Informatics Institute of Technology

In Collaboration With

The University of Westminster, UK

The University of Westminster, Coat of Arms

GenSum

A Generalized Text Summarization System using Optimized Transformers

A Product Specification & Prototype Design by

Mr. Nazhim Kalam

W1761265 | 2019281

Supervised by

Mr. Torin Wirasingha

February 2023

This Project Proposal is submitted in partial fulfilment of the requirements for the

BSc (Hons) Computer Science degree at

the University of Westminster.

ABSTRACT

Abstractive text summarization systems have been integrated with various application in the world to perform text summarization and its nothing new to the field. However, with the prior research it found that in the domain of movies the need for performance improvement is required using latest approaches than the current traditional ML & DL methods, movie review summarization plays a major role in helping users to make better decisions by matching their interest with the reviews of the movie, this saves a lot of time and also improves businesses in their sales.

In 2017 researches from Google Brain introduced NLP Transformers, which is a latest approach to solve NLP problems and its increasingly been known and used nowadays over traditional ML & DL approaches like using basic LSTM, RNN approaches. The author explored ways in which to get an optimal solution using Transformer for abstractive text summarization and yet making a generalized solution which can be adapted with respect to any domain (be it hotels, movies, restaurants) and increase its performance as the system gets used over with time.

The author was able to experiment with few of the top tier transformer architectures to filter out the optimal model and integrated an automated hyperparameter searching mechanism which will find the best set of hyperparameters to train the model. **ROUGE1 of <u>80.8</u>, ROUGE2 of <u>79.42</u>, ROUGEL of <u>80.8</u>, ROUGELSUM of <u>80.8</u>** was the optimal evaluation metric result achieved from the BART model giving the best result.

Keywords: Natural Language Processing (NLP), Machine Learning (ML), Deep Learning (DL), Recall-Oriented Understudy for Gisting Evaluation (ROUGE), Inductive logic programming (ILP)

Subject Descriptors:

- Computing methodologies → Artificial intelligence → Natural language processing → Natural language generation
- Theory of computation → Theory and algorithms for application domains → Machine learning theory → Semi-supervised learning.
- Information systems → Information systems applications → Management and querying of encrypted data.
- Security and privacy → Database and storage security → Data mining.

Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence.
DL	Deep Learning
GUI	Graphical User Interface
ML	Machine Learning
NLP	Natural Language Processing
ROUGE	Recall-Oriented Understudy for Gisting Evaluation.
BLEU	BiLingual Evaluation Understudy.
T5	Text to Transfer Transformer.
BART	Bidirectional Auto-Regressive Transformers.
BERT	Bidirectional Encoder Representations from Transformers.
PEGASUS	Pre-training with Extracted Gap-sentences for Abstractive Summarization Sequence-to-sequence
ILP	Inductive logic programming.
LSTM	Long Short-Term Memory.
RNN	Recurrent Neural Network.
CNN	Convolutional Neural Network.
SEQ2SEQ	Sequence to Sequence
RoBERTa	Robustly Optimized BERT Pre-training Approach
GPT-3	Third Generation Generative Pre-Trained Transformer
REST	Representational State Transfer
GPU	Graphical Processing Unit
API	Application Programming Interface

CHAPTER 01. INTRODUCTION

1.1. Chapter Overview

In this chapter, a series of top-tier pretrained transformer designs are optimized using automated search hyperparameter optimization in an effort to improve the performance of abstractive text summarization for movie reviews while developing a generalized solution that may be used in other domains. Along with a review of previous studies and a presentation of the anticipated project timetable, the research problem, gap, challenge, and method will be discussed in the work plan.

1.2. Problem Domain

1.2.1 Movie User Reviews

A growing number of websites, like Amazon and the Internet Movie Database (IMBD), a website for movie reviews, allow users to publish reviews for things they are interested in, along with the growth of Web 2.0, where user interaction is prioritized. (Khan, Gul, Zareei, et al., 2020)

Online movie reviews are evolving into an important information source for users, with the continuous increase in data on the web (M and Mehla, 2019). However, online users post a significant number of movies reviews every day, hence making it difficult for them to manually summarize the reviews and determine their interest in the film. One of the challenging problems in natural language processing is mining and summarizing movie reviews. (Khan, Gul, Uddin, et al., 2020).

Text summary assist users or business decision-makers by compiling and analyzing a significant number of online reviews. (Alsaqer and Sasi, 2017).

These days, the majority of people research a film's reviews before selecting or watching it on any platform, such Netflix or Amazon Prime, but we also come across conflicting reviews that can be either good or bad. While most reviews are detailed and require a significant amount of time to review, this develops a problem where users aren't able to make quicker decisions. Therefore, by summarizing the review makes it easier and faster for users to make decisions. This can also help streaming services like Netflix quickly discover the viewing habits or preferences of their users (Dashtipour et al., 2021)

1.2.2 Text Summarization

Today, there is a lot of textual material available, including news stories and reviews. Text summarizing helps us quickly find the key elements of the full piece by minimizing the quantity of text. (Mahajan et al., 2021).

Extractive summarization and abstractive summarization are typically the two methods of text summarization. When extractive summarizing, the most important lines from the context or article are plucked out without being altered in any way. Meanwhile, abstractive summarizing aims to create the sentences on its own and creates the summary; this is superior than extractive summarization since it is more meaningful to generate our own phrases inside the context rather than to utilize selected sentences from the context without any change. (Etemad, Abidi and Chhabra, 2021).

1.2.3 Transformers

Transformers in NLP is a novel architecture that aims to solve sequence-to-sequence tasks while handling long range dependencies with ease. It has surpassed competing neural models like CNN (Convolutional Neural Nets) and RNN (Recurrent Neural Nets) in terms of performance to appear as the dominant architecture for natural language processing (Wolf et al., 2020).

Transformers uses self-attention mechanism to target on selected areas of the input sentence followed by the encoder and decoder architecture (Etemad, Abidi and Chhabra, 2021).

1.3. Problem Definition

In the domain of movie review summarization, currently there are no researches done using the latest deep learning approaches (***such as Transformers***) to solve this problem, standard machine & deep learning algorithms such as Naïve Bayes, RNN have been used, the usage of advanced deep learning approaches can be utilized in order to enhance the quality/accuracy of the text summarization.

Deep learning models take longer to train but they provide greater accuracy since they can simultaneously automate feature extraction and classification, whereas machine learning algorithms require feature selection at first. Therefore, applying deep learning techniques will help to improve the quality of text summarization and help the user in making better decisions (Etemad, Abidi and Chhabra, 2021).

1.3.1 Problem Statement

No prior research has looked into applying cutting-edge deep learning methods like Transformers to produce abstractive summaries from movie reviews, which can improve text summarization. This solution aims to be generic and accessible to any sector. (Khan, Gul, Zareei, et al., 2020).

1.4. Research Questions

RQ1: What are the top tier transformer architectures widely used and know for NLP problems related to text summarization?

RQ2: How can a pretrained transformer architecture be fine-tuned to get the optimal hyper parameters and to automate it for model retraining?

RQ3: What kind of evaluations should we perform after fine-tuning to filter out the best transformer architecture?

RQ4: How can domain generalization be integrated for system?

1.5. Research aim & Objectives

1.5.1 Research Aim

The aim of this research is to design, develop and evaluate an optimal adaptive generalized transformer architecture from a range of popularly used architectures by fine-tuning via automated hyperparameter optimization, therefore obtaining the recommended architecture's optimum performance

A fully working system that can be utilized to perform abstractive text summarization based on the user input from any domain (movie, hotel, ecommerce etc.…) will be created by this research project. The quality of the resulting text summary or performance optimization will be the **main points** of emphasis. To get the best result, the usage of data preparation, data analysis, conducting hyperparameter tuning, and evaluating the models will be investigated.

Components will be built, necessary information will be gathered and researched, and performance will be assessed. The system may be utilized for private or public purposes on both a hosted server and a local browser. The data science models' source code will be made available in a public repository for future research and use. A research paper will be published and it end of this study.

1.5.2 Research Objectives

For the research to be considered successful, its goals must be fulfilled.

Table 1: Research Objectives (*Self-Composed*)

Objective	Description	LO	RQ
Literature Review	Complete a thorough critical review of earlier related work. **RO1:** Make a preliminary investigation on existing abstractive text summarization using deep learning approaches. **RO2:** Make a preliminary investigation on why transformers architecture was the chosen deep learning choice for this research. **RO3:** Analyze the top tier transformer architectures widely used. **RO4:** Analyzing how the models can be fine-tuned via hyperparameter optimization. **RO5:** Analyzing the different approaches used for model evaluation. **RO6:** Analyze how the model can be generalized for every other domain.	LO1, LO4, LO8	RQ1, RQ2, RQ3, RQ4
Methodology Selection and SLEP Framework	This defines the outline structure for the requirement analysis and the design process followed by the social legal ethical and professional issues. **RO1**: Analyzing the Research Methodology approaches. **RO2**: Analyzing the Development Methodology approaches. **RO3**: Analyzing the Project Management Methodology approaches. **RO4**: Analyzing the Solution Methodology approaches. **RO5**: Analyzing the Social, Legal Ethical and Professional Issues which could develop during the phase of the project.	LO2, LO6	RQ4, RQ2, RQ1

Requirement Elicitation	Establishing the project's requirements using appropriate methods and instruments to address the anticipated research gaps and challenges based on related past research. **RO1:** Gathering information related to the expected metadata required for the dataset to contain for the model training. **RO2:** Gathering the requirements of transformer architectures for fine-tuning and understand the end to end user expectations. **RO3:** Getting insights from domain experts to build a suitable system. **RO4:** Gathering the requirements for handling generalization.	LO1, LO3, LO5	**R**Q4, **R**Q2, **R**Q1
Design	Considering the following when developing the suggested system: **RO1:** Design a component to preprocess the dataset for the respective model inputs. **RO2:** Design a component to store the top tier transformer models with their respective metadata, to use throughout. **RO3:** Design a hyperparameter tuning component that can improve accuracy of the transformer model. **RO4**: Design high-level architecture for the system.	LO1, LO5	**R**Q2
Implementation	Setting up a mechanism capable of addressing the gaps that were intended to be covered. **RO1**: To develop data preprocessing component. **RO2**: To develop a component that handles and stores the top tier transformer architectures for fine-tuning. **RO3**: To develop the automated hyperparameter search component that handles all the top tier architectures assigned. **RO4**: To develop a component for the model evaluations for the measured hyperparameters	LO1, LO5, LO7	**R**Q2, **R**Q3

Evaluation	Testing and evaluating the developed system (including the data science models with the suitable metrices) **RO1**: Performing unit test, integration and performance testing along with a test plan created. **RO2**: Evaluating all the transformer architectures used for fine-tune experimentations, using recommended scores such as (ROUGE or BLEU SCORE).	LO1, LO5	**R**Q3
Documentation	Keeping track of and documenting the study project's ongoing progress and any challenges encountered.	LO6, LO8	-
Publication	Ensure that the documentation, reports, and papers are well-structured and include a critical analysis of the research. **RO1**: To publish a research paper on the related work done. **RO2**: To publish the testing & evaluation results of the work done. **RO3**: To publish the code implementation repository as public to be access by future research investigations, along with the models and datasets	LO4, LO8	-

1.6. Novelty of the Research

1.6.1 Problem Novelty

The problem novelty of this research is, the lack of attempt to increase transformer performance in order to get better textual summarizing outcomes (Khan, Gul, Zareei, et al., 2020).

1.6.2 Solution Novelty

The solution novelty for this problem is performing an automated approach for hyperparameter tuning & creating a retraining mechanism with newly exposed data to enhance its performance further using the optimal transformer (Etemad, Abidi and Chhabra, 2021)..

1.7. Research Gap

Based on previous work done (Khan, Gul, Zareei, et al., 2020) related to abstractive text summarization on movie reviews, the literature identify for the need of using advanced deep learning approaches to improve the performance of text summarization for this movie domain over traditional machine learning approach.

This project focuses on Empirical gap in the Movie Domain, as well as Theoretical and Performance gaps in the area of transformer optimization. Transformers plays a major role in the field of deep learning especially at problems related to Natural Language Processing, by performing hyperparameter optimization on several transformer architectures we can contribute to the enhanced quality of abstractive text summarization and create a generalized model which can be adapted with the respective domains usage and improve the performance.

1.8. Contribution to the Body of knowledge

Improving the performance of an existing solution is very common in the field of data science, as we can explore new algorithms or fine-tuning existing algorithms to get better results. The contributions for this project can be classified as theoretical contributions and domain contributions.

The following is a summarization of the authors contribution:

- ***Abstractive Text Summarization:*** *Automated Hyperparameter optimization + Model Retraining + Transformers + Deep Learning*
- ***Movie User Review & Generalization:*** *Research domain target is for Movie reviews, in addition the author makes the system generalized to adapt to any domain area*.

1.8.1 Research Domain Contribution

There are various deep learning techniques that can be used to handle abstractive text summarization, however with respect to previous researches done, (Zhang, Xu and Wang, 2019) it is found that ***transformers*** outperform most of the other deep learning approaches as of today but there was no much research on optimizing them for a much better performance.

This research will be focused on creating a ***generalized solution*** by achieving the optimized transformer architecture from a couple of the top tier existing architectures, via fine-tuning and performing hyperparameter optimization along with handling abstractive text summarization (Liu and Wang, 2021), therefore we are able to maximize the performance of the recommended architecture. The author plans out to make use of generalization where any domain when used the model will be optimizing and adapting towards their respective domain.

1.8.2 Problem Domain Contribution

Neural Networks makes up the backbone of deep learning algorithms which enables them to process complex unstructured data over normal means of machine learning algorithms (Mahajan et al., 2021). It is found that, the need for using advanced deep learning approaches has not been explored in the domain of movie review summarization.

Given that transformers perform well in this field, the proposed solution for this domain will be finding the recommended architecture along with hyper-parameter optimization, to reach its best performance. An additional contribution will be that, the proposed solution will be generalized to any other domain linked with the field of NLP text summarization.

1.9. Research Challenge

The main objective of this research is to achieve the generalized optimal transformer architecture for the field of NLP abstractive text summarization. Transformers were introduced in 2017 by a team at Google Brain and are the most used choice for NLP problems replacing RNN models, given that this architecture was introduced not much longer back brings to a point where there is a lack of research done in the area of transformer optimization for the purpose of abstractive text summarization. (Wolf et al., 2020).

Therefore, creating and finding the recommended transformer architecture along with the optimal parameters which also handles generalization becomes a challenge with very fewer resources to look up to.

Additionally, identifying suitable datasets for this domain (Movie Reviews Summarization) and Generalization is challenging and necessitates a substantial amount of effort in data preprocessing where it is important since we are dealing with NLP and performance optimization related domain.

1.10. Chapter Summary

The author outlined the research effort, explained why the research and problem were innovative, and discussed potential challenges that could develop while attempting to tackle them in this chapter. Additionally, the key goals that must be attained for the research to be deemed successful were outlined and connected to the necessary learning outcomes for the degree.

CHAPTER 02. SOFTWARE REQUIREMENTS SPECIFICATION

2.1. Chapter Overview

In this chapter, the author describes how to identify the essential needs and how to gather them. To carefully record the engagement of possible stakeholders, their interaction points, and their separate responsibilities, a rich picture diagram and stakeholder onion model are used. The chapter also discusses the methods used for requirement gathering and the results that were used to create functional and non-functional requirements, use case diagrams, and prototypes.

2.2. Rich Picture

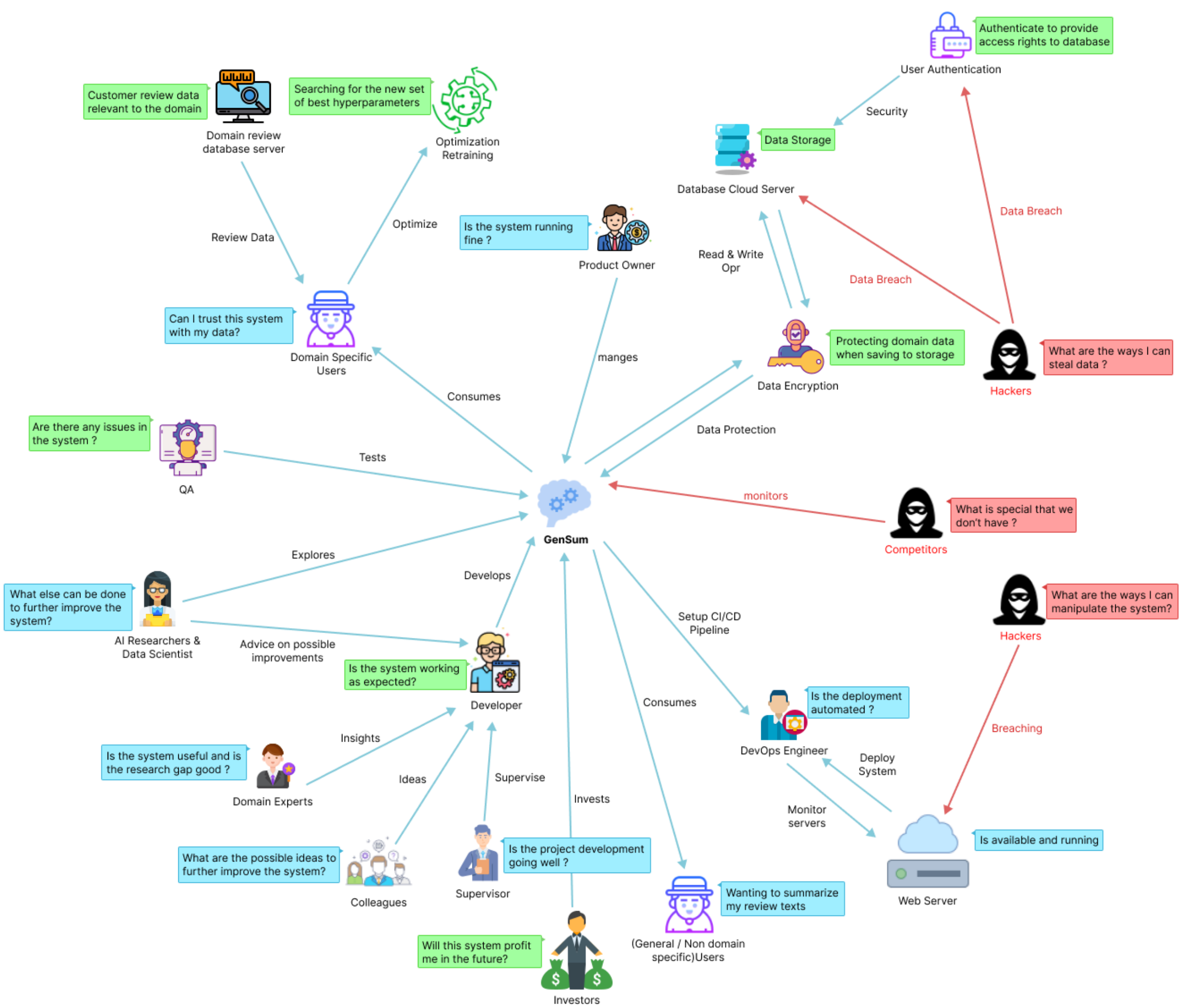


Figure 1: Rich picture diagram (*Self-Composed*)

The diagram above depicts a bird's-eye view of the surrounding region, as well as how certain stakeholders might interact with the system and profit from it. Along with the knowledge gained by the researcher to improve the system, the potential negative impacts on the design and prospective critical analyses are also identified.

2.3. Stakeholder Analysis

The section that follows acknowledges significant stakeholders involved with the system, their relationships, and their individual roles. The stakeholder onion model represents this information, and stakeholder perspectives elaborate on it.

2.3.1 Stakeholder Onion Model

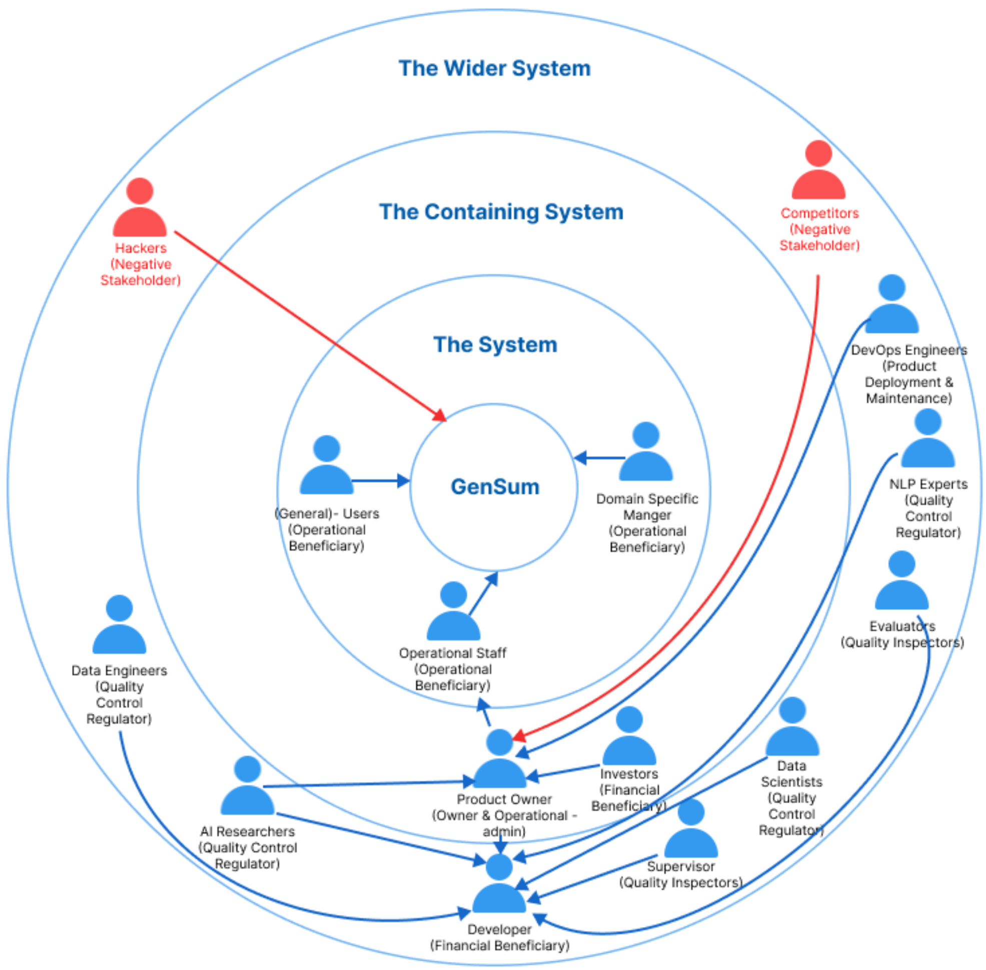


Figure 2: Stakeholder onion model (*self-Composed*)

2.3.2 Stakeholder Viewpoints

Table 2: Stakeholder viewpoints & Requirements (*self-Composed*)

Stakeholder	Role	Benefits/Description
Developer	Functional beneficiary	Works on developing the system
Investors		Profit is generated through system investment and money from marketing and user subscriptions.
Product Owner	Owner & Operational admin	Owns the system and has control over the system
Data Scientists	Quality Control Regulator	Provides performance optimizations for data science models and algorithms.
Data Engineers		Provides information on prospective data that might be utilized to provide the best possible suggestions.
AI Researchers		Conduct research in the specified area to enhance and implement reliable text summarizing models.
NLP Experts		Provides professional advice and insights into the subject to improve the system's performance.
Domain Specific Manager	Operational Beneficiary	Model retraining will be performed with the new data entered into the system for summarization.
General Users		Summarization output will be general for the user.
Operational Staff		Ensures that the system is up and functioning while responding to user requests and problems.
DevOps Engineers	Product Deployment & Maintenance	Makes ensuring the system is up and running in the cloud and is serving users without being throttled
Hackers	Negative Stakeholder	May manipulate the review data stored in the database which will affect the retraining process.
Competitors		May build competing systems that may outperform the existing system.
Evaluators	Quality Inspector	Analyzes and tests the system to verify if it is ready for production usage.

Supervisor		Checks to see if the system development is progressing well without any issues.

2.4. Selection of Requirement Elicitation Methodologies

Various methods were used, including literature reviews, surveys, interviews, prototyping, brainstorming, and self-evaluation, to gather the needs for the research project. Thereafter, the author explains why these particular requirement elicitation techniques were chosen.

Table 3: Requirement elicitation methodologies (*Self-Composed*)

Method	**Description**
Literature Review	In order to find research gaps in the area of interest they had selected and the planned topic of their study, the author first did a thorough literature review. To identify research gaps that can be filled with the use of technology, the author investigated current systems and associated technologies.
Survey	As a survey tool, a questionnaire was used to gather requirements and viewpoints from potential users of the recommended system. This kind of survey will help the author understand how people are thinking about and anticipating the prototype. It will also allow the author to justify whether or not the chosen solution would help the intended users.
Interviews	Interviews were conducted to get expert insight into domain-specific requirements and to choose the most effective strategy for resolving the current problem while expanding the corpus of knowledge. Since the field is new and the level of technical competence required is quite specific, interviews were found to be the best source of information. This method provided for a qualitative assessment of the proposed system, enabling the detection of any flaws or problems that would need to be fixed during prototyping.
Prototyping	Prototyping would enable the author to test and assess the prototype while repeatedly trying out numerous different implementations to uncover any possible areas for improvement because the project was chosen to follow the Agile Software Development Life-cycle.

Brainstorming	Whether you're attempting to come up with a broad subject, focus more narrowly, or decide what evidence to utilize for a single paragraph, brainstorming is a valuable strategy for generating ideas at every stage of the research process. The author conducted several brainstorming meetings with colleagues at various project phases to assess the system.
Self-Evaluation	Self-evaluation is done in order to examine the currently available applications, do competitor analyses on the current systems, and get insight into how negative stakeholders, such as hackers, can breach the system and find a way around to protect the data and the system.

2.5. Discussion of Findings

The relevant key stakeholders are split up into groups where the chosen best methodology was used for each group. **APPENDIX A.1** contains a complete breakdown of these stakeholders.

2.5.1 Literature Review

Table 4: Literature review findings (*Self-Composed*)

Discussion of Findings	**Citation**
In the completion of the literature review on the existing work done, it was identified that abstractive text summarization systems for customer reviews helps users to make better and quicker decisions on their actions let it be on buying products or watching a movie, user review summarization proves to save time for customers.	(Boorugu, Ramesh and Madhavi, 2019)
When exploring technologies that can be applied to achieve the required outcome, it was clear that traditional machine learning and deep learning approaches were only used for abstractive text summarization in the domain of movie reviews. Leaving the usage of advanced deep learning approaches such as Transformers untouched for this domain.	(Khan et al., 2020)
It was identified that transformer optimization has not been looked into when working with transformers in abstractive text summarization domain in general and not specific to the movie domain.	(Gupta et al., 2021)

Dataset related to working with model generalized has been used previously and is recommended to be used if researchers are willing to work with the idea of generalization for the domain of abstractive text summarization.	(Kouris, Alexandridis and Stafylopatis, 2019)

2.5.2 Brainstorming

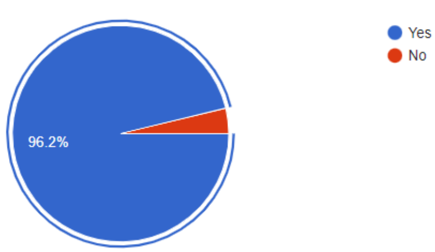
The author engaged in brainstorming across various project phases. These were carried out both with the authors' colleagues and supervisors as well as through a self-analysis process.

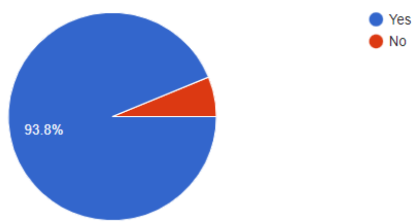
Table 5: Observations findings (*Self-Composed*)

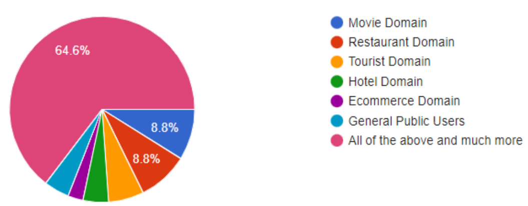
Criteria	Discussion of Findings
Able to figure out several other research gaps/ limitations which can be fit into the current project domain in order to increase the magnitude of research effort.	Multiple ideas were brought up as the result of the brainstorming session. The concept of creating a performance adaptive generalization model was brought up by the authors supervisor, along with several other approaches to increase the performance of the system exponentially such like making use of the new data from the domain users for retraining and combine all data with the common domain for retraining since the data count increases with respect to the common domain user.

2.5.3 Survey

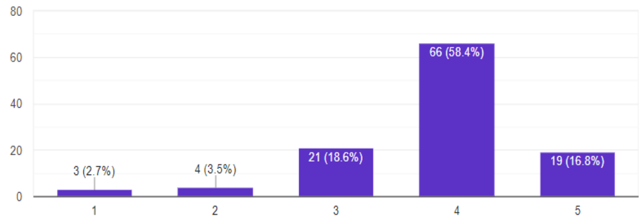
Table 5: Survey analysis (*Self-Composed*)

Question	Have you ever realized that reading lengthy reviews takes a significant amount of time?
Aim of question	To determine whether the audience as a whole considers reading lengthy reviews to be a time-consuming activity.
	**Findings & Conclusion** It can be concluded that a large part of the audience (more than 90% of the audience) finds that's reading lengthy reviews is a time-consuming hassle which also proves that they would appreciate if

<table>
<tr><td colspan="2">there would be a quicker approach for this problem, like a summarization. This also concludes to see a positive correlation from the results which was expected from the author of the project.</td></tr>
<tr><td>Question</td><td>Do you believe that developing a generic system for all domains would be a wise course of action?</td></tr>
<tr><td>Aim of question</td><td>Ensuring that developing a generic system would be beneficial in all domains</td></tr>
<tr><td colspan="2">Findings & Conclusion


It can be concluded that most of the participants (more than 90% of the audience) agrees that developing a generalized system which can adapt to the domain as they use, is beneficial and worth the effort to process with the project research. This also concludes to see a positive correlation from the results which was expected from the author of the project</td></tr>
<tr><td>Question</td><td>Who do you think will most benefit from this system?</td></tr>
<tr><td>Aim of question</td><td>Getting to know about the thoughts of the participants about to whom the system would mostly benefit from?</td></tr>
<tr><td colspan="2">Findings & Conclusion


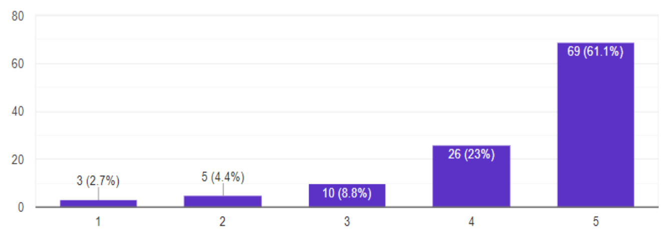
It can be concluded that a majority of the participants (more than 60%) finds that this system will benefit the movie, restaurant, tourist, hotel, ecommerce domains (these domains were considered since they are mostly interacted with the users on a daily bases and uses customer reviews for their domain as a part of their business) as well as the general users.</td></tr>
<tr><td>Question</td><td>How much do you think that this system would benefit you?</td></tr>
<tr><td>Aim of question</td><td>Getting to know how much the system would benefit the general participants which are NOT domain specific</td></tr>
<tr><td colspan="2">Findings & Conclusion</td></tr>
</table>



From the statistics graph, it can be concluded that roughly 75% of the audience finds that the system would benefit them for their general work or needs given that it's not domain specific to them, which is a positively correlated result from the achieved statistics.

Question	How much do you think that this system would benefit businesses?
Aim of question	Getting to know from the participants as to how much the system would benefit businesses/domains in solving this problem.

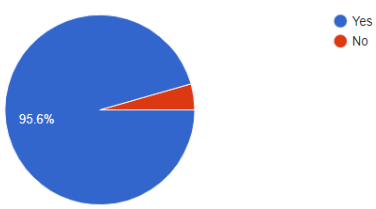
Findings & Conclusion



From the statistics graph, it can be concluded that roughly 84% of the audience finds that the system would benefit the businesses, which is a positively correlated results from the achieved statistics and that's what the author expected to achieve.

Question	Before making a reservation or booking a movie or a hotel, do you read the customer reviews?
Aim of question	Getting an idea from the audience if in general they give importance to customer/user reviews to any domain before consuming their product or services.

Findings & Conclusion



It can be concluded that a majority of the participants (more than 95% of the audience) agrees that they value and read customer reviews before they consume one's product or service. Therefore, making customer reviews a major contributing factor for business growth.

Question	How much you think customer reviews are important with respect to any domain?
Aim of question	Getting an idea from the audience to see how much they value customer reviews.

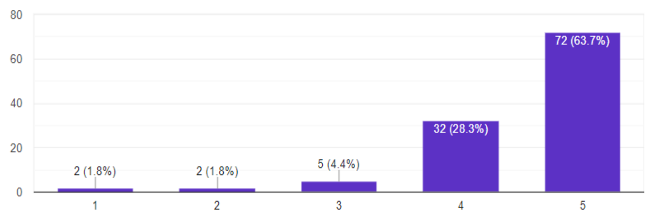
Findings & Conclusion  From the statistics graph, it can be concluded that roughly 90% of the audience finds that customer/user reviews are very important irrelevant to the domain, which is a positively correlated results from the achieved statistics and that's what the author expected to achieve.	
Question	Which additional features would you want to see in this system.
Aim of question	To identify the systems non-functional requirements which could potentially improve the system.
Findings & Conclusion The majority of participant responses were concerned with classifying the review text's sentiment after it had been summarized and with managing a list of review uploads so as to add filtering for the summarized review text based on the sentiment, whether it was positive or negative, along with the sentiment score.	

Table 6: Survey thematic analysis (*Self-Composed*)

Code	**Theme**
Convenience	User-friendly
Adjustability	Flexibility
Theme	**Conclusion**
Convenience	A group of participants required to upload more than one review and a time/bulk at once.
Adjustability	A majority of the participants requested for sentiment of the summary and the sentiment score to be also included with the output.

2.5.4 Interviews

Interviews with experts and researchers in the relevant domains were performed to obtain insights on the technical domain competence. To determine the project requirements, experts and researchers in ML and abstractive text summarizing systems were chosen. 2 PhD candidate in ML and Computational Linguistics, 1 NLP Researcher, 2 Software Architects, 1 Software Engineer and 1 Lecturer with MSc completion were interviewed. The interview outcomes were processed to a **thematic analysis** based on the following themes. Interview participant details can be found at over **APPENDIX A.2**

Table 7: Interview thematic analysis (*Self-Composed*)

Code	Theme
Data handling	Data Collection & Data Preprocessing
Transformer architectures	Best performing transformer architectures
Generalization	Handling adaptive generalization
Research scope	Research gap and scope
Hyperparameter tuning	Automatic hyperparameter tuning & model retraining
Hybrid transformers	Looking into hybrid transformer combinations
Custom transformers	Customizing the transformer architecture
Prototype	Prototype features and suggestions
Business benefits	Understanding which and how businesses would benefit
Evaluations	Understanding the importance and evaluation ways

Theme	Conclusion
Data handling	Data accessibility and data preparation techniques are crucial considerations for a data science project. Since every domain would initially employ the same model, PhD candidates proposed utilizing validated and well-researched datasets for the field of generalization to ensure the quality of data. Since text data may contain characters from other languages unless the project is restricted to English language support alone, NLP experts raised worry about the language of the text included in the project scope.
Transformer architectures	The interviewees stated that NLP tasks like text summarization and sentiment analysis may be successfully solved using transformer designs like BERT, GPT-2, Roberta, T5, and others. They advised using the most recent version of these models because they are frequently upgraded and developed in new, improved forms. To keep track of these upgrades, they advised examining

	daily analytics from websites like Hugging Face, such as download and like counts.
Generalization	Due to their scalability and performance advantages, the software engineers and architects suggested employing NoSQL databases like MongoDB or Firebase for storing data related to domain specific managers.
Research scope	Experts agree that using optimized transformers to solve the issue is a brilliant idea, but given the project's time constraints, they advise prioritizing the movie domain and postponing the task of developing a broader adaptive solution.
Hyperparameter tuning	The NLP researchers and Lectures suggested several ways of using tools and libraries to help with hyperparameter tuning since doing this manually is very time consuming and unnecessary effort.
Hybrid transformers	The hybrid transformer combination with ensemble techniques was well-loved by PhD applicants, but many believed the project scope was growing too large and risky for the time available.
Prototype	Interviewees are interested in the novel domain-specific retraining system and recommended incorporating a pretrained model for sentiment analysis if time allows.
Business benefits	Most of the interviewees suggested the Movie domain, Tourism, Ecommerce, Book, Researchers would find this useful in summarizing their customer reviews on their businesses.
Evaluations	PhD candidates and NLP experts emphasized the need for evaluations in the adaptive generalization model and suggested limiting the project to a maximum of three domains for easier comparison and clearer demonstration.

2.5.5 Self Evaluation

Comparing similar products from competitors and existing products gives the author an idea of making the project more unique and distinguish new approaches to solve the problem (**Competitor Analysis**). The author will also self-evaluate as to what data needs to be protected and how from the hackers. Few of the abstractive text summarization tools which are out there are listed and is available at **APPENDIX A.3**.

In the case of hackers stealing data from the database, **data encryption** can be applied therefore database will only contain the encrypted text data which will be then later decrypted from the decryption key when need, this will be most needed when performing the model retraining.

2.5.6 Prototyping

Table 6: Prototyping findings (*Self-Composed*)

Criteria	**Discussion of Findings**
In-order to look into the feasibility of continuing the project research a prototype was planned to be worked on.	During the iterative prototype process, the author encountered several requirements and obstacles, including the challenge of finding a suitable dataset with desired metadata in the movie domain. After intensive evaluation, the author discovered a large dataset with 8 million entries, but preprocessing it was difficult due to its size and noisy text. To automate the hyperparameter search, the author experimented with a framework called "Optuna." The system will be retrained using new data from the domain user, and the author plans to study at least three top-tier transformer designs to choose the best one.

2.5.7 Summary of Findings

Table 7: Summary of findings (*Self-Composed*)

Id	**Finding**	**LR**	**Survey**	**Self-Evaluation**	**Interview**	**Brainstorming**	**Prototyping**
1	The proposed system would benefit businesses (domain specific users) and general users (not domain specific)		✓			✓	
2	For the movie domain the limit of abstractive text summarization can be further pushed using optimized transformers to increase performance this being the existing limitation	✓			✓	✓	
3	It’s clear that customer/user reviews are valued and reviewed mostly by a vast majority of the audience before	✓	✓		✓	✓	

	they consume or use any product or service (applies to any domain)						
4	It’s clear that users spend lot of time review long reviews and they would like it being short to save time and make quicker decisions.	✓	✓			✓	
5	Hyperparameter tuning is one way to increase the performance of the system and it can be done both manually or by automated tools like Raytune, Optuna etc.…	✓			✓		✓
6	Data preprocessing for the domain of Movies and Generalization is requires a lot of effort since the datasets are mostly raw data difficult to find specially in the case of movie reviews (with expected metadata)	✓					✓
7	Additional features such as sentimental and sentimental score of the review summary is mostly expected from the users.		✓				
9	It’s clear on what are the suitable evaluation metrics to be used for abstractive text summarization.	✓			✓		
10	It’s clear on what the top tier transformer architecture that could be explored.	✓			✓		
11	Making use of larger new data for retraining for a specific domain, from companies/businesses who uses data which are of the same domain. (e.g.: - 50 different restaurants data can be combined for retraining give that the domain is “Restaurants”)				✓	✓	
12	Making use of data encryption to protect the data from hackers breaking into the database to steal data.			✓		✓	

2.6. Context Diagram

The boundaries and interactions of the system should be established before development. The graphic below shows how the system is situated.

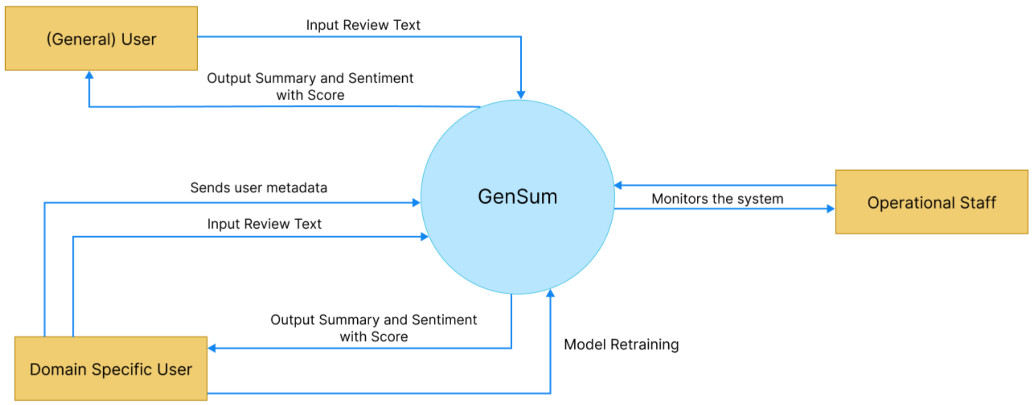


Figure 3: Context diagram (*Self-Composed*)

2.7. Use case Diagram

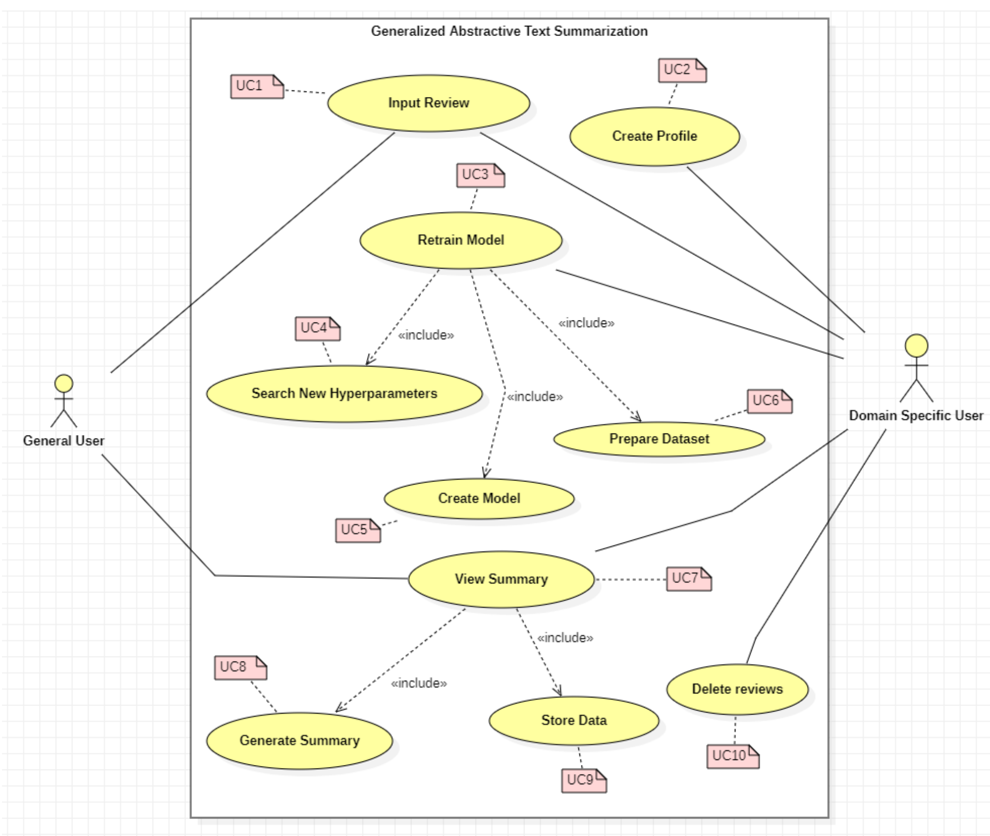


Figure 4: Use case diagram (*Self-Composed*)

2.8. Use case Descriptions

Usecase diagrams with the highest importance are given below, the rest of the Usecase descriptors are available at **APPENDIX A.4**.

Table 8: Use case description UC:07 (*Self-Composed*)

<table>
<tr><td>Use Case Name</td><td colspan="2">View Summary</td></tr>
<tr><td>Use Case Id</td><td colspan="2">UC:07</td></tr>
<tr><td>Description</td><td colspan="2">Displays a summarized version of the uploaded review text from the domain user’s end.</td></tr>
<tr><td>Primary Actor</td><td colspan="2">General User, Domain Specific User</td></tr>
<tr><td>Pre-Conditions</td><td colspan="2">The text review data must go through specific text preparation techniques before the summary can be produced.</td></tr>
<tr><td>Extended use cases</td><td colspan="2">None</td></tr>
<tr><td>Included use cases</td><td colspan="2">UC10, UC02</td></tr>
<tr><td>Trigger</td><td colspan="2">A user selects to summarize a given customer/user review text.</td></tr>
<tr><td rowspan="2">Main flow</td><td>Actor</td><td>System</td></tr>
<tr><td>1. The user enters the review text on the text field from the GUI.
2. Clicks on “Generate Summary” from the GUI</td><td>3. The system does the data preprocessing for the input review text.
4. Loads the generalized transformer model.
5. Generates the summary using the model.
6. (If Domain Specific User) stores the input review and summary into the database.
7. Returns the summary response back to the GUI</td></tr>
<tr><td>Alternative flows</td><td colspan="2">None</td></tr>
<tr><td>Expectational flows</td><td colspan="2">Displays an error message if the network request fails (server is down, or internet issues from client).</td></tr>
<tr><td>Post Conditions</td><td colspan="2">Success end condition: The user is presented with the summarized review text.</td></tr>
</table>

Table 9: Use case description UC:03 (*Self-Composed*)

<table>
<tr><td>Use Case Name</td><td colspan="2">Retrain Model</td></tr>
<tr><td>Use Case Id</td><td colspan="2">UC:03</td></tr>
<tr><td>Description</td><td colspan="2">Performs model retraining with the new data from the database, to find the new best set of hyperparameters.</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Domain Specific User</td></tr>
<tr><td>Pre-Conditions</td><td colspan="2">The actor should be a Domain Specific User and have an account created.</td></tr>
<tr><td>Extended use cases</td><td colspan="2">None</td></tr>
<tr><td>Included use cases</td><td colspan="2">UC05, UC06, UC07</td></tr>
<tr><td>Trigger</td><td colspan="2">The Domain Specific User clicks on the “Perform model retraining” button</td></tr>
<tr><td rowspan="2">Main flow</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Domain Specific logs into their account
2. Clicks on “Perform model retraining” from the GUI</td><td>3. The system pulls all the data with respect to the user id from the database.
4. Combines data of the common domains (only if user consent is given to use their data)
5. Finds new set of hyperparameters for the model with respect to new data.
6. Trains the model using the new hyperparameters.
7. Saves the model with the user Id
8. Updates the status in the database if succeed/fails</td></tr>
<tr><td>Alternative flows</td><td colspan="2">None</td></tr>
<tr><td>Expectational flows</td><td colspan="2">Displays an error message if the network request fails (server is down, or internet issues from client).</td></tr>
<tr><td>Post Conditions</td><td colspan="2">Success end condition: The user will be able to see the recent status of the model if the retraining is successful or failed</td></tr>
</table>

2.9. Requirements

2.9.1 Functional Requirements

Based on the significance of the system demands, the 'MoSCoW' approach was utilized to identify their priority levels. The details related to the priority levels are detailed at **APPENDIX A.5**.

The Usecase description along its mapping id is also listed at **APPENDIX A.6**

Table 10: Functional requirements (*Self-Composed*)

FR ID	Requirement	Priority Level	Use Case
FR1	Both general and domain specific users must be able to enter a review text from the GUI considering as the starting point of the summary generation.	M	UC01
FR2	Only Domain Specific Users should be able to sign up and create an account after entering the necessary details required	S	UC02
FR3	The system could allow the ability to update the account details of the domain user after creating the account	C	UC02
FR4	The system must undergo model retraining with the new data stored in the database for the specific domain user, when its triggered from the GUI with the user's consent.	M	UC03
FR5	The system could be able to perform model retraining automatically during off peak hours every day.	C	UC03
FR6	The system must be able to find the new set of best hyperparameters with the usage of the new data.	M	UC04
FR7	The system must be able to able to retrain the model with the new best hyperparameters and create the model	M	UC05
FR8	The system must be able to pull the new data from the database to recreate the new dataset for retraining.	M	UC06

FR9	The system should be able to combine all the data from a common group of domains when creating the dataset only given that the consent is approved to use their data	C	UC06
FR10	The system must be able to process the review text and display the summary output on the GUI	M	UC07
FR11	The system must be able to use the latest trained model to generate the summary for the review text	M	UC08
FR12	The system could generate the sentiment and the score for the summary generated	C	UC08
FR14	The system must store the entered user review and generated summary to be stored in the database for retraining purposes.	M	UC09
FR15	The system should encrypt the data when saving into the database (both the review and summary)	S	UC09
FR14	The system could allow the domain users to delete the reviews from the database.	C	UC10

2.9.2 Non-Functional Requirements

The non-functional requirements are prioritized into two level of which are “Important” and “Desirable”

Table 11: Non-functional requirements (*Self-Composed*)

NFR ID	Requirement	Specification	Priority Level
NFR1	The system needs to be simple enough for non-technical individuals to utilize without much effort.	Usability	Important
NFR2	Meaningful error messages should be displayed if anything goes wrong	Usability	Desirable
NFR3	Summary generation should be done within 3000ms	Performance	Important
NFR4	Following coding standards and best practices	Maintainability	Important

NFR5	Any domain users are able to use the application and model performance will adapt with respect to the domain	Generalization	Important
NFR6	The system should protect against data corruption by attackers, and testing can ensure this.	Security	Desirable
NFR7	The prototype can be used by several domains and multiple businesses under a single domain, then the system may have to support many concurrent user-requests.	Scalability	Desirable

2.10. Chapter Summary

To illustrate how the system interacts with society and the system stakeholders in this chapter, a Rich Picture Diagram was developed. The Saunder's Onion model, which took the influence flow from each stakeholder into account, was used to depict the stakeholders. Requirement gathering techniques were utilized to obtain all the essential data and the opinions of possible system stakeholders. Finally, the use cases, functional requirements, and non-functional requirements for the system were specified using the knowledge gathered from the requirement elicitation methodologies.

CHAPTER 03. DESIGN

3.1. Chapter Overview

This chapter discusses the design decisions made to establish an appropriate architecture for implementation based on the requirements obtained. High-level design, low-level design, design diagrams, and UI wireframes have been used to describe how the design goals are meant to be achieved while demonstrating the reasoning for selected design decisions.

3.2. Design Goals

The design goals of the proposed system along with the reasoning is available at **APPENDIX B.1**

3.3. High Level Design

3.3.1. Architecture Diagram

The system has a three-tier architecture that separates the data, logic, and presentation levels. The research contribution of the system includes generalization and domain-specific adaptive hyperparameter tuning and data preprocessing.

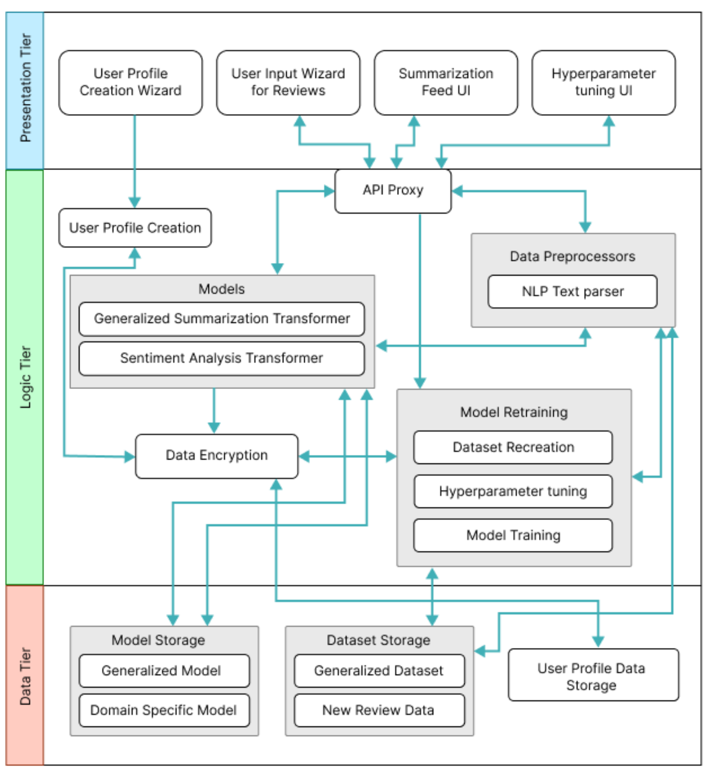


Figure 5: Three-tiered architecture (*Self-Composed*)

3.3.2. Discussion of Tiers of the Architecture

Data Tier

1. Model Storage - The text summarization models which will be used for both generalized text summarization and domain specific text summarization will be stored here.
    a) Generalized Model – The model which will be used by general users to generated review summarized, this model will be hyperparameter tuned for genialized purpose.
    b) Domain Specific Model – The model will be used by domain specific users for review summarization, this model will be replaced whenever the model retraining is triggered from the domain user.
2. Dataset Storage – The data which is required for model training will be available.
    a) Generalized Dataset – The data which is used for creating the generalized model will be stored for retraining when it comes to domain specific model retraining.
    b) New Review Data – The data stored here are from the domain users when they use the application, the data will be storage and used for retraining along with the generalized dataset.
3. User Profile Data Storage – The metadata data related to the domain specific user when creating the user profile will be stored, for updating and profile deletion.

Logic Tier

1. User Profile Creation – Allowing to create unique user profiles for each domain user, main purpose comes when working with model retraining to figure out the data to be used.
2. API Proxy – Interface which allows the frontend to communicate with the backend services via HTTP calls/ request.
3. Data Preprocessors – The text data that will be used as input for the text summarizer must be cleaned using the preprocessing code.
    a) NLP Text parser – Responsible for cleaning the input text review when received from the API endpoint.
4. Models – The model which will be responsible in generating the summary from the input review and find the sentiment of the summary generated.

    a) Generalized Summarization Transformer – This is the summarization model which will be used, an adaptive model depending on the domain and type of user interacting with the model with optimized hyperparameters.
    b) Sentiment Analysis Transformer – This model will be used to classify the generated summary into positive or negative sentiment.
5. Data Encryption – Data encryption is in charge of data protection/safety, keeping domain data extremely secure and leaving it useless even if it is stolen.
6. Model Retraining – Responsible for retraining the model with new data and finding new set of hyperparameters.
    a) Dataset Recreation – Responsible for recreating the dataset with new data which has been given as input from the domain users
    b) Hyperparameter tuning – Responsible for finding the new best set of hyperparameters using the new data.
    c) Model Training – Responsible for training the new model with the new set of hyperparameters found.

Presentation Tier

1. User Profile Creation Wizard – The UI that presents the user to create a new profile if they are planning to use the software for their domain business, or a general user to skip the sign up if only a generalized summary is required.
2. User Input Wizard for Reviews – The UI that will request the user to input the review which needs to be summarized.
3. Summarization Feed UI – The UI that displayed the summarized text for the input review.
4. Hyperparameter tuning UI – The UI that triggers model retraining when the domain user performs an action on it.

3.4. System Design

3.4.1. Choice of Design Paradigm

The main reason behind the author going ahead with **SSADM (Structured Systems Analysis and Design Method)** over **OOAD (Object-Oriented Analysis and Design)** to build the protype was due to the ease of ability to extend the system features when it comes to future developments of the system. Given below are the other factors as to why the choice of SSADM was considered:

- Object Oriented approaches will not make a greater benefit since the main core project research lies towards Data Science.
- Ability to demonstrate the MVP (Minimum Viable Product) prototype implementation for the research application is more convenient.
- More time efficient when concerned with the time constraint of having to complete the documentation research along with the project implementation.

3.5. Design diagrams

3.5.1. Data Flow diagrams

In order to show the relationships between components and provide a clearer understanding of how data flows across the whole system, the context diagram's components have been extensively broken down in the diagram below, which was detailed in the SRS previously.

3.5.1.1. Level 01 Data Flow diagram

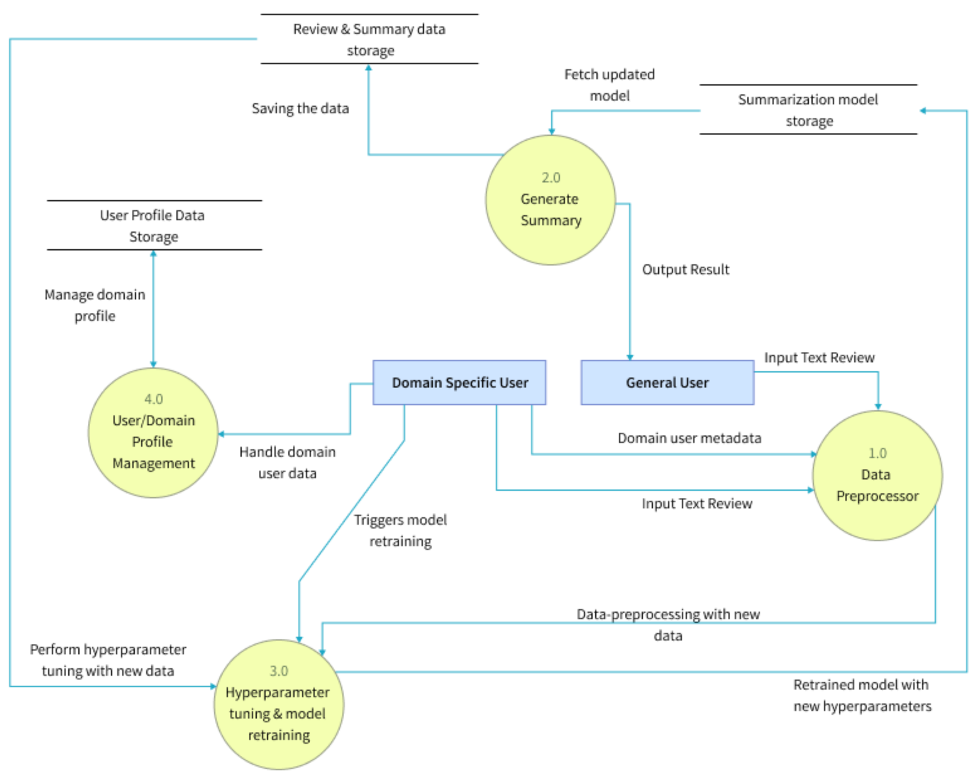


Figure 6: Data flow diagram - level 01 (*Self-Composed*)

3.5.1.2. Level 02 Data Flow diagram

The level 02 data flow diagram given below is a further breakdown of the core hyperparameter tuning and model retraining proposed in the level 01 data flow diagram.

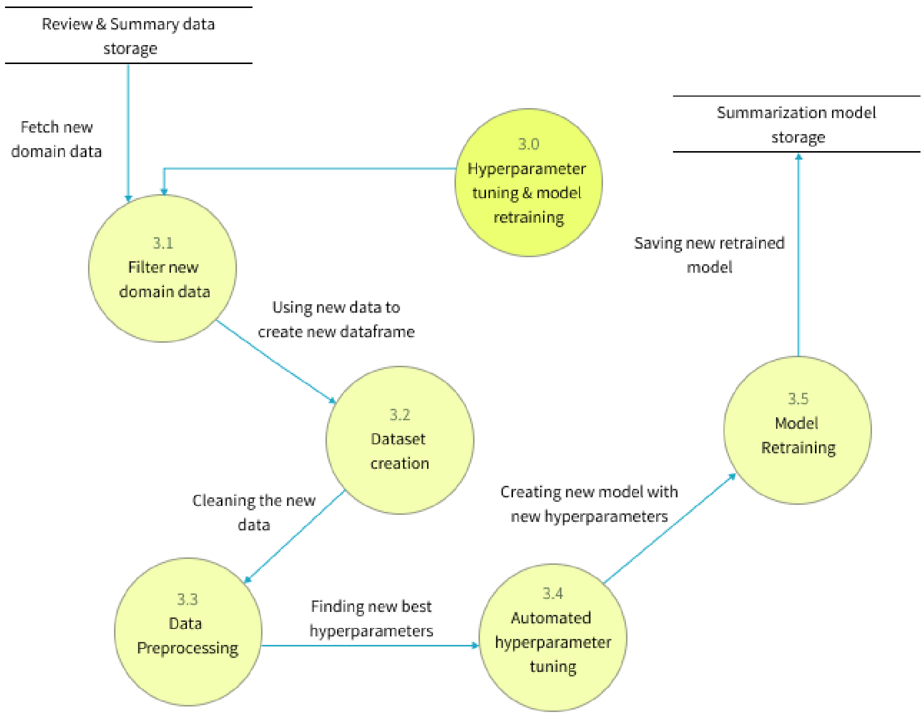


Figure 7: Data flow diagram - level 02 (*Self-Composed*)

3.5.2. System Process Activity Diagram

The flowchart given below represents the algorithm's flow and the decision structures which explains the flow of the system which is initially expected requirement.

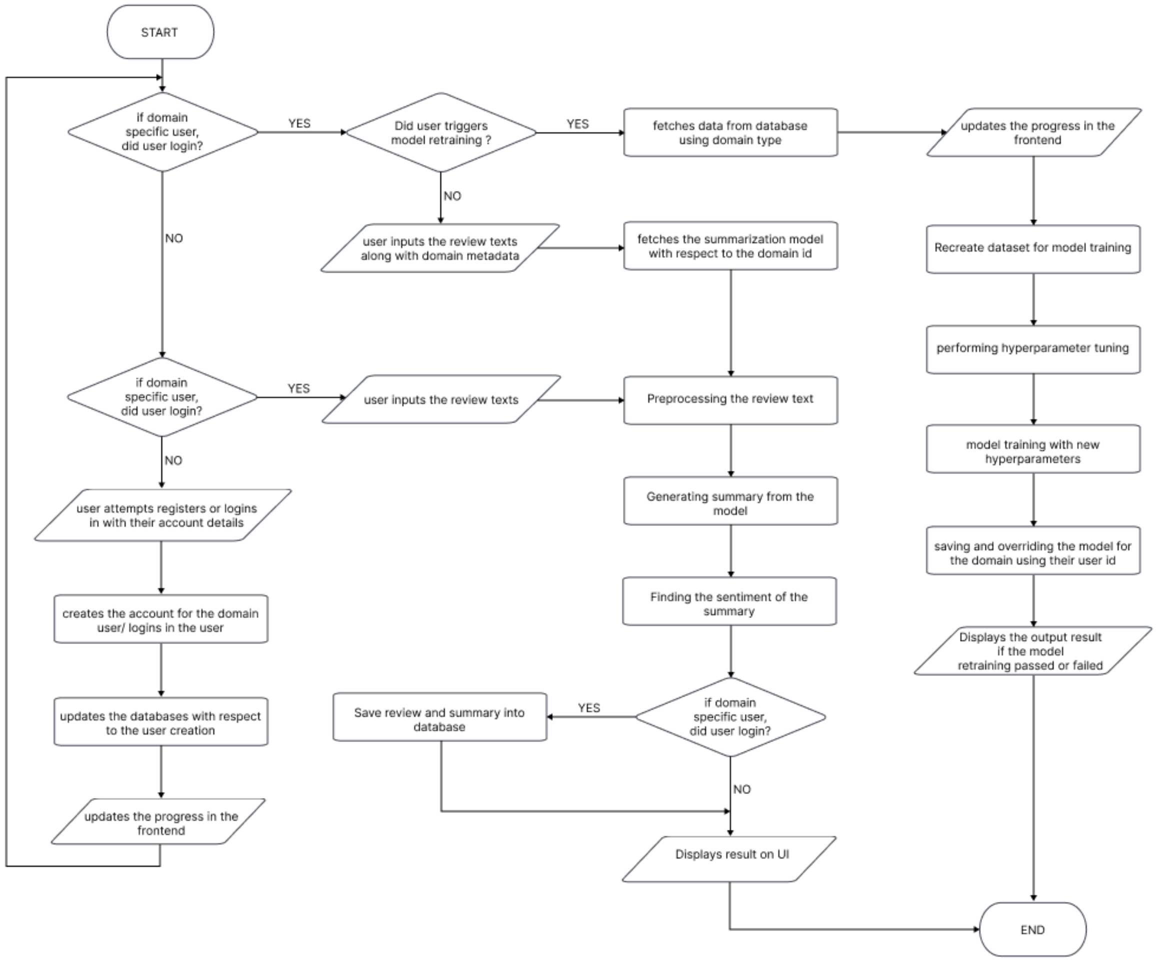


Figure 8: System process flow chart (*Self-Composed*)

3.5.4. UI Design

Given the specifications acquired from the target audience, the author chose a web application for the simulation of the proof-of-concept application. A wireframe design was created to depict the key user interface aspects in the system and is available in **APPENDIX B.2**

3.6. Chapter Summary

This chapter provides an in-depth examination of the project's design, including its architectural features and explains the core flow via data flow diagrams. The chapter concludes with a preview of the user interface wireframes that will be utilized to facilitate interaction between the end-user and the system.

CHAPTER 04. INITIAL IMPLEMENTATION

4.1. Chapter Overview

This chapter will provide a thorough overview of the technologies, supporting tools, and languages utilized for the project development, as well as the fundamental implementation of the research prototype.

4.2. Technology selection

4,2.1. Technology stack

The technologies utilized to implement the prototype at each tier are given below.

Figure 9: Technology stack (*Self-Composed*)

In preference to macOS and Linux, **Windows** will be the operating system used for project development and documentation. This is due to a wider variety of software available, which ensures that it has more industry-standard tools than Linux and macOS, along with better compatibility and familiarity, which make things simpler to use and manage.

4.2.2. Data selection

The project relies heavily on data science, so it's important to use trustworthy data sources to train the model for accurate results in text summarization. A generalized dataset for text summarization was needed to establish the base model for the project's goal of developing an adaptive generalized text summarization model. TensorFlow's datasets provided several options for this dataset and is considered a reputable source of data. The table below shows the datasets used by previous researchers, which were considered for the prototype development.

Table 8: Dataset sources (*Self-Composed*)

Dataset	**Source**
CNN Dailymail	TensorFlow Datasets
Gigaword	TensorFlow Datasets
Xsum	TensorFlow Datasets

All three datasets (CNN Dailymail, Gigaword, and Xsum) were used during the training process with various transformer architectures to evaluate which dataset produced the best results. After evaluation, Xsum had the best performance, so it was chosen as the final dataset for the project.

4.2.3. Programming Language Selection

Python was chosen as the programming language for implementing Machine Learning models and Backend APIs due to its simplicity, versatility, and wide range of use cases, including data science and machine learning. The availability of various libraries and active community support makes it a powerful tool for these tasks.

TypeScript (it’s a superset of JavsScript) was chosen for the frontend development in order to display dynamic content and create a highly interactive and engaging user experience.

4.2.4. Development Framework Selection

The author has chosen several development frameworks for the project covering all areas, the table given below describes the purpose of choosing each framework and whats it used for in the project.

Table 9: Development framework utilized (*Self-Composed*)

Framework	Reason for choosing
React	ReactJS offers reusable components for efficient application development and has strong community support, making it a valuable solution for developers.
Ant Design	Ant Design is a React UI framework with a large selection of pre-built components, customizable styles, and tree-shaking compatibility that reduces build time, making it an effective frontend development solution.
Flask	Flask is a lightweight and flexible Python web framework that's easy to learn and useful for creating backend APIs. It offers straightforward routing and request processing management, a built-in development server, and various extensions to enhance API capabilities.
Optuna	Optuna is a Python framework for hyperparameter optimization that's easy to use, efficient, and helps improve machine learning model performance. It has built-in parallelization, automated early halting, and distributed parallel optimization, and supports popular machine learning libraries.
PyTorch	PyTorch is a popular Python machine learning framework that's easy to use, has powerful features, and strong community support. It's built on Torch library and utilizes GPU capability, and is widely used in academia and business for developing machine learning models.

The data science core uses transformer models from Hugging Face, which are fine-tuned with the datasets used in the project. The goal of retraining the model is to test various hyperparameter changes.

4.2.5. Libraries Utilized

Table 10: Libraries used with reasonings (*Self-Composed*)

Library	Reasoning for selection
Firebase	Used for providing backend services for mobile and web application development.
Axios	Used for handling HTTP requests in JavaScript.

Redux	Used to control the state of JavaScript applications in a predictable manner by the use of actions, reducers, and a central store.
Hugging face Transformers	The Hugging Face transformers library provides pre-trained transformer models and tools for fine-tuning them on specific tasks in natural language processing.
NLTK	NLTK is a library for natural language processing that offers tools for various tasks and resources for training language models.
Rouge	Library for evaluating text summary quality. Used to compare machine-generated or peer summaries with one or more reference summaries.
Pandas	Pandas is a library for data manipulation and analysis, commonly used in data science for handling numerical tables and time series data
NumPy	NumPy is a Python library for scientific computing that supports large arrays and matrices of numerical data, along with a variety of mathematical functions for working with them.
Matplotlib & Seaborn	Used for creating static, animated, and interactive visualizations in Python
Gramformer	Hugging Face is an API for developers to fine-tune GPT-3 models and generate text for tasks like completion, generation, and classification.
Flask	Used for creating web APIs using Python to communicate with the transformer model and handling HTTP requests.

4.2.6. IDE's Utilized

Table 11: IDE's used along with justifications (*Self-Composed*)

IDE	**Justification for selection**
VSCode	It's highly adaptable and useful, with top-notch performance. It includes various capabilities like debugging, Git integration, syntax highlighting, and customizable extensions.
Google Colab	Due to its connection with Google Drive and availability of free GPUs, it's helpful for developing machine learning models via a cloud environment.
Jupyter Notebook	Due to their interactive and readable format, making it ideal for local experimentation, documentation and collaboration.

4.2.7. Summary of Technology Selection

Table 12: Summary of Technology selection (*Self-Composed*)

Component	Tools
Programming Languages	TypeScript, Python
Development Framework	Flask, PyTorch, Optuna
UI Framework	Ant Design, React
Libraries	NLTK, Rouge, React, Pandas, Gramformer, Matplotlib & Seaborn, Axios, Transformers (from hugging face)
IDE – Research	Google Colab, Jupyter Notebook
IDE – Product	VSCode
Version Control	Git, GitHub
Data storage	Firebase

4.3. Implementation of Core Functionalities

The project's main tasks involve experimenting with transformer architectures to find the best one, preprocessing data, automating hyperparameter search, retraining the model with new data and hyperparameters, and enabling it to summarize reviews from both domain and general users.

4.3.1. Automated Hyperparameter Search & Model Training

The author researched ways to automate hyperparameter searching as manual tuning is a waste of time. Despite various frameworks being available, Optuna was chosen for its flexibility and ease of use.

The code snippet given below shows how the Optuna framework is integrated with the model training code to perform automated hyperparameter search. The main performance contributing parameters are considered for the hyperparameter searching this includes learning rate, weight decay, num of training epochs, warmup ratio, batch size.

```python
print_custom('Performing hyperparameter training....')
def objective(trial: optuna.Trial):
    # Specify the training arguments and hyperparameter tune every arguments which are possible to tune
    training_args = Seq2SeqTrainingArguments(
        output_dir=SAVE_DIR,
        save_strategy="epoch",
        evaluation_strategy="epoch",
        learning_rate=trial.suggest_float("learning_rate", LR_MIN, LR_CEIL, log=True),
        weight_decay=trial.suggest_float("weight_decay", WD_MIN, WD_CEIL, log=True),
        num_train_epochs=trial.suggest_int("num_train_epochs", MIN_EPOCHS, MAX_EPOCHS),
        warmup_ratio=trial.suggest_float("warmup_ratio", 0.0, 1.0),
        per_device_train_batch_size=trial.suggest_int("per_device_train_batch_size", MIN_BATCH_SIZE, MAX_BATCH_SIZE),
        per_device_eval_batch_size=trial.suggest_int("per_device_eval_batch_size", MIN_BATCH_SIZE, MAX_BATCH_SIZE),
        save_total_limit=1,
        load_best_model_at_end=True,
        greater_is_better=True,
        predict_with_generate=True,
        run_name=MODEL_NAME,
        report_to="none",
    )

    # Create the trainer
    trainer = Seq2SeqTrainer(
        model=model,
        args=training_args,
        data_collator=data_collator,
        train_dataset=tokenize_data["train"],
        eval_dataset=tokenize_data["test"],
        tokenizer=tokenizer,
    )

    # Train the model
    trainer.train()

    # Evaluate the model
    metrics = trainer.evaluate()

    torch.cuda.empty_cache()

    # Return the loss
    return metrics["eval_loss"]
```

Figure 10: Hyperparameter search using Optuna (*Self-Composed*)

```python
# Hyperparameter results
learning_rate = study.best_params['learning_rate']
weight_decay = study.best_params['weight_decay']
num_train_epochs = study.best_params['num_train_epochs']
warmup_ratio = study.best_params['warmup_ratio']
per_device_train_batch_size = study.best_params['per_device_train_batch_size']
per_device_eval_batch_size = study.best_params['per_device_eval_batch_size']
```

```python
args = transformers.Seq2SeqTrainingArguments(
    'generalization-summary',
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    warmup_ratio=warmup_ratio,
    num_train_epochs=num_train_epochs,
    per_device_train_batch_size=per_device_train_batch_size,
    per_device_eval_batch_size= per_device_eval_batch_size,
    save_total_limit=2,
    eval_accumulation_steps=1,
    predict_with_generate=True,
    evaluation_strategy='epoch',
    gradient_accumulation_steps=2,
    fp16=True
)
```

Figure 11: Hyperparameter results and training arguments (*Self-Composed*)

The above snippet demonstrates how to result of the hyperparameter search is used within the training arguments for model training.

4.3.2. Model Usage General & Domain Specific Users.

```
@app.route('/text-summarizer/general', methods=['POST'])
def getGeneralizedSummary():
    try:
        data = request.get_json()
        review = data['review']
        inputs = generalized_tokenizer.encode(review, return_tensors='pt', max_length=MAX_INPUT, truncation=True)
        outputs = generalized_model.generate(inputs, max_length=150, min_length=40, length_penalty=2.0, num_beams=4,
        early_stopping=True)
        summary = generalized_tokenizer.decode(outputs[0], skip_special_tokens=True)

        sentimentAnalysisOutput = query({ "inputs": summary })
        sentiment, score = getOverallSentimentWithScore(sentimentAnalysisOutput)
        return {'summary': summary, 'sentment': {
            'sentiment': sentiment,
            'score': score
        } }, 200
    except Exception as e:
        return {'message': str(e)}, 500
```

Figure 12: General user review text summarization (*Self-Composed*)

The above code snippet is an API endpoint which handles text (review) summarization for the general users where they don’t need to create and account or have specialized model assigned to them instead the general model is utilized.

```
@app.route('/text-summarizer/domain', methods=['POST'])
def getDomainSpecificSummary():
    try:
        data = request.get_json()
        review = data['review']
        userId = data['userId']

        folder_path = 'model/' + userId
        model_path =  folder_path + '/' + MODEL_NAME
        tokenizer_path = folder_path + '/' + TOKENIZER_NAME

        if not os.path.exists(folder_path):
            return {'message': "Model not found"}, 404

        model = transformers.AutoModelForSeq2SeqLM.from_pretrained(model_path)
        tokenizer = transformers.AutoTokenizer.from_pretrained(tokenizer_path)

        inputs = tokenizer.encode(review, return_tensors='pt', max_length=MAX_INPUT, truncation=True)
        outputs = model.generate(inputs, max_length=150, min_length=40, length_penalty=2.0, num_beams=4, early_stopping=True)
        summary = tokenizer.decode(outputs[0], skip_special_tokens=True)

        sentimentAnalysisOutput = query({ "inputs": summary })
        sentiment, score = getOverallSentimentWithScore(sentimentAnalysisOutput)

        db.collection('domainUsers').document(userId).collection('reviewData').add({
            'review': review,
            'summary': summary,
            'sentiment': sentiment,
            'score': score
        })

        return {'summary': summary, 'sentment': {
            'sentiment': sentiment,
            'score': score
        } }, 200
    except Exception as e:
        return {'message': str(e)}, 500
```

Figure 13: Domain Specific text review summarization (*Self-Composed*)

The code snippet given above describes how the newly assigned domain specific model is used to generate the summary and store the input and outputs into the database along with returning the sentiment of the summary with the sentiment score. The sentiment analysis is done using a pretrained transformer directly from hugging face API.

4.3.3. Model Retraining.

```
@app.route('/domain-profile-retraining', methods=['POST'])
def retrainDomainSpecifcModel():
    try:
        data = request.get_json()
        newReviewSummaryData = []

        userId = data['userId'] # The user id is only needed to save the model in the respective folder
        domainType = data['domainType'] # Using the domainType, we can get all the data from other users which have been given
        access for retraining
        isUseOtherData = data['isUseOtherData'] # we can have a radio button in the frontend to select if the user wants to
        retrain only with their data or with the other users data as well

        # Steps to be considered for retraining the model and dataset recreation
        # 1. By checking the isAccessible flag, we can decide whether to use the data for model retraining, then we get all the
        data from the database which isAccessible = true for the given domainType
        print('Fetching data from the database...')
        if isUseOtherData == True:
            users = db.collection('domainUsers').where('domainType', '==', domainType).where('isAccessible', '==', True).get()
            for user in users:
                reviewData = db.collection('domainUsers').document(user.id).collection('reviewData').get()
                for review in reviewData:
                    newReviewSummaryData.append(review.to_dict())

        else:
            user = db.collection('domainUsers').document(userId).get()
            if user.exists:
                reviewData = db.collection('domainUsers').document(userId).collection('reviewData').get()
                for review in reviewData:
                    newReviewSummaryData.append(review.to_dict())
            else:
                return {'message': "User not found"}, 404
        print('Successfully fetched data from the database')
```

Figure 14: Fetching related data for model retraining (*Self-Composed*)

The code snippet above describes the necessary data fetched from the database to create the new dataset for model retraining, once the new dataset is created it is passed through a function to perform hyperparameter tuning and then retrain the model. Once completed retraining, the old model will be replaced with the new model in the folder path location.

4.3.4. Data Preprocessing

The raw dataset was contaminated with a lot of noise, numerous data preprocessing steps were required to clean the data before model training. The related preprocessing scripts can be found at **APPENDIX C.1**.

4.3. Chapter Summary

The chapter discusses the tools, technology, and languages utilized to create the research prototype. The fundamental functionality is covered, along with insights and samples of code for the implemented algorithms, moreover the testing and evaluation related code for the models is discussed.

CHAPTER 05. CONCLUSION

5.1. Chapter Overview

This chapter covers the preliminary conclusion of the research project, including the core functionality of its implementation for the MVP. Any deviations taken with in the project scope will be discussed and an initial evaluation test result will be attached. Any additional improvements planned for the project will be discussed. A demo of the project and the code reference for the project will also be included.

5.2. Deviations

The initial goal of the author was to create an optimized solution for movie review summarization using transformers, but after discussions made with supervisors the research gap of the author for the technical contribution being only hyperparameter tuning of transformer felt small in magnitude, therefore the idea of creating a ***performance adaptive generalized solution*** was considered to continue the research implementation on.

The only project schedule deviation is that the testing scripting like unit, integration and performance testing has not yet been started but will be able to cover up within the timeframe listed. The initial Gantt chart plan can be found at **APPENDIX D.1** and the current progressing one at **APPENDIX D.2**

5.3. Initial Test Results

The Xsum dataset was trained with three top tier transformer architecture which are BART, T5 and PEGASUS. The evaluations for all these architectures where conducted is given below.

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum	Gen Len
1	No log	0.377033	36.339200	16.680800	30.463900	30.470900	18.628000
2	0.973300	0.218723	48.158500	30.368000	43.379800	43.318400	19.164000
3	0.973300	0.124898	58.955100	46.272900	55.868600	55.895700	19.468000
4	0.248100	0.065275	71.579900	64.391400	69.579000	69.735200	19.748000
5	0.248100	0.033817	77.903500	74.465900	77.300500	77.366700	19.836000
6	0.103700	0.017642	80.060400	77.934100	79.934100	79.963800	19.836000
7	0.103700	0.012296	80.552900	78.879600	80.539700	80.581400	19.824000
8	0.048200	0.009251	80.776300	79.422400	80.802700	80.832400	19.828000

Figure 15: Evaluation result for bart-base model (*Self-Composed*)

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum	Gen Len
1	No log	0.022812	75.197600	70.900700	74.367300	74.487300	18.844000
2	0.063800	0.025599	75.297700	71.001800	74.531600	74.534500	18.892000
3	0.063800	0.014508	78.321900	75.546000	77.829600	77.873700	18.836000
4	0.067600	0.008198	78.461300	76.093400	78.239100	78.332400	18.824000
5	0.067600	0.005407	79.957700	78.359400	79.832400	79.902800	18.872000
6	0.032300	0.003742	80.078600	78.585000	79.945600	79.952800	18.864000
7	0.032300	0.003123	80.656400	79.608700	80.672000	80.676800	18.848000
8	0.017100	0.001702	80.661600	79.589900	80.614900	80.655100	18.864000
9	0.017100	0.001026	80.736900	79.734500	80.757600	80.767000	18.864000
10	0.010600	0.000348	80.753600	79.756300	80.771100	80.792700	18.864000
11	0.010600	0.000228	80.753600	79.756300	80.771100	80.792700	18.864000
12	0.006400	0.000341	80.753600	79.756300	80.771100	80.792700	18.864000
13	0.006400	0.000127	80.753600	79.756300	80.771100	80.792700	18.864000
14	0.004900	0.000115	80.753600	79.756300	80.771100	80.792700	18.864000
15	0.004900	0.000107	80.753600	79.756300	80.771100	80.792700	18.864000

Figure 16: Evaluation result for t5-base model (*Self-Composed*)

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum	Gen Len
1	No log	nan	10.069400	1.213700	8.335600	8.334600	32.000000
2	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
3	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
4	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
5	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
6	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
7	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
8	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
9	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
10	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
11	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000
12	0.000000	nan	10.069400	1.213700	8.335600	8.334600	32.000000

Figure 17: Evaluation result for pegasus-base model (*Self-Composed*)

The bart-base model outperforms the rest of the other two (t5 and pegasus) despite the fact the results of t5 was very closer to bart, it still gets ranked as the second due to evaluation result difference and the difference in the epoch taken to reach the result. However, the result of Pegasus was significantly low bringing it to the last. The validation accuracy and validation loss graphs can be found at **APPENDIX D.3**.

5.4. Required Improvements

A couple of improvements in-order to bring the project to a success are as follows:

- Integrating the developed APIs and models to the frontend UI: the UI has already been developed by the author at this point in time.
- Testing all areas of the project applications, this will include unit, performance and integration testing.
- Encrypting the data which is storing in the database for security purposes.
- Connecting a push notification service to the UI, to indicate when the retraining process is completed to the end user.

5.5. Demo of the Prototype

A functional prototype demo along with project presentation details was recorded and uploaded to YouTube as unlisted. The video for it can be found at https://youtu.be/NdXRkGFG9b8 and the presentation slides used in the demo can be found at https://tinyurl.com/3z3dj9nn

5.6. Code Reference

All related code and documentation material are made available in GitHub by the author at https://github.com/nazhimkalam/gensum/tree/main/Code

5.7. Chapter Summary

This chapter discuses about the core functionality completion of the project for the MVP. All deviations of the project and evaluation results have been discussed. The improvement of the project is also discussed by which the author will be working on. A demo about the project along with the code references is also listed.

REFERENCES

Abolghasemi, M., Dadkhah, C. and Tohidi, N. (2022). HTS-DL: Hybrid Text Summarization System using Deep Learning. *2022 27th International Computer Conference, Computer Society of Iran (CSICC)*. 23 February 2022. Tehran, Iran, Islamic Republic of: IEEE, 1–5. Available from https://doi.org/10.1109/CSICC55295.2022.9780395 [Accessed 26 October 2022].

Alsaqer, A.F. and Sasi, S. (2017). Movie review summarization and sentiment analysis using rapidminer. *2017 International Conference on Networks & Advances in Computational Technologies (NetACT)*. July 2017. Thiruvanthapuram, India: IEEE, 329–335. Available from https://doi.org/10.1109/NETACT.2017.8076790 [Accessed 10 October 2022].

Barna, N.H. and Heickal, H. (2022). An Automatic Abstractive Text Summarization System. *Dhaka University Journal of Applied Science and Engineering*, 6 (2), 39–48. Available from https://doi.org/10.3329/dujase.v6i2.59217.

Boorugu, R., Ramesh, G. and Madhavi, K. (2019). Summarizing Product Reviews Using Nlp Based Text Summarization. *International Journal of Scientific & Technology Research Volume*, 8 (10), 1127–1133.

Brasoveanu, A.M.P. and Andonie, R. (2020). Visualizing Transformers for NLP: A Brief Survey. *2020 24th International Conference Information Visualisation (IV)*. September 2020. Melbourne, Australia: IEEE, 270–279. Available from https://doi.org/10.1109/IV51561.2020.00051 [Accessed 2 November 2022].

Etemad, A.G., Abidi, A.I. and Chhabra, M. (2021). A Review on Abstractive Text Summarization Using Deep Learning. *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. 3 September 2021. Noida, India: IEEE, 1–6. Available from https://doi.org/10.1109/ICRITO51393.2021.9596500 [Accessed 10 October 2022].

Gupta, A. et al. (2021). Automated News Summarization Using Transformers. *ArXiv*, abs/2108.01064.

Gupta, V. and Lehal, G.S. (2010). A Survey of Text Summarization Extractive Techniques. *Journal of Emerging Technologies in Web Intelligence*, 2 (3), 258–268. Available from https://doi.org/10.4304/jetwi.2.3.258-268.

Joy, J. and Selvan, M.P. (2022). A comprehensive study on the performance of different Multi-class Classification Algorithms and Hyperparameter Tuning Techniques using Optuna. *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*. 23 June 2022. Kochi, India: IEEE, 1–5. Available from https://doi.org/10.1109/IC3SIS54991.2022.9885695 [Accessed 24 October 2022].

Khan, A. et al. (2020). Movie Review Summarization Using Supervised Learning and Graph-Based Ranking Algorithm. *Computational Intelligence and Neuroscience*, 2020, 7526580. Available from https://doi.org/10.1155/2020/7526580.

Kirmani, M. et al. (2019). Hybrid Text Summarization: A Survey. In: Ray, K. Sharma, T.K. Rawat, S. et al. (eds.). *Soft Computing: Theories and Applications*. Advances in Intelligent Systems and Computing. Singapore: Springer Singapore, 63–73. Available from https://doi.org/10.1007/978-981-13-0589-4_7 [Accessed 1 November 2022].

Kouris, P., Alexandridis, G. and Stafylopatis, A. (2019). Abstractive Text Summarization Based on Deep Learning and Semantic Content Generalization. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019. Florence, Italy: Association for Computational Linguistics, 5082–5092. Available from https://doi.org/10.18653/v1/P19-1501 [Accessed 24 October 2022].

Lackermair, G., Kailer, D. and Kanmaz, K. (2013). Importance of Online Product Reviews from a Consumer's Perspective. *Advances in Economics and Business*, 1 (1), 1–5. Available from https://doi.org/10.13189/aeb.2013.010101.

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. 8.

Liu, X. and Wang, C. (2021). An Empirical Study on Hyperparameter Optimization for Fine-Tuning Pre-trained Language Models. Available from http://arxiv.org/abs/2106.09204 [Accessed 24 October 2022].

Mahajan, R. et al. (2021). Text Summarization Using Deep Learning. *International Research Journal of Engineering and Technology (IRJET)*, 08 (05th May 2021), 1737–1740.

McAuley, J.J. and Leskovec, J. (2013). From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. *Proceedings of the 22nd international conference on World Wide Web - WWW '13*. 2013. Rio de Janeiro, Brazil: ACM Press, 897–908. Available from https://doi.org/10.1145/2488388.2488466 [Accessed 19 November 2022].

Mukherjee, R. et al. (2020). Read what you need: Controllable Aspect-based Opinion Summarization of Tourist Reviews. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 25 July 2020. 1825–1828. Available from https://doi.org/10.1145/3397271.3401269 [Accessed 10 October 2022].

Neyshabur, B. et al. (2017). Exploring Generalization in Deep Learning. *undefined*. Available from https://www.semanticscholar.org/reader/d53fb3feeeab07a0d70bf466dd473ec6052ecc07 [Accessed 9 November 2022].

Pai, A. (2014). Summarizer Using Abstractive and Extractive Method. *International Journal of Engineering Research*, 3 (5), 5.

Pizam, A. and Ellis, T. (1999). Customer satisfaction and its measurement in hospitality enterprises. *International Journal of Contemporary Hospitality Management*, 11 (7), 326–339. Available from https://doi.org/10.1108/09596119910293231.

Shi, T. et al. (2020). Neural Abstractive Text Summarization with Sequence-to-Sequence Models. Available from http://arxiv.org/abs/1812.02303 [Accessed 10 October 2022].

Shorten, C. and Khoshgoftaar, T.M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6 (1), 60. Available from https://doi.org/10.1186/s40537-019-0197-0.

Socher, R., Bengio, Y. and Manning, C.D. (2012). Deep Learning for NLP (without Magic). *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. July 2012. Jeju Island, Korea: Association for Computational Linguistics, 5. Available from https://aclanthology.org/P12-4005 [Accessed 2 November 2022].

Steinberger, J. and Jezek, K. (2009). Evaluation Measures for Text Summarization. *Comput. Informatics*, 28 (2), 251–275.

Wolf, T. et al. (2020). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020. Online: Association for Computational Linguistics, 38–45. Available from https://doi.org/10.18653/v1/2020.emnlp-demos.6 [Accessed 10 October 2022].

Zhang, J. et al. (2020). PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. Available from http://arxiv.org/abs/1912.08777 [Accessed 18 October 2022].

Zhou, K. et al. (2021). Domain Generalization with MixStyle. *undefined*. Available from https://www.semanticscholar.org/reader/4f6eafafc9563a5b904535078df7e74afe39ef59 [Accessed 5 December 2022].

aws.amazon.com. (2022). *Hyperparameter optimization for fine-tuning pre-trained transformer models from Hugging Face | AWS Machine Learning Blog*. [online] Available at: https://aws.amazon.com/blogs/machine-learning/hyperparameter-optimization-for-fine-tuning-pre-trained-transformer-models-from-hugging-face/ [Accessed 9 Nov. 2022].

APPENDIX A – SRS

A.1. Requirement Elicitation Methodologies

Table 13: Stakeholder groups (*Self-Composed*)

Group	Stakeholders	Reason	Instrument
G1	Domain experts (NLP Experts, AI Researchers, Data Scientists)	In order to respond to research questions and discover anything the author may have overlooked, gather any insights and information especially in the study area.	Interview
G2	Domain and General Users	Gather requirements which will help develop features expected in the application.	Survey & LR
G3	Competitors	Analyze any existing systems related to the research and understand how the project can be enhanced	Self-Evaluations & Brain Storming
G4	Developers	Cross checking if the project is feasible to be continued with.	Prototyping

A.2. Interview Participant Information

Table 14: Interview participant information (*Self-Composed*)

Participant ID	Name	Designation/Affiliations	Expertise
P1	Ms. Kanishka Silva	PhD Research Student in Computational Linguistics	NLP
P2	Mr. Nihal Kodikara	Machine Learning Expertise | Lecturer with PhD	ML and Neural Networks
P3	Ms. Rrubaa Panchendrarajan	NLP Researcher	NLP

P4	Mr. Pradeep Sanjaya	Software Architect	Algorithms
P5	Ms. Nelum Weerakoon	Software Architect & ML Researchers	ML & Algorithms
P6	Mr. Dinuka Piyadigama	VP Innovations, Software Engineer	ML & Neural Networks
P7	Ms. Krishna Kripa	Lecturer with MSc	NLP

A.3. Self-Evaluation (Competitor Analysis)

Table 15: Competitor Analysis (*Self-Composed*)

Competitor Analysis Table					
Tools / **Feature**	Summarize Bot	Resoomer	Smmry	Text Compactor	**GenSum**
Summarizing Text	✓	✓	✓	✓	✓
Domain Specific Generalization	✗	✗	✗	✗	✓
Ease of Use via GUI	✗	✓	✓	✓	✓
Summary sentiment and score	✗	✗	✗	✗	✓

A.4. Use case Descriptions

Table 16: Use case description UC:01 (*Self-Composed*)

Use Case Name	Input Review
Use Case Id	UC:01
Description	Requested the user to input a text review

Primary Actor	General User, Domain Specific User
Pre-Conditions	Domain Specific user needs to be login in before this action
Extended use cases	None
Included use cases	None
Trigger	A user selects the text input field to enter text review.
Main flow	The general user clicks on the input field to enter the review text, if it’s a domain specific user then user needs to login into the application for this action
Alternative flows	None
Expectational flows	Displays an error message if the network request fails (server is down, or internet issues from client).
Post Conditions	None

Table 17: Use case description UC:02 (*Self-Composed*)

<table>
<tr><td>Use Case Name</td><td colspan="2">Create Profile</td></tr>
<tr><td>Use Case Id</td><td colspan="2">UC:02</td></tr>
<tr><td>Description</td><td colspan="2">Domain users will be able to create a unique profile to manage their content</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Domain Specific User</td></tr>
<tr><td>Pre-Conditions</td><td colspan="2">None</td></tr>
<tr><td>Extended use cases</td><td colspan="2">None</td></tr>
<tr><td>Included use cases</td><td colspan="2">None</td></tr>
<tr><td>Trigger</td><td colspan="2">The domain user signups an account with in the system</td></tr>
<tr><td rowspan="2">Main flow</td><td>Actor</td><td>System</td></tr>
<tr><td>1. The domain user navigates to the sign-in page.
2. The domain user clicks on sign in, to register their self or login to the application</td><td>3. Create a new user in the database and notify the user.</td></tr>
</table>

Alternative flows	None
Expectational flows	Displays an error message if the network request fails (server is down, or internet issues from client).
Post Conditions	Success message displayed.

Table 18: Use case description UC:10 (*Self-Composed*)

<table>
<tr><td>Use Case Name</td><td colspan="2">Delete reviews</td></tr>
<tr><td>Use Case Id</td><td colspan="2">UC:10</td></tr>
<tr><td>Description</td><td colspan="2">Domain users will only be able to perform this action to mange their own data reviews and delete</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Domain Specific User</td></tr>
<tr><td>Pre-Conditions</td><td colspan="2">Domain user should be logged into the application</td></tr>
<tr><td>Extended use cases</td><td colspan="2">None</td></tr>
<tr><td>Included use cases</td><td colspan="2">None</td></tr>
<tr><td>Trigger</td><td colspan="2">Clicking on the delete action button on the review card list</td></tr>
<tr><td rowspan="2">Main flow</td><td>Actor</td><td>System</td></tr>
<tr><td>1. The domain user logins into the application
2. Navigates to the manage reviews area
3. Clicks on ‘Delete’ on the choice of review by the domain user</td><td>4. Searches for the review with the user id and the review id on the database.
5. Deletes the review from the database.</td></tr>
<tr><td>Alternative flows</td><td colspan="2">None</td></tr>
<tr><td>Expectational flows</td><td colspan="2">Displays an error message if the network request fails (server is down, or internet issues from client).</td></tr>
<tr><td>Post Conditions</td><td colspan="2">Success message displayed.</td></tr>
</table>

Table 19: Use case description UC:04 (*Self-Composed*)

Use Case Name	Search new hyperparameters

Use Case Id	UC:04
Description	Searching for new set of hyperparameters during model retraining process.
Primary Actor	Domain Specific User
Pre-Conditions	Domain user should have entered enough data into the system
Extended use cases	None
Included use cases	Retrain Model
Trigger	Domain user have triggered the model retraining from the UI by clicking on to the “Retrain model” button
Main flow	1. New unseen data is fetched from the database. 2. The data is used for automated hyperparameter model training. 3. New hyperparameter is used for model training
Alternative flows	None
Expectational flows	None
Post Conditions	None

Table 20: Use case description UC:05 (*Self-Composed*)

Use Case Name	Create model
Use Case Id	UC:05
Description	Using the new set of hyperparameters found the model is retrained to create a new updated version
Primary Actor	Domain Specific User
Pre-Conditions	Domain user should have entered enough data into the system
Extended use cases	None
Included use cases	Retrain Model
Trigger	Domain user have triggered the model retraining from the UI by clicking on to the “Retrain model” button

Main flow	1. Newly found hyperparameters are used to retrain the model. 2. Old model is replaced with the new model.
Alternative flows	None
Expectational flows	None
Post Conditions	None

Table 21: Use case description UC:06 (*Self-Composed*)

Use Case Name	Prepare dataset
Use Case Id	UC:06
Description	Pulling the new data from the database in order to create a new dataset for model retraining
Primary Actor	Domain Specific User
Pre-Conditions	Domain user should have entered enough data into the system
Extended use cases	None
Included use cases	Retrain Model
Trigger	Domain user have triggered the model retraining from the UI by clicking on to the “Retrain model” button
Main flow	1. Gets the parameters sent from the request body. 2. Fetches data from the database related to the parameters. 3. Creating new dataset using the data.
Alternative flows	None
Expectational flows	None
Post Conditions	None

Table 22: Use case description UC:08 (*Self-Composed*)

Use Case Name	Generate Summary
Use Case Id	UC:08
Description	Generating summary for the input review using the latest model saved.
Primary Actor	Domain Specific User, General User

Pre-Conditions	User should have entered a review text from the frontend to generate a summary for.	
Extended use cases	None	
Included use cases	View Summary	
Trigger	User clicked on “Generate summary” after using the review text as input.	
Main flow	Actor	System
	1. User should have entered a text from the frontend in the input field requested. 2. User clicks on “General summary”	3. System uses the input review to perform data preprocessing. 4. System uses the preprocessed text review to generate the summary
Alternative flows	None	
Expectational flows	Displays an error message if the network request fails (server is down, or internet issues from client).	
Post Conditions	Success message displayed.	

Table 23: Use case description UC:09 (*Self-Composed*)

Use Case Name	Store data	
Use Case Id	UC:09	
Description	Storing the review and summary data along with the sentiment.	
Primary Actor	Domain Specific User	
Pre-Conditions	Domain user should have entered input review and requested	
Extended use cases	None	
Included use cases	View summary	
Trigger	Domain user clicks on ‘Generate summary’ after adding a review text	
Main flow	Actor	System

	1. User should have entered a text from the frontend in the input field requested. 2. User clicks on "General summary"	3. The review data is used to generate the summary. 4. Using the generated summary to get the sentiment and sentiment score. 5. The result of all these will be written into the database
Alternative flows	None	
Expectational flows	None	
Post Conditions	None	

A.5. Functional Requirements

Table 24: 'MoSCoW' priority levels (*Self-Composed*)

Priority Level	**Description**
Must have (M)	The demand at this level is the fundamental functional requirement for a prototype, and it must be carried out.
Should have (S)	Although not strictly required for the anticipated prototype to function, important criteria do provide a lot of value.
Could have (C)	Optional, non-essential desirable needs are crucial to the project's scope.
Will not have (W)	Requirements that the system might not meet right now and that are not given first consideration.

A.6. Usecase mappings

Table 25: Usecase mappings (*Self-Composed*)

Use case Id	**Use case name**
UC01	Input Review
UC02	Create Profile
UC03	Retrain Model

UC04	Search New Hyperparameters
UC05	Create Model
UC06	Prepare Dataset
UC07	View Summary
UC08	Generate Summary
UC09	Store Data
UC10	Delete reviews

APPENDIX B – DESIGN

B.1. Design goals

Table 26: Design goals of the proposed system (*Self-Composed*)

Design Goal	Description
Performance	To find the new set of hyperparameters with the new data, model retraining requires a significant amount of time. As a result, the newly created dataset (with unseen data) should be accurately made, and it is best if it takes the least amount of time to query the data from various businesses within the same domain to create the dataset. Moreover, other core functionalities should be designed effectively to increase overall performance.
Correctness	The correctness & quality of the output should be of the highest possible level utilizing the optimized transformer architecture. Since several approaches are considered in order to get the optimized solution the expected output should of the best possible form.
Usability	The system's usability must be straightforward for users of all levels of knowledge because its primary function is to summarize review text for any domain, including movies and general users.
Adaptability	Adopting new features or components need to be a simple procedure. The system shouldn't be broken if a component is added or removed, and it shouldn't be affected overall.
Scalability	In a production environment, the system may need to accommodate a large number of concurrent user requests. This should be manageable by the backend. The system should be easily expandable to accommodate new data.

B.2. UI Wireframes

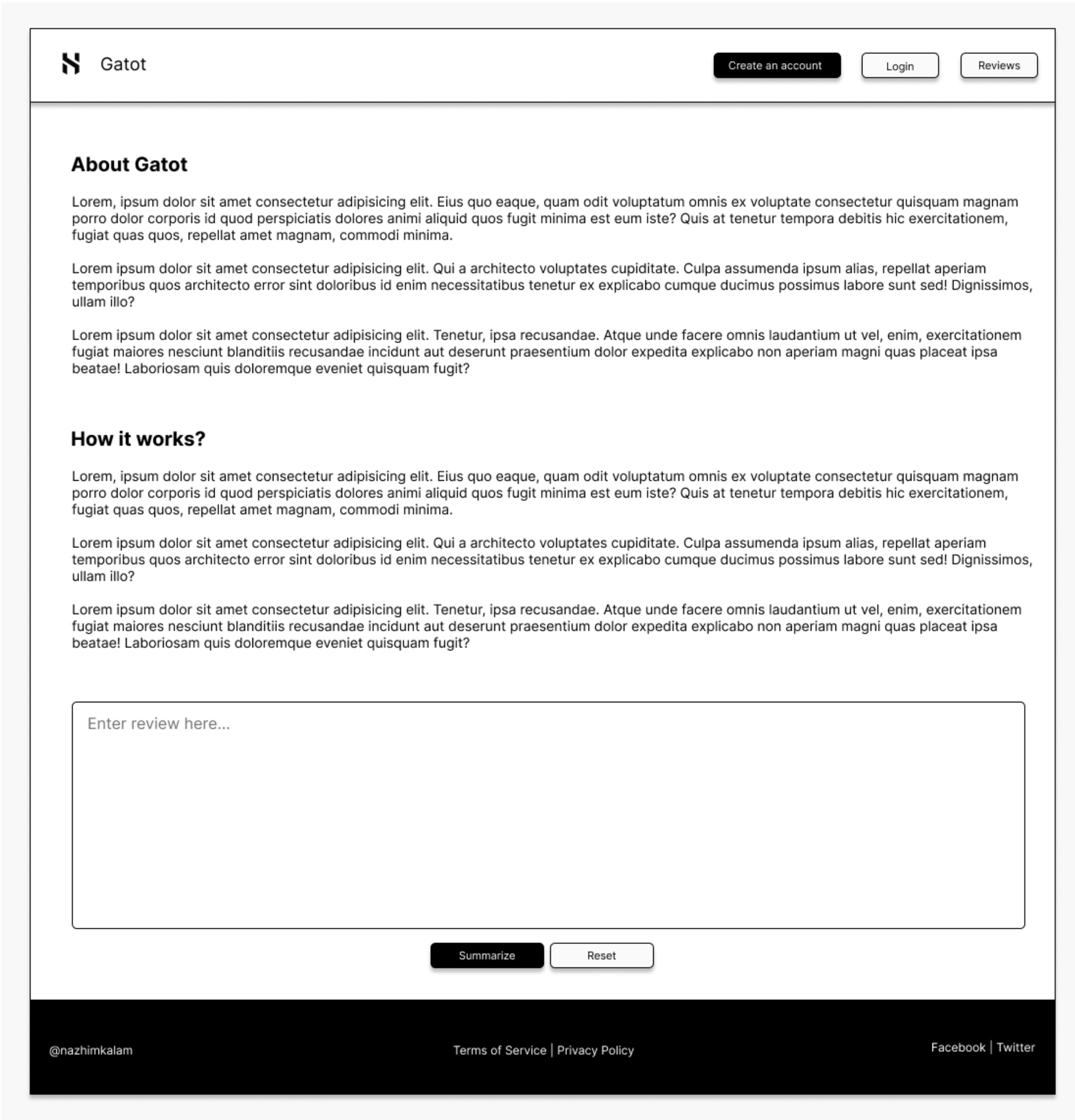


Figure 18: UI – Home page (*Self-Composed*)

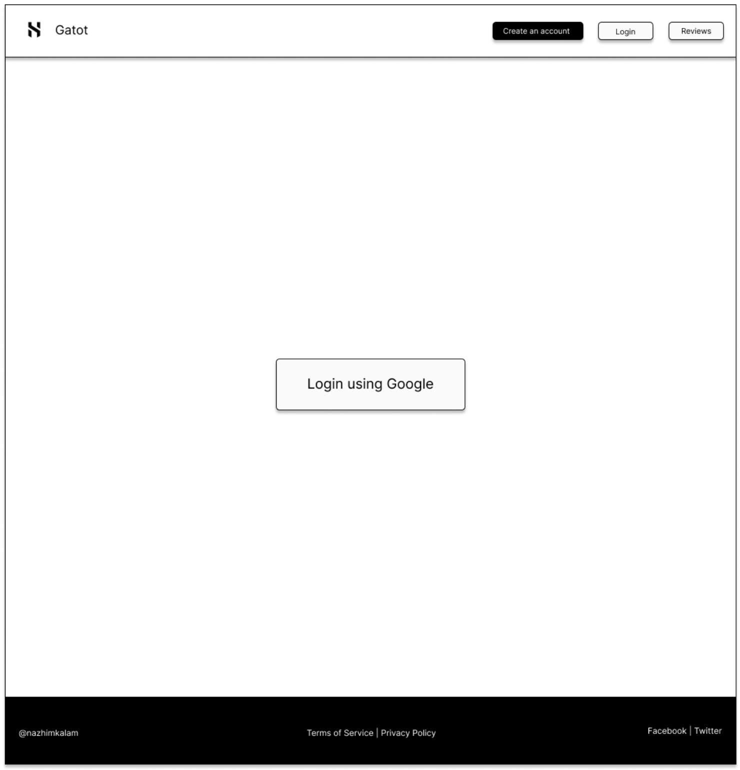


Figure 19: UI – Login page (*Self-Composed*)

Figure 20: UI – Register page (*Self-Composed*)

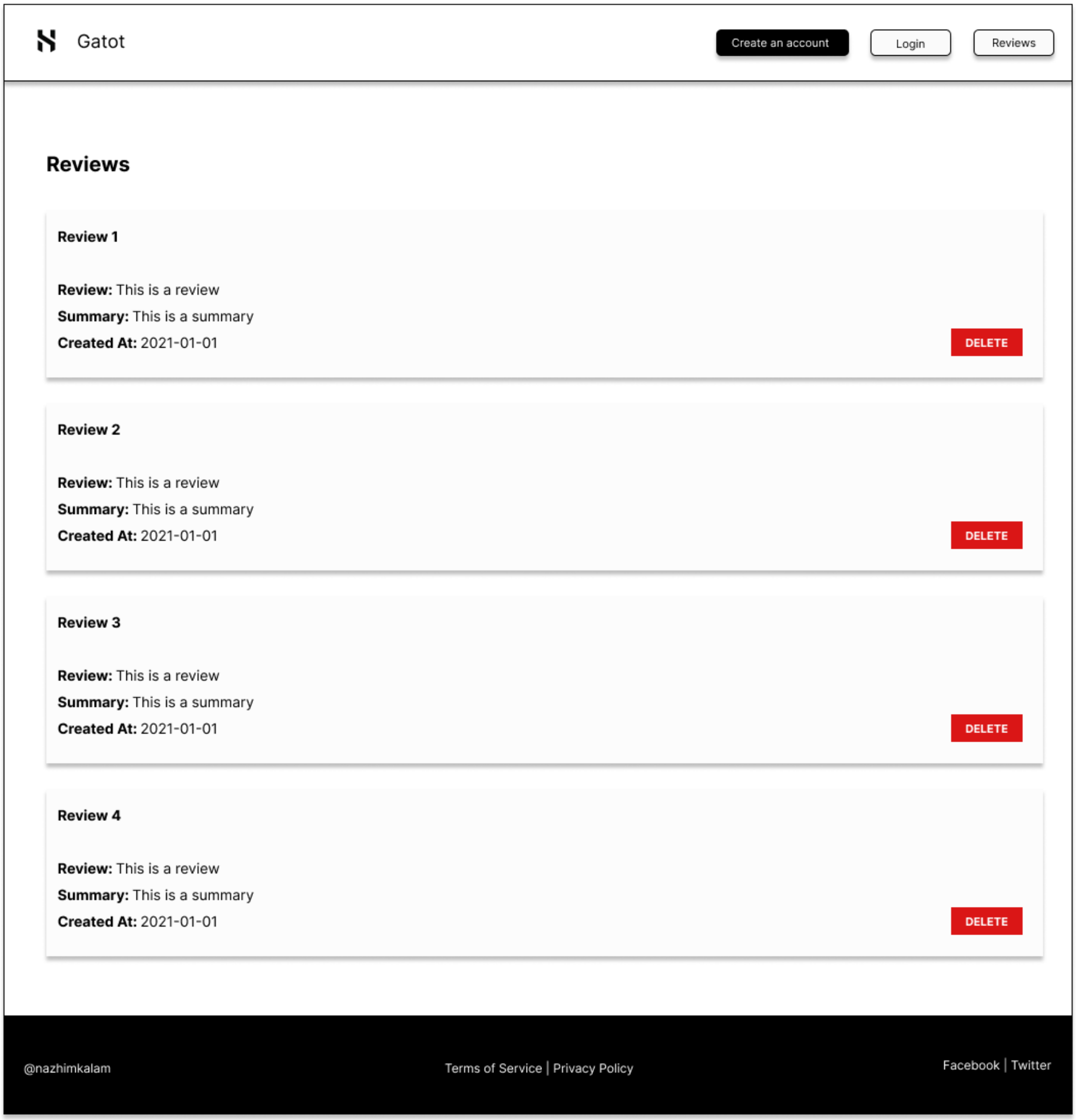


Figure 21: UI – Review history page (*Self-Composed*)

APPENDIX C – IMPLEMENTATION

C.1. Data Preprocessing

```
def md_links(text: Text) -> Text:
    markdown_link=re.compile(r'\[.*?\]\(.*?\)')
    return markdown_link.sub(r'',text)
```

```
df['text'] = df['text'].parallel_apply(lambda sentence: md_links(sentence))
df['summary'] = df['summary'].parallel_apply(lambda sentence: md_links(sentence))
```

Figure 20: Preprocessing: Remove markdown (*Self-Composed*)

```
def scrape_links(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'',text)
```

```
df['text'] = df['text'].parallel_apply(lambda sentence: scrape_links(sentence))
df['summary'] = df['summary'].parallel_apply(lambda sentence: scrape_links(sentence))
```

Figure 22: Preprocessing – Remove hyperlinks (*Self-Composed*)

```
def remove_html_tags(text: Text) -> Text:
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)
```

```
df['text'] = df['text'].parallel_apply(lambda sentence: remove_html_tags(sentence))
df['summary'] = df['summary'].parallel_apply(lambda sentence: remove_html_tags(sentence))
```

Figure 23: Preprocessing: Remove html tags (*Self-Composed*)

```
def chat_words_conversion(text: Text) -> Text:
    new_text = []
    for word in text.split():
        if word.upper() in chat_words_map_dict:
            new_text.append(chat_words_map_dict[word.upper()])
        else:
            new_text.append(word)
    return " ".join(new_text)
```

```
df['text'] = df['text'].parallel_apply(lambda sentence: chat_words_conversion(sentence))
df['summary'] = df['summary'].parallel_apply(lambda sentence: chat_words_conversion(sentence))
```

Figure 24: Preprocessing: Char words extension (*Self-Composed*)

The above code snippets are used to convert the short key words into longer form, such as e.g.: ‘ATM’ is converted into ‘At the moment’

```
def en_contractions(text: Text) -> Text:
    return ' '.join([contractions.fix(word)
                     if word in contractions.contractions_dict else word
                     for word in text.split()])
```

```
df['text'] = df['text'].parallel_apply(lambda sentence: en_contractions(sentence))
df['summary'] = df['summary'].parallel_apply(lambda sentence: en_contractions(sentence))
```

Figure 25: Preprocessing: Handling common contractions (*Self-Composed*)

The above code snippets are used to handle/extend common contractions such as e.g.: ‘They’re’ into ‘They are’

```
s_chars = '¥ҎÏïŰŬĎŸæ₿₩ÚÑÀèÅ”ĜåŽÖéříÿý€ŝĤ₹áŜŮÂ≥ûÌÇšŘúüëÓđŠčÎŤÆÒœ₩öËäøÍťìĈôàĥÝ¢ç“žðÙÊĉŭÈŒÐÉÔĵùÁů„âÄűĴóêĝÞîØòď₿ČÜþňÛ'
PUNC = '+@«#_\-!$%%^&*¬()£<>?/\\|}\]\[{;\,~:\"\''
```

```
def special_char(text: Text) -> Text:
    # first, let's remove any unicode strings
    text = text.encode('ascii', 'ignore').decode()
    # remove printable bachslashes
    text = re.sub(r'[\t\s\n\r\b\a]', ' ', text)
    # Special letters
    text = re.sub(r'[{}]'.format(s_chars), '', text)
    # Punctuation [remove punctuation between spaces only which represent noises]
    text = re.sub(r'\s[{}]\s'.format(PUNC), ' ', text)
    # space at the start or the end of the context
    text = re.sub(r'(^\s)|(\s$)', '', text)
    # Single character
    text = re.sub(r'(\s[^iIaA]\s)', ' ', text)
    return text
```

```
df['text'] = df['text'].parallel_apply(lambda sentence: special_char(sentence))
df['summary'] = df['summary'].parallel_apply(lambda sentence: special_char(sentence))
```

```
df.head(3)
```

Figure 26: Preprocessing: Removing special characters (*Self-Composed*)

```
from textblob import TextBlob

def spell_correction(df):
    # creating a new column for the corrected text
    df['corrected_text'] = df['text']
    # creating a new column for the corrected summary
    df['corrected_summary'] = df['summary']
    # creating a for loop for the entire dataset
    for i in range(len(df)):
        # Records
        print('Counter: ' + str(i+1) + '/' + str(len(df)+1))
        # creating a variable for the text of the current row
        text = df['corrected_text'][i]
        # creating a variable for the summary of the current row
        summary = df['corrected_summary'][i]
        # creating a variable for the corrected text of the current row
        corrected_text = TextBlob(text).correct()
        # creating a variable for the corrected summary of the current row
        corrected_summary = TextBlob(summary).correct()
        # updating the corrected text column with the corrected text
        df['corrected_text'][i] = str(corrected_text)
        # updating the corrected summary column with the corrected summary
        df['corrected_summary'][i] = str(corrected_summary)
    # returning the dataset with the new columns
    return df

spell_correction(df)
df_copy_correction = df.copy()
```

Figure 27: Preprocessing: Resolving spelling mistakes (*Self-Composed*)

```
def rm_duplicates(text: Text) -> Text:
    return re.sub(r'\b(\w+\s*)\1{1,}', '\\1', text)
```

```
df_copy_correction['corrected_text'] = df_copy_correction['corrected_text'].parallel_apply(lambda sentence: rm_duplicates(sentence))
df_copy_correction['corrected_summary'] = df_copy_correction['corrected_summary'].parallel_apply(lambda sentence: rm_duplicates(sentence))
```

Figure 28: Preprocessing: Removing duplicates (*Self-Composed*)

```
args = InferenceArguments(
                model_name_or_path="Qishuai/distilbert_punctuator_en",
                tokenizer_name="Qishuai/distilbert_punctuator_en",
                tag2punctuator=DEFAULT_ENGLISH_TAG_PUNCTUATOR_MAP,
)
inference = Inference(inference_args=args, verbose=False)
```

```
def punct_restoration(list_of_text: List[Text], name: Text) -> List[Text]:
    list_of_texts = []
    for text in tqdm(list_of_text, desc=f"Auto Punctuation for {name}"):
        list_of_texts.append(
            inference.punctuation([text])[0][0]
        )
    return list_of_texts
```

```
df_copy_correction['punc_corrected_text'] = punct_restoration(df_copy_correction['corrected_text'].values.tolist(), "text")
df_copy_correction['punc_corrected_summary'] = punct_restoration(df_copy_correction['corrected_summary'].values.tolist(), 'summary')
```

Figure 29: Preprocessing: Restoring missing punctuations (*Self-Composed*)

```
def grammely_correction(list_text: List[Text], name: Text) -> List[Text]:
    list_of_correction = []
    for text in tqdm(list_text, desc=f'Grammerly Correction for {name}'):
        if len(text.split()) < 50:
            list_of_correction.append(list(gf.correct(text, max_candidates=1))[0])
        else:
            list_of_correction.append(" ".join([list(gf.correct(sentence, max_candidates=1))[0]
                                                for sentence in tokenizer.tokenize(text)]))
    return list_of_correction
```

```
df_copy_correction['gram_corrected_text'] = grammely_correction(df_copy_correction['punc_corrected_text'].values.tolist(), "text")
df_copy_correction['gram_corrected_summary'] = grammely_correction(df_copy_correction['punc_corrected_summary'].values.tolist(), 'summary')
```

Figure 30: Preprocessing: Grammarly correction (*Self-Composed*)

APPENDIX D – CONCLUSION

D.1. Project Initial Plan

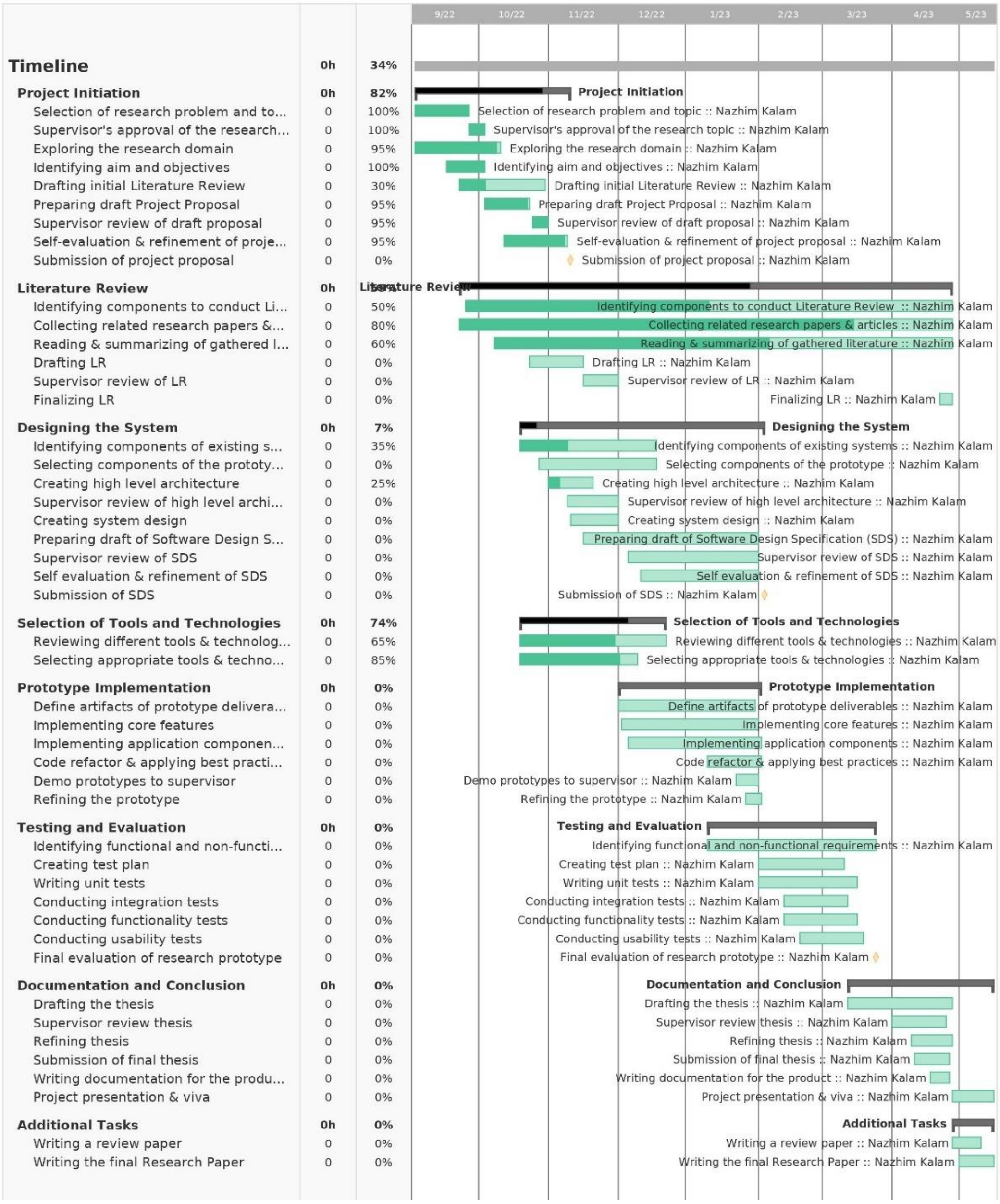


Figure 31: Gantt chart: Initial plan (*Self-Composed*)

D.2. Project Current Progress

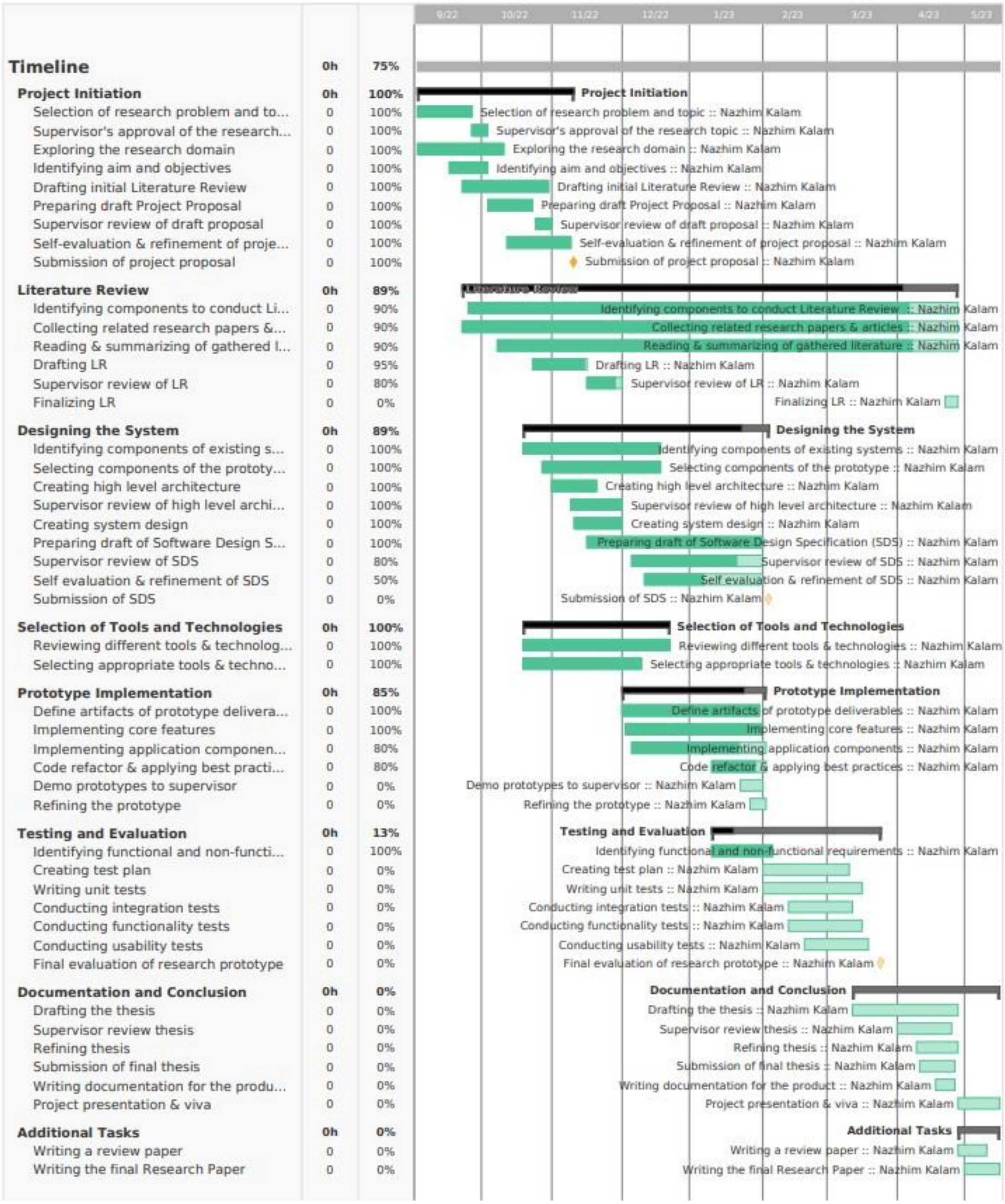


Figure 32: Gantt chart: Current plan (*Self-Composed*)

D.3. Initial Test Evaluation Results.

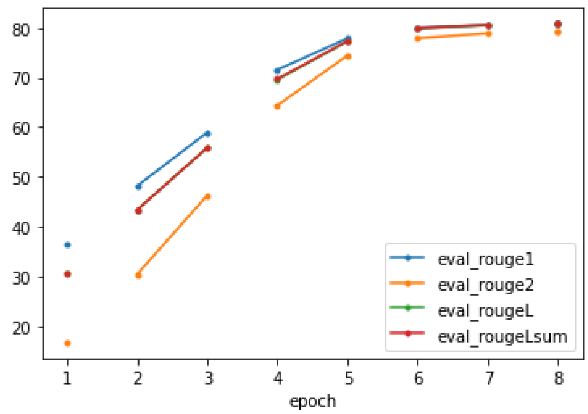


Figure 33: bart-base validation accuracy graph (*Self-Composed*)

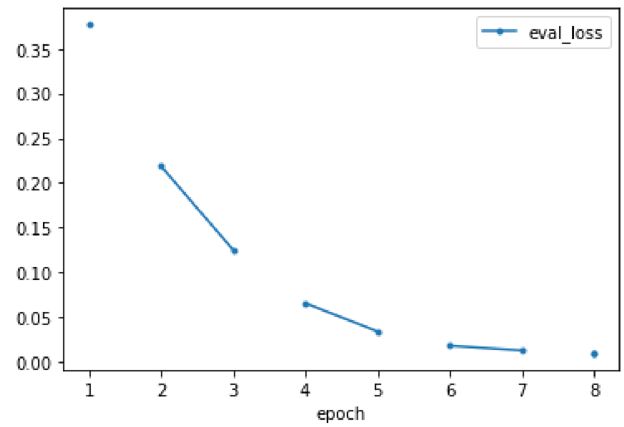


Figure 34: bart-base validation loss graph (*Self-Composed)*

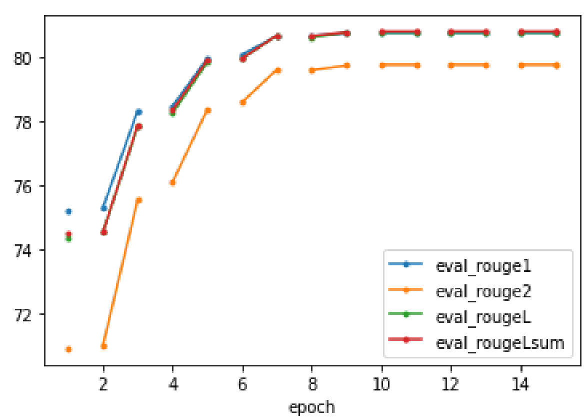


Figure 35: t5-base validation accuracy graph (*Self-Composed*)

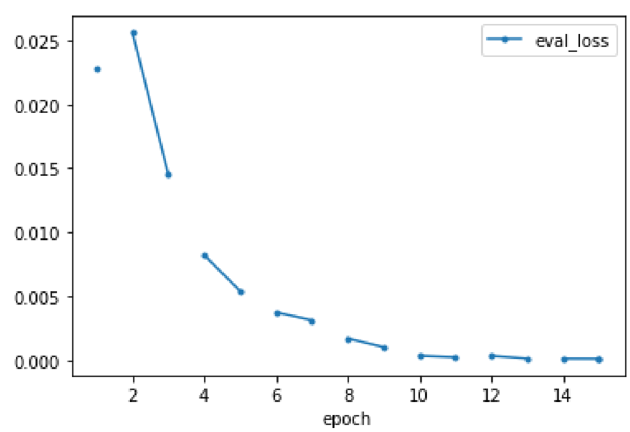


Figure 36: t5-base validation loss graph (*Self-Composed*)

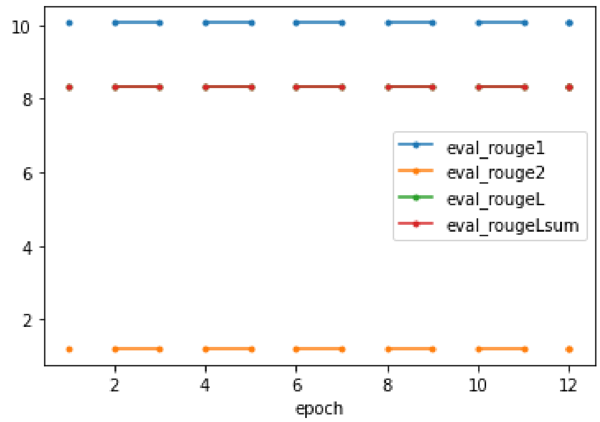


Figure 37: pegasus-base validation accuracy graph (*Self-Composed*)

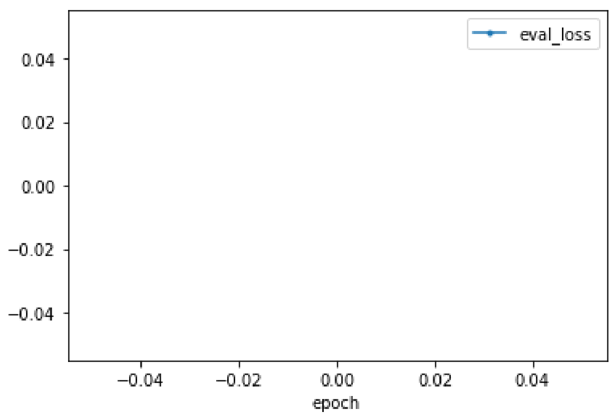


Figure 38: pegasus-base validation accuracy graph (*Self-Composed*)

END