed in our literature review. In addition to collecting live user feedback, we collected data for our final design using a *System Usability Scale* (SUS) and a project-specific feedback form.

SUS quantitatively measures subjective user feedback, focusing on effectiveness, efficiency, and satisfaction. Below, each statement corresponds with those in Appendix A.2.1.

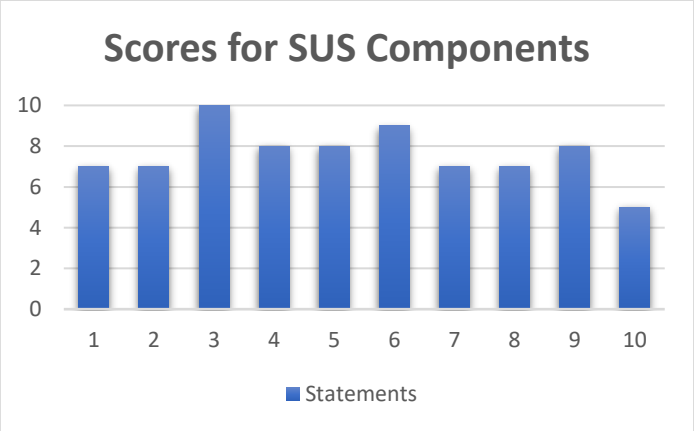



*Figure 41 SUS Scores*

The study was conducted on 20 participants who used the system for the first time. In interpreting the results of the SUS, it is essential to consider both the overall score and individual item responses. For comparative purposes, we have scaled each component score out of 10. Overall, we have performed well, with our most robust components being ease of use and consistency. The mode score is 7, with most of the remaining scores above this, suggesting that the system generally met or exceeded user expectations. Our lowest score has been on users needing to learn things before being able to use our application, something we anticipated. We will go into detail about this metric in our discussion. Our SUS score is 76, which falls in the upper end of a 'B' category (68 – 80.3). A score of 80.3 or above is considered excellent and is graded 'A' [41]. While we were eager to reach this bracket, we adopted a mindset that considers this a learning experience, helping us improve our application for future work. As SUS is intended for new users, we used Google Forms to get insights from ex-participants involved during the testing phases. Overall, we received positive feedback supporting the application; see Appendix A.2.2, where we also provided comments.

## 6.3 Evaluating Requirements

This section evaluates how our application has met its specified functional and non-functional requirements (Appendix B). Throughout this study, our requirements have been fine-tuned in response to user feedback and insights, changing priorities, and unforeseen challenges. No requirements were removed, but their details and priority levels were adjusted per evolving needs and circumstances. We refrained from adjusting 'must' requirements unless there was a critical reason, allowing us to maintain the core functionality of our application. Below are graphical representations showing the proportions of how many requirements were completed.

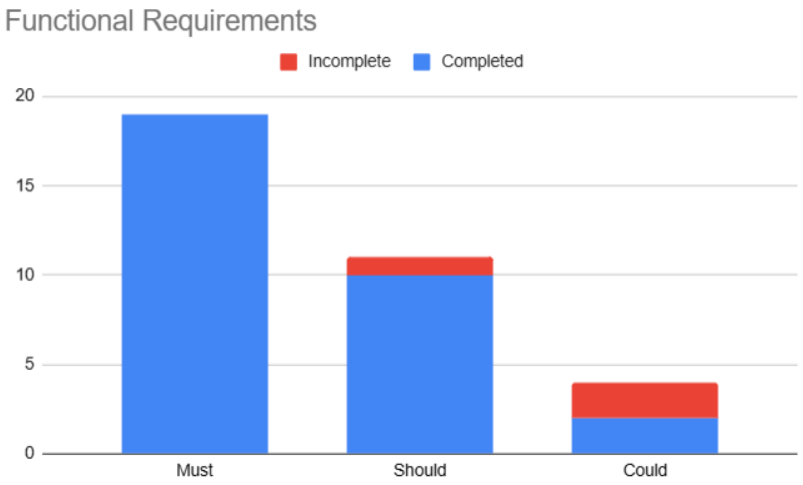



*Figure 42 Graph showing the proportion of completed Functional Requirements*

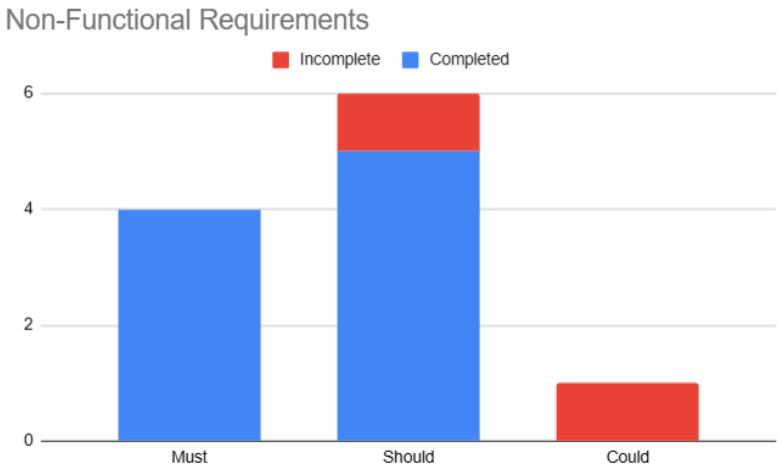



*Figure 43 Graph showing the proportion of completed Non-Functional Requirements*

Out of the 45 implementable requirements (excluding 'Won't') outlined in Appendix B, we successfully met 40, representing a fulfilment rate of 89%. These include all 'must' requirements and most others, meaning our project aligns closely with the specified criteria.

To be considered complete, each requirement was rigorously tested and validated using manual and automated testing methods, ensuring complete coverage of each requirement. We will now discuss the incomplete requirements:

- FR 1.4 (could): This requirement was initially implemented but later removed from the system after user testing. First, it was redundant, as a user could use the navigation bar to access the feature. Second, providing an additional path to direct users away from the tiles interrupted their flow when reading all the feature descriptions. Given that the feature tiles were implemented to encourage engagement, it was deemed counterproductive to include this option.
- FR 7.5 (could): We decided against implementing a chatbot due to time constraints and its lack of necessity. Implementing a chatbot would have required significant time investment at the cost of not being able to spend valuable time on user testing. Additionally, our FAQ only needs ten questions to answer the most common queries. The original intention for the chatbot was to help users receive quick answers if they were struggling to navigate through many questions in the FAQ.
- FR 8.4 (should): Due to the minimalism of our UI, providing a dropdown menu would not have significantly improved our navigation process. Additionally, the application is designed to guide users through a sequential flow, ensuring they navigate through features in a predetermined order for optimal user experience.
- NFR 1.4 (should): Initially, a tutorial on using the application was produced, but this was removed as the UI was intuitive enough not to need it. It turned out that it was much more efficient for users to self-discover application elements rather than following a guide. However, as explained by our user feedback, the application lacks in explaining general concepts such as Training. This will be discussed in the next section.
- NFR 7.1 (should): Regretfully, this requirement was not fulfilled, as a more relaxed approach was taken. Due to our fast pace for iterative development, we did not always follow the best coding practices, such as using comments and code refactoring. Additionally, as this was an independent project, we did not fully consider its implications. One implication had been readability when writing this report. Now, limited by time, this issue can be addressed by dedicating time to future work to improve code quality and adherence to coding standards.

These requirements could be fulfilled or modified in more time to suit our application better. Moreover, the structure of our code allows for implementations to be added without requiring significant deviations from the existing architecture and extensive rewriting of the current feature components.

## 6.4 Discussion

At its current state, our application proposes a credible solution that allows non-technical users to use state-of-the-art summarisation technology effectively. Our application has demonstrated to be comprehensive and intuitive, incorporating multiple functionalities that serve both practical and experimental needs. As verified by Appendix A.1, by our final iteration, all users could attain accurate results across each of our features without encountering any problems. This is despite the large scope of potential errors that would typically form a barrier for users. As evidenced by Figure 41, our users were satisfied with our system's ease of use and integration of elements, scoring an overall SUS value of 76. This is further supported by Appendix A.2.2, where each of the W3C metrics scored highly, with an average of 8/10. Additionally, through intensive testing and UI optimisation to improve efficiency, we have significantly reduced the required time to successfully complete tasks across all features (Figure 40). Nevertheless, we acknowledge several limitations that could be addressed in future work.

From our evidence (Appendix A.2.4), our main limitation has been our lack of focus on providing pedagogical value. Throughout our study, we prioritised ensuring that non-technical users could obtain accurate values from our features and as efficiently as possible. In addition to optimising our UI and mitigating sources of error, we should have equally valued feedback in trying to comprehend a user's understanding of each feature and its purpose. Instead, we naively assumed that obtaining a final result equated to them understanding its underlying processes and implications. This oversight led to a significant gap in user understanding, as evidenced by the finding that 40% of our users felt the need for background research to confidently utilise our features (Appendix A.2.2). This further suggests our current attempts at using tooltips, descriptions, and FAQ answers were insufficient. We now realise this was because these resources operated independently, which did not allow users to form comprehensive connections and develop a holistic understanding of our application's features. Subsequently, users were not motivated to contribute to the field without a proper understanding of a feature's purpose, as comments in Appendix A.2.4 suggest. This could be addressed in future work by implementing a more integrated and cohesive approach to educational resources within our application.

Another limitation has been our approach in collecting user data, as it is not quality controlled. For example, after assessment, many of the submitted user summaries contain grammatical and spelling errors, which could negatively impact training models if supplied with low-quality data. To address this, our application could incorporate methods highlighting these areas to flag them to users. Additionally, as our system currently has a feature to rate machine-generated summaries, the feature could be easily adapted to allow the rating of these user summaries, which can serve as a crowd-sourced validation process. Summaries that users agree to be of higher quality and relevance can be used as training data if they pass some threshold. Additionally, if we had more time, we could have taken the opportunity to capture more insightful information from our users to support developers better. This could have been through integrated feedback forms or methods to differentiate preferred characteristics of models better.

Another aspect worth discussing is the localised approach to our project. Due to the iterative nature of our project, we decided to keep our application local to the developer's machine instead of deployment. While this brought disadvantages, such as only being able to run the application when the developer was present and hence constraining the feedback we could receive, it also offered advantages that aligned with the project's specific goals. Working locally simplified the development process, leading to faster development cycles and quicker feedback loops from testing and user feedback. Additionally, as the developer was always present during use, it allowed him to examine

the user's workflow and develop better user-centric solutions. With consent, each interaction was screen recorded with the microphone enabled (in-person and remotely), meaning we captured every intricate detail and could perform detailed post-analysis, e.g. Figure 40. However, in a future version, we expect the application to be available on the web, which can be achieved by hosting it on a web server. We will not expect users to use their CPU for 'Model Training' features, but we will provide cloud-based services to accelerate the training process.

In comparing our work with SummerTime [8], our evidence suggests that we have significantly simplified text summarisation and evaluation requirements. To get a quantitative evaluation, participants with technical backgrounds were asked to use SummerTime and our application, and 100% agreed that our application was more appealing and straightforward. Apart from offering different variations of text summarisation, such as single-doc and multi-doc, a purposeful decision we discussed in the literature review, our application provides all of SummerTime's features except the automatic model selector. However, our users can achieve the same results through their own critical evaluation and assessments, which we believe to be more beneficial. Rather than automatic selection, our users can use quantitative results, compare model outputs, and judge based on their needs and preferences. Though SummerTime attempts to improve usability by hiding background cells in its notebook and using functions to automate complex tasks, compared to our UI, it still proved overwhelming to our users who tested it and struggled to navigate the code. In addition to our extra features and intuitive UI, we provide better descriptions and graphs to facilitate side-by-side comparison. Below, we have provided screenshots of SummerTime to showcase how we have improved its features.

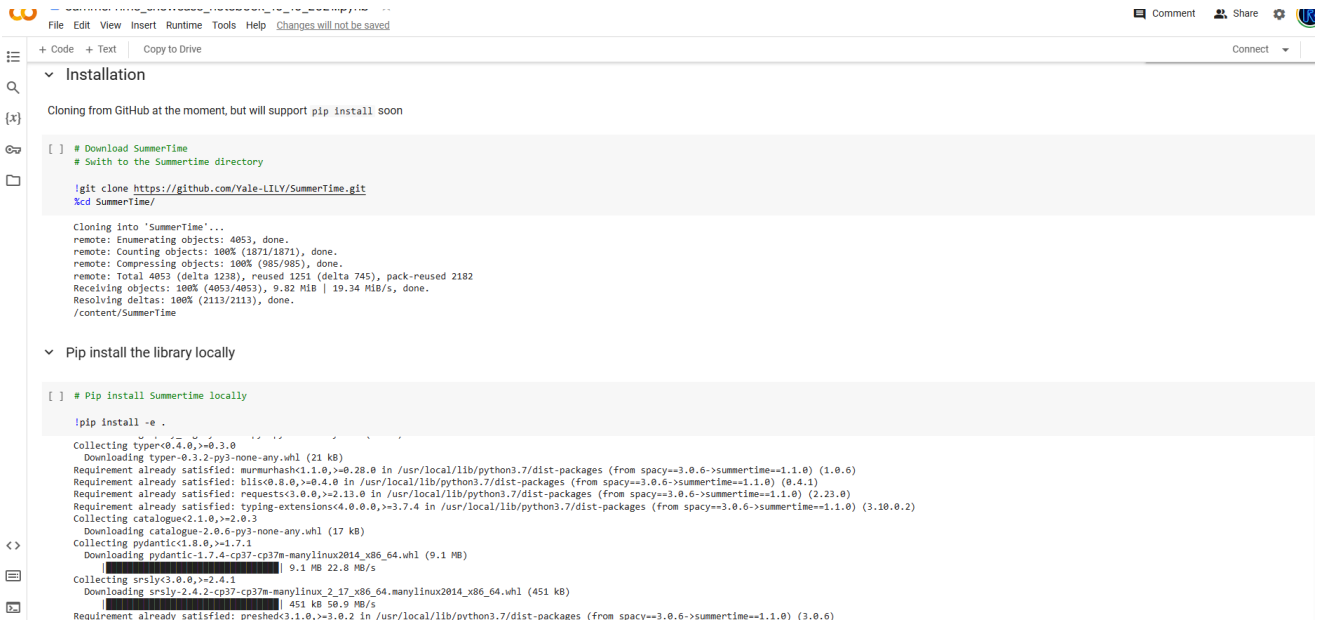



*Figure 44 Screenshot of SummerTime UI*

Figure 44 shows the format of SummerTime, which lacks a user-friendly UI and relies on modifications of its code to perform experimentation. Although the paper emphasises that users can achieve results 'with a few lines of code', even if they made minor changes, this would still be error-prone for non-technical users without programming experience. In contrast, our application completely removes the requirement to understand code details and the risk of syntax errors.

```
# Users can easily access documentation to assist with model selection
sample_model.show_capability()
```

```
Pegasus is the default singe-document summarization model.
Pegasus is a abstractive, neural model for summarization.
####################
Introduced in 2019, a large neural abstractive summarization model trained on web crawl and news data.
Strengths:
- High accuracy
- Performs well on almost all kinds of non-literary written text
Weaknesses:
- High memory usage
Initialization arguments:
- `device = 'cpu'` specifies the device the model is stored on and uses for computation. Use `device='gpu'` to run on an Nvidia GPU.
```

*Figure 45 Example of a SummerTime Description*

Report:

BLEU (Bilingual Evaluation Understudy) is a popular metric usually used to evaluate the quality of machine-generated translations, but it is also useful to evaluate text summaries. It calculates a score between 0 and 1, where higher scores indicate better quality.

BLEU Score: This is the main output which tells you how closely the candidate summary matches human-generated reference summaries. It is calculated based on the precision of n-grams (usually 1-grams and 2-grams) in the candidate summary compared to the reference summaries.
Brevity Penalty: BLEU incorporates a penalty for very short translations to ensure fairness in evaluation. It unfavours models from generating very brief summaries that may unfairly increase scores. The penalty is based on the length ratio of the candidate summary to the reference summaries.

Length Ratio: This is the ratio of the length of the candidate summary to the length of the reference summaries. It is used in calculating the brevity penalty. A length ratio closer to 1 indicates that the candidate summary has a similar length to the reference summaries.

1-gram: 1-gram is a single word. In the context of BLEU, 1-gram precision measures how many individual words in the candidate summary match those in the reference summaries.

2-gram: A 2-gram is a sequence of two words. 2-gram precision measures how many two-word sequences in the candidate summary match those in the reference summaries.

*Figure 46 Example of a Description of our Application*

Above, we have randomly chosen two descriptions, with Figure 45 belonging to SummerTime and Figure 46 belonging to our application. As can be seen, our descriptions are much more in-depth and descriptive, and our evidence supported that it enhanced our user's engagement and understanding.

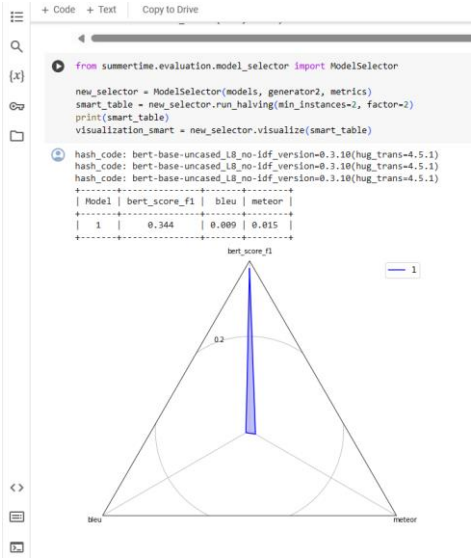



*Figure 46 SummerTime Graph*

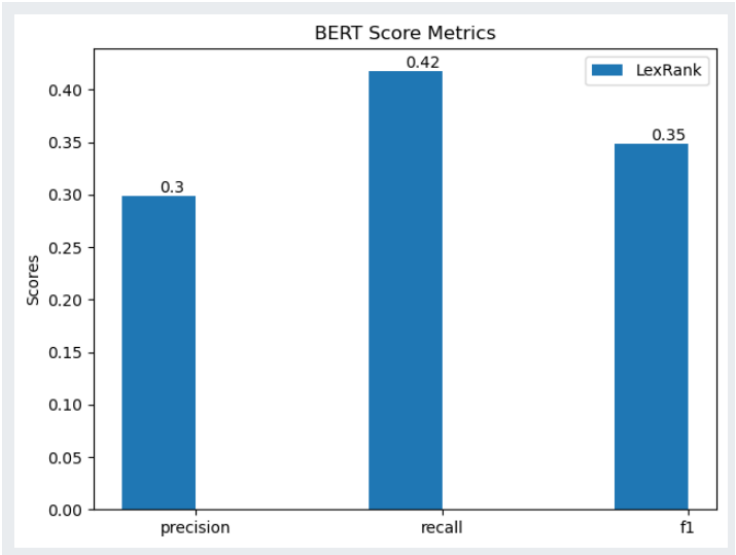



*Figure 47 Our Application's Graph*

Above, we provide two graphs representing how information is graphically presented to a user. As expressed by our users, radar plots were complicated when making side-by-side comparisons without relying on table values. However, comparing bars in a bar chart has been very simple visually. It is also worth noting that we offer each of their sub-metrics for each metric. For example, Figure 46 only displays BERTScore F1, but we provide this with precision and recall, offering more insights that we clearly explain in the associated report. We have considered our users' needs and simplified their evaluation process.

# Chapter 7: Conclusion

To conclude, the goal of this project has been to create a web application that broadens the accessibility of text summarisation technology to non-technical users. Our platform empowers users to perform comprehensive text summarisation, evaluation and testing using state-of-the-art models and metrics. By providing our platform, we have offered additional summarisation functionality, improved user trust with summarisation technology, and allowed for collaborative improvement among developers and non-technical users. Our features were carefully selected and designed to provide optimal coverage of text summarisation techniques that would be of significant interest and utility to our users.

Throughout our study, we have leveraged HCI methodologies to conduct rigorous user testing and evaluation, which has guided the implementations as part of our Agile software development workflow. We have used methods to quantitively measure our improvements throughout the project, including accuracy and efficiency. As a result, our final application is intuitive, user-centric, and user-friendly, with features that serve both practical and experimental purposes. Our main features have been 'Summariser', which allows users to change model parameters to achieve customised summaries; 'Train Model', which enables users to train a model using their own training data; 'Output Analysis', which allows users to visually inspect how summaries have been constructed and also rate and produce their own summaries, 'Score Analysis' where users can quantitively evaluate model outputs and assess their associated reports with graphical representations, and Web-Enhanced summarisation which provides users with data from the internet to supplement their summaries.

While users can interact with the features and obtain results, our most significant limitation has been providing insufficient information to our users. This has meant users have struggled to find the motivation to fully leverage our application and provide user data to support the field. Although we did create dedicated preparation materials for user training, they often became the focal point of discussions and hindered our user testing. Hence, they were removed, and we decided that it would be more productive to defer the final development of these materials to future work and focus on functionality. Now the application is functionally complete; if we had more time, it would be dedicated to enhancing and integrating these materials to improve our users' understanding. Another concern users often had was the quality of some summary results, which were significantly poor and not due to factors such as parameter adjustments and data quality. This is despite telling our users that our focus was on providing accessibility for testing existing models instead of optimising their summary quality, which was not our aim. In future work, we could further clarify this or change the project's direction to also focus on producing consistent, higher-quality summarisation results. Additionally, our application could be improved by integrating feedback forms to provide further insights, which could inform developers directly about opinions of particular models; this could include quantitative and qualitative feedback. While our study represents a significant step forward compared to current offerings, users cannot fully appreciate our platform's potential without a critical understanding of our features and its purpose. With these improvements, we believe we can contribute significantly to the field of ATS.

In terms of self-development, this project has been an incredible experience that has matured me academically and professionally. In addition to learning a lot about myself, I have learned how to conduct critical research and enhance my written and technical skills. Overall, this project has been a significant milestone in my self-development, and it has given me many skills that will help me build the required confidence and competence to be a successful technical consultant. Please see Appendix F for a complete personal reflection.

# References


[1] *www.youtube.com. (n.d.). What is NLP (Natural Language Processing)? [online] Available at: https://www.youtube.com/watch?v=fLvJ8VdHLA0.*

[2] Kavlakoglu, E. (2020). *NLP vs. NLU vs. NLG: the differences between three natural language processing concepts*. [online] IBM Blog. Available at: https://www.ibm.com/blog/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/.

[3] Myers (1999). *Overview of HCI Design and Implementation*. [online] Available at: https://www.cs.cmu.edu/~bam/uicourse/special/.

[4] Statista (2023). *Data Created Worldwide 2010-2025* . [online] Statista. Available at: https://www.statista.com/statistics/871513/worldwide-data-created/.

[5] Neabjorkqvist (2023). The average time spent on websites: 3 tips. [online] Contentsquare. Available at: https://contentsquare.com/blog/average-time-spent-on-websites-is-dropping/.

[6] Elhadad, N., 2004. User-Sensitive Text Summarization-Thesis Summary.

[7] Loem, M., Takase, S., Kaneko, M. and Okazaki, N., 2022. Extraphrase: Efficient data augmentation for abstractive summarization. *arXiv preprint arXiv:2201.05313*.

[8] Ni, A., Azerbayev, Z., Mutuma, M., Feng, T., Zhang, Y., Yu, T., Awadallah, A.H. and Radev, D., 2021. SummerTime: Text summarization toolkit for non-experts. *arXiv preprint arXiv:2108.12738*.

[9] Barbella, M., Risi, M., Tortora, G. and Citarella, A.A., 2022. Different Metrics Results in Text Summarization Approaches. In *DATA* (pp. 31-39).

[10] Ferreira, R., de Souza Cabral, L., Lins, R.D., e Silva, G.P., Freitas, F., Cavalcanti, G.D., Lima, R., Simske, S.J. and Favaro, L., 2013. Assessing sentence scoring techniques for extractive text summarization. *Expert systems with applications*, *40*(14), pp.5755-5764.

[11] Erkan, G. and Radev, D.R., 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, *22*, pp.457-479.

[12] Dlikman, A. and Last, M., 2016, January. Using Machine Learning Methods and Linguistic Features in Single-Document Extractive Summarization. In *DMNLP@ PKDD/ECML* (pp. 1-8).

[13] Shi, T., Keneshloo, Y., Ramakrishnan, N. and Reddy, C.K., 2021. Neural abstractive text summarization with sequence-to-sequence models. *ACM Transactions on Data Science*, *2*(1), pp.1-37.

[14] Kumar, S. and Solanki, A., 2023. An abstractive text summarization technique using transformer model with self-attention mechanism. *Neural Computing and Applications*, *35*(25), pp.18603-18622.

[15] Yadav, S., Gupta, D., Abacha, A.B. and Demner-Fushman, D., 2021. Reinforcement learning for abstractive question summarization with question-aware semantic rewards. *arXiv preprint arXiv:2107.00176*.

[16] Zhang, J., Zhao, Y., Saleh, M. and Liu, P., 2020, November. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International conference on machine learning* (pp. 11328-11339). PMLR.

[17] research.google. (n.d.). *PEGASUS: A State-of-the-Art Model for Abstractive Text Summarization*. [online] Available at: https://research.google/blog/pegasus-a-state-of-the-art-model-for-abstractive-text-summarization/.

[18] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L., 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[19] Sharma, S. (2024). *Text Summarization with BART Model*. [online] Medium. Available at: https://medium.com/@sandyeep70/demystifying-text-summarization-with-deep-learning-ce08d99eda97.

[20] Miller, D., 2019. Leveraging BERT for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*.

[21] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), pp.1-67.

[22] *research.google. (n.d.). Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer. [online] Available at: https://research.google/blog/exploring-transfer-learning-with-t5-the-text-to-text-transfer-transformer/.*

[23] Lin, C.Y., 2004, July. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[24] Papineni, K., Roukos, S., Ward, T. and Zhu, W.J., 2002, July. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

[25] Paperspace Blog. (2022). *Automated metrics for evaluating the quality of text generation*. [online] Available at: https://blog.paperspace.com/automated-metrics-for-evaluating-generated-text/.

[26] Banerjee, S. and Lavie, A., 2005, June. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* (pp. 65-72).

[27] Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q. and Artzi, Y., 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

[28] Bell, T.E. and Thayer, T.A., 1976, October. Software requirements: Are they really a problem?. In *Proceedings of the 2nd international conference on Software engineering* (pp. 61-68).

[29] Moreno, A.M., Seffah, A., Capilla, R. and Sanchez-Segura, M.I., 2013. HCI practices for building usable software. *Computer*, *46*(04), pp.100-102.

[30] Majid, R.A., Noor, N.L.M. and Adnan, W.A.W., 2012. Strengthening the HCI approaches in the software development process. *International Journal of Computer and Systems Engineering*, *6*(4), pp.470-474.

[31] Ledo, D., Houben, S., Vermeulen, J., Marquardt, N., Oehlberg, L. and Greenberg, S., 2018, April. Evaluation strategies for HCI toolkit research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (pp. 1-17).

[32] Pitale, A. and Bhumgara, A., 2019, November. Human computer interaction strategies—designing the user interface. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 752-758). IEEE.

[33] Initiative (WAI), W.W.A. (2023). *Designing for Web Accessibility – Tips for Getting Started*. [online] Web Accessibility Initiative (WAI). Available at: https://www.w3.org/WAI/tips/designing/#provide-clear-and-consistent-navigation-options.

[34] Initiative (WAI), W.W.A. (n.d.). *Accessibility Principles*. [online] Web Accessibility Initiative (WAI). Available at: https://www.w3.org/WAI/fundamentals/accessibility-principles/#top.

[35] www.w3.org. (n.d.). *Exploring Usability Enhancements in W3C Process - slide 'Exploring Usability Enhancements in W3C Process'*. [online] Available at: https://www.w3.org/2002/Talks/0104-usabilityprocess/slide1-0.html.

[36] Grudin, J., 1989. The case against user interface consistency. *Communications of the ACM*, *32*(10), pp.1164-1173.

[37] Alférez, G.H., Esteban, O.A., Clausen, B.L. and Ardila, A.M.M., 2022. Automated machine learning pipeline for geochemical analysis. *Earth Science Informatics*, *15*(3), pp.1683-1698.

[38] Cao, M., Dong, Y. and Cheung, J.C.K., 2021. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. *arXiv preprint arXiv:2109.09784*.

[39] Devaraj, K. (2023). *Levels of Testing: A Complete Approach to Quality Assurance*. [online] Testsigma Blog. Available at: https://testsigma.com/blog/levels-of-testing/.

[40] none (n.d.). *Keystroke Level Model (KLM) Calculator*. [online] courses.csail.mit.edu. Available at: https://courses.csail.mit.edu/6.831/2009/handouts/ac18-predictive-evaluation/klm.shtml.

[41] T, W. (2017). *Measuring and Interpreting System Usability Scale (SUS) - UIUX Trend*. [online] UIUX Trend. Available at: https://uiuxtrend.com/measuring-system-usability-scale-sus/.

[42] Brooke, J., 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), pp.4-7.

[43] docs.anaconda.com. (n.d.). *Installing on Windows — Anaconda documentation*. [online] Available at: https://docs.anaconda.com/free/anaconda/install/windows/.

[44] Radigan, D. (2022). *Project management with agile principles*. [online] Atlassian. Available at: https://www.atlassian.com/agile/project-management/project-management-intro.

[45] blog.iil.com. (2021). *Project Management: Agile and Waterfall Explained*. [online] Available at: https://blog.iil.com/project-management-agile-and-waterfall-explained/.

[46] Text Summarization by TextRank! - Data Wow blog – Data Science Consultant Thailand | Data Wow in Bangkok. (n.d.). *Text Summarization by TextRank! - Data Wow blog – Data Science Consultant Thailand | Data Wow in Bangkok*. [online] Available at: https://www.datawow.io/blogs/text-summarization-by-textran.

[47] https://www.flaticon.com/free-icons/parameters

[48] https://app.gazerecorder.com/

[49] https://brackets.io/?lang=en

[50] Card, S.K., Moran, T.P. and Newell, A., 1980. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, *23*(7), pp.396-410.

[51] https://chromewebstore.google.com/detail/axe-devtools-web-accessib/lhdoppojpmngadmnindnejefpokejbdd

# Appendix

## Appendix A - Data Collection

### A.1 System Functionality Data Collection

The below study was conducted over five sprints, which we refer to as iterations in the report. For each iteration, ten new users were recruited for each user testing phase, where approximately 20 minutes were spent with them to collect the following data. To achieve the following, we provided users with five tasks that were heuristically chosen and required the most complex set of actions to reach. If they were completed, it was most probable that users had fully grasped all other aspects of the feature. While intended to represent the whole feature, we had to be general enough to cover all implementations from future iterations. To note, users were first required to have data stored in their clipboard to avoid spending time on creating input text. The following were the tasks, as provided to the user:

1. **Home Screen** & **FAQ**: N/A, no task set.
2. **Summariser**: 'For this task, using the Summariser tool, please choose the **Pegasus** model under Abstractive Summarisation. Then, paste your input into the respective field and provide the following parametric adjustments:
    a. Length Pentalty = 1.6
    b. Min Length = 20, Max Length = 30
    c. Number of Beams = 6
    d. Turn on GPU.

    Then, generate a summary.
3. **Train Model**: 'In this task, you will not perform any actual training due to time constraints. However, you are required to initiate a training process. Using the **T5** model, upload the training data from the PC's desktop. Then, label the respective columns to tell the Trainer which the input and target columns are. Once columns the columns are recognised, use the following training splits:
    a. Train Split: 63%
    b. Validate Split: 24%
    c. Test Split: 13%

    Then, please select the following parameters:
    a. Turn on GPU.
    b. Learning Rate = 0.04
    c. Number of Epochs = 6
    d. Batch Number = 16
    e. Model name: 'test_model'. Once initiated, let the conductor know.'
4. **Visual Output Analysis**: 'Your task is understanding how a summary was constructed from input sentences. Using the LexRank model, paste your input and generate visualisations. Then, strategically, discover where each 'important' word from the summary originates. Once you have found the source sentence for each word, the task is completed.'
5. **Score Analysis**: 'Your task is to derive quantitative values from a dataset we provide, which has multiple entries. Compare Bart, Bert, and LexRank outputs using the ROUGE metric.'
6. **Web-Enhanced Summarisation**: 'Your task is to gain web insights directly from our application. Using **LexRank**, try to obtain three sets of images (and background information if available) on your constructed summary.'

Results for each of the tasks:

Average Task Success Rate (%):

| Iteration | Home Screen | Summariser | Model Training | Output Analysis | Score Analysis | Web-Enhancement | FAQ |
|---|---|---|---|---|---|---|---|
| **1** | 100 | 20 | 10 | 20 | 20 | N/A | 100 |
| **2** | 100 | 40 | 60 | 30 | 40 | 70 | 100 |
| **3** | 100 | 60 | 70 | 80 | 70 | 50 | 100 |
| **4** | 100 | 80 | 100 | 70 | 70 | 80 | 100 |
| **5** | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

These are the average percentages of users who could perform a task successfully. A task was marked as a 'Success' if the end goal was reached, verified by the conductor. As previously noted, no tasks were required for the Home Screen and FAQ, so they were defaulted to 100%.

Average Task Completion Time (S):

| Iteration | Home Screen | Summariser | Model Training | Output Analysis | Score Analysis | Web-Enhancement | FAQ |
|---|---|---|---|---|---|---|---|
| **1** | 1 | 227 | 264 | 243 | 312 | N/A | 1 |
| **2** | 1 | 162 | 185 | 240 | 246 | 89 | 1 |
| **3** | 1 | 82 | 143 | 106 | 107 | 86 | 1 |
| **4** | 1 | 43 | 64 | 54 | 49 | 43 | 1 |
| **5** | 1 | 17 | 27 | 17 | 37 | 19 | 1 |

As no tasks were designed for the Home Screen and FAQ, we have assigned constant values of 1. These are a subset as they only include the time of a verified successful task. If a task was marked as 'success', its time was recorded to complete. These have all been averaged for representation.

Identified Errors (n):

| Iteration | Home Screen | Summariser | Model Training | Output Analysis | Score Analysis | Web-Enhancement | FAQ |
|---|---|---|---|---|---|---|---|
| **1** | 0 | 12 | 13 | 11 | 14 | N/A | 0 |
| **2** | 0 | 6 | 7 | 7 | 9 | 7 | 0 |
| **3** | 0 | 2 | 3 | 5 | 6 | 7 | 0 |
| **4** | 0 | 0 | 0 | 2 | 1 | 3 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

As opposed to averaging across our groups, the number of errors collectively represents all errors identified by all users in an iteration.

Average Time on Features (S):

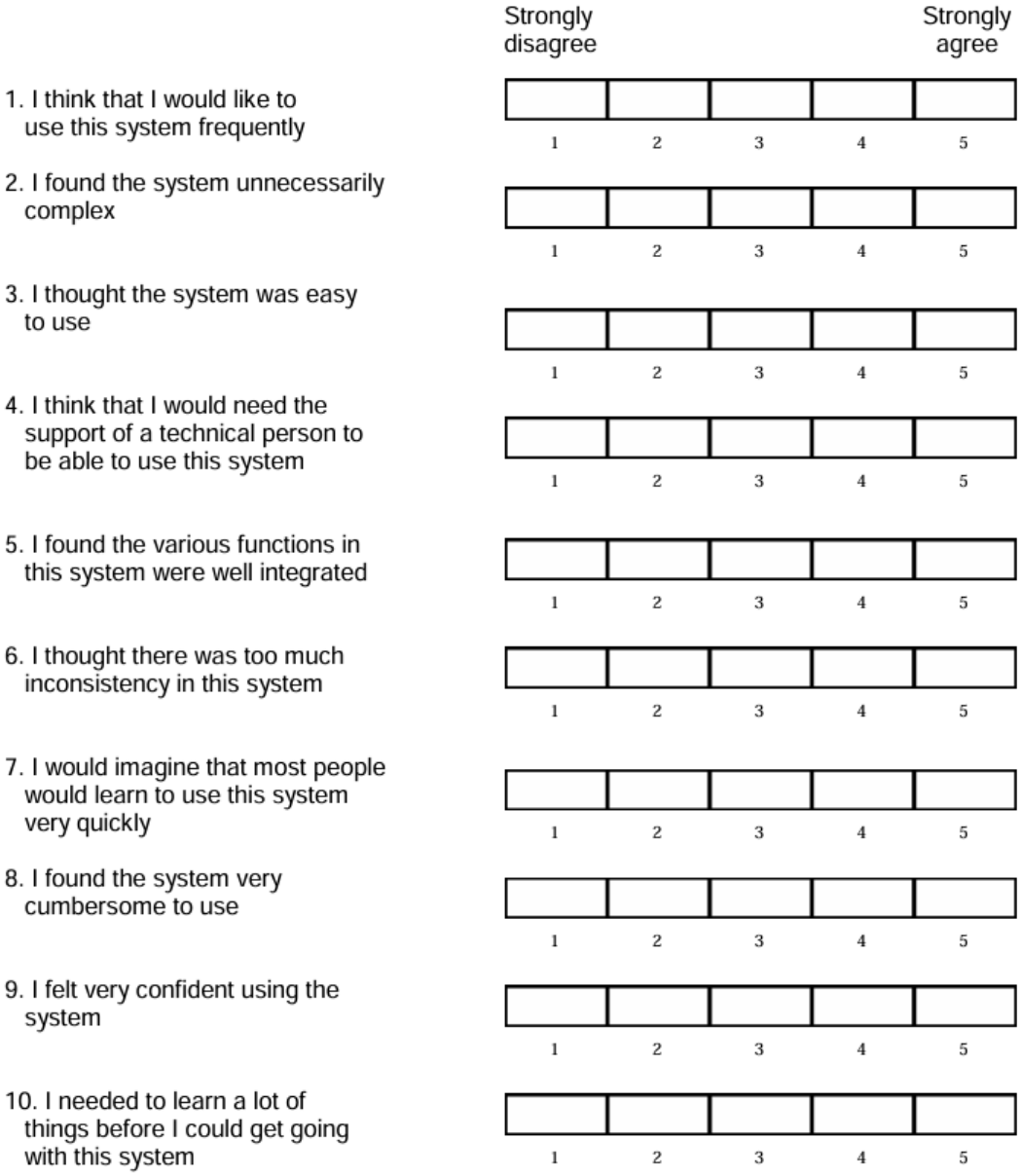
| Iteration | Home Screen | Summariser | Model Training | Output Analysis | Score Analysis | Web-Enhancement | FAQ |
|---|---|---|---|---|---|---|---|
| **1** | 24 | 67 | 24 | 47 | 56 | N/A | 68 |
| **2** | 56 | 103 | 27 | 96 | 67 | 46 | 34 |
| **3** | 62 | 119 | 37 | 154 | 88 | 78 | 42 |
| **4** | 60 | 116 | 24 | 153 | 74 | 85 | 29 |
| **5** | 72 | 176 | 31 | 178 | 97 | 128 | 28 |

This table represents how long our users spent on each feature after completing the tasks. As the average time to complete a task does not represent how much time a user wants to spend on a feature, this data attempts to address this.

## A.2 User Data Collection

### A.2.1 System Usability Scale (SUS)

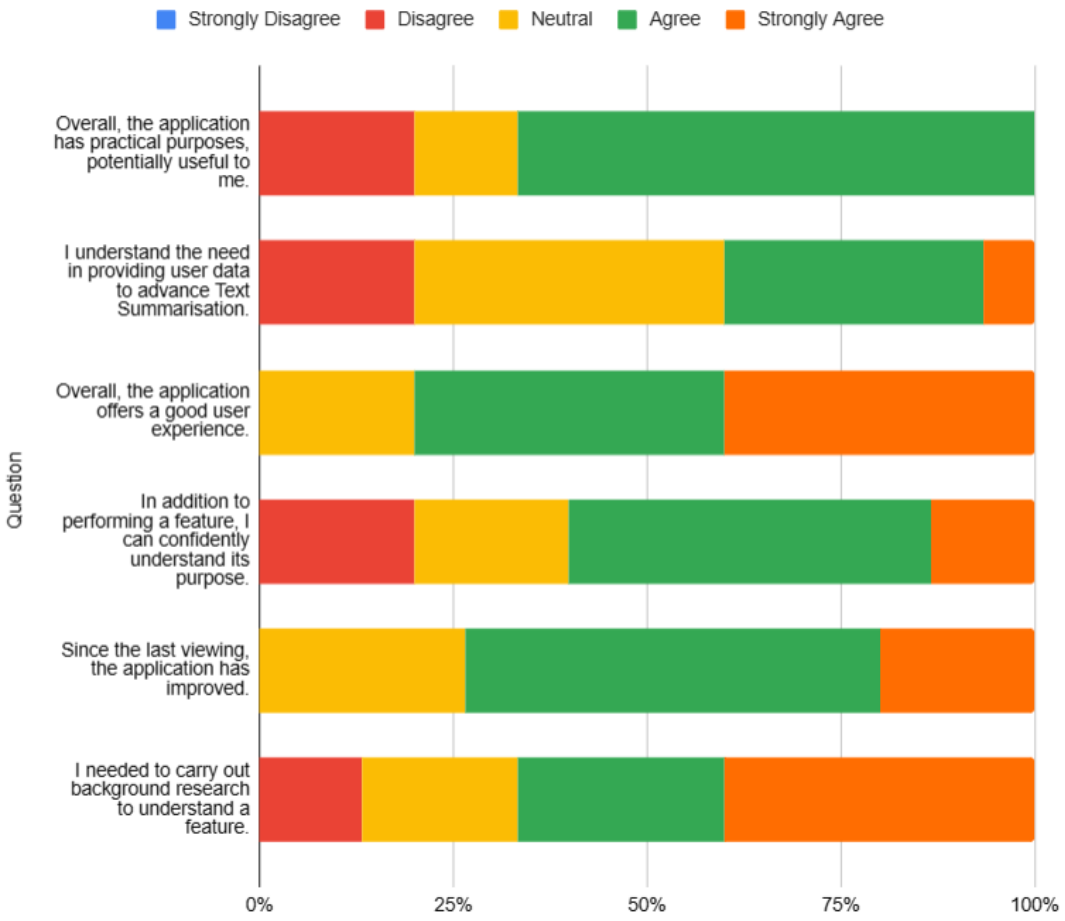
SUS [42] is a Likert scale that users provide once they have tested a complete application. It is scored such that the contribution is from 0 to 4 for each item. For odd numbers, it is the scale position subtract 1. Whereas for even numbers, the scale is five minus the position in the scale. Then, each score is summed, and the overall score (out of 40) is multiplied by 2.5. This gives a score out of 100, which is interpreted. This is discussed in the evaluation.



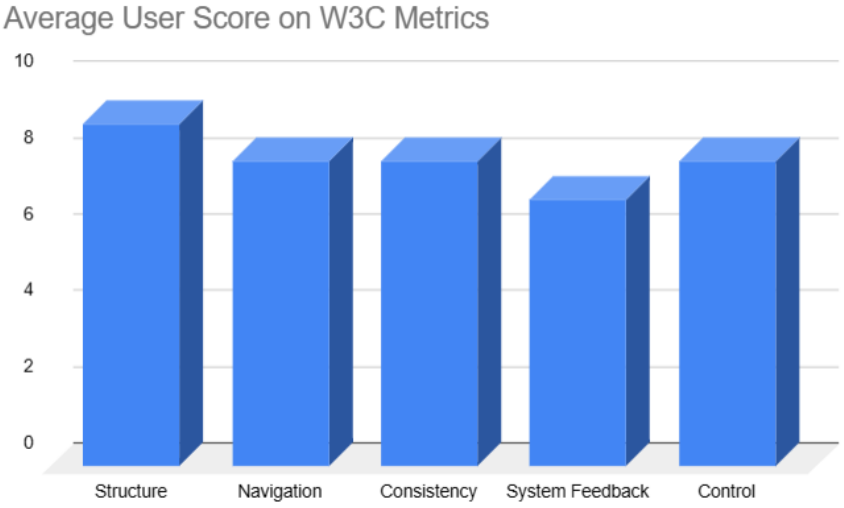

As stated in the paper, to be used effectively, users should be asked to give a score immediately after using the system. This means the developer should not influence them, only through the application. Also, they should not spend time thinking about each statement but provide a response immediately using their intuition.

### A.2.2 Google Feedback Form Results

We recruited 15 ex-participants from iterations 1 to 5 who opted to return to the study and were interested in seeing the application's progression. Unlike the SUS, participants in this survey were free to spend as long as they wanted. Generally, we can see a positive stance from our users, who often have a positive sentiment. Similar to question 10 from the SUS, we see support in the statement that more needs to be done in providing background material to support the use of each feature. An overwhelming majority of 40% of users strongly agreed that background research needed to be done.

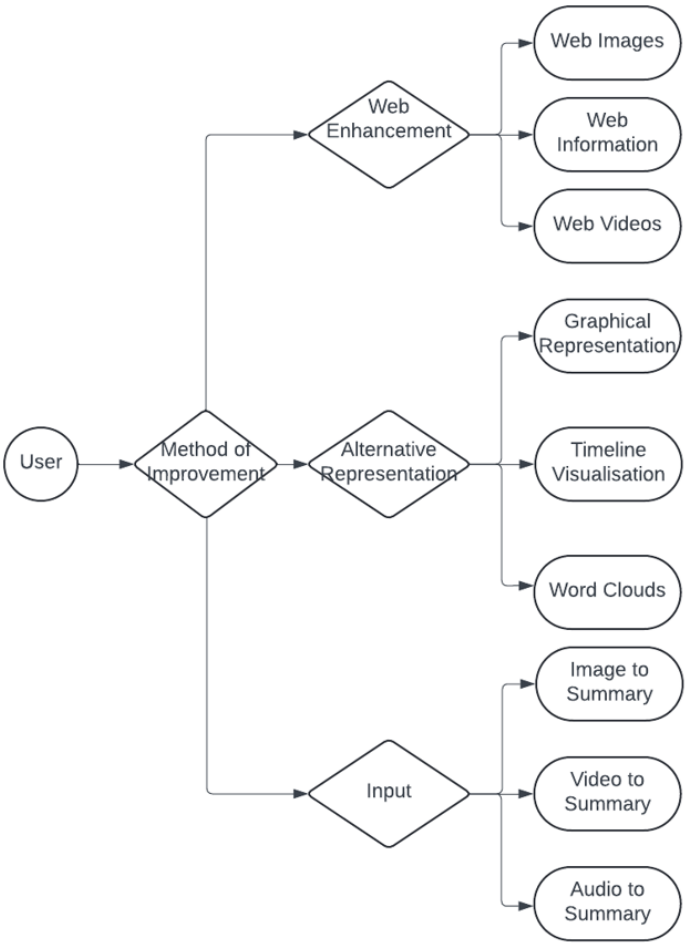



Concerning [35], we asked the participants to rate each common area on which websites tend to perform poorly. The mean score is 8, which we infer is good, with the application's structure being the most preferred aspect. As is consistent with our other feedback, system feedback requires improvement to provide a better understanding.

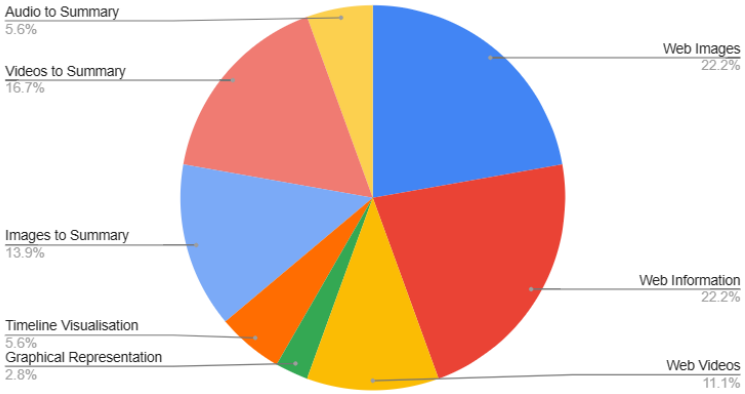


### A.2.3 User-Inspired Method of Summarisation

As part of our project, we wanted to create a feature developed and fully inspired by our users. During our first iteration, we gathered insights and discussed how Text Summarisation currently lacks. This involved asking our stakeholders for ideas on how to make improvements. Users explained that loss of contextual information was often an issue. After numerous discussions, we extracted a set of plausible ideas that we believed could be implemented within the project's timeframe. These ideas were then proposed to users through a questionnaire, where they could vote for their preference. Below is a component diagram which provides an overview of their choices.



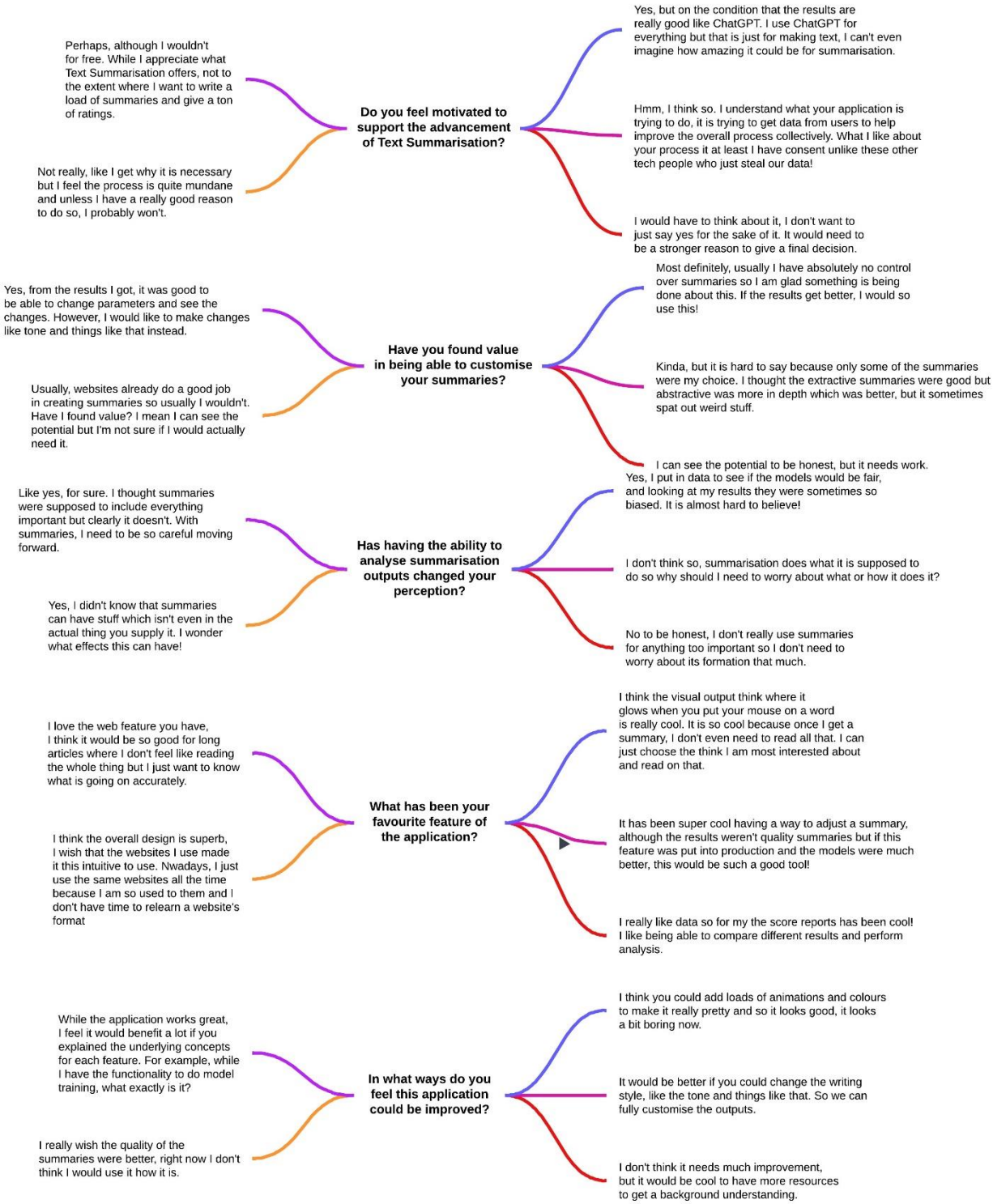

34 participants voted, and the results were as follows:




This data suggested our users had a clear preference for web enhancement, which would be of the utmost practical use. With Web Images and Information being equal and the two highest scores, we used these as goals for our implementation. Tailored by user feedback, the final feature meets the needs of our users.

### A.2.4 Final User Acceptance Testing Comments

Users were also asked to add their final comments on the complete version of the application through an online form. Using a random generator, we selected five questions out of those provided for each. Ten questions were asked, but for brevity, we have only shown five that closely relate to our project aims. Additionally, we have randomly chosen five randomly selected comments for each question.



## Appendix B - Functional & Non-Functional Requirements

Functional Requirements

This section formally introduces our system components and details how we plan to implement them. These have been adapted throughout the project and are the final requirements. In red, we highlight those which were not met.

1. **Home Screen**:
   1.1 The Home Screen **must** include a title and an application description.
   1.2 The Home Screen **must** include tiles containing an icon representing each feature.
   1.3 Upon hovering over a tile, the application **must** display additional information about the corresponding feature/tool.
   1.4 Clicking on a tile **could** direct users to the feature/tool section.

2. **Summariser**:
   2.1 The Summariser feature **must** provide functionality to generate summaries of input text.
   2.2 Users **must** be able to select between Abstractive and Extractive summarisation and then be provided with the correct options amongst the models: BART, Pegasus, T5, Bert, and LexRank.
   2.3 Upon inputting text and selecting a model, users **must** be able to adjust parameters.
   2.4 The generated summary **must** be displayed to the user.

3. **Model Training**:
   3.1 The Model Training feature **must** allow users to train custom summarisation models.
   3.2 Users **must** be able to input training data and specify parameters for model training.
   3.3 Progress and results of the model training process **should** be displayed to the user.
   3.4 Users **must** be able to test a trained model.

4. **Output Analysis**:
   4.1 The Output Analysis feature **must** provide visualisations to compare the outputs of different summarisation models.
   4.2 Visualizations **should** include word-level analysis, highlighting words and their likely originating sentences.
   4.3 Users **should** be able to hover over words to view additional contextual information.
   4.4 Users **must** submit ratings; the system should store and update the rating data accordingly.
   4.5 Users **should** be able to rate summaries based on coherence, conciseness, readability, content, and overall quality.

5. **Score Analysis**:
   5.1 The Score Analysis feature **must** display analytics of user ratings for provided summaries.
   5.2 Users **must** be able to compare models using a range of metrics and data sources.
   5.3 Scores **must** be averaged using multiple data sources to ensure robustness.
   5.4 Users **must** be provided graphs and descriptions explaining each metric and score.

6. **Web-Enhanced Summaries**:
   6.1 The Web-Enhanced Summaries feature **should** enhance summaries with web information and images (if available) for better understanding.
   6.2 Users **should** be able to view summaries with clickable words that reveal additional contextual information.
   6.3 The system **should** analyse and provide context for words in the summary.
   6.4 Users **could** interact with the enhanced summaries to gain deeper insights.

7. **FAQ**:
7.1 The FAQ feature **should** answer frequently asked questions about the application and its functionalities.
7.2 Users **should** access the FAQ section to find solutions to common queries.
7.3 The FAQ section **should** be organised and easily navigable.
7.4 Users **could** search for specific questions or browse through categories.
7.5 This feature **could** be implemented as a Chatbot

8. **Navigation**:
8.1 The application **must** provide a navigation bar to access different features easily.
8.2 Each feature **must** have a dedicated section accessible via the navigation bar.
8.3 The navigation bar **must** be responsive and visible across all application pages.
8.4 A dropdown menu **should** appear when hovering over a feature in the navigation bar, providing access to related sub-sections or options.

9. We **won't** be optimising any of the summarisation models to improve their quality. Instead, we present existing models on which users can train themselves.

Non-Functional Requirements

This section will outline our non-functional requirements and specify various system qualities and attributes.

**1. Usability**:
1.1 The application **must** have an intuitive, user-friendly interface to facilitate easy navigation and interaction.
1.2 All features **must** be clearly labelled and accessible from the main page to ensure ease of use.
1.3 The application **should** provide tooltips or hints for complex functionalities to assist users in understanding the system.
1.4 The application **should** include clear onboarding tutorials or walkthroughs for new users, guiding them through essential functionalities.

**Performance**:
2.1 The application **must** be responsive and provide real-time feedback to user actions, such as text input and button clicks.
2.2 Summarisation and analysis processes **should** be executed efficiently, with minimal latency, to enhance user experience.
2.3 Visualisations and interactive elements **should** load quickly to prevent user frustration and ensure smooth usability.

**Accessibility**:
6.1 The application **must** comply with accessibility standards, such as WCAG (Web Content Accessibility Guidelines), to ensure that it is usable by people with disabilities.
6.2 Text elements, buttons, and interactive components **should** have sufficient colour contrast and size to be easily distinguishable and clickable.

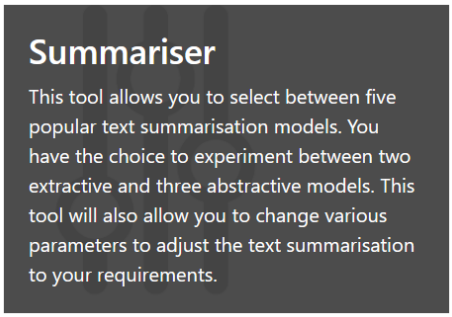
**Maintainability**:
7.1 The application codebase **should** be well-structured and thoroughly documented to help developers make future maintenance and enhancements.
7.2 Version control systems, such as Git, **should** be used to track changes to the codebase and enable collaboration among development teams for future work.

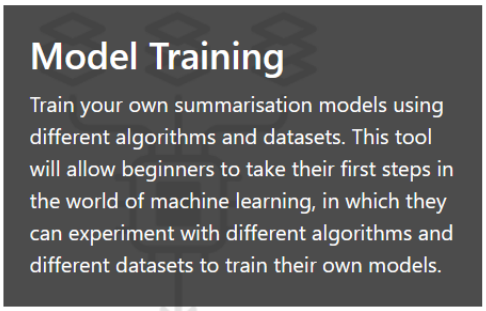
## Appendix C - Full Page Screenshots of UI

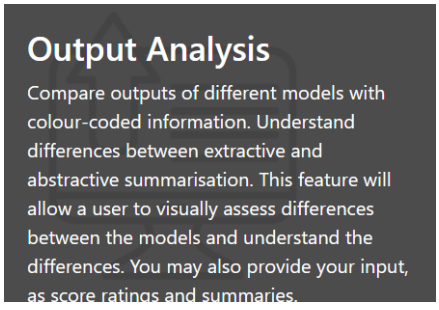
Home Screen

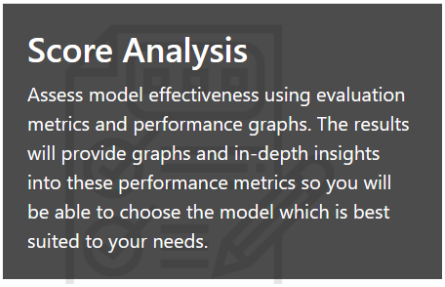

Icon sources: [46][47]

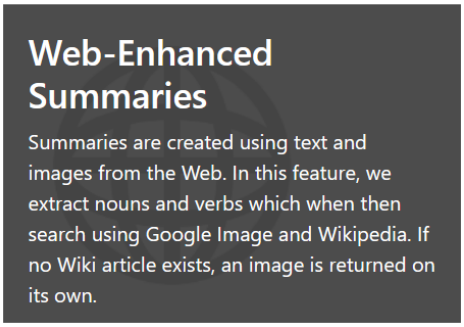
Hovered Icons Descriptions

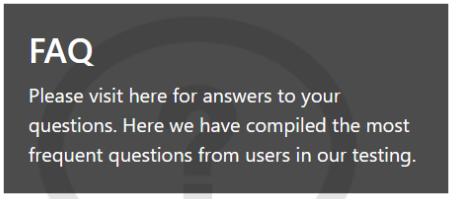


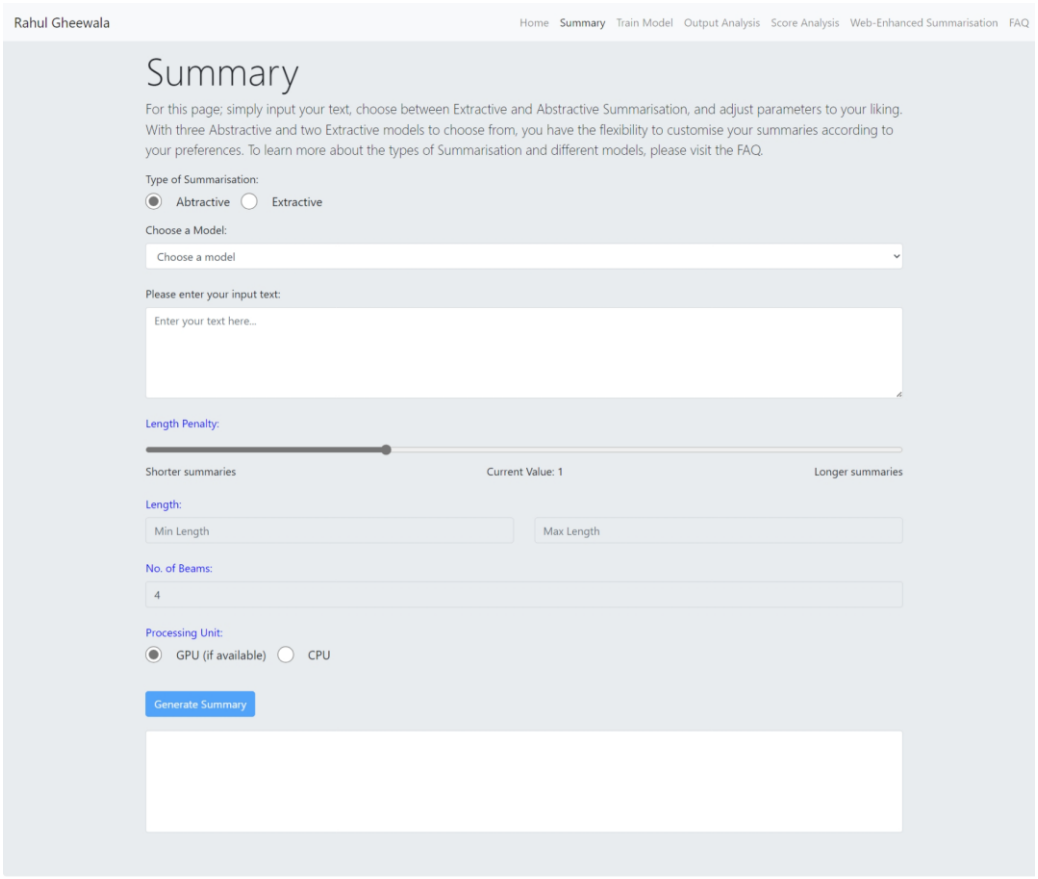



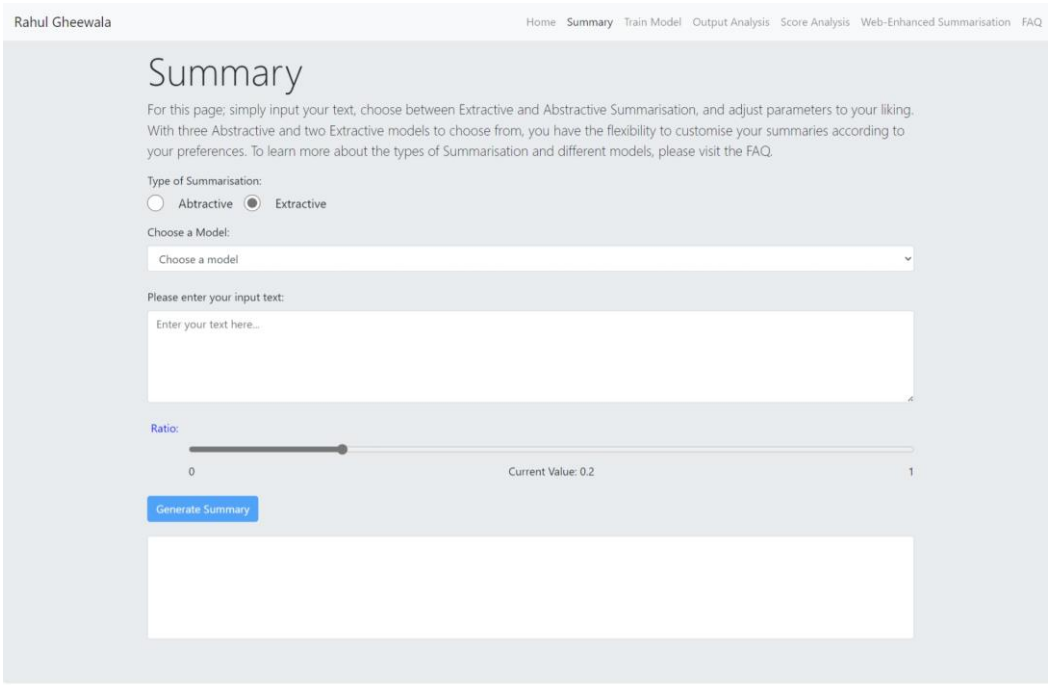



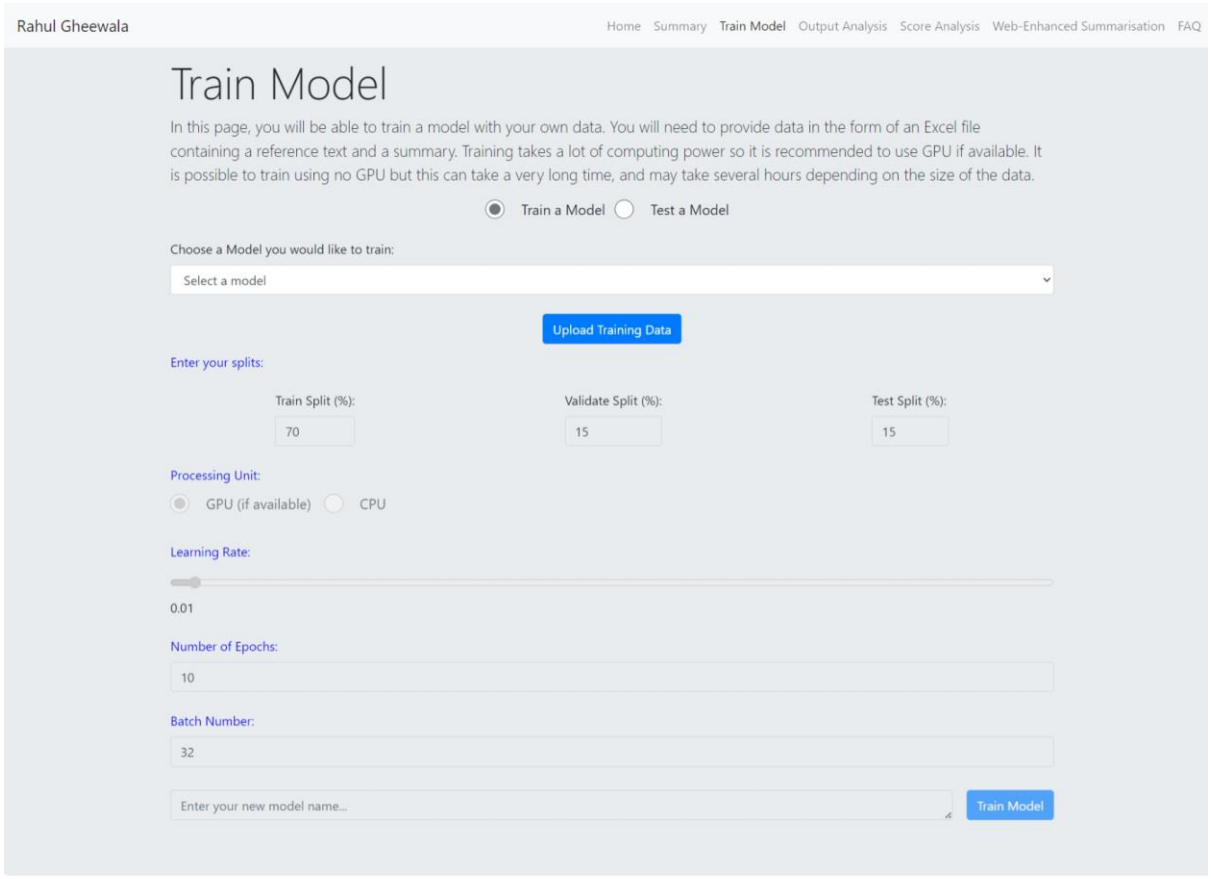



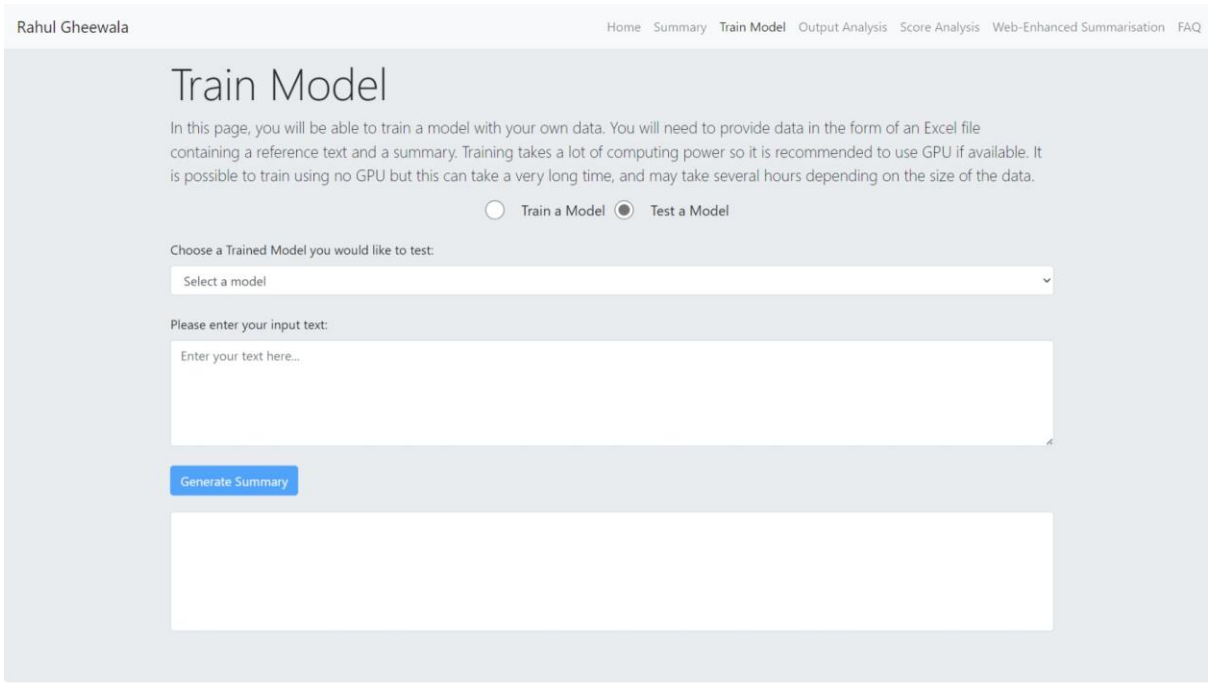
Summary


Rahul Gheewala
Home Summary Train Model Output Analysis Score Analysis Web-Enhanced Summarisation FAQ
Summary
For this page; simply input your text, choose between Extractive and Abstractive Summarisation, and adjust parameters to your liking. With three Abstractive and two Extractive models to choose from, you have the flexibility to customise your summaries according to your preferences. To learn more about the types of Summarisation and different models, please visit the FAQ.
Type of Summarisation:
Abtractive Extractive
Choose a Model:
Choose a model
Please enter your input text:
Enter your text here...
Length Penalty:
Shorter summaries Current Value: 1 Longer summaries
Length:
Min Length
Max Length
No. of Beams:
4
Processing Unit:
GPU (if available) CPU
Generate Summary

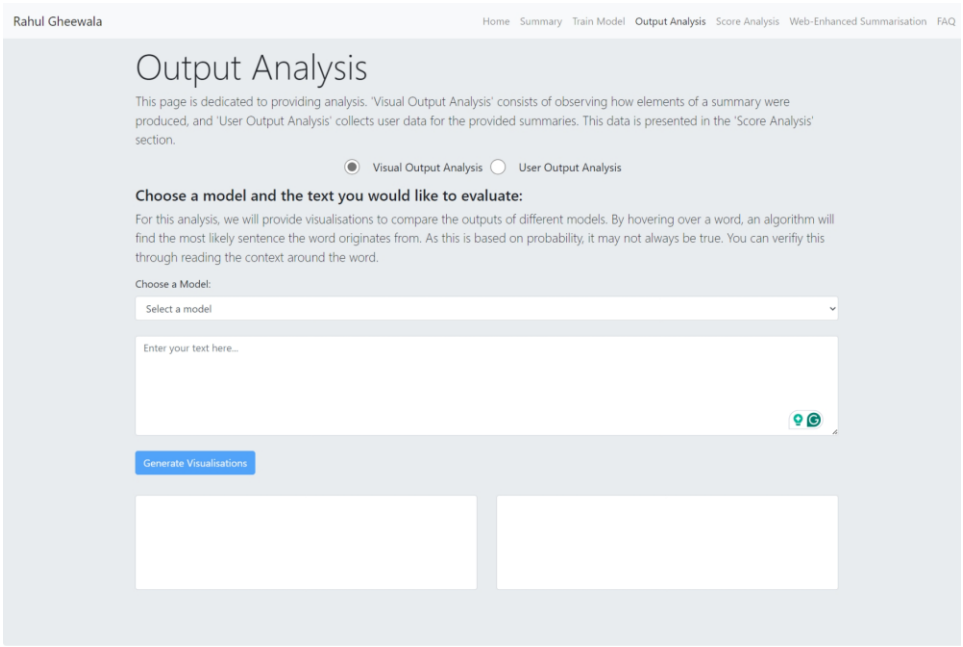

Rahul Gheewala
Home Summary Train Model Output Analysis Score Analysis Web-Enhanced Summarisation FAQ
Summary
For this page; simply input your text, choose between Extractive and Abstractive Summarisation, and adjust parameters to your liking. With three Abstractive and two Extractive models to choose from, you have the flexibility to customise your summaries according to your preferences. To learn more about the types of Summarisation and different models, please visit the FAQ.
Type of Summarisation:
Abtractive Extractive
Choose a Model:
Choose a model
Please enter your input text:
Enter your text here...
Ratio:
0 Current Value: 0.2 1
Generate Summary

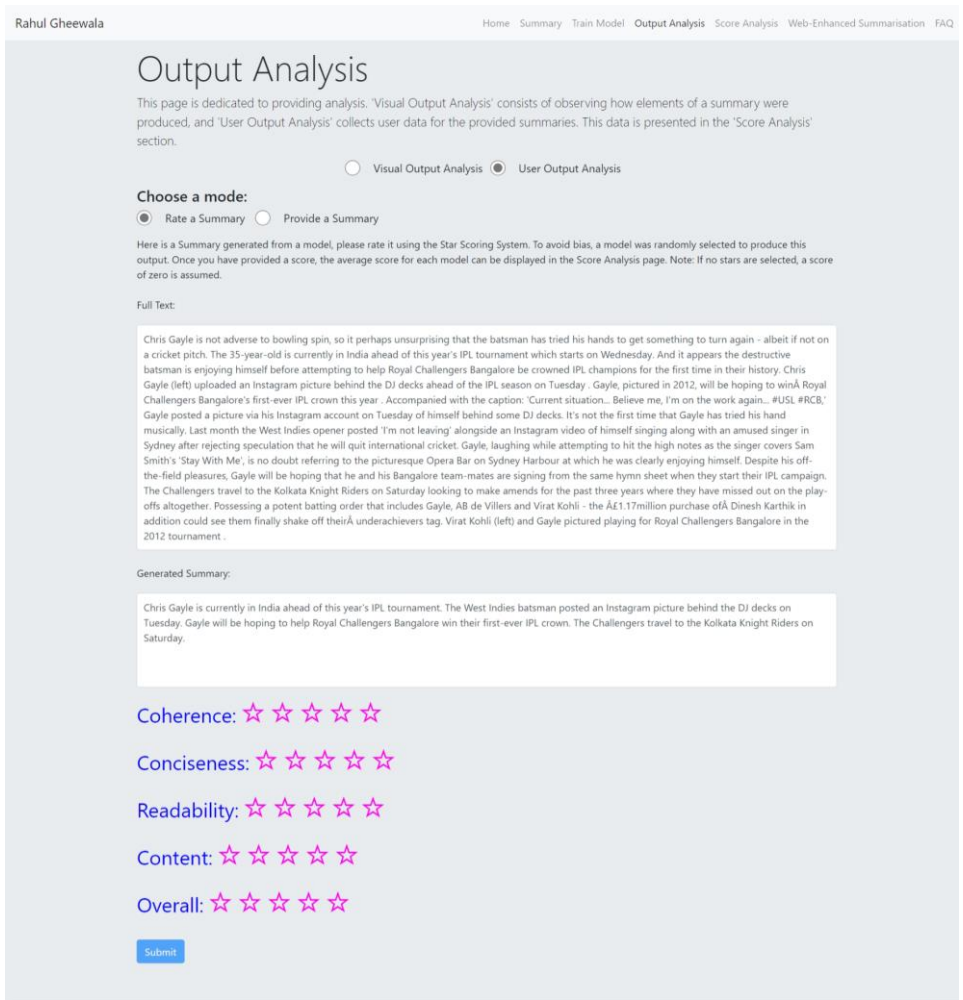
# Train Model

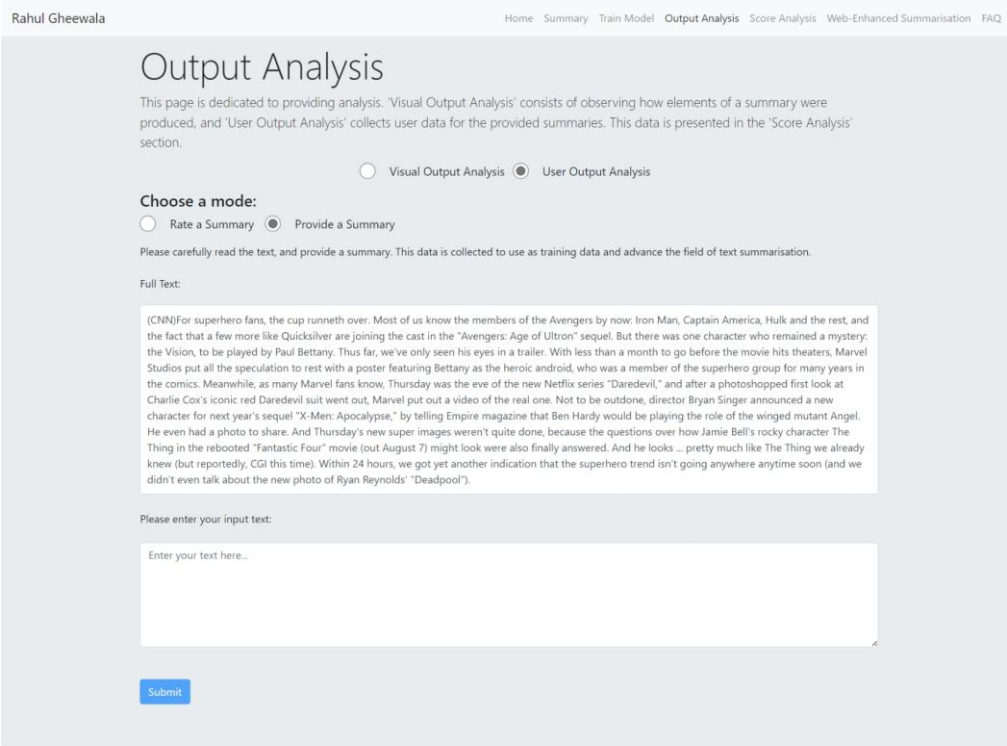

Rahul Gheewala
Home Summary Train Model Output Analysis Score Analysis Web-Enhanced Summarisation FAQ
Train Model
In this page, you will be able to train a model with your own data. You will need to provide data in the form of an Excel file containing a reference text and a summary. Training takes a lot of computing power so it is recommended to use GPU if available. It is possible to train using no GPU but this can take a very long time, and may take several hours depending on the size of the data.
Train a Model
Test a Model
Choose a Model you would like to train:
Select a model
Upload Training Data
Enter your splits:
Train Split (%):
70
Validate Split (%):
15
Test Split (%):
15
Processing Unit:
GPU (if available)
CPU
Learning Rate:
0.01
Number of Epochs:
10
Batch Number:
32
Enter your new model name...
Train Model


Rahul Gheewala
Home Summary Train Model Output Analysis Score Analysis Web-Enhanced Summarisation FAQ
Train Model
In this page, you will be able to train a model with your own data. You will need to provide data in the form of an Excel file containing a reference text and a summary. Training takes a lot of computing power so it is recommended to use GPU if available. It is possible to train using no GPU but this can take a very long time, and may take several hours depending on the size of the data.
Train a Model
Test a Model
Choose a Trained Model you would like to test:
Select a model
Please enter your input text:
Enter your text here...
Generate Summary

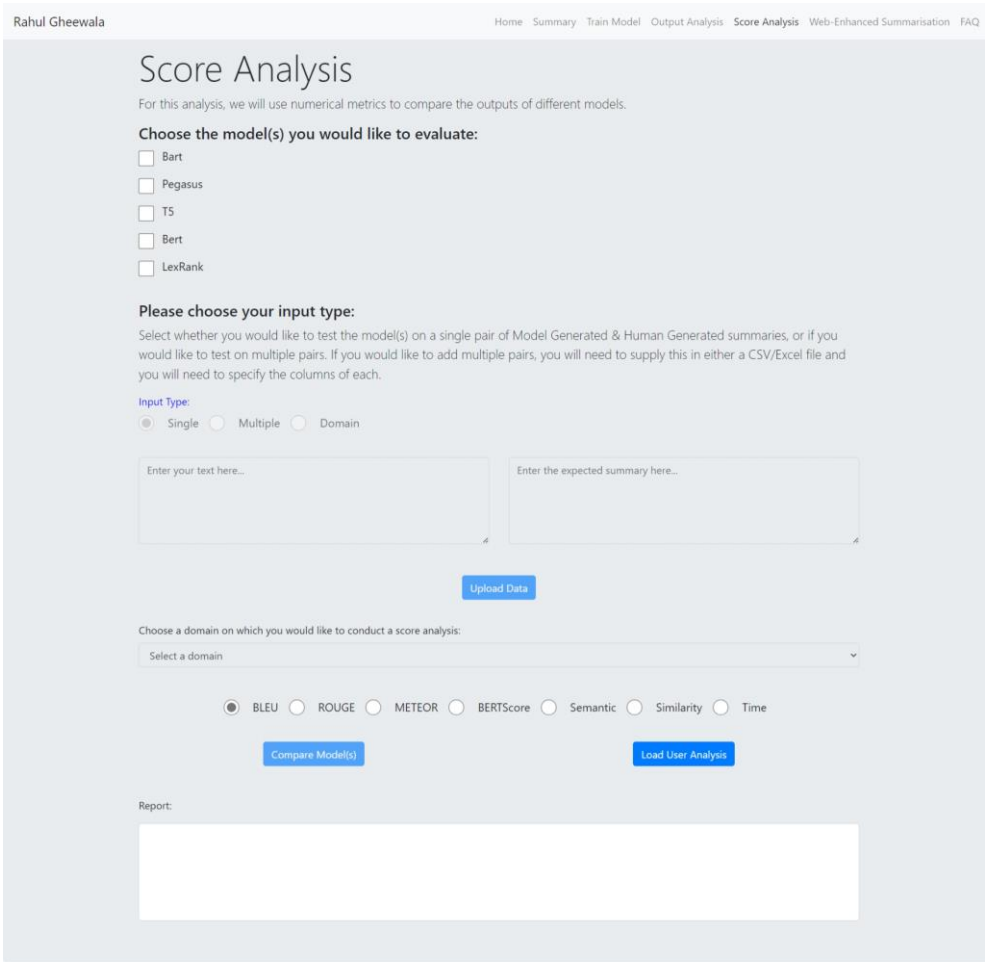
# Output Analysis



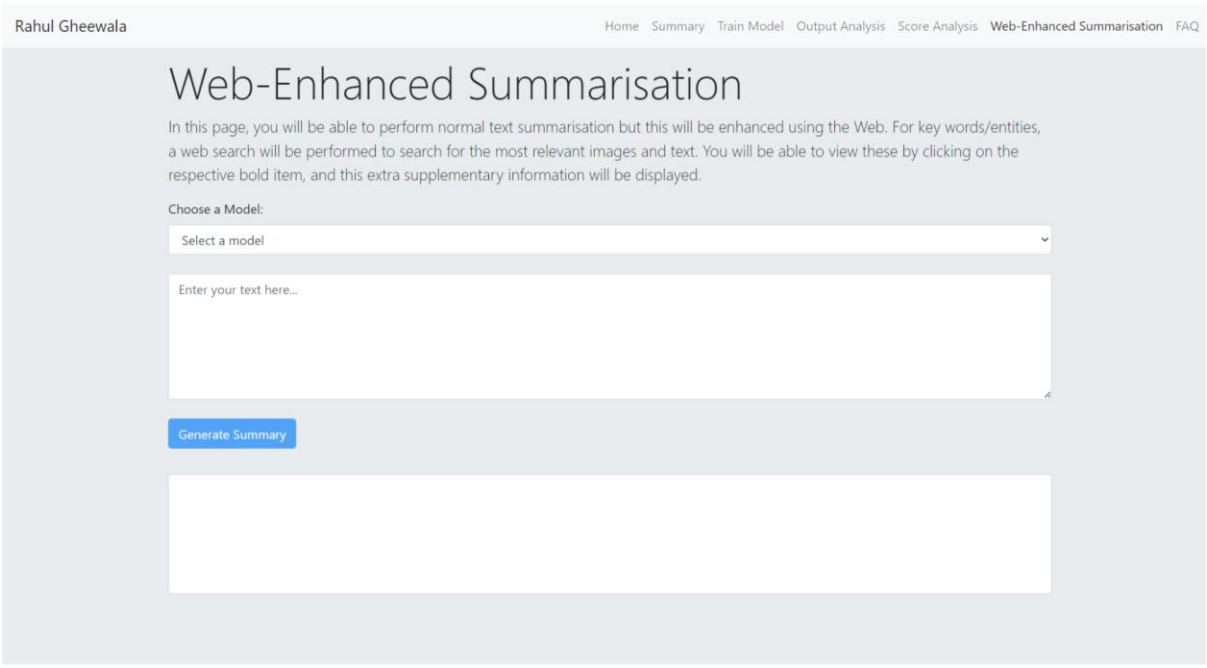





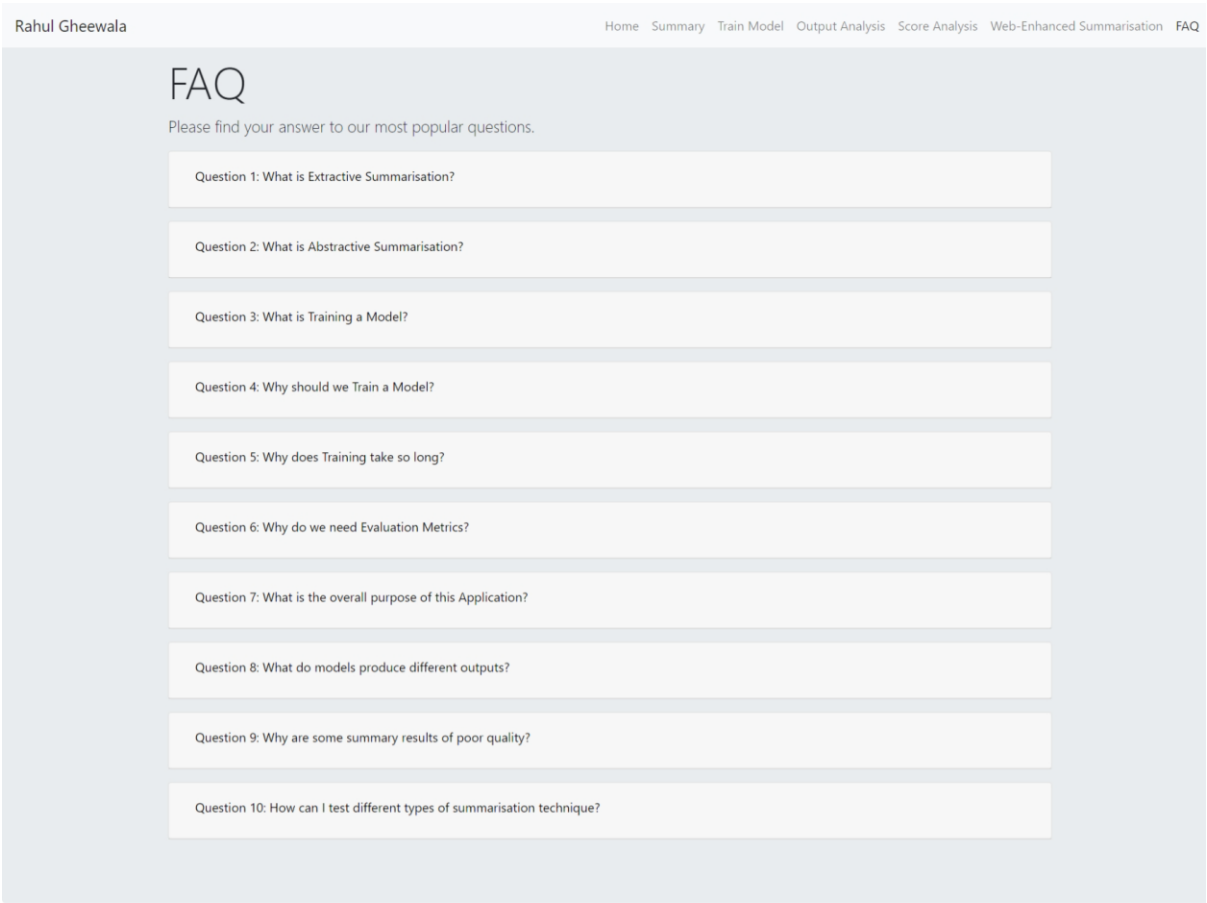

Score Analysis



Web-Enhanced Summarisation



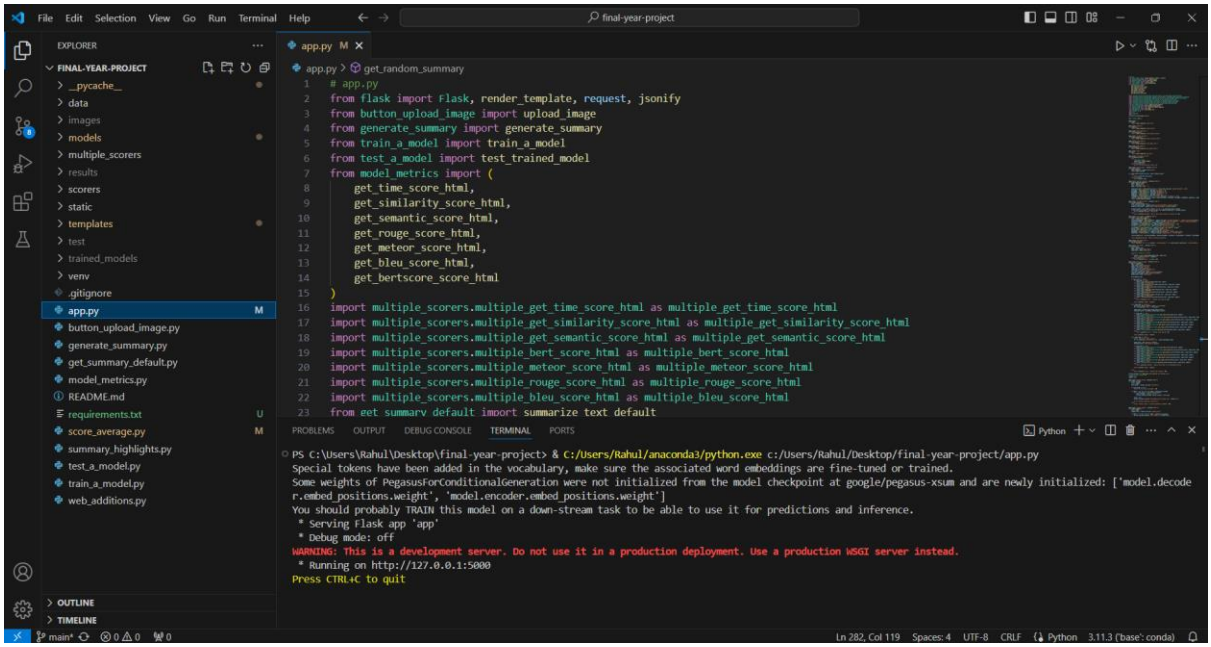

FAQ



## Appendix D- Running the Project

Repository: [Student Projects 2023-24 / rxg949 · GitLab (bham.ac.uk)]

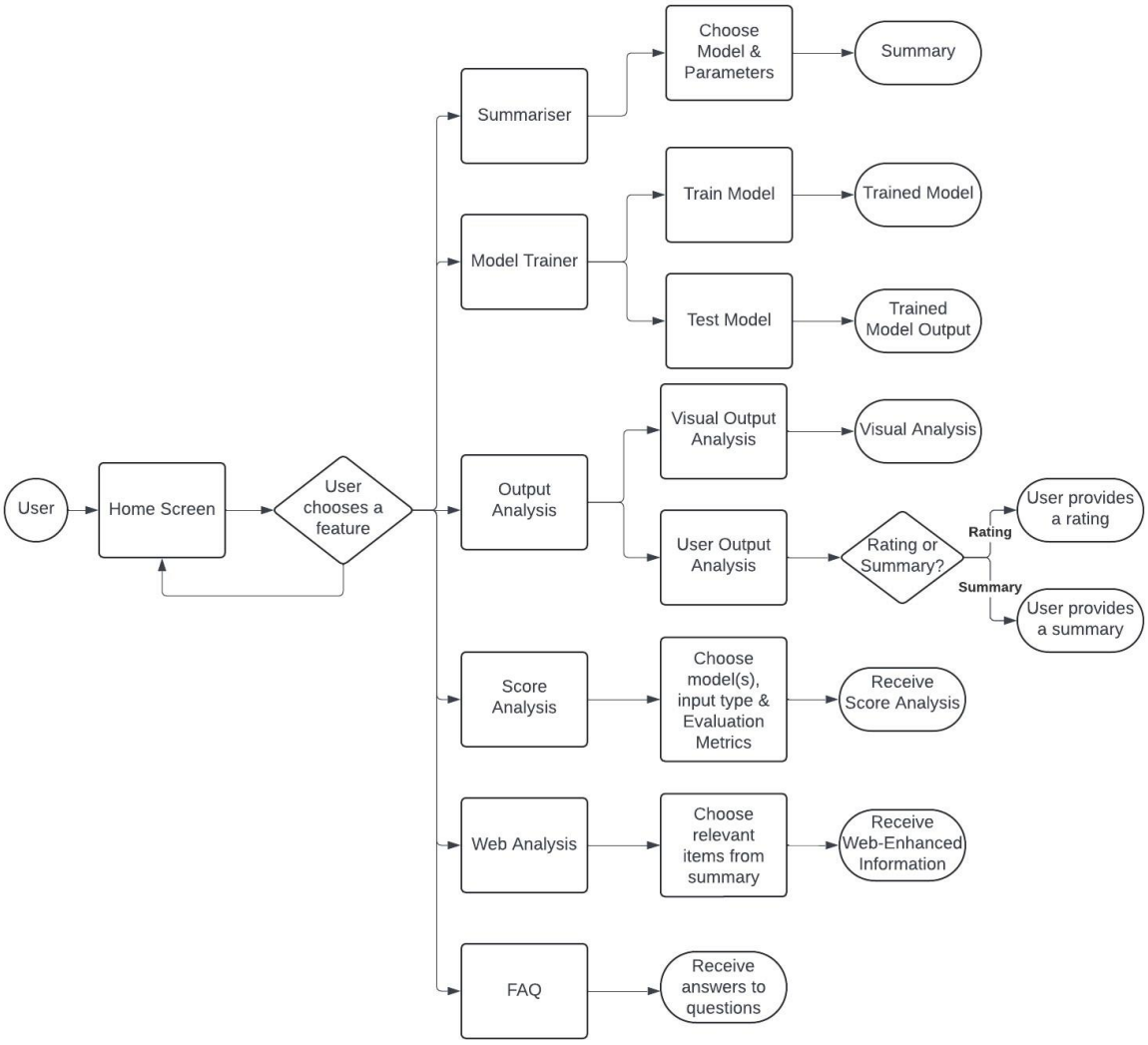
1. Clone the repository
2. Download Anaconda [43]
3. Open the repository using VSCode (or preferred IDE)
4. Activate virtual environment
5. Install packages from requirements.txt
   If any manual installation is required, please follow the instructions on the terminal. Models have been cached, but fresh downloads may still be needed.
6. Run app.py, and wait about a minute until you see this running message in the Terminal:




7. Enter the URL (http://127.0.0.1:5000) into a Browser.

## Appendix E - Full Component Diagram

An overview of our application's functionalities:




We have used terminal shapes in our example. However, this is to demonstrate the outcome of a process when using a feature. In our application, users can use the navigation bar to return to the Home Screen or directly to a new feature during or after completing a process.

## Appendix F – Personal Reflection

This project has provided invaluable lessons and experiences that will help me beyond my degree. This is possibly the most significant independent project I will ever undertake, and I am proud of my accomplishments. Having completed a year-long placement as a Software Developer with IBM, I learned my true passion was in making powerful technology accessible to clients to improve their business operations. For this reason, I chose and accepted a return offer to become a Data & AI consultant, where I can focus solely on translating complex technologies into practical solutions.

This project has prepared me for my upcoming role in many ways; for example, I have made text summarisation technology accessible to non-technical users concerning their preferences and recommendations. This can be likened to my working with clients to understand their business needs and presenting viable technological solutions. Through working with many users to test my application, I practised the skills I will use in communicating with potential clients to understand their underlying needs and requirements. This project also allowed me to hone my skills in time management, independence, resilience, adaptability, and technical abilities, which are essential to succeed in technical consulting.

I am grateful to the University of Birmingham, as my degree has prepared me significantly for this project and my upcoming role. The 'Team Project' and 'Software Engineering' modules gave me the foundations to conduct a software development report and apply software engineering principles in practice. The 'HCI' module prepared me by teaching effective user-centric methodologies, which were pivotal in helping me get started in designing an intuitive and user-friendly interface. Additionally, the 'NLP' module helped me sincerely appreciate the impact of data and how it can affect integrity. This largely steered the direction of this project, where we prioritised allowing users to perform testing and evaluation of summarisation models, allowing them to understand the impact of biased or domain-specific data on results.