

INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with UNIVERSITY OF WESTMINSTER

**Detecting Code Quality Issues in AI-generated code: A deep learning approach**

A Product Specification, Design and Prototype by

Mr Muhammad Areeb Niyas

W1809939 / 20200129

Supervised by Mr Jihan Jeeth

Submitted in partial fulfilment of the requirements for the BSc(Hons) in Computer Science degree at the University of Westminster.

**Date: February 2024**

Table of Contents

[CHAPTER 01: PROBLEM 1](#_bookmark0)

* 1. [Chapter Overview 1](#_bookmark1)
  2. [Introduction 1](#_bookmark2)
  3. [Problem Background 1](#_bookmark3)
     1. [AI-generated Code 2](#_bookmark4)
     2. [Code Quality Issues 2](#_bookmark5)
     3. [Deep Learning 2](#_bookmark6)
  4. [Problem Definition 3](#_bookmark7)
     1. [Problem Statement 3](#_bookmark8)
  5. [Research Motivation 3](#_bookmark9)
  6. [Existing Work 4](#_bookmark10)
  7. [Research Gap 5](#_bookmark11)
  8. [Contribution to the Body of Knowledge 6](#_bookmark13)
     1. [Contribution to Research Domain 6](#_bookmark14)
     2. [Contribution to Problem Domain 6](#_bookmark15)
  9. [Research Challenge 7](#_bookmark16)
     1. [Research Questions 8](#_bookmark17)
  10. [Research Aim 8](#_bookmark18)
  11. [Research Objectives 8](#_bookmark19)

[1.12 Chapter Summary 11](#_bookmark20)

[CHAPTER 02: METHODOLOGY 12](#_bookmark22)

* 1. [Chapter Overview 12](#_bookmark23)
  2. [Research Methodology 12](#_bookmark24)
  3. [Development Methodology 13](#_bookmark26)
     1. [Design Methodology 13](#_bookmark27)
     2. [Evaluation Methodology 13](#_bookmark28)
     3. [Solution Methodology 14](#_bookmark29)
  4. [Project Management Methodology 15](#_bookmark31)
     1. [Project Scope 15](#_bookmark32)
        1. [In-scope 15](#_bookmark33)
        2. [Out-scope 15](#_bookmark34)
     2. [Schedule 15](#_bookmark35)
        1. [Gantt Chart 16](#_bookmark36)
        2. [Deliverables 17](#_bookmark38)
     3. [Resource Requirements 17](#_bookmark40)
        1. [Hardware Requirements 17](#_bookmark41)
        2. [Software Requirements 18](#_bookmark42)
        3. [Data Requirements 18](#_bookmark43)

[2.4.3.3 Skill Requirements 18](#_bookmark44)

* + 1. [Risks and Mitigation 19](#_bookmark45)
  1. [Chapter Summary 19](#_bookmark47)

[References i](#_bookmark48)

## List of Figures

[Figure 1: Workflow diagram (Self-Composed) 14](#_bookmark30)

[Figure 2: Gantt chart 16](#_bookmark37)

**List of Tables**

[Table 1: Related work 5](#_bookmark12)

[Table 2: Research objectives 11](#_bookmark21)

[Table 3: Research methodologies 13](#_bookmark25)

[Table 4: Deliverables 17](#_bookmark39)

[Table 5: Risks and mitigation 19](#_bookmark46)

**List of Abbreviations**

**AI** Artificial Intelligence

**DL** Deep Learning

**ML** Machine Learning

**ANN** Artificial Neural Networks

**RNN** Recurrent Neural Networks

**CNN** Convolutional Neural Networks

**LLM** Large Language Model

# CHAPTER 01: PROBLEM

## Chapter Overview

This chapter provides an overview of the project that has been undertaken by the author. First, the author will be diving into the project background in detail and discuss the problem. Then the existing work and its limitations will be discussed which leads to the research motivation along with the challenges faced, aims and objectives of this research project.

## Introduction

In this research project, the author tries to identify a novel deep learning approach for the code quality issues that can be highlighted in AI-Generated code and to introduce a novel system to detect these issues prior to a developer’s implementation.

## Problem Background

The current tech era is rapidly evolving, especially with generative AI. This is mainly because of the potential and ability of AI to carry out tasks that require high skill more reliably and quickly than humans (Harding et al., 2023).

Code generation is being widely used and spread with many generative tools which can be accessed and integrated seamlessly into the daily lives of a programmer. With the widespread and ease of use due to the outstanding performance, code generating models are released into the market swiftly in this era (Barke, James and Polikarpova, 2023). Therefore, the increase in AI- generated code in the industry gives rise to questions about the nature and quality of AI-assisted programming. Especially when the future is headed towards; the majority of code in the world being AI-generated.

### AI-generated Code

ChatGPT and generative AI experienced a quick rise in daily use and popularity since its release in November 2022. Consequently, it has helped a lot of people with its amazing ability with human-like responses. The GPT 3.5 has shown significant potential for redefining a number of research domains including code generation.

The productivity of developers increases with automatic code generation by producing or completing the source/executable code automatically based on programming patterns or specifications.

However, issues concerning potential hazards are prevalent with the widespread and adoption of AI-Generated code. This is due to the lack of studies which formally investigate the reliability and quality of the code generated (Y Liu et al., 2023).

### Code Quality Issues

AI-generated code especially ChatGPT commonly contains quality issues such as incorrect outputs, underperforming, maintainability and compilation errors (Y Liu et al., 2023). While many existing tools attempt to solve code quality issues and highlight common mistakes, current code quality tools in the industry do not use datasets from AI-generated code and code quality tools have not been built to target AI-generated code. Some common code quality issues include: syntax errors, duplicates, unsecure code, incorrect output and overly complex code.

### Deep Learning

Machine Learning (ML) is a vital component in today’s world of artificial intelligence which uses algorithms to imitate and find patterns in data, similar to how humans learn. Deep Learning (DL) is a subset of machine learning which uses multi layered neural networks which ideally tries to imitate the way a human brain functions. It has become very popular over the years due to the increase and widespread of big data and performing better than traditional ML techniques (Alzubaidi et al., 2021).

While machine learning methods have been used extensively to identify issues in code, recent research using deep learning methods, such as convolutional neural networks (CNNs), artificial neural networks (ANNs), and recurrent neural networks (RNNs), has shown a higher accuracy rate

in detecting code smells when compared to traditional ML methods (Ho et al., 2023). However, code quality detectors targeting AI-generated code or built on top of less natural language text LLMs to avoid biases are not available. So, the author mainly considers using deep learning approaches to identify code quality issues in AI-generated code in this research as the risk of attacks and maintainability of code is a rising concern in the near future as developers enter an era of AI-generated code.

## Problem Definition

There is a gap between an experience coder compared to generated code and current LLMs cannot entirely replace professional developers. In a prominent level, LLMs currently have a gap in identifying requirements and context of a problem clearly. (Wu, 2023).

AI-generated code is frequently used in modern software development to automate repetitive operations and increase productivity. However, AI-generated code may experience a number of quality problems that result in bugs, inefficiencies, and difficulties (Y Liu et al., 2023). Therefore, consequently there is an increasing demand for automated tools that can evaluate the quality of code produced by AI models like ChatGPT.

A deep learning approach that provides a solution for the breakdown of above-mentioned problems is required to ensure a novelty prototype. This study therefore would aim to create a tool that can be used by developers seamlessly to identify code quality issues in AI-generated code prior to deployment.

### Problem Statement

The majority of code written in the future will be AI-generated but there are code quality issues in such code and the current tools are not accurate analyzers therefore new novel systems are required to detect code quality issues catered to AI-specific code.

## Research Motivation

The author believes that in the near future, the majority of code will be AI-generated. Current existing code analysis tools and detection systems are mainly built on top of human-written code and current tools are focused on detecting issues in the nature of humans.

However, with the evolution of AI the industry needs to prepare for the future that is within reach. Therefore, adapting to a forthcoming era of AI. The author believes the proposed research of using deep learning to detect code quality issues in AI-generated code will be a novelty and stepping- stone in the future of code analysis and code quality issues detection.

## Existing Work

|  |  |  |  |
| --- | --- | --- | --- |
| Citation | Brief Description | Limitations | Contribution |
| (Ho et al., 2023) | Uses a 3-method approach using CNN, LSTM and neural network with weighted loss function to accurately detect code smells. | Can be further improved using pre- trained models.  Does not use AI- generated code in dataset. | Provide a solution using different deep learning techniques to detect code smells which is effective. |
| (Birillo et al., 2023) | Proposed an algorithm to detect code quality issues in online programming tasks and pre- written templates which is a manual validation process targeting learning platforms. | Manual validator to label issues during training so there could be biases. A ML or DL approach would be more accurate.  Does not look into AI-generated code. | First implementation to detect code quality issues targeting educational and learning platforms/ |
| (Y Liu et al., 2023) | Provides an insight into  ChatGPT’s limitations and identifies issues with quality of generated code. Produces a roadmap for future work in AI- generated code. | There is no tool specific to check and verify code quality issues in AI- generated code. | Study proves that ChatGPT generated code often include code quality issues such as compilation errors, incorrect outputs and maintainability problems along with hindered |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | performances. |
| (Z Liu et al., 2023) | Provides an analysis of code generation from ChatGPT. But ChatGPT can not directly fix its issues and in most scenarios, it has vulnerabilities so suggests multi round fixing. | ChatGPT is evolving so findings can be soon outdated.  May not be generalizable to other LLMS. | Uncover potential issues and problems in code generated by ChatGPT and provide a n insight into improving code generation techniques. |
| (Das, Yadav and Dhal, 2019) | Provides a deep learning based solution to detect two types of code smells (Brain Class and Brain Method) | There is not much studies that investigate the use of deep learning techniques to detect code smells.  May perform poorly with AI-generated code. | Accurately were able to detect two types of code smells, namely Brain Class and Brain Method code smells using deep learning. |

## Research Gap

*Table 1: Related work*

Current code analysis tools analyze code against a pre-set of coding rules and validates according to given guidelines. This helps identify subtle defects or vulnerabilities. Low quality code can cause many problems in software development due to maintainability problems, security issues and inefficient code. Therefore, developers need to pay attention and take precautions about code quality prior to deployment (Vable, Diehl and Glymour, 2021). Recently, LLMs which contain much more natural language text than executable code (industry standard code) is used in code generation systems. This leads to biases which can reduce the quality of generated code (Mouselinos, Malinowski and Michalewski, 2023).

While state-of-the-art code generation tools such as ChatGPT can generate effective code, the AI- generated code commonly contains quality issues such as incorrect outputs, underperforming, maintainability and compilation errors (Y Liu et al., 2023). So, code quality issues in AI-generated code is a critical issue in the field of code generation and no studies have been conducted to detect code quality issues specific to AI-generated code. Thus, a deep learning approach can be developed to fill this gap.

## Contribution to the Body of Knowledge

The contributions to the body of knowledge will be achieved by creating a novel solution which is a deep learning-based model to identify code quality issues in AI-generated code before the code goes into deployment.

### Contribution to Research Domain

While machine learning (ML) methods have been used extensively to identify issues in code, recent research using deep learning methods, such as convolutional neural networks (CNNs), artificial neural networks (ANNs), and recurrent neural networks (RNNs), has shown a higher accuracy rate in detecting code smells and code quality issues when compared to traditional ML methods (Ho et al., 2023). Therefore, it is proposed to explore a deep learning based approach in order to identify code quality issues accurately and precisely targeting AI-generated code.

### Contribution to Problem Domain

Currently available solutions are primarily focused and built on top of code which is not AI- generated. Majority of the code quality tools developed still have limitations and constraints in their ability (Das, Yadav and Dhal, 2019). Therefore, a tool that highlights code quality issues targeting AI-generated code that hasn’t been done before in previous research will be investigated so that developers can detect such issues prior to deployment.

## Research Challenge

This research project’s primary objective is to improve the accuracy and widespread use of deep learning techniques in code quality tools while overcoming the limitations in literature in this new code generation domain. The following is a list of research challenges:

1. **Finding research materials** – Due to the constant changes in the code generation domain it will be highly difficult to stay up to date on new case studies and constant changes. New techniques for data preparation should also be incorporated in order to raise the standard of publicly available datasets (Sharma, Sinha and Sharma, 2022). It is highly difficult to gather datasets that would provide successful results in the domain of using deep learning to detect code quality issues (Raychev, 2021).
2. **Developing and improving accuracy of deep learning model** – Using deep learning is a new approach in the domain of code quality. Even though machine learning has great potential in solving the issue of code quality, there has not been many successful outcomes as of yet (Raychev, 2021). The volume of data that specific ML algorithms can handle as well as specific data type limitations that some ML algorithms impose is a key challenge (Juddoo and George, 2020).
3. **Code quality analysis for AI-generated code** – At the point of initiation of this research project, there is no research that standardly investigates the quality of code and reliability of AI-generated code (Y Liu et al., 2023). Current static analysis tools in the industry are built using a variety of different object oriented metrics and templates that check against a preset of coding rules and guidelines .

According to these challenges, it will be complex to construct a prototype that uses artificial intelligence to detect issues in code quality in AI-generated code.

### Research Questions

**RQ1**: How can a DL approach be used to detect code quality issues in AI-generated code?

**RQ2**: How effective would a DL approach be in detecting code quality issues in AI-generated code compared to other tools?

**RQ3**: What are the latest advancements in deep learning that can be used to perform code analysis?

## Research Aim

*The aim of this research is to design, develop and evaluate a novel analysis tool that will provide relevant, up-to-date, and accurate highlighting of code quality issues in AI-generated code by using machine learning and deep learning techniques.*

To further elaborate, this research project will aim to provide a system that can be used by developers where they can input the AI-generated code and test for any code quality issues in the generated code. The use of data mining techniques, deep learning (DL) methods, data analysis, machine learning techniques will be researched to make the best possible analysis.

Further investigation to provide a prompt based on feedback from the analysis tool will also be explored. The required knowledge and expertise will be researched and the tool will be thoroughly tested to evaluate the accuracy.

## Research Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| Research Objectives | Explanation | Learning Outcome | Research Questions |
| Problem Identification | Carry out thorough preliminary research.  **RO1**: To conduct thorough research on previous work to get a proper understanding of the problem. | LO1, LO2 | RQ1, RQ3 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **RO2**: To evaluate current methodologies used to solve the problem. |  |  |
| Literature Review | Do an in-depth study on existing solutions and how deep learning and machine learning can be used to build a more accurate system.  **RO3**: To learn about current existing analysis tools and solutions.  **RO4**: To compare and evaluate current tools with AI-Generated code.  **RO5**: To discover limitations of existing analysis tools. | LO1, LO4, LO8 | RQ1, RQ2, RQ3 |
| Data Gathering and Analysis | **RO6:** To gather information about the need for analysis tools for AI-Generated code.  **RO7:** To collect requirements for user expectations for an analysis system that would detect code quality issues.  **RO8:** To get feedback and opinions from domain experts to build a good overall system. | LO2, LO3 | RQ1, RQ2, RQ3 |
| Research Design | **RO9:** To state and identify the design objectives. | LO1, LO5,  LO8 | RQ1, RQ3 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R10:** To create a high-level architecture diagram to analyze the structure.  **R11:** To create a data flow diagram to evaluate the flow. |  |  |
| Implementat ion | **R12:** To select the most suited technologies and tools for implementation.  **R13:** To identify and create an accurate model that would detect multiple code quality issues in AI-generated code.  **R14:** To make a user interface that would have good UX and integrate the core functionality | LO1, LO5, LO7, LO8 | RQ1, RQ2, RQ3 |
| Testing and Evaluation | **R15:** To create appropriate test plans for the user interface and the model.  **R16:** To utilize appropriate methods, evaluate the model and the user interface and document any limitations | LO5, LO8 | RQ2 |
| Publish Findings | Create well-organized documentation, reports, and articles that analyze the research findings critically.  **R17**: To publish a research paper based on findings. | LO4, LO8 | RQ1, RQ2, RQ3 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R18**: To publish analysis and test results identified through the research.  **R19**: To make the code or model  open source so that further research may be pursued. |  |  |

## 1.12 Chapter Summary

*Table 2: Research objectives*

This chapter covered a comprehensive overview of the problem domain and background of the proposed solution which is to build a novel code analysis tool catered to address issues in AI- generated code. The research gap, aims and objectives being addressed along with its upcoming challenges were discussed in detail. The methodologies used will be discussed in the following chapter.

# CHAPTER 02: METHODOLOGY

## Chapter Overview

This chapter will cover the research, design, development, evaluation, solution and project management methodologies chosen by the author for this research project along with its available options and justifications as to why the author chose particular methodologies. Additionally, the project scopes will be discussed along with the planned schedule and deliverables. Later in the chapter, the hardware, software, data and skill requirements required for the project will be explored along with the risks and mitigations to be anticipated.

## Research Methodology

|  |  |
| --- | --- |
| Research Philosophy | Out of the 4 research philosophies: positivism, interpretivism, pragmatism and realism, **Pragmatism** was chosen. This is because the author will evaluate both quantitative and qualitative methodologies to experiment and determine which would produce best performance in achieving the research goal. |
| Research Approach | As this research aims to comprehensively apply a variety of theories and evaluate the hypothetical statement – the **deductive** approach was chosen out of the two available research approaches which are inductive and deductive. |
| Research Choice | From the given options: mono method, multi method and mixed method  – the **mixed method** was picked. This is because both methods – qualitative and quantitative are used through the intended use of interviews, experiments and surveys to collect data throughout the research process and evaluate. |
| Research Strategy | **Surveys**, **experiments** and **interviews** will be conducted amongst  potential candidates based on performance and evaluation metrics. The |

|  |  |
| --- | --- |
|  | research strategy would help address and provide insights into answering the research questions of the project. |
| Time Horizon | Amongst the available time horizons namely: longitudinal and cross- sectional, the **cross-sectional** approach was chosen because evaluations and data collection would be carried out on one point in time during the evaluation phase of the project. |
| Data Collection and Analysis | Interviews, surveys, trial and error observations along with multiple other techniques would be used in order to collect and analyze data. Also, insights gained from similar work and literature will be applied. |

*Table 3: Research methodologies*

## Development Methodology

The **prototype** development methodology was chosen amongst the various software lifecycle development methodologies because this project aims to develop, design and evaluate the prototype as needed and accordingly to new findings and evaluations in the industry as it’s an emerging and rapidly growing research area.

### Design Methodology

**Object Oriented Analysis and Design** (OOAD), out of the available design methodologies, was chosen. In order to adapt to frequent changes and updates in the industry, especially with LLMs and AI-generated code, new information will be gathered and updated regularly throughout this research project. Therefore, in order to accommodate the need for constant changes and requirements, this method was chosen for its adaptable and iterative aspects.

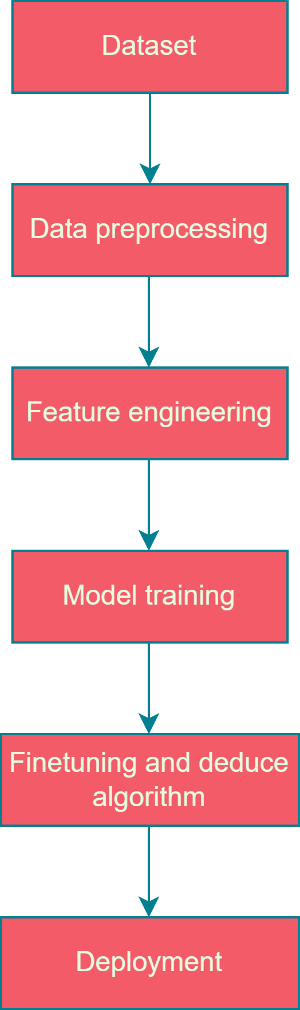
### Evaluation Methodology

The following evaluation metrics were chosen by the author from the deep learning and code analysis domain in order to quantitively evaluate the performance:

* + - * Precision and Recall
      * F1 Score
      * Classification Accuracy
      * ROC curve and AOC

### Solution Methodology

The dataset that contains only AI-generated code needs to be gathered during the primary and secondary data collection process in order to identify code quality issues specific to AI-generated code. Then the data should move into the data processing stage where the data cleaning and handling of missing data is done. After the feature engineering and model training, the algorithm is chosen after fine tuning through evaluation, the model is then deployed.



*Figure 1: Workflow diagram (Self-Composed)*

## Project Management Methodology

From the available options **Agile Prince 2** methodology was chosen for project management. Agile Prince 2 allows the author to be more open and flexible to improvements and swift changes which is a crucial aspect to the execution of this research project.

* + 1. **Project Scope**

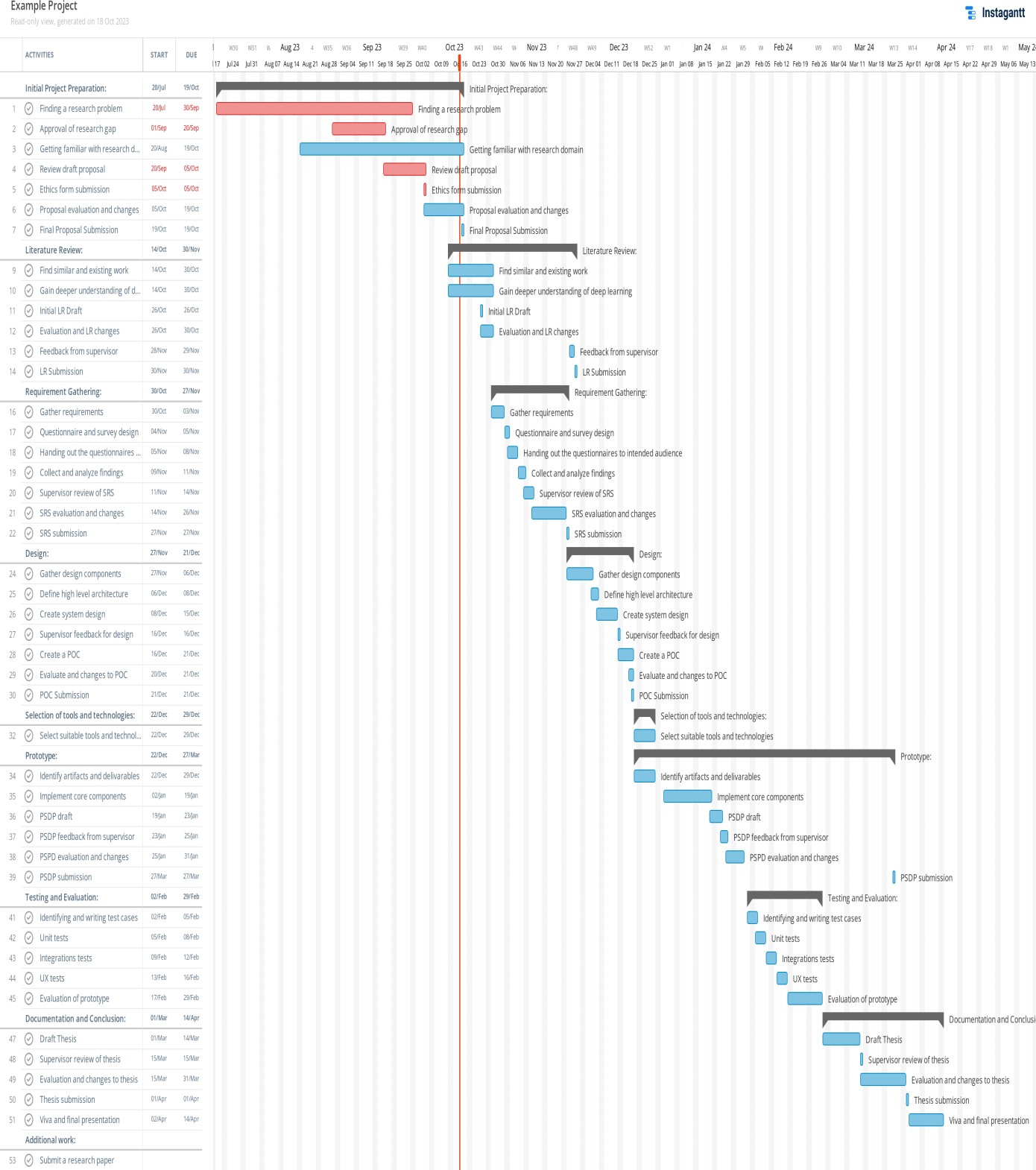
### In-scope

* + - * + Developing an algorithm to detect code quality issues in AI-generated code.
        + A method to classify different types of code quality issues.
        + A web interface to display and evaluate the proposed deep learning solution.

### Out-scope

* + - * + The proposed solution will only detect code quality issues.
        + Built for only limited programming languages.
        + Produce a prompt that would automatically resolve the issues in the code.
    1. **Schedule**

### Gantt Chart



*Figure 2: Gantt chart*

### Deliverables

|  |  |
| --- | --- |
| **Deliverable** | **Date** |
| Final Project Proposal | 19th October 2023 |
| Literature Review  Thorough analysis and critical evaluation of existing and related work in the domain. | 30th October 2023 |
| System Requirement Specification  Specifies the requirements to be met by the proposed prototype. | 27th November 2023 |
| Proof of Concept Submission  An initial draft implementation of the proposed solution. | 21st December 2023 |
| Final Project Specifications Design and Prototype (PSDP)  Documentation and prototype of the implemented solution along with its core features. | 1st February 2024 |
| Minimum Viable Product Submission | 7th March 2024 |
| Final Thesis Submission | 1st April 2024 |

*Table 4: Deliverables*

* + 1. **Resource Requirements**

In order to meet the expectations and requirements of the aforementioned methodologies and data gathering techniques, the skills, software, hardware and data requirements are as follows:

### Hardware Requirements

* + - * + **Intel I5 or I7 CPU** – To handle high processing power to train and test deep learning models.
        + **8GB RAM or above** – To control and manage huge datasets
        + **50GB free disk space** – To store datasets and models prior and post training along with application data.

If hardware requirements are not met or sufficient, the recommended approach of using cloud technologies such as google collab would be used.

### Software Requirements

* + - * + **Python** – Efficient to use for machine learning and deep learning projects as it allows effective training of models.
        + **React** – To build the front end of the proposed prototype and provide good user experience.
        + **Zotero** – To organize, build and manage research papers and citations.
        + **GitHub** – Sufficient and organized version control.
        + **Google Drive** – To keep regular backups of the project.
        + **MS Word/Google Docs** – To provide clear documentation.
        + **Jupiter Notebook/PyCharm IDE** – To develop and maintain code sufficiently.
        + **Figma** – To create and design UI prototypes to display the prototype.

### Data Requirements

In order to build a successful model, datasets that contain AI-generated code would be required which is readily available from Codex and figshare. This can be used to test and evaluate the proposed prototype.

### Skill Requirements

* + - * + Machine Learning/Deep Learning expertise – A thorough understanding of ML and DL techniques and theory are required. Domain knowledge of code analysis and its link to ML and DL is also a vital aspect for this research project.
        + Ability to design user friendly interfaces to display the prototype.
        + Presentation skills to pitch and present the project adequately.
        + Documentation skills to create and maintain proper standard documentation for the prototype and provide the project deliverables.
    1. **Risks and Mitigation**

|  |  |  |  |
| --- | --- | --- | --- |
| Risk Item | Severity | Frequency | Mitigation Plan |
| No sufficient knowledge on technical and domain areas. | 5 | 5 | Read and gain insights on domain area from experts. |
| Losing documentation, data and code. | 4 | 2 | Take regular backups and maintain a GitHub repository. |
| Lack of hardware resources | 4 | 3 | Switch to cloud-based solutions and development environments such as Google Colab. |
| Project requirements changing due to the project being a part of a rapidly growing domain. | 4 | 5 | Stay updated on the latest changes by staying in touch with domain experts and latest research papers. |
| Hypothesis turning out to be contradicting or invalid. | 4 | 3 | Continue and keep exploring the research field as any strong output would be a contribution. |

*Table 5: Risks and mitigation*

## Chapter Summary

In this chapter, the methodologies followed by the author for this project were discussed. Diving into the research methodologies and the prototype development along with the OOAD design approaches were chosen amongst the available options. The evaluation metrics used for assessing the project were outlined. Moreover, the solution methodology used for a DL approach was introduced with the integration of Agile Prince-2 project management with the justifications as to why they were chosen. Additionally, this chapter explored the project scopes covered, schedules, deliverables, required skills, data and resource needs and the risks and mitigations that should be anticipate.

# References

Alzubaidi, L. et al. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8 (1), 53. Available from https://doi.org/10.1186/s40537-021-00444-8.

Barke, S., James, M.B. and Polikarpova, N. (2023). Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proceedings of the ACM on Programming Languages*, 7 (OOPSLA1), 85–111. Available from https://doi.org/10.1145/3586030.

Birillo, A. et al. (2023). Detecting Code Quality Issues in Pre-written Templates of Programming Tasks in Online Courses. *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, 152–158. Available from https://doi.org/10.1145/3587102.3588800.

Das, A.K., Yadav, S. and Dhal, S. (2019). Detecting Code Smells using Deep Learning. *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*. October 2019. Kochi, India: IEEE, 2081–2086. Available from https://doi.org/10.1109/TENCON.2019.8929628 [Accessed 19 October 2023].

Harding, J. et al. (2023). AI language models cannot replace human research participants. *AI & SOCIETY*. Available from https://doi.org/10.1007/s00146- 023-01725-x [Accessed 3 October 2023].

Ho, A. et al. (2023). Fusion of deep convolutional and LSTM recurrent neural networks for automated detection of code smells. *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*. 14 June 2023. Oulu Finland: ACM, 229–234. Available from https://doi.org/10.1145/3593434.3593476 [Accessed 3 October 2023].

Juddoo, S. and George, C. (2020). A Qualitative Assessment of Machine Learning Support for Detecting Data Completeness and Accuracy Issues to Improve Data Analytics in Big Data for the Healthcare Industry. *2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM)*. 25 November 2020. Balaclava, Mauritius: IEEE, 58–

66. Available from https://doi.org/10.1109/ELECOM49001.2020.9297009 [Accessed 19 October 2023].

Liu, Y. et al. (2023). Refining ChatGPT-Generated Code: Characterizing and Mitigating Code Quality Issues. *ArXiv*. Available from https:/[/www.semanticscholar.org/paper/Refining](http://www.semanticscholar.org/paper/Refining-ChatGPT-Generated-)-[ChatGPT-Generated-](http://www.semanticscholar.org/paper/Refining-ChatGPT-Generated-) Code%3A-Characterizing-and-Liu-Le- Cong/41a2e7c079179ae94557d3198de674a16a5987a6 [Accessed 31 August 2023].

Liu, Z. et al. (2023). No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT. Available from https://doi.org/10.48550/ARXIV.2308.04838 [Accessed 31 August 2023].

Mouselinos, S., Malinowski, M. and Michalewski, H. (2023). A Simple, Yet Effective Approach to Finding Biases in Code Generation. Available from <http://arxiv.org/abs/2211.00609> [Accessed 21 September 2023].

Raychev, V. (2021). Learning to Find Bugs and Code Quality Problems - What Worked and What not? *2021 International Conference on Code Quality (ICCQ)*. 27 March 2021. Moscow, Russia: IEEE, 1–5. Available from https://doi.org/10.1109/ICCQ51190.2021.9392977 [Accessed 19 October 2023].

Sharma, K.K., Sinha, A. and Sharma, A. (2022). Software Defect Prediction using Deep Learning by Correlation Clustering of Testing Metrics. *International Journal of Electrical and Computer Engineering Systems*, 13 (10), 953–960. Available from https://doi.org/10.32985/ijeces.13.10.15.

Vable, A.M., Diehl, S.F. and Glymour, M.M. (2021). Code Review as a Simple Trick to Enhance Reproducibility, Accelerate Learning, and Improve the Quality of Your Team’s Research. *American Journal of Epidemiology*, 190 (10), 2172– 2177. Available from https://doi.org/10.1093/aje/kwab092.

Wu, J.J. (2023). Does Asking Clarifying Questions Increases Confidence in Generated Code? On the Communication Skills of Large Language Models. Available from <http://arxiv.org/abs/2308.13507> [Accessed 4 September 2023].