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**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst. Integer euismod, nisi a vulputate malesuada, velit velit accumsan nibh, a blandit justo lacus vel enim. Sed ullamcorper commodo eros, non scelerisque justo dictum id. Aenean faucibus congue elit, id commodo ligula convallis a. Nullam facilisis libero id libero semper, eu pellentesque ante dapibus. Proin bibendum dictum mauris, a commodo quam dictum id. Nam placerat, nisl non congue viverra, elit quam congue dui, non congue sem nisi id metus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst. Integer euismod, nisi a vulputate malesuada, velit velit accumsan nibh, a blandit justo lacus vel enim. Sed ullamcorper commodo eros, non scelerisque justo dictum id. Aenean faucibus congue elit, id commodo ligula convallis a. Nullam facilisis libero id libero semper, eu pellentesque ante dapibus. Proin bibendum dictum mauris, a commodo quam dictum id. Nam placerat, nisl non congue viverra, elit quam congue dui, non congue sem nisi id metus.

**Keywords:** Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst

**Subject Descriptors:**

* Theory of computation 🡪 Theory and algorithms for application domains 🡪 Machine learning theory 🡪 Inductive inference.
* Theory of computation 🡪 Design and analysis of algorithms 🡪 Approximation algorithms analysis 🡪 Stochastic approximation.
* Mathematics of computing 🡪 Probability and statistics 🡪 Stochastic processes.
* Information systems 🡪 Information systems applications 🡪 Data mining.
* Computation methodologies 🡪 Machine learning 🡪 Machine learning algorithms 🡪 Ensemble methods

Contents

[**ABSTRACT** i](#_Toc126571403)

[List of Tables v](#_Toc126571404)

[List of Figures vi](#_Toc126571405)

[**CHAPTER 01. INTRODUCTION** 1](#_Toc126571406)

[**1.1. Chapter overview** 1](#_Toc126571407)

[**1.2. Problem domain** 1](#_Toc126571408)

[**1.2.1 Movie User Reviews** 1](#_Toc126571409)

[**1.2.2 Text Summarization** 2](#_Toc126571410)

[**1.2.3 Transformers** 2](#_Toc126571411)

[**1.3. Problem definition** 2](#_Toc126571412)

[**1.3.1 Problem statement** 3](#_Toc126571413)

[**1.4. Research questions** 3](#_Toc126571414)

[**1.5. Research aim & objectives** 3](#_Toc126571415)

[**1.5.1 Research aim** 3](#_Toc126571416)

[**1.5.2 Research objectives** 4](#_Toc126571417)

[**1.6. Novelty of the research** 6](#_Toc126571418)

[**1.6.1 Problem novelty** 6](#_Toc126571419)

[**1.6.2 Solution novelty** 6](#_Toc126571420)

[**1.7. Research gap** 6](#_Toc126571421)

[**1.8. Contribution to the body of knowledge** 7](#_Toc126571422)

[**1.8.1 Contribution to the research domain** 7](#_Toc126571423)

[**1.8.2 Contribution to the problem domain** 8](#_Toc126571424)

[**1.9. Research challenge** 8](#_Toc126571425)

[**1.10. Chapter summary** 8](#_Toc126571426)

[**CHAPTER 02. SOFTWARE REQUIREMENTS SPECIFICATION** 9](#_Toc126571427)

[**2.1. Chapter overview** 9](#_Toc126571428)

[**2.2. Rich picture** 9](#_Toc126571429)

[**2.3. Stakeholder analysis** 10](#_Toc126571430)

[2.3.1 Stakeholder onion model 10](#_Toc126571431)

[2.3.2 Stakeholder viewpoints 11](#_Toc126571432)

[**2.4. Selection of requirement elicitation methodologies** 11](#_Toc126571433)

[**2.5. Discussion of findings** 13](#_Toc126571434)

[2.5.1 Literature review 13](#_Toc126571435)

[2.5.2 Observations 13](#_Toc126571436)

[2.5.3 Survey 14](#_Toc126571437)

[2.5.4 Interviews 14](#_Toc126571438)

[2.5.5 Prototyping 14](#_Toc126571439)

[2.5.6 Summary of findings 15](#_Toc126571440)

[**2.6. Context diagram** 16](#_Toc126571441)

[**2.7. Use case diagram** 17](#_Toc126571442)

[**2.8. Use case descriptions** 17](#_Toc126571443)

[**2.9. Requirements** 19](#_Toc126571444)

[2.9.1 Functional requirements 19](#_Toc126571445)

[2.9.2 Non-functional requirements 20](#_Toc126571446)

[**2.10. Chapter summary** 21](#_Toc126571447)

[**CHAPTER 03. DESIGN** 22](#_Toc126571448)

[**3.1. Chapter overview** 22](#_Toc126571449)

[**3.2. Design goals** 22](#_Toc126571450)

[**3.3. High level design** 23](#_Toc126571451)

[**3.3.1. Architecture diagram** 23](#_Toc126571452)

[**3.3.2. Discussion of tiers of the architecture** 24](#_Toc126571453)

[**3.4. System design** 25](#_Toc126571454)

[**3.4.1. Choice of design paradigm** 25](#_Toc126571455)

[**3.5. Design diagrams** 25](#_Toc126571456)

[**3.5.1. Data flow diagrams** 25](#_Toc126571457)

[**3.5.1.1. Level 01 data flow diagram** 25](#_Toc126571458)

[**3.5.1.2. Level 02 data flow diagram** 26](#_Toc126571459)

[**3.5.2. Algorithmic design** 27](#_Toc126571460)

[**3.5.2.1. Existing LTC architecture** 27](#_Toc126571461)

[**3.5.2.2. Algorithm proposed by the author** 27](#_Toc126571462)

[**3.5.3. Algorithmic analysis** 31](#_Toc126571463)

[**3.5.4. System process activity diagram** 31](#_Toc126571464)

[**3.5.5. UI design** 32](#_Toc126571465)

[**3.6. Chapter summary** 32](#_Toc126571466)

[**CHAPTER 04. INITIAL IMPLEMENTATION** 33](#_Toc126571467)

[**4.1. Chapter overview** 33](#_Toc126571468)

[**4.2. Technology selection** 33](#_Toc126571469)

[**4,2.1. Technology stack** 33](#_Toc126571470)

[**4.2.2. Selection of data** 34](#_Toc126571471)

[**4.2.3. Selection of programming language** 34](#_Toc126571472)

[**4.2.4. Selection of development framework** 36](#_Toc126571473)

[**4.2.4.1 DL framework** 36](#_Toc126571474)

[**4.2.4.2. UI framework** 36](#_Toc126571475)

[**4.2.4.3. API web framework** 37](#_Toc126571476)

[**4.2.5. Other libraries & tools** 38](#_Toc126571477)

[**4.2.6. Integrated Development Environment (IDE)** 39](#_Toc126571478)

[**4.2.7. Summary of chosen tools & technologies** 39](#_Toc126571479)

[**4.3. Implementation of core functionalities** 39](#_Toc126571480)

[**4.3.1. Algorithm implementation** 39](#_Toc126571481)

[**4.3.2. Data fetchers** 43](#_Toc126571482)

[**4.3.3. Preprocessing** 43](#_Toc126571483)

[**4.4. Chapter summary** 43](#_Toc126571484)

[**CHAPTER 05. CONCLUSION** 44](#_Toc126571485)

[**5.1. Chapter overview** 44](#_Toc126571486)

[**5.2. Deviations** 44](#_Toc126571487)

[**5.2.1. Scope related deviations** 44](#_Toc126571488)

[**5.2.2. Schedule related deviations** 44](#_Toc126571489)

[**5.3. Initial test results** 44](#_Toc126571490)

[**5.4. Required improvements** 44](#_Toc126571491)

[**5.5. Demo of the prototype** 45](#_Toc126571492)

[**5.6. Chapter summary** 45](#_Toc126571493)

[**REFERENCES** I](#_Toc126571494)

[**APPENDIX A – INTRODUCTION** IV](#_Toc126571495)

[**A.1. Research questions** IV](#_Toc126571496)

[**APPENDIX B – SRS** IV](#_Toc126571497)

[**B.1. Requirement elicitation methodologies** IV](#_Toc126571498)

[**B.2. Survey analysis** V](#_Toc126571499)

[**B.3. Interview analysis** XI](#_Toc126571500)

[**B.4. Use case descriptions** XIII](#_Toc126571501)

[**B.5. Functional requirements** XIV](#_Toc126571502)

[**APPENDIX C – DESIGN** XV](#_Toc126571503)

[**C.1. Algorithm intuition** XV](#_Toc126571504)

[**C.2. UI wireframes** XVI](#_Toc126571505)

[**APPENDIX D – IMPLEMENTATION** XVIII](#_Toc126571506)

[**D.1. Fetch data** XVIII](#_Toc126571507)

[**D.2. Preprocessing** XXII](#_Toc126571508)

[**APPENDIX E – CONCLUSION** XXV](#_Toc126571509)

[**E.1. Project scope** XXV](#_Toc126571510)

[**E.2. Project schedule** XXVI](#_Toc126571511)

[**E.3. Project progress** XXVII](#_Toc126571512)

# List of Tables

[Table 1: Research Objectives (*Self-Composed*) 4](#_Toc125649728)

[Table 2: Stakeholder viewpoints (*self-Composed*) 11](#_Toc125649729)

[Table 3: Requirement elicitation methodologies (*Self-Composed*) 12](#_Toc125649730)

[Table 4: Observations findings (*Self-Composed*) 13](#_Toc125649731)

[Table 5: Prototyping findings (*Self-Composed*) 14](#_Toc125649732)

[Table 6: Use case description UC:03; UC:04 (*Self-Composed*) 17](#_Toc125649733)

[Table 7: Use case description UC:05; UC:06 (*Self-Composed*) 18](#_Toc125649734)

[Table 8: Functional requirements (*Self-Composed*) 19](#_Toc125649735)

[Table 9: Non-functional requirements (*Self-Composed*) 21](#_Toc125649736)

[Table 10: Design goals of the proposed system (*Self-Composed*) 22](#_Toc125649737)

[Table 11: Complexities of BPTT and adjoint sensitivity 31](#_Toc125649738)

[Table 12: Dataset sources (*Self-Composed*) 34](#_Toc125649739)

[Table 13: Selection of data science language (*Self-Composed*) 35](#_Toc125649740)

[Table 14: Selection of DL framework (*Self-Composed*) 36](#_Toc125649741)

[Table 15: Selection of UI framework (*Self-Composed*) 37](#_Toc125649742)

[Table 16: Selection of web framework (*Self-Composed*) 37](#_Toc125649743)

[Table 17: Chosen libraries (*Self-Composed*) 38](#_Toc125649744)

[Table 18: Chosen IDEs (*Self-Composed*) 39](#_Toc125649745)

[Table 19: Chosen tools & technologies (*Self-Composed*) 39](#_Toc125649746)

[Table 20: Stakeholder groups (*Self-Composed*) IV](#_Toc125649747)

[Table 21: Survey analysis (*Self-Composed*) V](#_Toc125649748)

[Table 22: Survey thematic analysis codes, themes & conclusions (*Self-Composed*) IX](#_Toc125649749)

[Table 23: Interview thematic analysis codes, themes & conclusions (*Self-Composed*) XI](#_Toc125649750)

[Table 24: Interview participant details (*Self-Composed*) XIII](#_Toc125649751)

[Table 25: Use case description UC:07 (*Self-Composed*) XIII](#_Toc125649752)

[Table 26: ‘MoSCoW’ technique of requirement prioritization (*Self-Composed*) XIV](#_Toc125649753)

# List of Figures

[Figure 1: Rich picture diagram (*Self-Composed*) 9](#_Toc125649754)

[Figure 2: Stakeholder onion model (*self-Composed*) 10](#_Toc125649755)

[Figure 3: Context diagram (*Self-Composed*) 16](#_Toc125649756)

[Figure 4: Use case diagram (*Self-Composed*) 17](#_Toc125649757)

[Figure 5: Three-tiered architecture (*Self-Composed*) 23](#_Toc125649758)

[Figure 6: Data flow diagram - level 01 (*Self-Composed*) 26](#_Toc125649759)

[Figure 7: Data flow diagram - level 02 (*Self-Composed*) 26](#_Toc125649760)

[Figure 8: System process activity diagram (*Self-Composed*) 32](#_Toc125649761)

[Figure 9: Tech stack (*Self-Composed*) 33](#_Toc125649762)

[Figure 10: Initialize algorithm (*Self-Composed*) 40](#_Toc125649763)

[Figure 11: Build algorithm (*Self-Composed*) 40](#_Toc125649764)

[Figure 12: Algorithm – sensory, stochastic and leakage variables (*Self-Composed*) 41](#_Toc125649765)

[Figure 13: Algorithm – forward propagation (*Self-Composed*) 42](#_Toc125649766)

[Figure 14: Algorithm – define weights and biases (*Self-Composed*) 42](#_Toc125649767)

[Figure 15: Algorithm – Euler-Maruyama SDE solver (*Self-Composed*) 43](#_Toc125649768)

[Figure 16: Algorithm intuition (*Self-Composed*) XV](#_Toc125649769)

[Figure 17: UI wireframes – Home (*Self-Composed*) XVI](#_Toc125649770)

[Figure 18: UI wireframes – News (*Self-Composed*) XVI](#_Toc125649771)

[Figure 19: UI wireframes – Cryptocurrencies (*Self-Composed*) XVI](#_Toc125649772)

[Figure 20: UI wireframes – Cryptocurrency (*Self-Composed*) XVI](#_Toc125649773)

[Figure 21: UI wireframes – Admin login (*Self-Composed*) XVII](#_Toc125649774)

[Figure 22: UI wireframes – Admin model configuration (*Self-Composed*) XVII](#_Toc125649775)

[Figure 23: UI wireframes – Forecast (*Self-Composed*) XVII](#_Toc125649776)

[Figure 24: Fetch historical prices (*Self-Composed*) XVIII](#_Toc125649777)

[Figure 25: Fetch Twitter volume (*Self-Composed*) XIX](#_Toc125649778)

[Figure 26: Fetch block reward size (*Self-Composed*) XIX](#_Toc125649779)

[Figure 27: Scrape tweets (*Self-Composed*) XIX](#_Toc125649780)

[Figure 28: Clean tweets (*Self-Composed*) XX](#_Toc125649781)

[Figure 29: Fetch Google Trends (*Self-Composed*) XXI](#_Toc125649782)

[Figure 30: Analyze sentiments (*Self-Composed*) XXII](#_Toc125649783)

[Figure 31: Combine and condense tweets (*Self-Composed*) XXIII](#_Toc125649784)

[Figure 32: Combine all datasets (*Self-Composed*) XXIV](#_Toc125649785)

[Figure 33: Initial Gantt chart (*Self-Composed*) XXVI](#_Toc125649786)

[Figure 34: Current Gantt chart (*Self-Composed*) XXVII](#_Toc125649787)

**Acronyms**

|  |  |
| --- | --- |
| **AI** | Artificial Intelligence. |
| **API** | Application Programming Interface. |
| **AD** | Automatic Differentiation. |
| **ARIMA** | Autoregressive Integrated Moving Average. |
| **BPTT** | Back-Propagation Through Time. |
| **BTC** | Bitcoin. |
| **CT-GRU / RNN** | Continuous-time Gated Recurrent Unit / Recurrent Neural Network. |
| **DL** | Deep Learning. |
| **GPU** | Graphics Processing Unit. |
| **LSTM** | Long Short-Term Memory. |
| **LTC** | Liquid Time-constant. |
| **ML** | Machine Learning. |
| **(s)MAPE** | Symmetric Mean Absolute Product Error. |
| **MASE** | Mean Absolute Scaled Error. |
| **MSE** | Mean Squared Error. |
| **MVP** | Minimal Viable Product. |
| **N-BEATS** | Neural Basis Expansion Analysis for interpretable Time Series. |
| **NLP** | Natural Language Processing. |
| **ODE** | Ordinary Differential Equations. |
| **POC** | Proof-Of-Concept. |
| **REST** | Representational State Transfer. |
| **RMSE** | Root Mean Squared Error. |
| **RNN** | Recurrent Neural Network. |
| **SOTA** | State Of the Art. |
| **SDE** | Stochastic Differential Equations. |
| **SGD** | Stochastic Gradient Descent. |
| **TS** | Time Series. |
| **UI** | User Interface. |
| **XAI** | Explainable Artificial Intelligence. |

# **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**

The research questions proposed are available in [**APPENDIX A.1**](#_A.1._Research_questions).

# **1.5. Research aim & objectives**

## **1.5.1 Research aim**

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

To further explain the objective, 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.

To confirm or disprove the selected hypothesis, the necessary information will be obtained and investigated, components will be built, and performance will be evaluated. Both a hosted server and a local browser will be able to execute the system for private or public usage. The data science models and their source code will be made accessible for future study and usage in a public repository. The information gleaned from the literature review will be published in a review paper.

## **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 | [**R**Q1](#researchq1),[**R**Q2](#researchq1),[**R**Q3](#researchq1),[**R**Q4](#researchq1) |
| 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 | [**R**Q4](#researchq1),[**R**Q2](#researchq1),[**R**Q1](#researchq1) |
| Requirement Elicitation | Defining the project's needs utilizing relevant approaches and tools in order to solve the projected research gaps and obstacles based on prior related 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](#researchq1),[**R**Q2](#researchq1),[**R**Q1](#researchq1) |
| 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](#researchq2) |
| 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](#researchq2),[**R**Q3](#researchq2) |
| 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, BERT SCORE). | LO1,  LO5 | [**R**Q3](#researchq3) |
| 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.

## **1.6.2 Solution novelty**

The solution novelty for this problem has several approaches few of which performing automated hyperparameter tuning, creating a retraining mechanism with newly exposed data and exploring any changes in the model architecture to enhance its performance further.

# **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 Contribution to the research domain**

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 Contribution to the problem domain**

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**

In this chapter, the author gave an outline of the research project that was carried, the reasons why the research and problem were innovative, and the difficulties that could arise while trying to solve them. In addition, the essential objectives that must be pursued for the study to be considered effective were put out and linked to the degree's required learning outcomes.

# **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. |
| Data Scientists | Quality Control Regulator | Provides performance enhancements for the models and algorithms used in data science. |
| Data Engineers |  | Gives guidance on potential data that may be used to generate the best suggestions possible. |
| AI Researchers |  | Conduct research in the specified area to enhance and implement reliable text summarizing models. |
| NLP Experts |  | Offers specialized guidance and insights on the field  knowledge, to enhance the functionality of the system. |
| Domain Specific Manager | Operational Beneficiary | Text reviews are used as inputs for abstractive summarization, and the model is retrained with prior inputs as new data to increase performance. |
| General Users | Unless specifically assigned or retrained, typical users will utilize a general abstractive summarization model. |
| 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 | Checks to see if the system is ready for production use and puts it through its paces. |

# **2.4. Selection of requirement elicitation methodologies**

There were several requirement elicitation approaches used to collect needs for the creation of the research project. The approaches selected for this were literature review, survey, interviews, prototyping, brainstorming and self-evaluation. The following is a discussion of the rationales behind selecting the mentioned requirement elicitation approaches.

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| Literature Review | To determine research gaps in the chosen domain of interest and the intended topic of study at the project's outset, the author conducted a thorough literature analysis. Current systems were researched together with comparable technologies that might be applied to the existing systems that were referenced in literature in order to discover research gaps available in technologies that can be used. |
| Survey | A questionnaire was utilized as a survey instrument to obtain requirements and opinions from possible users of the suggested system. The author will benefit from this sort of poll in understanding people's thought processes and expectations for the prototype. It will also enable the author to explain whether or not the targeted users will benefit from the suggested solution. |
| Interviews | Interviews were performed to gain expert insight into domain-specific requirements and to determine the best method to address the issue at hand while adding to the body of knowledge through research. Interviews were determined to be the greatest source of information because the field is new and the technical expertise needed is very precise. Additionally, this technique allowed for the qualitative evaluation of the suggested system, allowing for the identification of any shortcomings or difficulties that could need to be resolved during prototyping. |
| Prototyping | The project was chosen to follow the Agile Software Development Life-cycle, thus prototyping would allow the author to test and evaluate the prototype while iteratively trying out several alternative implementations to find any potential areas for improvement. |
| Brainstorming | Whether you're attempting to come up with a broad subject before you start your research, you're trying to focus more specifically, or you're determining what evidence to use for a particular paragraph, brainstorming is a useful technique to produce ideas at every step of the process. In order to assess the system for personally, the author has a number of brainstorming sessions with his colleagues at various project stages. |
| 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 B.1**](#_B.1._Requirement_elicitation) 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

In-order to gather requirements from the target audience to list the functionalities needed for the project develop, a survey was conducted. The result analysis is available at [**APPENDIX B.2**](#_B.2._Survey_analysis).

## 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 and is available at [**APPENDIX B.3**](#_B.3._Interview_analysis)..

## 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 B.4**](#_B.4._Self-Evaluation_(Competitor).

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. | Numerous requirements and obstacles were made clear during the iterative prototype process. Especially in the area of movies, finding a good dataset with the desired metadata was a significant challenge. The author was able to uncover a significant dataset with around 8 million entries after completing an intensive evaluation of research papers. The dataset has to be split into smaller segments for usage, nevertheless, due to its size. Preprocessing the data was difficult since it was not just enormous but also noisy text data, which required a lot of cleaning. The author experimented with automatic hyperparameter search approaches because manual hyperparameter tweaking required a significant amount of time and is not practical when working with automated model retraining process. He discovered that a framework named "Optuna" was useful for automatically improving and retraining the model. In order to retrain the system, new data entered by the domain user will be incorporated. The author has to study at least three top-tier transformer designs in order to choose the optimal 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. |  | ✓ |  |  |  |  |
| 8 | Creating a hybrid transformer model to further increase the performance is a suggested improved. |  |  |  | ✓ | ✓ |  |
| 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.

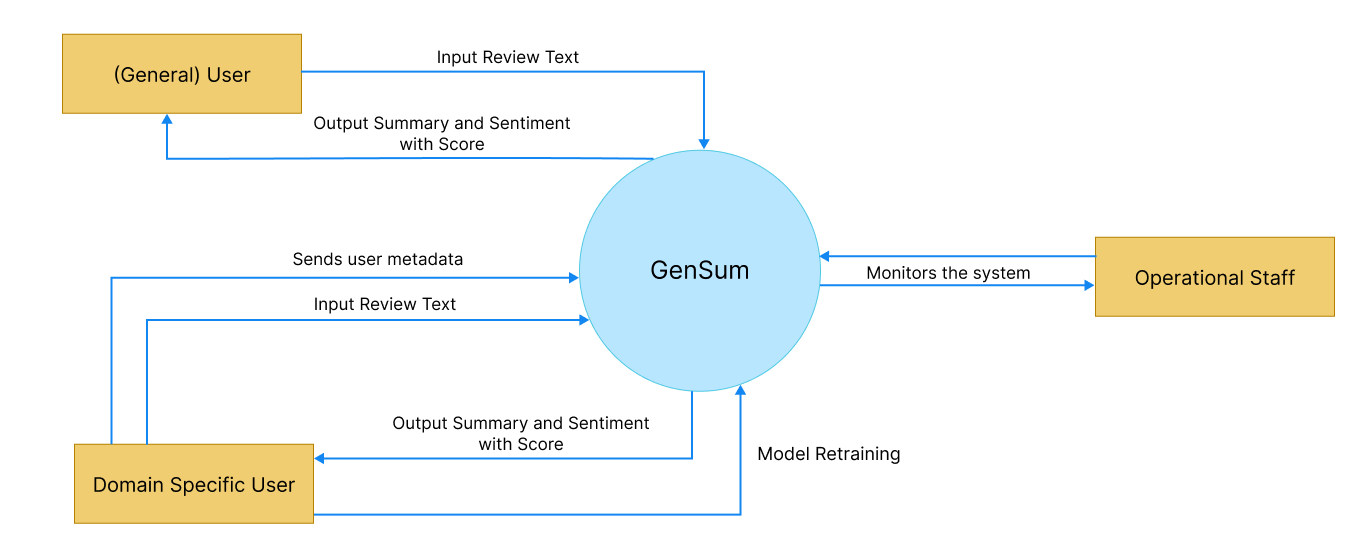


Figure 3: Context diagram (*Self-Composed*)

# **2.7. Use case diagram**

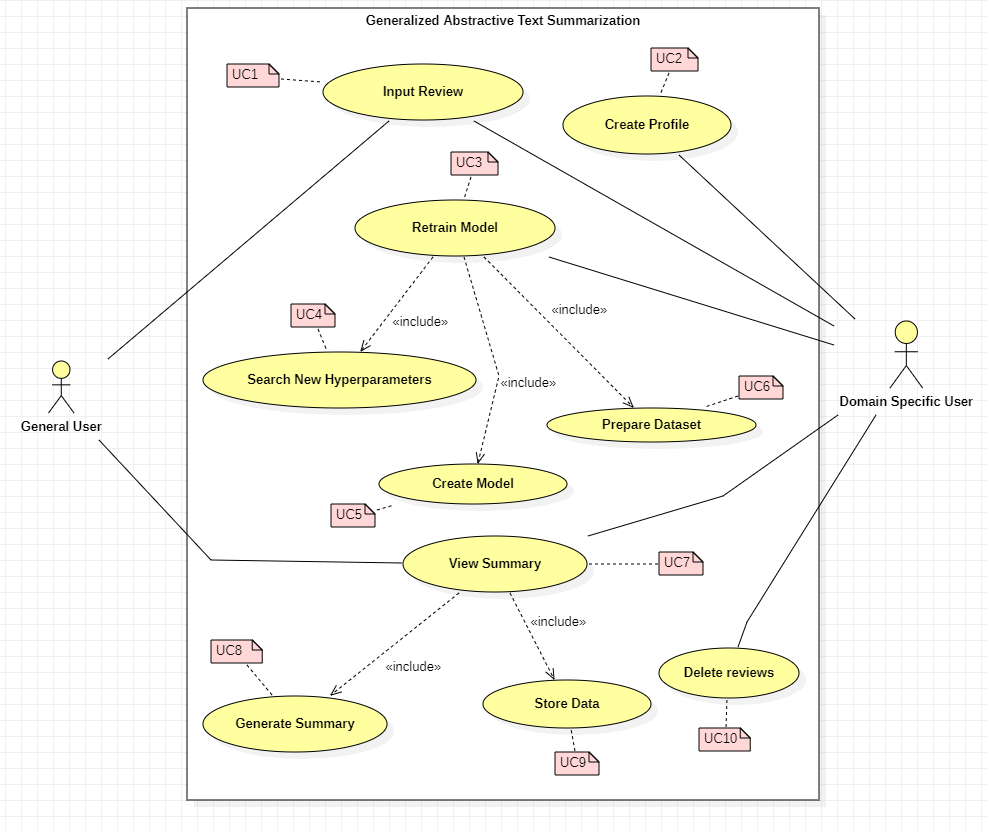


Figure 4: Use case diagram (*Self-Composed*)

# **2.8. Use case descriptions**

Usecase diagrams with the highest importance are given below, the rest of the Usecase descriptors are available at [**APPENDIX B.5**](#_B.5._Use_case).

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

|  |  |  |
| --- | --- | --- |
| Use Case Name | View Summary | |
| Use Case Id | UC:07 | |
| Description | Displays a summarized version of the uploaded review text from the domain user’s end. | |
| Primary Actor | General User, Domain Specific User | |
| Pre-Conditions | The text review data must go through specific text preparation techniques before the summary can be produced. | |
| Extended use cases | None | |
| Included use cases | UC10, UC02 | |
| Trigger | A user selects to summarize a given customer/user review text. | |
| Main flow | **Actor** | **System** |
| 1. The user enters the review text on the text field from the GUI. 2. Clicks on “Generate Summary” from the GUI | 1. The system does the data preprocessing for the input review text. 2. Loads the generalized transformer model. 3. Generates the summary using the model. 4. **(If Domain Specific User)** stores the input review and summary into the database. 5. Returns the summary response back to the GUI |
| 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 end condition: The user is presented with the summarized review text. | |

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

|  |  |  |
| --- | --- | --- |
| Use Case Name | Retrain Model | |
| Use Case Id | UC:03 | |
| Description | Performs model retraining with the new data from the database, to find the new best set of hyperparameters. | |
| Primary Actor | Domain Specific User | |
| Pre-Conditions | The actor should be a Domain Specific User and have an account created. | |
| Extended use cases | None | |
| Included use cases | UC05, UC06, UC07 | |
| Trigger | The Domain Specific User clicks on the “Perform model retraining” button | |
| Main flow | **Actor** | **System** |
| 1. Domain Specific logs into their account 2. Clicks on “Perform model retraining” from the GUI | 1. The system pulls all the data with respect to the user id from the database. 2. Combines data of the common domains (only if user consent is given to use their data) 3. Finds new set of hyperparameters for the model with respect to new data. 4. Trains the model using the new hyperparameters. 5. Saves the model with the user Id 6. Updates the status in the database if succeed/fails |
| 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 end condition: The user will be able to see the recent status of the model if the retraining is successful or failed | |

# **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 priotity levels are detailed at [**APPENDIX B.6**](#_B.5._Functional_requirements).

The Usecase description along its mapping id is also listed at [**APPENDIX B.7**](#_B.5._Functional_requirements)

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 also find the sentiment of the generated summary if its positive or negative and return the result. | C | UC08 |
| FR13 | The system could make use of a hybrid model for the text summarization. | 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**

In this chapter, a Rich Picture Diagram was created to show how the system interacts with society and the system stakeholders. The stakeholders were represented using Saunder's Onion model, which included the flow of influence from each stakeholder. To acquire all the necessary information and the opinions of potential system stakeholders, requirement gathering approaches were used. Lastly, the insights gained from the requirement elicitation approaches were used to specify the system's use cases, functional requirements, and non-functional requirements.

# **CHAPTER 03. DESIGN**

# **3.1. Chapter overview**

The design choices taken to create a suitable architecture for implementation, depending on the requirements received, are discussed in this chapter. To explain how the design goals are intended to be accomplished while outlining the justification for selected design decisions, high-level design, low-level design, design diagrams, and UI wireframes have been utilized.

# **3.2. Design goals**

Table 12: 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. |

# **3.3. High level design**

## **3.3.1. Architecture diagram**

The image below depicts the architecture of the system. Three tiers of architecture separate the data, logic, and presentation levels. The system's generalization and domain specific adaptive hyperparameter tuning and data preprocessing represent the research contribution.

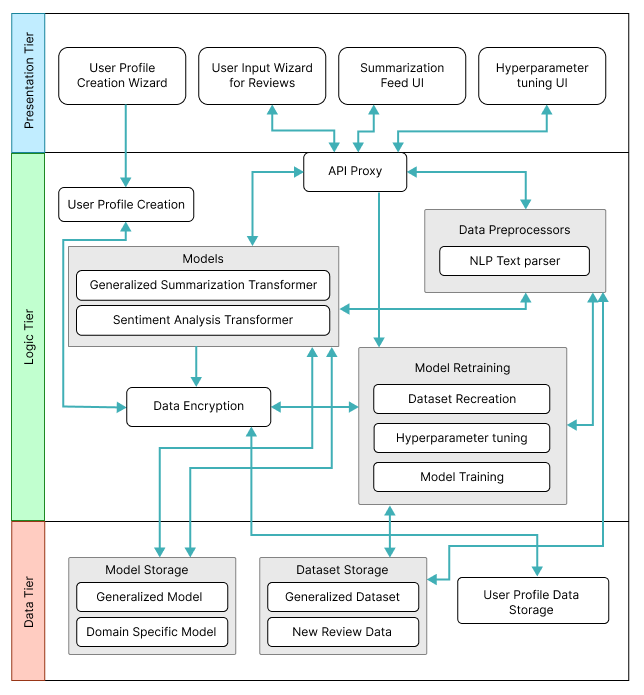


Figure 5: Three-tiered architecture (*Self-Composed*)

## **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.
2. Generalized Model – The model which will be used by general users to generated review summarized, this model will be hyperparameter tuned for genialized purpose.
3. 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.
4. Dataset Storage – The data which is required for model training will be available.
5. 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.
6. 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.
7. 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.
4. NLP Text parser – Responsible for cleaning the input text review when received from the API endpoint.
5. Models – The model which will be responsible in generating the summary from the input review and find the sentiment of the summary generated.
6. 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.
7. Sentiment Analysis Transformer – This model will be used to classify the generated summary into positive or negative sentiment.
8. 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.
9. Model Retraining – Responsible for retraining the model with new data and finding new set of hyperparameters.
10. Dataset Recreation – Responsible for recreating the dataset with new data which has been given as input from the domain users
11. Hyperparameter tuning – Responsible for finding the new best set of hyperparameters using the new data.
12. 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**

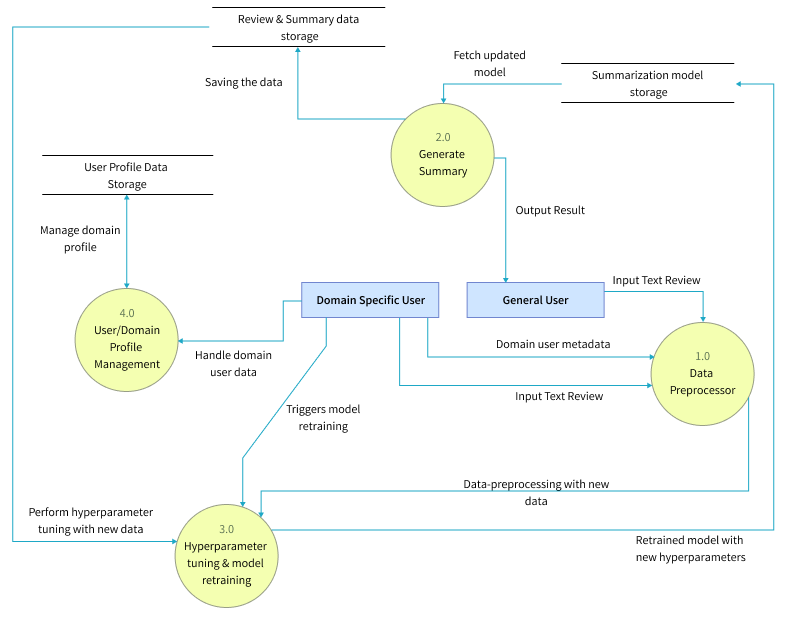


Figure 6: Data flow diagram - level 01 (*Self-Composed*)

### **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.

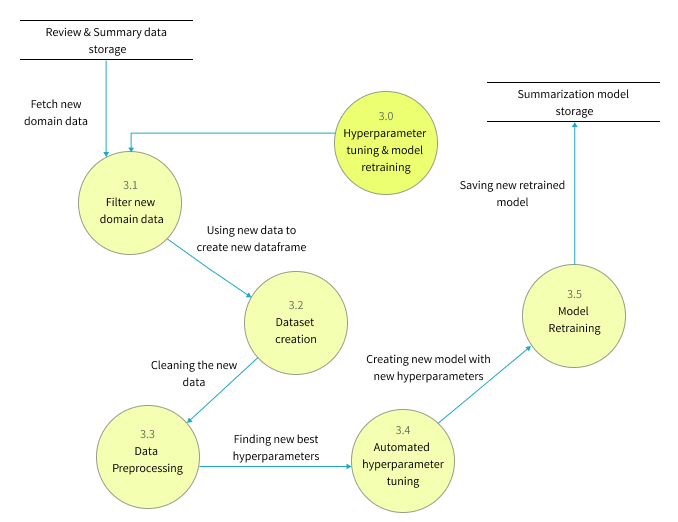


Figure 7: Data flow diagram - level 02 (*Self-Composed*)

## **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.

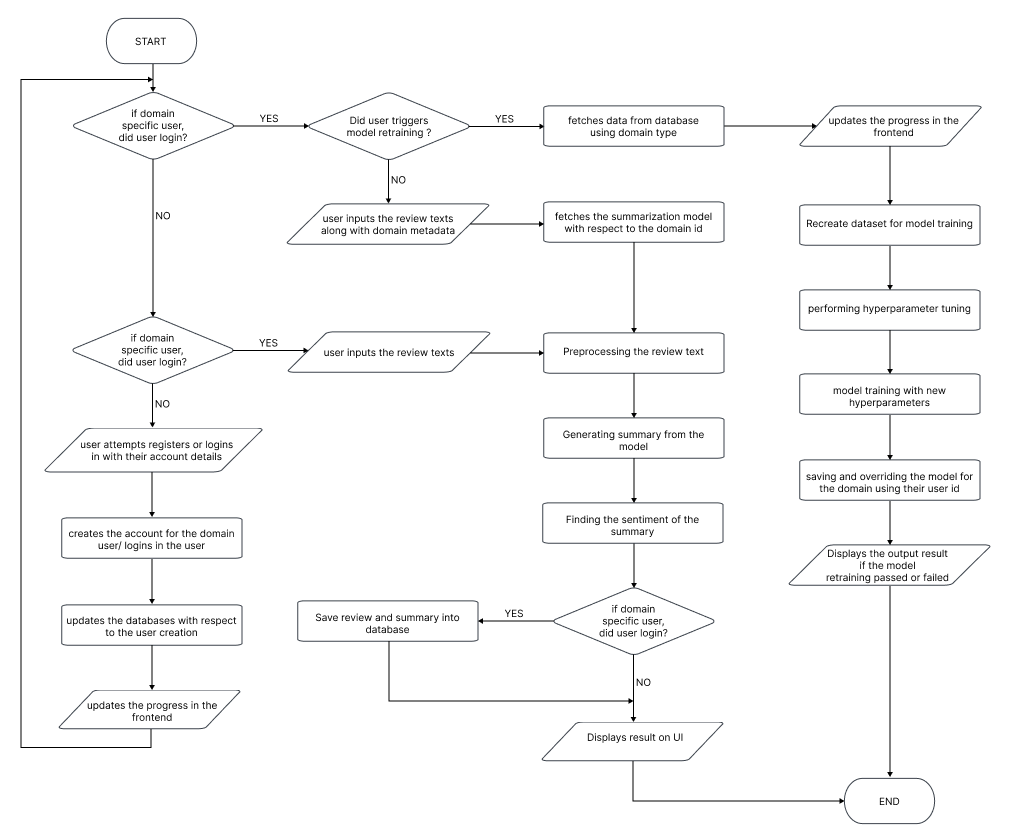


Figure 8: System process flow chart (*Self-Composed*)

## **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 C.2**](#_C.2._UI_wireframes)

# **3.6. Chapter summary**

This chapter provides an in-depth examination of the project's design, including its architectural features and overall flow. Algorithm design and the process is also discussed. 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**

In this chapter, the author describes the core implementation of the system and the necessary decisions taken to approach that implementation. Moreover, the chosen tools, languages, and technologies are presented alongside their reasoning.

# **4.2. Technology selection**

## **4,2.1. Technology stack**

The chosen technologies are depicted in the diagram below.



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

## **4.2.2. Selection of data**

As this is a data science project, the highest quality of data is a necessity. The author utilized multiple sources of data that are potential contributions to the target inference; the following were required:

* BTC historical data
* BTC block reward size
* BTC tweets
* BTC Twitter volume
* BTC Google Trends

The univariate single horizon forecasting model utilized the above data in a combination, while the multivariate multi-horizon forecasting model utilized solely the historical data. The below table describes the sources of each respective dataset.

Table 12: Dataset sources (*Self-Composed*)

|  |  |
| --- | --- |
| **Dataset** | **Source** |
| BTC historical data | From a third-party investing.com API. |
| BTC block reward size, BTC Twitter volume | From a public dashboard that provides multiple different information about a specific cryptocurrency. |
| BTC tweets | Tweets from 2014-2019 were downloaded from Kaggle – the remaining till date were extracted from a Twitter tweet scraper. |
| BTC Google Trends | From the PyTrends library that provides Google Trends data. |

Gathering the data was a long and arduous process as it was not as simple as downloading available datasets, and certain APIs being rate-limited. Dedicated python scripts were written to extract the data and to streamline updating available data. The author will publicize these scripts and the data to facilitate future research.

## **4.2.3. Selection of programming language**

Programming languages were analyzed prior to development. Specifically, for three main aspects: the client, the data science component, and the API communicating between the model and the client.

The below table summarizes the analysis for the language chosen for the data science component; where each option was given a score within H – High, M – Medium, and L – Low.

Table 13: Selection of data science language (*Self-Composed*)

|  |  |  |  |
| --- | --- | --- | --- |
| **Data science**  To implement the core data science components two of the most popular languages that are used widely for data science were analyzed. | | | |
| **Aspect** | **Relevance** | **Python** | **R** |
| Availability of libraries. | A language that supports multiple libraries is paramount as the author would require multiple different techniques to gather the required data and streamline the model and algorithm development. | H | M |
| Author familiarity and ease of implementation. | Implementing the algorithm, the mathematical intricacies, and the respective model should be as simple as possible. It is an additional benefit if the author has hands-on experience with the chosen language, | H | M |
| Learning curve | The difficulty of the chosen language must not be a hindrance as the goal is to utilize the tool to implement a system rather than spending time learning the language. | L | M |
| Community and documentation. | Community support and well-written documentation is paramount as the author will not have time to debug trivial issues. | H | M |
| **Conclusion**  Based on the analysis, the author decided to use **Python**, as it was more relevant. | | | |

To develop the user interface not much competition is present to analyze. **JavaScript** is the stand-alone leader and is the choice of the author as it is dynamic and can handle user interactions seamlessly. Although recent technology has presented the usage of C# for frontend development, high latency issues and lack of community knowledge are a downfall.

To setup the communication between the model and the user interface APIs are required. Multiple technologies are available for API development. The author chose **Python** as their core data science component is also built using Python; therefore, utilizing the same language would reduce the time taken to learn new languages for insignificant reasons.

## **4.2.4. Selection of development framework**

### **4.2.4.1 DL framework**

The author chose Python for developing the core data science component. As the core algorithm and model will be DL-based, DL frameworks must be meticulously analyzed to choose the most relevant framework. The two most popular frameworks, TensorFlow and PyTorch, were analyzed.

Table 14: Selection of DL framework (*Self-Composed*)

|  |  |
| --- | --- |
| **Framework** | **Description** |
| TensorFlow | Used for production level applications, has detailed documentation, community support and handles large datasets. It also provides better visualization options which makes it easy to debug and monitor training, which is important as a novel algorithm is being built and no comparison is present. |
| PyTorch | Is more lightweight and developer-friendly, as it provides a more higher-level development. Therefore, has a much smaller learning curve, easier to get started, and feels more intuitive as it is simpler to build models. |
| **Conclusion**  The author opted to use **TensorFlow**. Although it is more complicated, the higher-level API: Keras, is now officially a part of TensorFlow. Therefore, model development has become much simpler. Additionally, building the algorithm requires more low-level details.  (PyTorch vs. TensorFlow: 2022 Deep Learning Comparison | Built In, 2022) | |

### **4.2.4.2. UI framework**

As JavaScript was chosen for developing the UI, respective JavaScript frontend frameworks and libraries must be analyzed. There is an ocean of JavaScript libraries- the top four were chosen for evaluation; the four being Angular, Vue, Svelte, and React.

Table 15: Selection of UI framework (*Self-Composed*)

|  |  |
| --- | --- |
| **Framework** | **Description** |
| Angular | Suitable for large scale applications with dedicated submodules for particular functionalities. However, can be less performant in comparison and unnecessarily heavy. |
| Vue | Tiny framework that takes little to no time to startup, and is much more intuitive as the code is simple and straightforward. Additionally, based on simulations, it has been identified to perform better than Angular and React. However, has much fewer resources. |
| Svelte | Most lightweight and truly reactive. Much more performant than the rest; however, has a small community of developers and is relatively new. |
| React | Customizable and promotes code reusability via functions as components. Carries a large community and is open-source while being SEO friendly. Additionally, the React developer tools is a very handy tool. |
| **Conclusion**  Based on the analysis, the author chose **React** as the GUI built will be simple and there is no requirement for large-scale applications, as it is not the primary focus.  (Angular vs React | Angular vs Vue | React vs Vue – Know the Difference, 2021) | |

### **4.2.4.3. API web framework**

As python was chosen for the API development, respective Python web frameworks must be analyzed to choose the more relevant one. Analysis was conducted between Django and Flask as they are the two most popular frameworks.

Table 16: Selection of web framework (*Self-Composed*)

|  |  |
| --- | --- |
| **Framework** | **Description** |
| Flask | A very lightweight framework that provides only the simplest of functionalities. However, is the preferred choice for ML API development due to it being lightweight. |
| Django | Suitable for more larger scaled applications that provides a vast range of functionalities and is stricter and less flexible. Therefore, is much more demanding and heavier. |
| **Conclusion**  The author chose **Flask** as it provides only the necessities in exposing an ML model and since the luxury features provided by Django (ex: authentication) were not required.  (Flask Vs Django: Which Python Framework to Choose?, 2021) | |

## **4.2.5. Other libraries & tools**

Table 17: Chosen libraries (*Self-Composed*)

|  |  |
| --- | --- |
| **Library** | **Justification** |
| NumPy | Facilitates mathematical functions and calculations that is immensely required when building the algorithm. |
| Pandas | To create dataframes to perform analysis, cleaning, transformations, filtration etc. on the datasets. |
| Scikit-learn | To create data splits and feature scaling. |
| Lingua | To detect the language of the tweets. As this project is limited to using only English tweets, they must first be identified. |
| SpacCy | To perform NER to extract entities that could potentially be within the pre-defined impactful index. |
| Matplotlib + Seaborn | For analysis, visualizations and dashboarding. |
| Beautiful Soup | For scraping the block reward size and the Twitter volume from the public dashboard. |
| VADER | Perform sentiment analysis on the tweets. |
| TensorBoard | Visualize and obtain insights of the model training process associated evaluation metrics and additional dashboarding. |
| Redux | For API requests from the client. |
| Ant design | Makes creating appealing user interfaces hassle-free. |

## **4.2.6. Integrated Development Environment (IDE)**

Table 18: Chosen IDEs (*Self-Composed*)

|  |  |
| --- | --- |
| **IDE** | **Justification** |
| Kaggle | Consists of 32GB of RAM; therefore, all datasets can be loaded and processed at once without needing to process sections of data at a time. Additionally, provides easy integration with existing Kaggle datasets and user-uploaded datasets. |
| Jupyter | For local trials and testing. |
| VSCode | Lightweight and extremely powerful. Consists of multiple shortcuts, extensions and snippets that can significantly boost development productivity. |

## **4.2.7. Summary of chosen tools & technologies**

Table 19: Chosen tools & technologies (*Self-Composed*)

|  |  |
| --- | --- |
| **Component** | **Tools** |
| Programming languages | Python, JavaScript |
| Development framework | Flask, TensorFlow |
| UI development framework | Ant design |
| Libraries | React, NumPy, Pandas, Scikit-learn, Beautiful Soup, Lingua, Matplotlib, Seaborn, VADER sentiment analyzer. |
| IDEs | Kaggle and Jupyter notebooks; VSCode. |
| Version control | Git + GitHub |

# **4.3. Implementation of core functionalities**

The novel algorithm, the scripts to fetch the required data, and the preprocessing performed can be considered as the core functionalities of the project.

## **4.3.1. Algorithm implementation**

The author initially implemented the LTC architecture since there is no modern reference utilizing recommended best practices and approaches. The author then built on this architecture, replacing the underlying ODEs with SDEs.

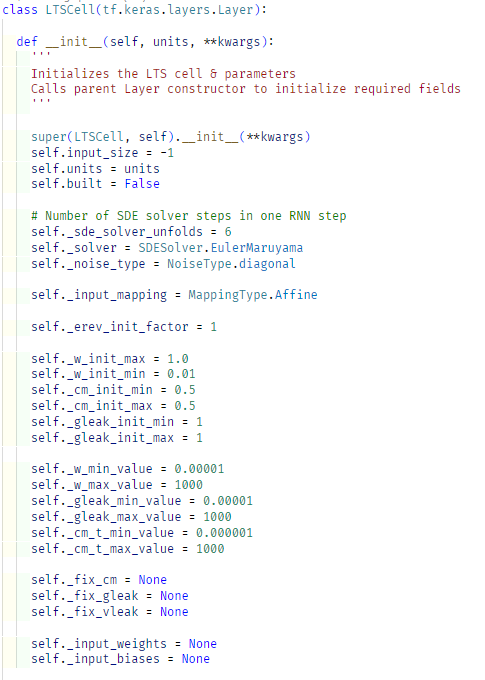


Figure 10: Initialize algorithm (*Self-Composed*)

The above code snippet initializes the algorithm cell with the necessary variable maximum and minimum values. In the above method, the built model can perform input-independent initializations. By inheriting from the base Keras Layer class, the ability to be used in the higher level of the model’s layer definition is obtained (as existing LSTM and RNN cells).

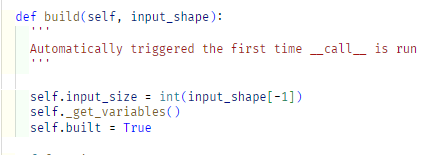
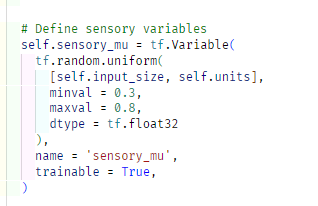
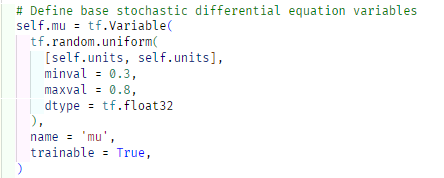


Figure 11: Build algorithm (*Self-Composed*)

The above snippet defines what occurs upon initialization; in other words, it “builds” the algorithm cell. A helper function is utilized here that defines the variables (sigma, mu, weights, and leakage conductance variables (Hasani et al., 2020)). The input shape is available within the above function; therefore, the model can initialize the variables used here. The below snippet demonstrates how some of these variables are initialized.

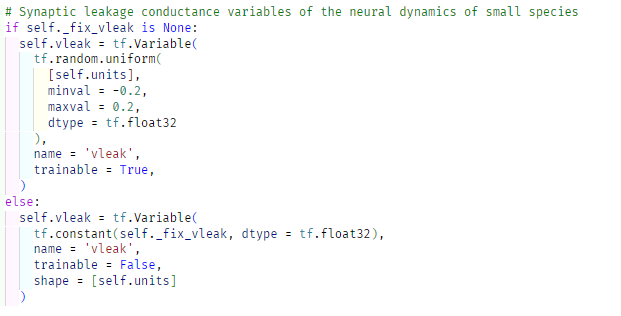


Figure 12: Algorithm – sensory, stochastic and leakage variables (*Self-Composed*)

The final step is the forward computation process that will occur on each epoch, in other words, the forward propagation process.

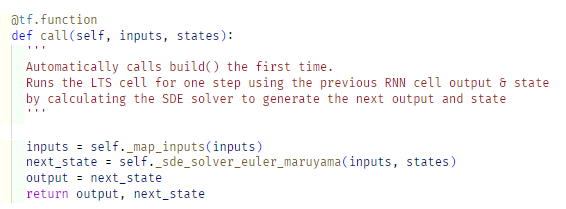


Figure 13: Algorithm – forward propagation (*Self-Composed*)

The above function is run automatically on each epoch. Initially, a helper function defines the weights and biases of the network, as demonstrated below.

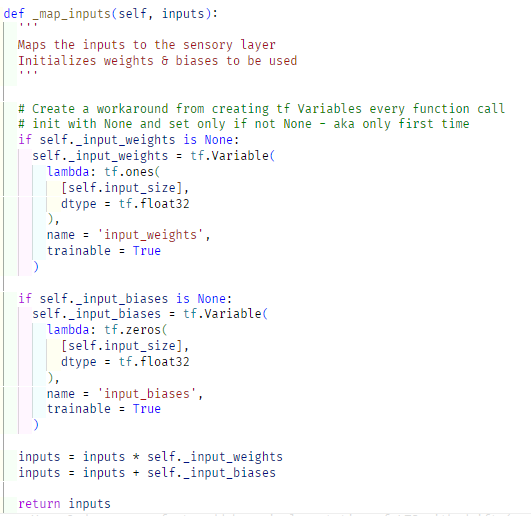


Figure 14: Algorithm – define weights and biases (*Self-Composed*)

As determined in previous chapters, the optimal way of performing the forward computation of SDEs is to use the Euler-Maruyama method. The below code snippet is an implementation of the Euler-Maruyama SDE solver used by the author utilizing Brownian motion as the noise, as demonstrated by Duvenaud (2021).

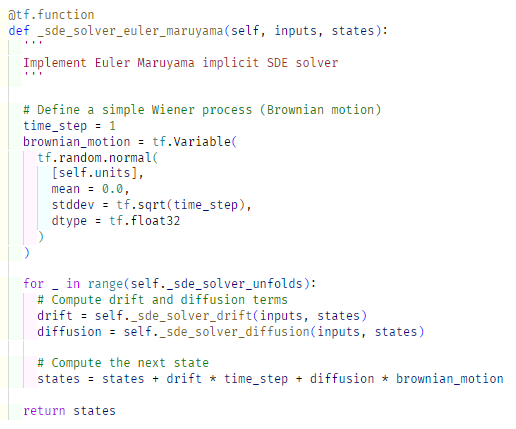


Figure 15: Algorithm – Euler-Maruyama SDE solver (*Self-Composed*)

## **4.3.2. Data fetchers**

The data fetchers are scripts that are used to extract the data to be used by the model. The scripts are placed under [**APPENDIX D.1**](#_D.1._Fetch_data).

## **4.3.3. Preprocessing**

Preprocessing steps are required to prepare the data fetched from the data fetchers before being used by the model. The preprocessing scripts are placed under [**APPENDIX D.2**](#_D.2._Preprocessing).

# **4.4. Chapter summary**

This chapter focused on defining the technologies and tools that facilitate the software development that would demonstrate the research. Additionally, the implementation of the core features is demonstrated with accompanying code snippets.

# **CHAPTER 05. CONCLUSION**

# **5.1. Chapter overview**

This chapter provides an initial conclusion to the research project focused on implementing the core components required to consider it a functional prototype. In detail, any deviations taken from the proposed scope and the schedule in the project proposal are mentioned. Moreover, any additional improvements required to produce an MVP alongside the current evaluation results are specified.

# **5.2. Deviations**

## **5.2.1. Scope related deviations**

The features in scope proposed in the project proposal are available in [**APPENDIX E.1**](#_E.1._Project_scope). Based on the proposed scope, no deviations have been taken.

## **5.2.2. Schedule related deviations**

The schedule proposed by the author is available in **APPENDIX E.2**. Based on the proposed Gantt chart, the author’s journey so far has not had any major deviations. However, a single task (no. 45) that mentions “implementing supplementary components” scheduled to be completed by January 23rd is still in progress. The progress of the Gantt chart with the updated dates provided is available in [**APPENDIX E.3**](#_E.3._Project_roadmap).

# **5.3. Initial test results**

# **5.4. Required improvements**

To consider this research successful, a couple of improvements are required.

* Enhance the performance of the system to the best possible accuracy – attempt more optimization procedures.
* Integrate the model in use to a GUI – GUI has been prepared; a simple Flask API should be created to establish a communication.
* Perform testing for each section of the application – conduct unit, performance, and integration testing.
* Compare the system’s performance with existing solutions.

# **5.5. Demo of the prototype**

A demo of the prototype was recorded and uploaded as an unlisted video on YouTube, the link can be found here.

# **5.6. Chapter summary**

This chapter provided the reader with an overview of the current status of the ongoing research project, including, but not limited to - deviations taken from the proposed features and schedule, the evaluation results, and any further improvements required.

# **REFERENCES**

S. Nakamoto, (2020). *Bitcoin: A peer-to-peer electronic cash system*. Available from <https://bitcoin.org/bitcoin.pdf> [Accessed 25 September 2022].

Rahouti, M., Xiong, K. and Ghani, N. (2018). Bitcoin Concepts, Threats, and Machine-Learning Security Solutions. *IEEE Access*, 6, 67189–67205. Available from <https://doi.org/10.1109/ACCESS.2018.2874539> [Accessed 25 September 2022].

Kervanci, I. sibel and Akay, F. (2020). Review on Bitcoin Price Prediction Using Machine Learning and Statistical Methods. *Sakarya University Journal of Computer and Information Sciences*. Available from <https://doi.org/10.35377/saucis.03.03.774276> [Accessed 25 September 2022].

Makridakis, S., Spiliotis, E. and Assimakopoulos, V. (2018a). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13 (3), e0194889. Available from <https://doi.org/10.1371/journal.pone.0194889> [Accessed 25 September 2022].

Makridakis, S., Spiliotis, E. and Assimakopoulos, V. (2018b). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34 (4), 802–808. Available from <https://doi.org/10.1016/j.ijforecast.2018.06.001> [Accessed 25 September 2022].

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36 (1), 75–85. Available from <https://doi.org/10.1016/j.ijforecast.2019.03.017> [Accessed 25 September 2022].

Hasani, R. et al. (2020). Liquid Time-constant Networks. Available from <https://doi.org/10.48550/arXiv.2006.04439> [Accessed 25 September 2022].

Chen, R.T.Q. et al. (2019). Neural Ordinary Differential Equations. Available from <https://doi.org/10.48550/arXiv.1806.07366> [Accessed 25 September 2022].

Poulopoulos, D. (2021). Is “Liquid” ML the answer to autonomous driving? *Medium*. Available from <https://towardsdatascience.com/is-liquid-ml-the-answer-to-autonomous-driving-bf2e899a9065> [Accessed 25 September 2022].

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9 (8), 1735–1780. Available from <https://doi.org/10.1162/neco.1997.9.8.1735> [Accessed 25 September 2022].

BI4ALL. (2021). Supervised Machine Learning in Time Series Forecasting. *BI4ALL – Turning Data Into Insights*. Available from <https://www.bi4all.pt/en/news/en-blog/supervised-machine-learning-in-time-series-forecasting/> [Accessed 12 October 2022].

Hasani, R. et al. (2021). Liquid Neural Networks. *YouTube*. Available from <https://www.youtube.com/watch?v=IlliqYiRhMU&t=350s>. [Accessed on 30 Sep. 2022].

Mozer, M.C., Kazakov, D. and Lindsey, R.V. (2017). Discrete Event, Continuous Time RNNs. Available from <https://doi.org/10.48550/ARXIV.1710.04110> [Accessed 14 October 2022].

Funahashi, K. and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6 (6), 801–806. Available from <https://doi.org/10.1016/S0893-6080(05)80125-X> [Accessed 14 October 2022].

A naive forecast is not necessarily bad. (2014). *The Business Forecasting Deal*. Available from <https://blogs.sas.com/content/forecasting/2014/04/30/a-naive-forecast-is-not-necessarily-bad/> [Accessed 15 October 2022].

Rubanova, Y., Chen, R.T.Q. and Duvenaud, D. (2019). Latent ODEs for Irregularly-Sampled Time Series. [Available from https://doi.org/10.48550/ARXIV.1907.03907](Available%20from%20https:/doi.org/10.48550/ARXIV.1907.03907) [Accessed 18 October 2022].

Duvenaud, D (2021). Directions in ML: Latent Stochastic Differential Equations: An Unexplored Model Class. *YouTube*. Available from <https://www.youtube.com/watch?v=6iEjF08xgBg>. [Accessed on 30 Sep. 2022].

Valencia, F., Gómez-Espinosa, A. and Valdés-Aguirre, B. (2019). Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning. *Entropy*, 21 (6), 589. Available from <https://doi.org/10.3390/e21060589> [Accessed 16 October 2022].

Serafini, G. et al. (2020). Sentiment-Driven Price Prediction of the Bitcoin based on Statistical and Deep Learning Approaches. *2020 International Joint Conference on Neural Networks (IJCNN)*. July 2020. Glasgow, United Kingdom: IEEE, 1–8. Available from <https://doi.org/10.1109/IJCNN48605.2020.9206704> [Accessed 16 October 2022].

Fleischer, J.P. et al. (2022). Time Series Analysis of Cryptocurrency Prices Using Long Short-Term Memory. *Algorithms*, 15 (7), 230. Available from <https://doi.org/10.3390/a15070230> [Accessed 26 September 2022].

Pontryagin, L.S., Boltyanskii, V.G., et al. (1962) The Mathematical Theory of Optimal Processes. Interscience Publishers John Wiley & Sons, Inc., New York-London, Translated from the Russian by K.N., Trirogoff.

Tzen, B. and Raginsky, M. (2019). Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit. Available from <http://arxiv.org/abs/1905.09883> [Accessed 2 December 2022].

Press, W.H. (ed.). (2007). *Numerical recipes: the art of scientific computing*, 3rd ed. Cambridge, UK ; New York: Cambridge University Press.

Lapicque, L. 1907. Recherches quantitatives sur l’excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generalej* 9: 620–635.

Li, X. et al. (2020). Scalable Gradients for Stochastic Differential Equations. Available from <http://arxiv.org/abs/2001.01328> [Accessed 18 January 2023].

Flask Vs Django: Which Python Framework to Choose? (2021). *InterviewBit*. Available from <https://www.interviewbit.com/blog/flask-vs-django/> [Accessed 12 December 2022].

Angular vs React | Angular vs Vue | React vs Vue - Know the Difference. (2021). *Radixweb*. Available from <https://radixweb.com/blog/angular-vs-react-vs-vue> [Accessed 12 December 2022].

PyTorch vs. TensorFlow: 2022 Deep Learning Comparison | Built In. (2022). Available from https://builtin.com/data-science/pytorch-vs-tensorflow [Accessed 12 December 2022].

# **APPENDIX A – INTRODUCTION**

# **A.1. Research questions**

**R**Q1: What are the top tier transformer architectures widely used and know for NLP problems related to text summarization?

**R**Q2: How can a pretrained transformer architecture be fine-tuned to get the optimal hyper parameters?

**R**Q3: What kind of evaluations should we perform after fine-tuning to filter out the best transformer architecture?

**R**Q4: How can domain generalization be integrated for system?

# **APPENDIX B – SRS**

# **B.1. Requirement elicitation methodologies**

Table 20: 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 |

# **B.2. Survey analysis**

Table 21: 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 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. | |
|  | |
| **Question** | Do you believe that developing a generic system for all domains would be a wise course of action? |
| **Aim of question** | Ensuring that developing a generic system would be beneficial in all domains |
| **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 | |
|  | |
| **Question** | Who do you think will most benefit from this system? |
| **Aim of question** | Getting to know about the thoughts of the participants about to whom the system would mostly benefit from? |
| **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. | |
|  | |
| **Question** | How much do you think that this system would benefit you? |
| **Aim of question** | Getting to know how much the system would benefit the general participants which are NOT domain specific |
| **Findings & Conclusion**  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 22: 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. |

# **B.3. Interview analysis**

Table 23: 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 | Since this is a project connected to data science, the availability of data and the data preparation methods to be used are the main concerns. PhD candidates suggested to make use of verified and well researched datasets for the area of generalization since every domain will be using the same model initially to start off with, therefore the quality of data should be considered, it was recommended to use datasets that have already been studied and utilized by other researchers since they have done so and verified their findings. NLP researches were concerned on the language of text the project scope is into when performing text preprocessing, since text data can also contain other language characters unless the project is scoped down to only English language supportive. |
| Transformer architectures | Most of the interviewees pointed out similar transformers architectures which they have used and found impressive results, which are mostly BERT, GPT-2, Roberta, T5 etc... where they have explored not only with text summarization but also when other NLP areas such as sentiment analysis, proving again that transformers are well known for solving NLP problems. They also stated to check up with the daily stats (most downloads and likes) about the transformer architectures from Hugging Face, this is because new better versions of the transformers are always been produced/updated. |
| Generalization | The Software Engineers and Architects suggested to make use of document-oriented NoSQL database management system to handling data storage for the domain specific managers, this is because its easily scalable and provider superior performance especially for the idea of adaptative generalization for this project. Such services are like MongoDB, Firebase NoSQL DB etc. |
| Research scope | The technology exports and research experts find that the solution of solving this problem using optimized transformers is great but they find that creating a generalized adaptive solution would be challenging with the time frame of the project but also advised to solve for the domain of movies first and then get into the others if time permits. |
| 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 | PhD candidates liked the idea of using hybrid transformer combination by using ensemble approaches to combine the top best two transformer architecture but it seems the scope of the project for the time frame is becoming bigger and riskier. |
| Custom transformers | The NLP researchers recommended to customize the existing transformer architecture instead of Hybrid model creation because of the project scope. |
| Prototype | The interviewees are interested to see how the generalization system for domain specific retraining is going to work together since they haven’t seen any such approach earlier from their experience. They also suggested if time permits to make use of a pretrained model to get the sentiment of the summary aswell to be displayed on the GUI. |
| 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 | The PhD candidates and NLP experts suggested the importance of evaluations when it comes to dealing with the adaptive generalization model since this can be used in any domain, therefore suggesting the author of the project to explore maximum of 3 domains when working with so its easier to compare the evaluation results else it will be confusing when demonstrating the work to anyone. |

Table 24: 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 |

# **B.4. Self-Evaluation (Competitor Analysis)**

Table 25: 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 | 🗶 | 🗶 | 🗶 | 🗶 | **✓** |

# **B.5. Use case descriptions**

Table 25: Use case description UC:07 (*Self-Composed*)

|  |  |
| --- | --- |
| Use case | Update model hyperparameters |
| Id | UC:07 |
| Description | Manually change the hyperparameters used by the model. |
| Actor | Admin |
| Supporting actor (if any) | None |
| Stakeholders (if any) | None |
| Pre-conditions | All the data must be scraped and preprocessed (as the model would ideally need to be retrained upon hyperparameter tuning). |
| Main flow | 1. Admin authorizes themselves. 2. Admin can change the hyperparameters in use to a set of predefined values. 3. The system ensures data available is up-to-date (must be in this case, as the script will run periodically automatically). If not:    1. Obtains the latest available data.    2. Performs sentiment analysis and self-retrains. 4. The system retrains itself with the data and new hyperparameters. |
| Alternative flows | None |
| Exceptional flows | None |
| Post-conditions | The model is updated with the chosen hyperparameters. |

# **B.6. Functional requirements**

Table 26: ‘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. |

Table 27: 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 C – DESIGN**

# **C.1. UI wireframes**

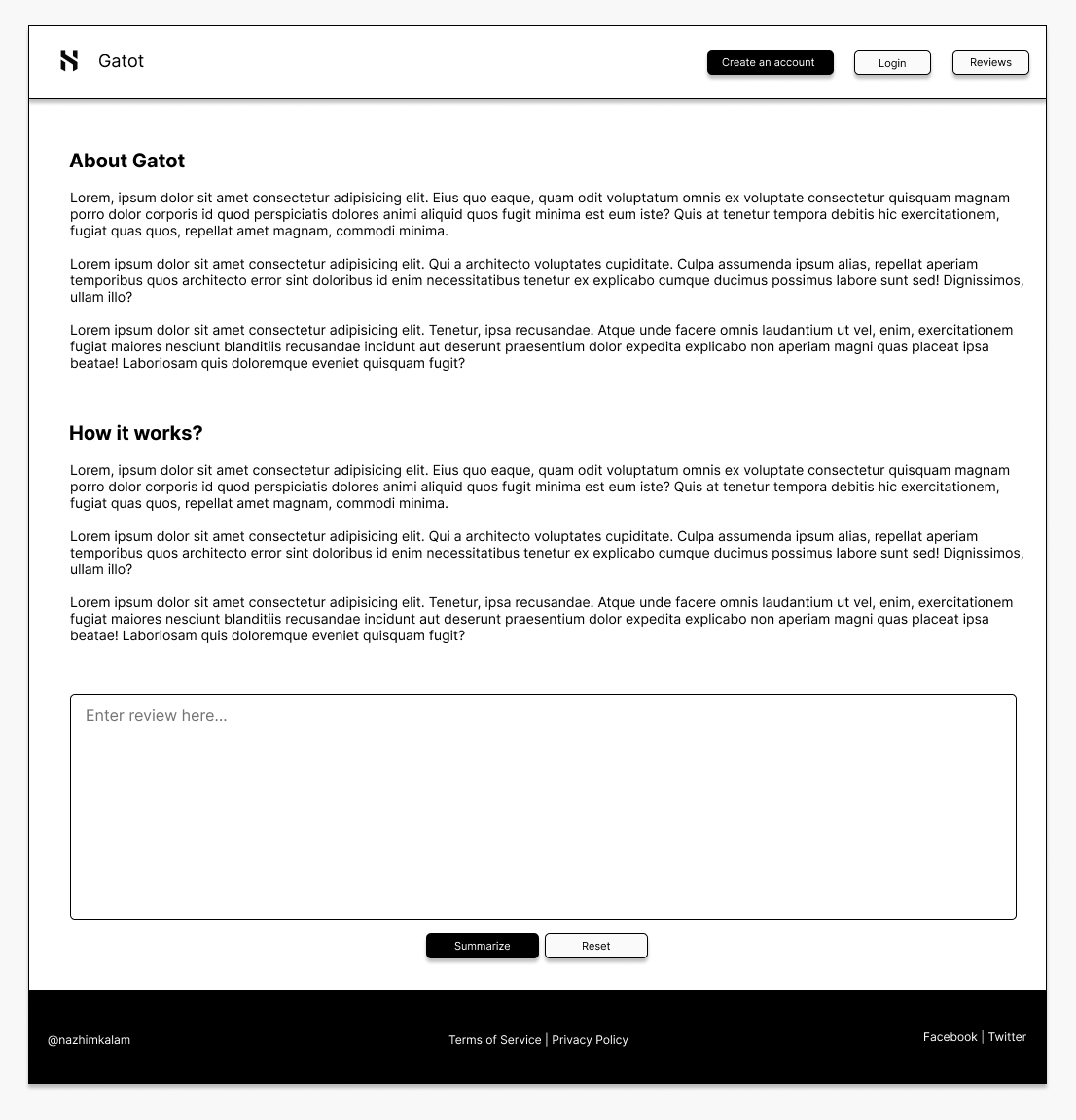


Figure 16: UI – Home page (*Self-Composed*)



Figure 17: UI – Login page (*Self-Composed*)

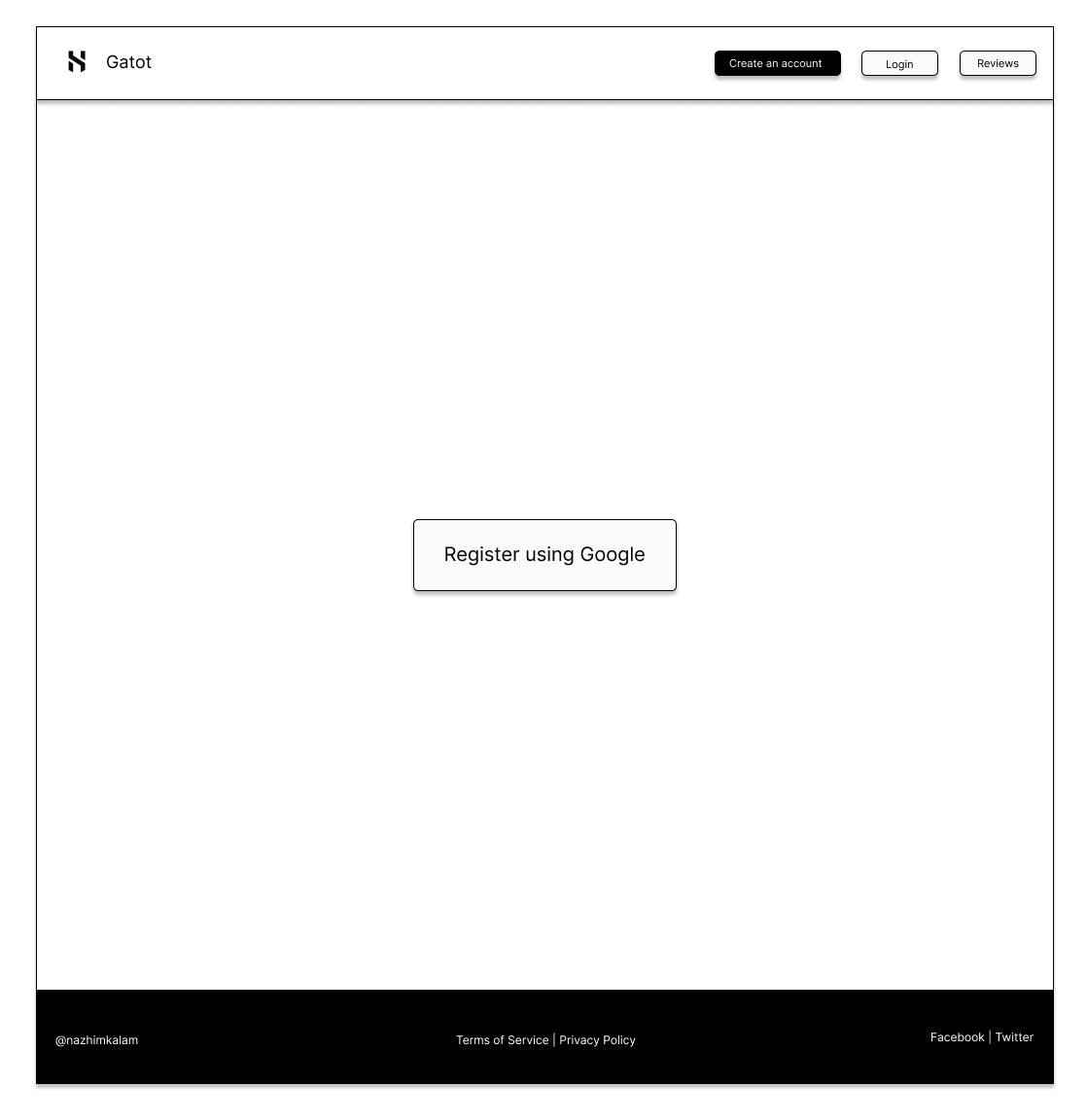


Figure 18: UI – Register page (*Self-Composed*)

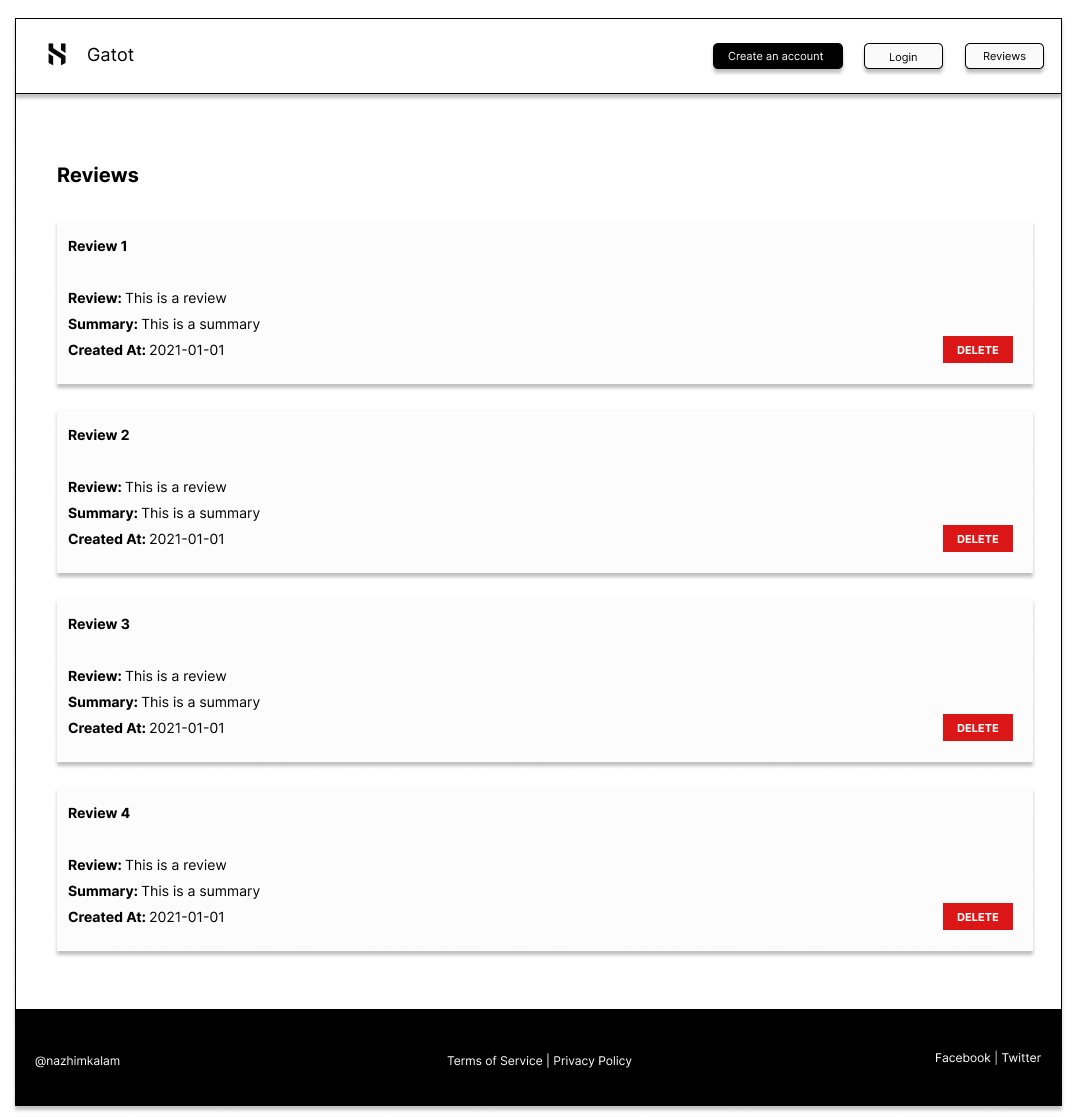


Figure 19: UI – Review history page (*Self-Composed*)

# **APPENDIX D – IMPLEMENTATION**

# **D.1. Fetch data**

**Fetch historical prices**



Figure 24: Fetch historical prices (*Self-Composed*)

The above script describes a couple of functions that can be used to fetch the latest BTC historical prices data and create a new updated CSV file that can be later read from by the model. A third-party API was used to fetch the data as existing APIs are all discontinued.

**Fetch Twitter volume & block reward size**

|  |  |
| --- | --- |
| Figure 25: Fetch Twitter volume (*Self-Composed*) | Figure 26: Fetch block reward size (*Self-Composed*) |

The above scripts fetch the Twitter volume and block reward, that were fetched from a website that exposes this data publicly. Therefore, a simple website scraping tool can be used without requiring any authentication or authorization.

**Fetch tweet data**

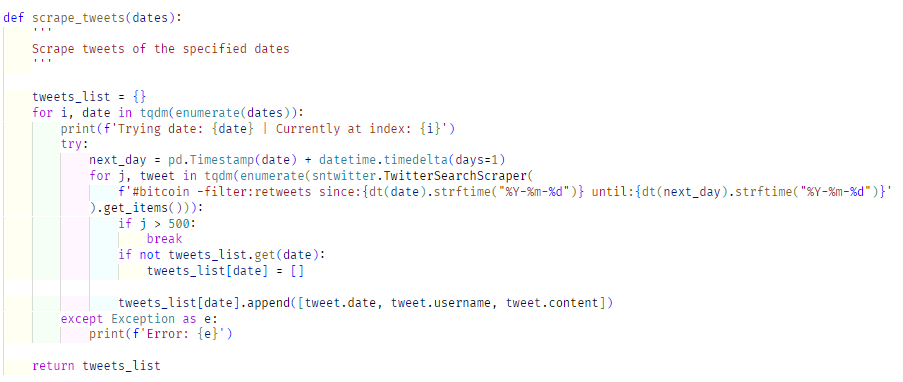


Figure 27: Scrape tweets (*Self-Composed*)

Obtaining the tweet data required a more tedious process as the Twitter API had been updated to only provide tweets for the past week. However, third-party libraries provide this functionality. Tweets fetched were limited to 500 for a single day due to time, performance, and storage constraints, and as the application is not the core contribution. Initially, tweets were fetched up to a specific time point; in future, the above script could be run to scrape tweets of specific dates that are described to be from the days that are currently existing in the data folder up to the day at which the script is run. There is a further limitation as only ‘#bitcoin’ is searched.



Figure 28: Clean tweets (*Self-Composed*)

As this research is currently limited to only English, the tweets are filtered and non-English tweets are removed.

**Fetch Google Trends**

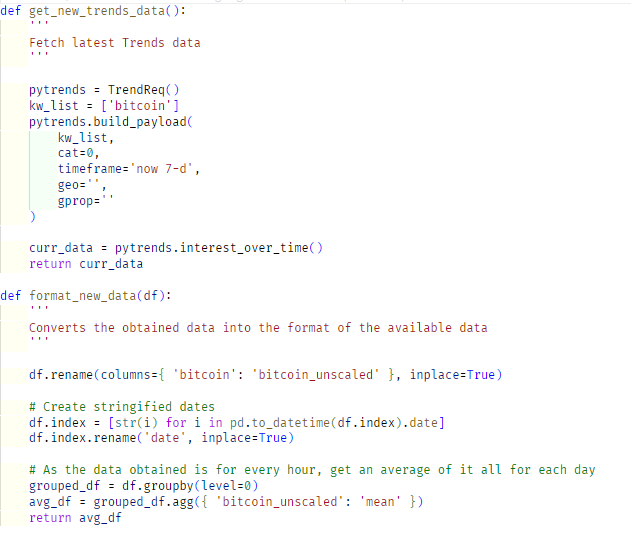


Figure 29: Fetch Google Trends (*Self-Composed*)

Fetching Google Trends data was also a relatively straightforward procedure, as Python exposes a library specifically for this purpose. However, rate-limitations had to be overcome by running the script multiple times for specific data ranges at a time rather than the entire history.

# **D.2. Preprocessing**

**Tweet sentiment analysis**

The main step of preprocessing is to perform sentiment analysis on the obtained tweet data. In this research, VADER sentiment analyzer is used as determined in previous chapters.

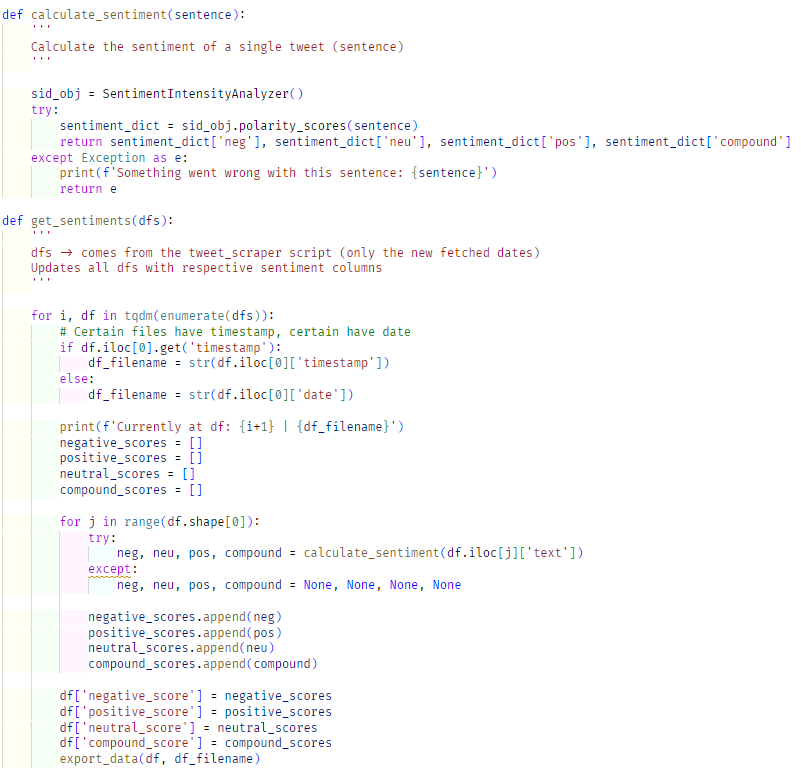


Figure 30: Analyze sentiments (*Self-Composed*)

The above script is used to perform sentiment analysis on the tweets and concatenates the negative, positive, neutral, and compound scores into the existing tweet dataset, which can then be condensed down to create an average score for a single day.

**Tweet dataset condensation**

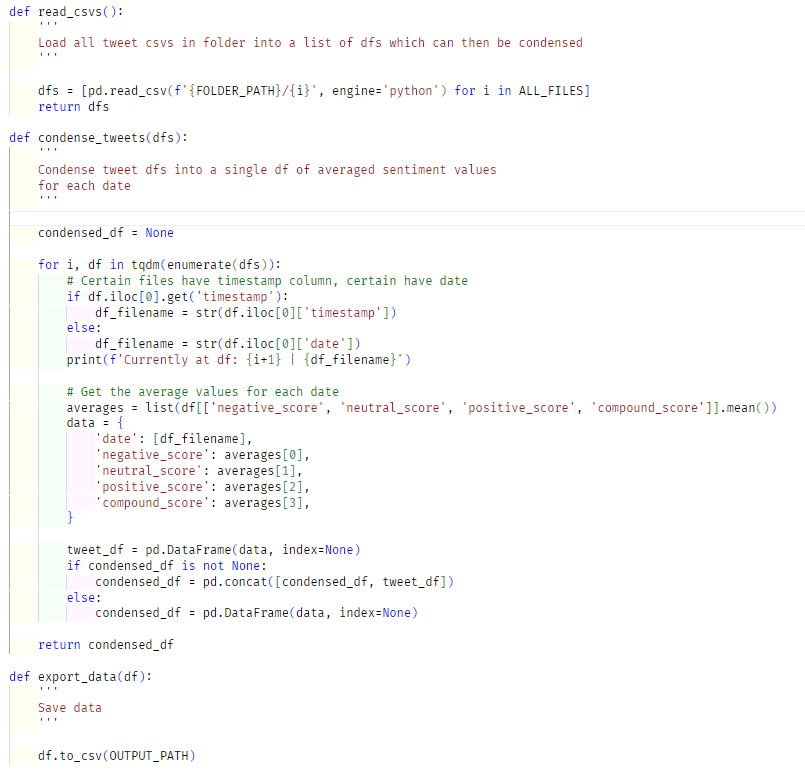


Figure 31: Combine and condense tweets (*Self-Composed*)

As the other data being used directly create a single CSV file with a row for each date, the condensation process is not required. However, as the tweet data fetched consists of a separate CSV file for each date, this data must be compressed to the same format as other datasets.

The above script condenses the tweet dataset into a single CSV file by averaging the sentiment scores for each day.

**Final dataset creation**

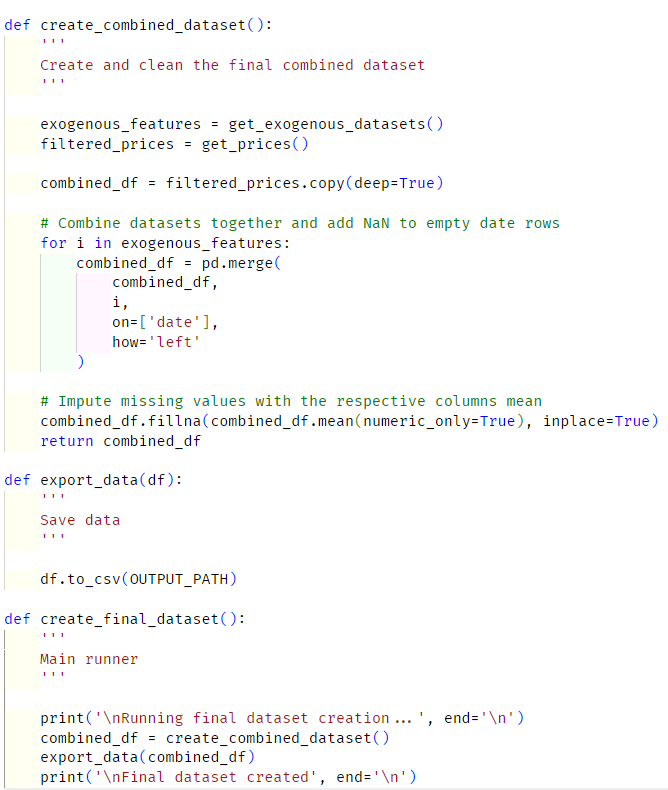


Figure 32: Combine all datasets (*Self-Composed*)

The above script is used to create the final dataset that is used by the model. It fetches all the datasets and combines them into a single data frame. Initially, a helper function is called that removes unneeded columns from the data files, which were decided upon conducting correlation tests. Missing values of each feature of specific dates are imputed by the mean of their respective columns. This combined dataset can then be saved so that the model can finally utilize it.

# **APPENDIX E – CONCLUSION**

# **E.1. Project scope**

**In scope**

* Implementing a novel LTC architecture capable of being used as currently existing solutions and the corresponding creation of a system.
* Periodical updates of the model with the latest available data.
* Evaluate and compare the implemented system against existing solutions to validate or invalidate hypothesis [**H**01](#myhypothesis).
* Ability to display a range of predictions for the chosen horizon.
* By combining them with the BTC historical data, consider Twitter sentiment, volume, and the ‘block reward size’ as external factors.

# **E.2. Project schedule**

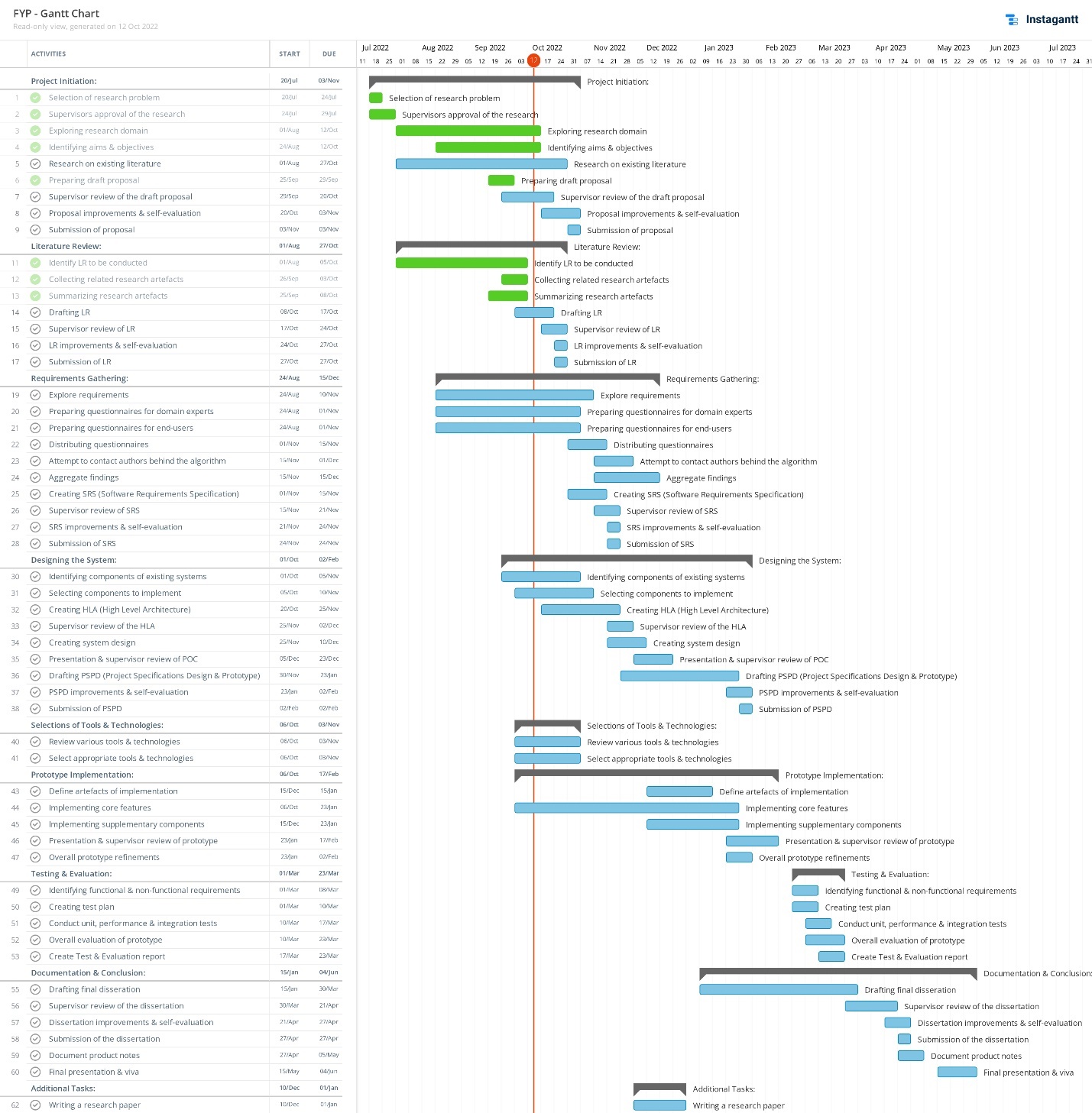
****

Figure 33: Initial Gantt chart (*Self-Composed*)

*A clearer version can be found* [*Here*](https://drive.google.com/file/d/1fpAf_W51Hc4CMBcM5A6T_b77BcfpDzYM/view?usp=sharing)

# **E.3. Project progress**

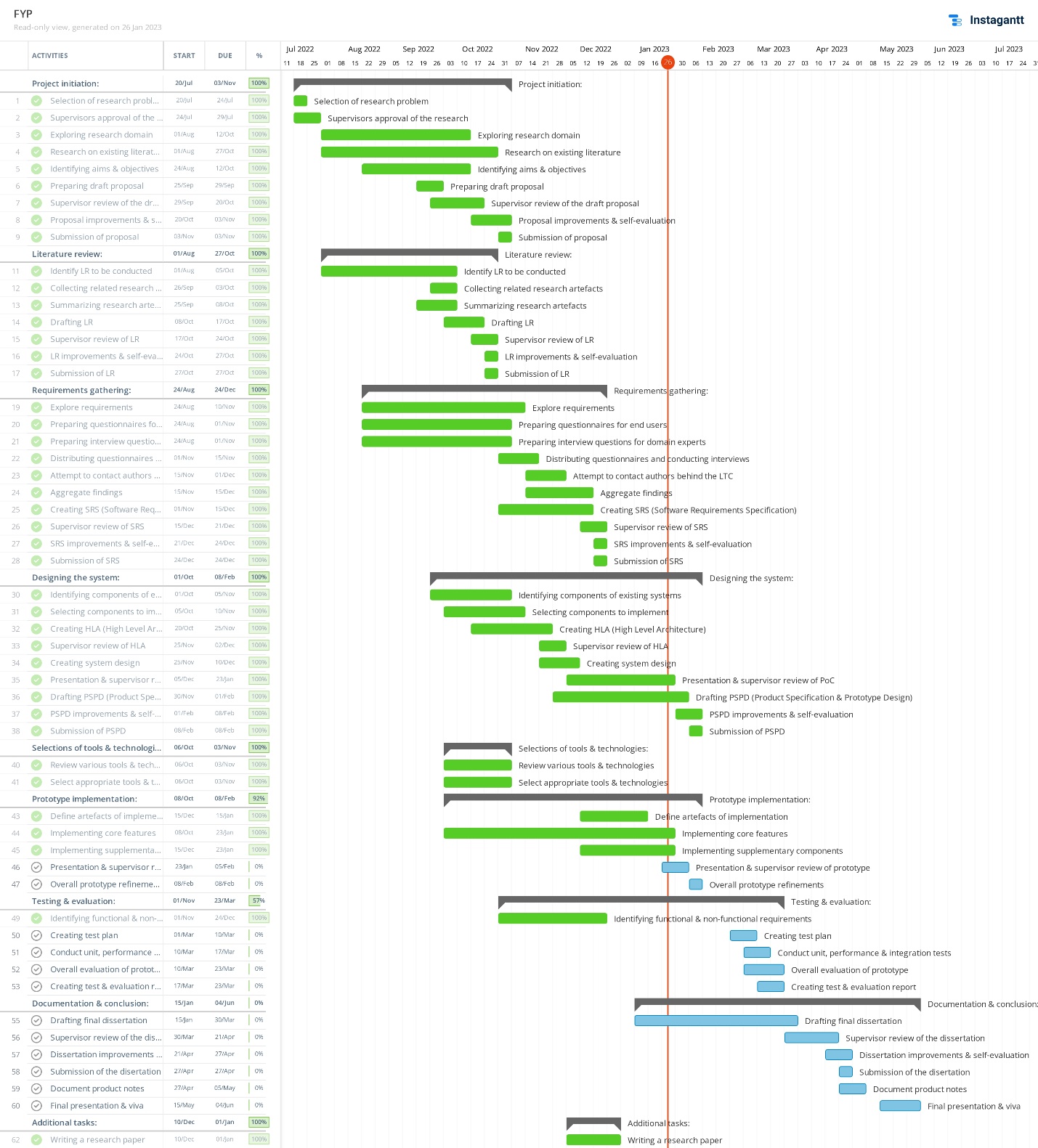


Figure 34: Current Gantt chart (*Self-Composed*)

*A clearer version can be found* [*Here*](https://drive.google.com/file/d/1hxyFfM2JPT-MGs1n7RSaJu0vGc9jKXvD/view?usp=sharing)