

Automatic exam essay grader with artificial intelligence feedback

Final Project Report

TU857

BSc in Computer Science (Infrastructure)

**Joseph Egan**

**C20389531**

**Supervisor: Dr Aneel Rahim**

School of Computer Science

Technological University, Dublin

**Date: 11/04/2025**

Abstract

Since 2016, there has been a 13.8% increase in the number of enrolled students in Irish universities (HEA Statistics), reflecting a growing student population. This increase has contributed to a widening lecturer-to-student ratio problem in Ireland. Currently, Ireland has a student-to-staff ratio of 22.38:1 (OECD Data), significantly higher than the OECD average of 15.58:1. This disparity places additional pressure on lecturers, particularly for labour-intensive tasks such as grading essays and providing personalised feedback.

Detailed feedback on written work is essential for students, as it supports their academic growth and development. However, due to increased workloads, lecturers may struggle to offer the level of individualised feedback that helps students improve key skills such as writing and critical analysis.

An AI-assisted essay grading and feedback system could help address this issue. By automating aspects of the grading process, such a system can provide timely, constructive feedback on elements such as grammar, argument structure, and coherence. This would enable lecturers to focus on more complex, personalised teaching while ensuring students receive consistent, high-quality feedback. Implementing an AI-driven system in this context could enhance the learning experience and help educators manage the demands of increasing class sizes across Irish higher education.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Joseph Egan

Date: 11/04/2025

Acknowledgements

I would like to thank my supervisor Dr Aneel Rahim for his guidance throughout the course of the project. I am also grateful to my family for supporting me.

Table of Contents

[1. Introduction 7](#_Toc157603088)

[1.1. Project Background 7](#_Toc157603089)

[1.2. Project Description 7](#_Toc157603090)

[1.3. Project Aims and Objectives 7](#_Toc157603091)

[1.4. Project Scope 7](#_Toc157603092)

[1.5. Thesis Roadmap 7](#_Toc157603093)

[2. Literature Review 8](#_Toc157603094)

[2.1. Introduction 8](#_Toc157603095)

[2.2. Alternative Existing Solutions 8](#_Toc157603096)

[2.3. Technologies Researched 8](#_Toc157603097)

[2.4. Other Relevant Research 8](#_Toc157603098)

[2.5. Existing Final Year Projects 8](#_Toc157603099)

[2.6. Conclusions 8](#_Toc157603100)

[3. Experiment / Software Design 9](#_Toc157603101)

[3.1 Introduction 9](#_Toc157603102)

[3.2. Software Methodology 9](#_Toc157603103)

[3.3. Requirements Gathering 9](#_Toc157603104)

[3.4. Overview of System 9](#_Toc157603105)

[3.X. Oher Sections 9](#_Toc157603106)

[3.X. Conclusions 9](#_Toc157603107)

[4. Experiment / Software Development 10](#_Toc157603108)

[4.1. Introduction 10](#_Toc157603109)

[4.2. Software Development 10](#_Toc157603110)

[4.X. Oher Sections 10](#_Toc157603111)

[4.X. Conclusions 10](#_Toc157603112)

[5. Testing and Evaluation 11](#_Toc157603113)

[5.1. Introduction 11](#_Toc157603114)

[5.2. System Testing 11](#_Toc157603115)

[5.3. System Evaluation 11](#_Toc157603116)

[5.4. Conclusions 11](#_Toc157603117)

[6. Conclusions and Future Work 12](#_Toc157603118)

[6.1. Introduction 12](#_Toc157603119)

[6.2. Future Work 12](#_Toc157603120)

[6.3. Conclusions 12](#_Toc157603121)

[Bibliography 13](#_Toc157603122)

[Appendices 14](#_Toc157603123)

# 1. Introduction

This section will cover the introduction, description, and the scope of my project.

## Project Background

In the modernising world there is a growing need for third level education in degrees to enter the working world as can be seen through the education by generation graph below. This is causing issues and strains on the Irish University education system.

A graph of a number of people

AI-generated content may be incorrect.

(<https://www.cso.ie/en/releasesandpublications/ep/p-cpp8/censusofpopulation2022profile8-theirishlanguageandeducation/levelofeducation/>)

## Ireland was one of the founding members of the Organisation of Economic Co-operation and Education (OECD) in 1960 and currently faces significant challenge with its staff-to-student ratios with one of the worst students to lecturer ratios of the current organization members. This means that lecturers are, relatively speaking, dealing with more students compared to lecturers in other countries, for example Ireland has 22.38:1 student to lecturers whereas the United States has 13.63, and the UK has a ratio of 13.50. (TODO OCD data)

## This ratio imbalance creates excess strain on lectures, especially for labour intensive tasks such as grading essays and providing detailed personalised feedback. The growing number of Irish students and limited ability to increase lecture staff creates a need for tools that can solve this problem.

## Project Description

## This project involved the design and development of a web-based application that uses artificial intelligence to automatically evaluate and provide constructive feedback on student essays. The system combines state-of-the-art machine learning models to emulate human-like assessment and improve the feedback process in educational environments.

## The grading system integrates both **BERT** and **LSTM** models to score essays based on specific marking criteria. BERT was used for its contextual understanding of language, while LSTM provided sequence-based evaluation, ensuring the system could detect both semantic and structural qualities of writing. These models produced numeric and qualitative scores that reflect coherence, grammar, argument strength, and clarity.

## To enhance student learning, the system uses **GPT-2** to generate personalized, natural-language feedback. I developed a prompt engineering framework that extracts key portions of the essay and model-generated scores, then formats them into optimized prompts to generate human-like suggestions and explanations. Token limits were dynamically managed to comply with model constraints while preserving contextual integrity.

## From a technical perspective, the application was built using **Python**, with **Flask** as the web framework. I implemented backend services for text extraction from .docx and .pdf files, grading logic, and API integration with Hugging Face and OpenAI libraries. The frontend interface allows users to upload essays, view automated grades, and receive actionable feedback instantly.

## The system simulates the role of a human examiner, making grading scalable, consistent, and accessible. It was particularly valuable for educational institutions and teachers seeking to supplement manual grading with AI-driven support.

## 

## Project Aims and Objectives

Overall **Project Aims:**

1. To design and implement an AI-driven system that automates the grading of student essays with high accuracy and consistency.
2. To enhance the educational feedback process through natural language generation, providing students with personalized, constructive advice.
3. To combine multiple deep learning models (BERT, LSTM, and GPT-2) in a cohesive pipeline for robust evaluation and contextual understanding.
4. To reduce the workload on human educators by supporting scalable and unbiased essay assessments.
5. To create an intuitive web-based application for seamless user interaction with the grading system.

**Project Objectives:**

**Machine Learning Objectives:**

* Implement and fine-tune a BERT model for contextual grading of essay submissions.
* Integrate an LSTM model to evaluate syntactic and sequential quality of writing.
* Compare performance across models to ensure balanced and interpretable grading outcomes.
* Combine outputs of BERT and LSTM to generate a composite score for each submission.

**Natural Language Generation Objectives:**

* Utilize GPT-2 to create human-like feedback based on essay content and grading scores.
* Engineer dynamic prompts that guide GPT-2 to provide feedback that is helpful, specific, and non-repetitive.
* Manage token length constraints by intelligently chunking input and optimizing prompt formatting.

**Software Development Objectives:**

* Develop a Flask-based web application for file upload, essay extraction, grading display, and feedback presentation.
* Build robust backend functionality to handle .docx and .pdf file parsing using libraries like python-docx.
* Set up HTML-based front-end components to visualize grading results and AI-generated feedback.

**Data & Processing Objectives:**

* Extract raw essay text from uploaded documents and clean it for model input.
* Tokenize and preprocess text appropriately for each model used (BERT, LSTM, GPT-2).
* Handle data throughput efficiently to support multiple essay submissions in real-time.

**User Experience Objectives:**

* Provide users with both numerical scores and narrative feedback to support learning.
* Ensure a clean, accessible interface suitable for students and educators with varying technical backgrounds.
* Minimize feedback latency and optimize prompt design for fast model inference.

**Evaluation & Ethics Objectives:**

* Ensure feedback remains constructive, unbiased, and sensitive to student effort.
* Evaluate the system’s feedback quality against sample essays manually reviewed by educators.
* Maintain data privacy and confidentiality for uploaded student content.

**Deployment Objectives:**

* Ensure the application is modular and extensible for future improvements (e.g., additional grading rubrics or languages).
* Prepare the system for potential deployment in real-world educational settings or integration into learning platforms.

## Project Scope

This project focuses on the development of an AI-enhanced web application that automates the assessment of student essays. The application utilizes deep learning models to grade essays based on qualitative criteria and generates constructive, human-like feedback using natural language processing techniques. It is designed to assist educators by streamlining the grading process while offering students actionable insights into their writing.

**In-Scope Items:**

**1. Essay Grading System**

* Integration of pre-trained **BERT** and **LSTM** models to analyse and score essays.
* Implementation of algorithms to evaluate language fluency, grammar, argument structure, and coherence.
* Combination of scores from both models into a composite grading report.

**2. Feedback Generation**

* Use of **GPT-2** to generate tailored written feedback for students based on essay content and scores.
* Development of custom prompt templates to ensure clarity, usefulness, and tone appropriateness.
* Token management and chunking logic to comply with model input constraints.

**3. Web Application Development**

* Creation of a **Flask-based web application** to allow users to:
  + Upload essay documents in .docx or .pdf formats.
  + Receive instant grading scores and AI-generated feedback.
* Development of HTML/CSS frontend templates to display results clearly and interactively.

**4. File and Data Handling**

* Extraction of text from uploaded documents using python-docx and PyMuPDF.
* Preprocessing of essay content for compatibility with NLP models (tokenization, cleaning, truncation).
* Management of essay content and feedback generation in a modular and efficient backend.

**5. User Interface and Experience**

* Design of a minimal and intuitive UI suitable for students and teachers.
* Clear separation of grading criteria, scores, and narrative feedback in the result display.
* Seamless feedback loop for end-users without requiring technical knowledge.

**6. Evaluation & Testing**

* Manual comparison of model-generated feedback with educator-reviewed samples.
* Performance testing of grading and generation time.
* Evaluation of model accuracy and relevance of feedback.

**Out of Scope:**

* Real-time collaborative grading between students and teachers.
* Grading of multimedia content or handwriting (only typed text essays supported).
* Multilingual essay grading (initial implementation is English-only).
* Integration into third-party Learning Management Systems (e.g., Moodle, Blackboard).
* Creation of a mobile version of the web app.

## Thesis Roadmap

One sentence summary of the following chapters

# Literature Review

This chapter provides a literature review on related works and technologies, including existing AI-based and machine learning based grading systems.

# Experiment / Software Design

This chapter discusses software design, methodology, and system architecture.

# Experiment / Software Development

This chapter outlines the development process and key implementation details.

# Testing and Evaluation

This chapter covers testing and evaluation, detailing system performance and user feedback.

# 6. Conclusions and Future Work

This chapter concludes the report and discusses potential areas for future development.

# 2. Literature Review

## 2.1. Introduction

In this chapter …

This chapter provides an overview of machine learning models, a brief overview of artificial intelligence and their applications in automated essay grading is provided in this section. It examines existing solutions, technologies researched, and relevant academic studies that inform this project’s approach. Additionally, a comparison with other AI grading systems highlights key strengths and limitations as well as the problems and challenges related to it.

The literature review provides the theoretical and technological context for this project, exploring how artificial intelligence, machine learning, and natural language processing have been applied to educational assessment. This section begins by identifying existing solutions in the domain of automated essay grading, highlighting key tools and systems that are either commercially available or open source. It examines how these tools function, the models they use, and the challenges they face in replicating human-level understanding and grading accuracy. The review also covers the motivations for AI grading systems, such as reducing lecturer workload and improving feedback consistency.

In addition to surveying existing systems, the review discusses various technologies and methodologies that were evaluated or implemented in this project. This includes transformer-based models like BERT and GPT-2, as well as sequential models like LSTM. The analysis extends to domain-specific studies and scientific papers that informed the project's direction, helping define its scope, limitations, and innovation potential. The literature review ultimately illustrates the project's niche: creating an AI-assisted grading system designed to support, not replace, human evaluators, with an emphasis on explainable, constructive, and student-friendly feedback.

## 2.2. Alternative Existing Solutions

Various AI-powered essay grading tools, AI tools, and machine learning tools already exist, offering different levels of automation and accuracy. Examples include:

* + 1. **ChatGPT**: A general-purpose conversational AI model designed to generate human-like text based on user prompts. While primarily a chatbot, it can assist with drafting, editing, and providing feedback on written content. It has problems with reading vectors of wording to predicate responses instead of thoroughly answering questions though, for example answering 20 \* 8 it might answer like 20 \* 10 is 200, -50 is 150, this is close enough to answer the question, instead of directly calculation sums.

<https://chat.openai.com/>

* + 1. **Tarekyehya’s Automated Essay Scoring Project**: An open-source initiative that uses machine learning neural networks to score essays based on predefined rubrics. This is a open source project for an essay grader but has problems with user friendly interfaces and lack of personalised feedback.

[tarekyehya/automated-essay-scoring: This repository contains code and resources for my solutions of Kaggle competition focused on training a model to score student essays automatically, The goal here is to make the solutions in the production not only notebook, following MVC pattern and some SOLID.](https://github.com/tarekyehya/automated-essay-scoring)

* + 1. **Turnitin Feedback Studio**: This is a widely used plagiarism detection and grading tool that provides automated feedback on writing mechanics but lacks deeper conceptual understanding. It is useful for detecting misquoted or improperly quoted references and is useful for tracking the progress of students in their writing.

[Plagiarism Detector: Reliable Detection Tools | Turnitin](https://www.turnitin.com/products/feedback-studio/)

* + 1. **Grammerly**: An English language writing assistant software tool. It reviews the spelling, grammar, and tone of a piece of writing as well as identifying possible instances of plagiarism. It can also suggest style and tonal recommendations to users and produce writing from prompts with its AI.

<https://www.grammarly.com/>

* + 1. **EssayGrader**: this is an American pure artificial intelligence grading system used by teachers and lecturers at multiple educational institutions levels, for example elementary school (American primary education), middle school (American junior cert), high school (American leaving cert). Similar to my application EssayGrader requires a lecturer to import a grading rubric to work, but unlike my system it is artificial intelligence specific.

[Essay Grader AI | The AI Essay Grader for Teachers | Free Plans](https://www.essaygrader.ai/)

* + 1. AWSAF49's Automated Essay Scoring 2.0: This is an open source project to develop an AI model that can score student essays by fine-tuning the DeBERTaV3 model using ordinal regression/classification techniques. This is similar to my application by using advanced transformer models for essay scoring and using Keras models, but it lacks usable user interfaces, does not incorporate BERT models and is still in development.

[awsaf49/automated-essay-scoring-2: Automated Essay Scoring 2.0 | Improve upon essay scoring algorithms to improve student learning outcomes](https://github.com/awsaf49/automated-essay-scoring-2)

* + 1. myEssai: This is an artificial intelligence tool used to grade the essays of students. It was designed as an AI tutor providing iterative, detailed essay feedback aimed at improving student writing or accurately grading their essays.

[myEssai | AI Powered Essay Tutor](https://myess.ai/)

* + 1. CoGrader: This is an AI-driven grading system designed to reduce grading time for teachers and lecturers. This tool offers shallow narrative feedback for teachers and deals with a lot of the surface level work involved in grading essays.

[CoGrader | AI Essay Grader | Spend Less Time Grading, More Time Teaching](https://cograder.com/)

* + 1. IntelliMetric: Thos is a proprietary AI essay scoring engine that uses natural language processing and deep linguistic analysis to predicate essay scores based on complex models trained on human-graded examples. Similar to my application it uses deep learning models to predict scores, however unlike mine it lacks a method of student-friendly explanation output.

[IntelliMetric – Vantage Learning](https://www.intellimetric.com)

* + 1. Kolly: This is an AI-driven platform that specialises in helping students improve their college admission essays for Ivy league universities. While my application focuses on general academic essays, Kolly is specifically tailored for college entrance writing and offers feedback to enhance persuasiveness and narrative quality. Unlike my tool, Kolly is not meant for structured academic assessment or rubric-based grading. Its strength is its niche specialization, making it ideal for college applicants seeking high-quality edits.

[Kolly – AI College Essay Feedback](https://www.kolly.ai)

* + 1. Classx: This is less a grading tool, but more of a generic artificial intelligence education support tool. Classx has support tools for students homework like sentence generators, summarizers and a chatbox essay grader, but is not focused on grading essays like my tool.

[Essay Grader - ClassX](https://classx.org/chatbot/essay-grader/)

[AI Tools for Students - ClassX](https://classx.org/ai-tools-for-students/)

* + 1. Paperpal: This is an academic writing tool designed for researchers and students, offering grammar checks, paraphrasing, and structured improvement suggestions. While it supports essay improvement through AI, it lacks the grading and scoring functionality central to my application. Unlike my tool, Paperpal is positioned more as an academic editor than a grader.

[AI Academic Writing Tool - Online English Language Check | Paperpal](https://paperpal.com/)

* + 1. E-rater scoring engine: This is machine learning and artificial intelligence essay grader similar to my application, that uses artificial intelligence and natural language processing to evaluate students essays through automatic scoring and feedback for assessments like the Test of English as a Foreign Language exam.

<https://www.ets.org/erater/about.html>

* + 1. HyperWrite: This is a general-purpose AI writing assistant tool that can generate, rewrite, summarize, and assist with a wide range of writing tasks. While it can help revise essays and provide writing suggestions, it doesn’t assign scores or follow grading rubrics like my appliciation. This is an interesting tool similar to a more specialised ChatGPT.

[HyperWrite | AI Writing Assistant](https://www.hyperwriteai.com/)

* + 1. JFYNet: This is an AI-based feedback tools for teachers and lecturers to better understand student writing performance. It provides targeted suggestions that help teachers plan instruction. While my application is a student-facing tool with immediate feedback, JFYNet is a more teacher-oriented support tool for helping them grade essays.

[JFY AI Writing Assessment Tool, AI Teacher Assistant Program](https://jfynet.org/ai-writing-assessment-tool/)

* + 1. All Day TA: This is an AI chatbot assistant developed in the University of Toronto that answers student questions and provides writing guidance. Similar to my application it supports students with feedback, but it operates more as a conversational learning tool rather than an essay grader. Its strength is accessibility and personalized interaction, but a it has an informal structure and a lack of formal scoring.

[All Day TA](https://www.alldayta.com/)

While these solutions demonstrate the potential of AI in education, they often fall short in terms of personalised feedback, adaptability to specific curricula, and transparency in scoring criteria. This project seeks to address these gaps by integrating explainable AI techniques and tailored feedback generation.

## 2.3. Technologies Researched

As part of this I have investigated several different technologies which have been useful for my project:

* + 1. **TensorFlow 2.14**
* **Purpose:** Backend framework for machine learning operations, including training and running LSTM models used in your essay grading system.
* **Why I Used It:** TensorFlow 2.x offers eager execution, strong support for custom layers, and integration with Keras 2.x, making it ideal for text classification tasks.
* **Alternative:**
  + **TensorFlow 2.15+**: Includes performance enhancements.
  + **JAX** (for experimental performance-focused ML pipelines).

I did not choose this because Keras 2.14 automatically downloads the version 2.14 and I left it as is for package compatibility.

* + 1. **Keras 2.14**
* **Purpose:** High-level API for building and training deep learning models in TensorFlow; used for the LSTM-based essay scoring model.
* **Why I Used It:** Simple syntax, tight integration with TensorFlow, and good support for text processing layers like Embedding, LSTM, and Dense.
* **Alternative:**
  + **Keras 3**: Offers better compatibility with PyTorch and JAX
  + **PyTorch Lightning**: More flexible and modular, especially for research-grade NLP.

**I did not use Keras 3 because it was incompatible with many of my other packages, like Tensorflow, and would have required a major rewrite of my application to be functional that I did not have time for.**

* + 1. **Hugging Face Transformers**
* **Purpose:** To develop pretrained transformer Bidirectional encoder representations from transformers models (BERT) used for essay grading and feedback generation (GPT-2).
* **Why I Used It:** Provides easy access to cutting-edge language models with pretrained weights and tokenizer utilities, and allows me to use artificial intelligence to translate the scores from my natural language processing models into more human readable feedback.
* **Alternative:**
  + **GPT-3.5 / GPT-4o** via OpenAI API (for higher-quality generation).
  + **DistilBERT / RoBERTa** (faster or more accurate alternatives to BERT).

I chose not to use APIs due to financial considerations and complications with the OpenAI website, and I choose to use BERT models instead of DistilBERT or RoBERTa models do to previous familiarity and experience.

* + 1. Large Language Models (via Hugging Face pipeline)
* **Purpose:** Generative artificial intelligence model used to produce student feedback based on essay input, exam paper, grading rubric, and grading score.
* **Why I Used It:** Local Large Language models were an easier solution to the use of low volume of queries for the purpose of testing and simpler to use than APIs of larger models.

|  |  |  |  |
| --- | --- | --- | --- |
| Model Name | Parameters | Download Size (approx) | Link |
| GPT-4 (OpenAI) | Unknown (est. 1.5T) | Cloud only | https://chat.openai.com |
| GPT-3.5 (OpenAI) | 175B | Cloud only | https://chat.openai.com |
| LLaMA 2 7B | 7B | ~13.5 GB | https://huggingface.co/meta-llama/Llama-2-7b |
| LLaMA 2 13B | 13B | ~25 GB | https://huggingface.co/meta-llama/Llama-2-13b |
| LLaMA 2 70B | 70B | ~140 GB | https://huggingface.co/meta-llama/Llama-2-70b |
| Mistral 7B | 7B | ~13 GB | https://huggingface.co/mistralai/Mistral-7B-v0.1 |
| Mixtral (MoE 2x7B) | 12.9B active | ~12 GB | https://huggingface.co/mistralai/Mixtral-8x7B-v0.1 |
| Phi-2 (Microsoft) | 2.7B | ~1.7 GB | https://huggingface.co/microsoft/phi-2 |
| Gemma 2B (Google) | 2B | ~1.4 GB | https://huggingface.co/google/gemma-2b |
| Gemma 7B (Google) | 7B | ~4.5 GB | https://huggingface.co/google/gemma-7b |
| GPT-J 6B | 6B | ~24 GB | https://huggingface.co/EleutherAI/gpt-j-6B |
| GPT-NeoX 20B | 20B | ~80 GB | https://huggingface.co/EleutherAI/gpt-neox-20b |
| TinyLlama | 1.1B | ~1 GB | https://huggingface.co/cognitivecomputations/TinyLlama-1.1B-Chat-v1.0 |
| OpenChat 3.5 | Based on 7B | ~4–6 GB | https://huggingface.co/openchat/openchat-3.5-1210 |
| Code LLaMA 7B | 7B | ~13 GB | https://huggingface.co/codellama/CodeLlama-7b-hf |

* **Which one did you used:** I chose to use GPT-2, because it is lightweight at 500 megabytes, open-source, and can be fine-tuned or used off-the-shelf for constrained text generation.
* **Alternative:**
  + **GPT-3.5 Turbo** or **GPT-4**: Better fluency, more relevant output, but requires API access and quota.
  + **The table above shows a list of alternatives in an easier to digest format.**
    1. **Flask**
* **Purpose:** Lightweight Python web framework for building the frontend interface and managing essay uploads, processing, and result display.
* **Why I Used It:** Simple setup, integrates well with Python-based AI pipelines, and perfect for a single-page application.
* **Alternative:**
  + **FastAPI**: more modern, async support, better for scaling RESTful APIs.
  + **Streamlit / Gradio**: if I was building a quick demo or GUI dashboard.
  + ReactNative: this would have been better if I was integrating a firebase or other online database.
    1. HTML
* **Purpose:** Used for the frontend of my website. Used in Flask to render dynamic essay results, grading scores, and feedback.
* **Why I Used It:** Works natively with Flask and supports clean UI rendering for server-side apps.
* **Alternative:**
  + **React / Vue.js**: If I was building a more interactive, client-heavy frontend.
  + **TailwindCSS**: For improved UI styling and layout design.
    1. **Tokenizer Utilities (GPT2Tokenizer, BertTokenizer)**
* **Purpose:** Manages token counts and input slicing to comply with model input size limits, for example 1024 for GPT-2.
* **Why I Used It:** It ensures input doesn’t exceed model constraints while preserving essay integrity and context.
* **Alternative:**
  + **Tokenizers (by Hugging Face)**: Faster, more configurable tokenization backends.
  + **Tiktoken**: Used for precise token counting with OpenAI models.
    1. **GitHub**
* **Purpose:** Version control software, especially useful during iterative model development and deployment.
* **Why I Used It:** Industry-standard tool for source management and deployment tracking. It was useful for testing features, for example BERT or LSTM models, without compromising the application by having stable versions of the application.
* Alternative:
  + Gitbucket: This is a website made by Scala compatible with Git.
  + Bitbucket: This is a Git code management tool integrated with other features for team planning made by Atlassian.
    1. **Scikit-learn**
* **Purpose:** A popular machine learning library that could have been used to develop classifiers for grading purposes.
* **Why I did not Use It:** Scikit-learn was a potential technology that I could have used in my application that I did research. I did not find a use for it for my application within the lifetime of the project, but it could of helped students as a visual tool see where they compare to their peers or as an estimate tool.
  + 1. **Docker**
* **Purpose:** I could have used Docker to compartmentalise my application into a lightweight, more easily accessible format for deployment for others to download and use, version control, and testing on other operating systems and devices.
* **Why I did not Use It:** Converting my application into a Dockerfile and a Docker image would have been a new experience for me and I am inexperienced with Docker in general. I did research Docker as a possibility, but I lacked the time to experiment with the software in time for this project.

## 2.4. Other Relevant Research

Domain specific research

Scientific paper specific research:

Strategies for Deploying Unreliable AI Graders in High-Transparency High-Stakes Exams (“Strategies for Deploying Unreliable AI Graders in High-Transparency High-Stakes Exams,” n.d.), is a paper regarding possible ways to use a NPL AI grading system designed to grade midterm tests regarding high level explanations of code. This is a similar topic to mine in the context that we are both developing systems to grade students work, but we differ in core objects of or projects. They want to develop a system that would officially grade a student’s test instead of a human, such as a lecturer or teacher’s assistant, instead of my objective of producing a tool to assist students and lecturers grading practice exam papers. I feel my project covers the niche of a tool to assist lecturers and an exam preparation and self-improvement tool to assist students, whereas their paper discusses ways to develop a tool to replace humans from grading papers.

College Exam Grader using LLM AI models (“College Exam Grader using LLM AI models,” n.d.), is a paper regarding an artificial intelligence exam grader based on ChatGPT-4.0. This is a project with the exact same goal as mine. The development of an AI based exam grader that can automate lecturers and teachers grading papers, but also produce accurate, consistent, and precise grading results. Their approach is different than mine, because they are using the AI to grade the exam paper and offer feedback, whereas I am using a machine learning model to grade the papers and the AI to personalise the feedback to the students.

Automatic Grading of Computer Programs: A Machine Learning Approach (“Automatic Grading of Computer Programs: A Machine Learning Approach,” n.d.), is a paper regarding a machine learning model to assess the competency of a Programmer by grading their computer programs. This is similar to my project in the context that it uses a machine learning model to grade assessed work but differs in that it is used to grade the individual who write the programs through their programs. This is an example to learn how to implement a machine learning model, but it is more related to professional workplace application compared to my academic focused project.

Beyond human subjectivity and error: a novel AI grading system (Gobrecht et al., 2024) is a paper

based on developing automatic short answer grading system using an open-source transformer

model to grade students answers to open ended questions. This is similar to my project in automatically grading students work but is more ambitious than the scope of my project. This paper in particular offers a comparison and possibly a template for expanding my project from grading answers to past exams papers to answers to actual exam papers, i.e. the difference between a study tool and a replacement to grading real papers.

ChatGPT as a Solver and Grader of Programming Exams written in Spanish (Saborido-Fernández et al., 2025) is a paper that assesses ChatGPT's ability to solve and grade real programming exams from a Computer Science BSc program. Findings indicate that while ChatGPT is effective for simple coding tasks, it struggles with complex problems and evaluating others' solutions. This paper suggests that while AI can assist in grading, human oversight remains crucial, aligning with my project's goal of creating a supportive tool rather than a complete replacement for human graders.

Are Large Language Models Good Essay Graders? (Kundu and Barbosa, 2024) is a paper that evaluates the effectiveness of Large Language Models like ChatGPT and Llama in Automated Essay Scoring. The study finds that LLMs tend to assign lower scores than human raters and exhibit poor correlation with human evaluations, suggesting they are not yet suitable replacements for human grading similar to the pervious paper ChatGPT as a Solver and Grader of Programming Exams written in Spanish.

Grading exams using large language models: A comparison between human and AI grading of exams in higher education using ChatGPT (Flodén, 2025) is an article from the British Educational Research journal that compares the performance of ChatGPT in grading university exams to that of human teachers. It examines the reliability and consistency of AI grading, highlighting areas where AI aligns with or deviates from human assessments. In particular this article discusses a problem that I have noticed myself using model application that AI grading systems tend to give grades with a range, i.e. that the models tend to not show very high or low grades.

Machine Learning-Based Automated Grading and Feedback Tools for Programming: A Meta-Analysis (Messer et al., 2023) this paper analyses how machine learning has been applied to grade and provide feedback on programming assignments. It discusses various approaches, their effectiveness, and the challenges faced in automating the grading process. The paper offers a comprehensive overview of existing methods and tools, helping identify best practices and potential pitfalls to avoid in developing my grading assistance tool.

Assessing Confidence in AI-Assisted Grading of Physics Exams through Psychometrics: An Exploratory Study (Kortemeyer and Nöhl, 2024) is a paper that discusses reliability and confidence that can be placed in the accuracy of artificial intelligence grading of real exams using the Item Response Theory. This paper helps show the relevance of my project in that automatic grading tools are useful and also shows why I choose the scope of my project, because of the ethics and significance of failure in real world applications.

Strategies for Deploying Unreliable AI Graders in High-Transparency High-Stakes Exams (Azad et al., 2020) this paper describes the deployment of an imperfect NLP-based automatic short answer grading system in a large-enrolment introductory college course. It characterizes the deployment as both high stakes and high transparency, discussing the challenges and strategies involved. The strategies discussed for deploying AI graders in sensitive academic settings can inform the implementation of my tool.

## 2.5. Existing Final Year Projects

Title: Machine Learning for Predictive Analysis and Recommender System Student: Glory Pierce Eguare Description (brief): An application that generates a predictive algorithm for the fuel consumption efficiency of automotives What is complex in this project: The What technical architecture was used: Model template view architecture Explain key strengths and weaknesses of this project, as you see it. Strengths: The use of a detailed dataset auto-mpg, to train the machine learning algorithm. Weaknesses: The normalisation of the ranges of his dataset to facilitate easier training, may 12 have led to skewed data biases to his results. This would have disproportionally punished more specialised vehicles, i.e. an unusually heavy car would look like it would have unexpectedly terrible mpg and raise the value of more average cars.

Title: Generating expenditure tracking and analysis from hard copy receipts Student: Cillian Keohane Description (brief): A system to scan receipts and process them into an expenditure tracker What is complex in this project: The use of a Wikimedia API and the machine learning algorithm to process the receipts. What technical architecture was used: Model, View, and Controller architecture Explain key strengths and weaknesses of this project, as you see it. Strengths: the use of the scanner and photo processor, and the machine learning algorithm that adapts to the individual user and improves over time. Weakness: It is not synchronized to other banking devices to automate the process of taking in receipts.

## 2.6. Conclusions

The literature review has revealed a dynamic and rapidly evolving field surrounding AI-driven assessment and feedback tools, with applications ranging from simple grammar checking to fully automated grading systems for high-stakes exams. Throughout this research, it became clear that while many tools already exist—such as ChatGPT, Turnitin Feedback Studio, and IntelliMetric—few directly address the unique blend of goals at the heart of this project: accurate AI grading tailored for academic essays, and personalized feedback generation to support both lecturers and students in formative assessment environments.

Unlike large-scale systems built to replace human judgment entirely, such as the College Exam Grader using LLM AI models, or tools designed primarily for assessment standardization in high-stakes settings (e.g., Azad et al.'s Strategies for Deploying Unreliable AI Graders in High-Transparency High-Stakes Exams), this project focuses on assisting rather than replacing educators. It recognizes the pedagogical importance of human oversight while offering a scalable solution to the growing lecturer-to-student imbalance in Ireland and other education systems. This approach maintains the balance between automation and accountability, leveraging the strengths of machine learning without discarding the contextual sensitivity of human feedback.

What distinguishes this project further is its modular, hybrid architecture, combining transformer-based (BERT) and sequential (LSTM) models for scoring, with a generative language model (GPT-2) for narrative feedback. Many existing systems use black-box grading models with little transparency or flexibility in how feedback is constructed. In contrast, this system aims to be explainable and adaptable—highlighting individual strengths and areas for improvement in student writing, aligned to defined grading rubrics. Tools like Grammarly, Paperpal, or myEssai provide helpful suggestions, but lack contextual rubric-aligned scoring and true educational framing in their outputs.

The reviewed literature also revealed practical limitations that helped refine the project’s scope. For example, challenges with LLM reliability, evaluation bias, token limits, and the interpretability of feedback emphasized the importance of human-in-the-loop systems and iterative testing. These insights directly influenced the model choices, evaluation methods, and feedback strategies adopted in the final system.

Ultimately, this project positions itself in a niche that prioritizes educational enhancement over efficiency alone. By building a system that bridges the gap between automated grading and human teaching, the project contributes meaningfully to the broader discourse on AI in education. The foundation laid here has strong potential for future expansion—including multilingual grading, integration into virtual learning environments, and development of a custom GPT-style model—ensuring it remains relevant in the next generation of academic AI tools.

# 3. Experiment / Software Design

## 3.1 Introduction

This section focuses on the design and planning process behind the development of the AI grading system. It begins by outlining the chosen software development methodology, which in this case is the Waterfall model. The rationale for this approach is explained in terms of project scope, individual workload, and structured progression through phases such as planning, design, implementation, testing, and deployment. Following this, the section details how requirements were gathered through feasibility studies, tool evaluations, and analysis of potential use cases by educators and students.

The design chapter also introduces the high-level architecture of the application. It explains the interaction between the grading models, essay input system, feedback generation module, and the web application interface. Diagrams and descriptions are included to illustrate how the components are integrated, from text extraction to model inference and final result display. Each system element was designed with modularity and scalability in mind, allowing for future additions such as support for more grading rubrics, additional essay formats, or multilingual capabilities.

## 3.2. Software Methodology

## Waterfall Methodology - The Waterfall model is a traditional, linear approach where each of the 5 phases of development, Requirements, Design, Implementation, Testing, and Deployment, is completed before moving on to the next. It works well for projects with clear, well-defined goals and minimal expected changes during development.

## This is the methodology I chose to follow for my project. Because I was working alone and had clear goals from the start, the structured nature of Waterfall helped me break the project into manageable phases. I began with requirements gathering and design, then moved into implementation and testing in a sequential manner. It also allowed me to focus deeply on one stage at a time without needing to switch between tasks constantly. For a solo project with limited collaboration and a fixed deadline, Waterfall provided the structure I needed to stay organised and on track.

## Agile Methodology – The Agile is an iterative and flexible approach that promotes working in small, repeatable sprints with regular feedback and incremental updates. It’s ideal for dynamic projects where requirements might evolve, or when working closely with users or a team.

## If I had more time or access to other teammates, Agile would have been a great fit for my project. I could have broken my work into two-week sprints—one for setting up the document parser, another for testing the grading models, and another for tuning feedback generation. After each sprint, I could have gathered user feedback and used that to refine the next phase of development.

* + 1. Spiral Methodology - The Spiral Model combines structured planning with iterative development, with a focus on identifying and reducing risk through repeated cycles of development. Each loop involves planning, risk analysis, engineering, and evaluation.

Given the complexity of my project, using multiple different machine learning models like LSTM and BERT models and integrating artificial intelligence models like GPT-2, the Spiral Model could have helped me manage potential risks more effectively. I could have started with a small prototype that graded basic essays, then iteratively added features like feedback generation or rubric alignment. In each cycle, I would evaluate what worked, identify any performance issues or limitations, and use those findings to guide the next stage. This would have allowed me to address technical challenges early and adapt the system gradually based on testing and feedback.

* + 1. V-model Methodology - The V-Model is a development approach that mirrors the Waterfall model but places heavy emphasis on validation and testing. For every development stage, there is a corresponding testing phase, ensuring that quality assurance is built into every step.

The V-Model would have been ideal for a more test-driven approach to my project. For example, while designing the feedback generation module, I could have written validation tests to compare the AI's output against educator-reviewed responses. Similarly, during the grading model development, I could have aligned every step with test cases that verified accuracy and performance. This would have made it easier to ensure that each part of the system met specific goals and functioned as expected. Given the importance of accuracy and fairness in educational tools, using the V-Model could have helped make my application more reliable and robust.

## 3.3. Requirements Gathering

I did an initial feasibility study over September and October, which can be seen in my GitHub repository of my final year project at [C20389531Joseph/FYP at FeasibilityStudy](https://github.com/C20389531Joseph/FYP/tree/FeasibilityStudy).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE** | | 1 | AI feedback generator | |
| **Description of Goal in Context** | | The program sends the essay, grading system, and grade to an api for chatgpt. The chatgpt will offer feedback, that will be presented to the user. | | |
| **Preconditions** | | Api is set up, the input function to receive and process the essay is set up, the ML algorithim is set up and functional, the systems ability to send the data to the api is functional, and the ability to receive input from the api and present it to the user is set up. | | |
| **Post Conditions, Success End Condition** | | Working feedback from the AI, and the grade is presented to the student regarding their essay. | | |
| **DESCRIPTION of Scenario** | | The system will put the essay through the ML algorithim to grade it. The system will then automatically sent the essay, grading system, and grade to the api. The api sends the request to an existing ai like chat gpt or gemini to translate into human readably comments about the grading of their essay and ways to improve. This feedback from the api and the grade from the ML algorithim will be presented to the user. | | |
| **Main Flow** | | | | | |
| **Step** | **Action** | | | **Alternate** | |
| 1.1 | The ML algorithim grades the essay | | |  | |
| 1.2 | The system sendsthe essay, grading system, and grade to the api automatically. | | |  | |
| 1.3 | The api makes the request to the ai with the information from step 1.2 | | |  | |
| 1.4 | The ai receives the request | | | 2.4 | |
| 1.5 | The AI generates feedback | | |  | |
| 1.6 | The AI sends the feedback through the api back into the system | | |  | |
| 1.7 | The system receives the response from the ai | | | 2.7 | |
| 1.8 | The system presents the feedback from the api and the grade from the ML algorithim to the user | | |  | |
|  | | | | | |
| **EXCEPTIONS or ERROR Flow**  **Description** | | | | | |
| **Step** | **Branching Action**  < Exception number m of Use Case n> | | | **Alternate** | |
| 2.4 | Error sending the request to the AI  Skip steps 1.5 to 1.7 | | | 1.4 | |
|  | | | | | |
| **ALTERNATIVE or VARIATION Flow**  **Description**  <condition causing alternative>  <list of variation> | | | | | |
| **Step** | **Branching Action** | | | **Alternate** | |
| 2.7.1 | If there is an issue with receving the feeback from the AI skip 1.8 | | | 1.7 | |
| 2.7.2 | The system presents the grade from the ML algorithim to the user with a message indicating an issue with the AI | | |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE** | | 2 | LSTM and BERT grader | |
| **Description of Goal in Context** | | The application receives the essay document from the user, loads the correct models, for example CMPU4007 2023, uses the models to grade the essay, and returns the feedback. | | |
| **Preconditions** | | The input function to receive and process the essay is set up, the ML models is set up and functional, the systems ability to send the data to the models is functional, and the ability to receive feedback from the models is functional. | | |
| **Post Conditions, Success End Condition** | | Working feedback from the models, and the grade is presented to the student regarding their essay. | | |
| **DESCRIPTION of Scenario** | | The system will put the essay into the ML models to grade it. The models will grade the essay and send feedback. This feedback from the a ML model will be presented to the user. | | |
| **Main Flow** | | | | | |
| **Step** | **Action** | | | **Alternate** | |
| 1.1 | The application processes the essay. | | |  | |
| 1.2 | The application sends the text and exam paper to use to the LSTM file and the BERT file to grade the essay | | |  | |
| 1.3 | The model files load the correct model, e.g. CMPU 4025 year 2023. | | |  | |
| 1.4 | The application uses the model to grade the essay | | |  | |
| 1.5 | The models send feedback to the controller file | | |  | |
| 1.6 | The application receives the feedback from the models | | | 2.6 | |
| 1.7 | The application presents grading from models to user | | |  | |
| **EXCEPTIONS or ERROR Flow**  **Description** | | | | | |
| **Step** | | | | | |
|  | **Branching Action**  < Exception number m of Use Case n> | | | **Alternate** | |
| 2.6 | One or more of the feebacks are not received from the models  Use feedback from any possible model | | | 1.7 | |
| **ALTERNATIVE or VARIATION Flow**  **Description**  <condition causing alternative>  <list of variation> | | | | | |
| **Step** | | | | | |
| 2.7.1 | **Branching Action** | | | **Alternate** | |
|  |  | | |  | |

## 3.4. Overview of System

The Automatic Essay Grader with AI Feedback (AEGAF) is a web-based application designed to assist both students and educators by automating the process of essay evaluation and feedback. Its primary goal is to estimate the quality of student-written essays and generate constructive, tailored feedback that highlights both strengths and areas for improvement. The system is built with a strong pedagogical focus, aiming to support students in self-assessment and revision while reducing the marking burden on lecturers. By acting as a formative tool, AEGAF bridges the gap between traditional human evaluation and modern AI capabilities.

The process begins when a user selects a specific exam or mock question from a predefined list and uploads their essay in either .docx or .pdf format. The system extracts the raw text content from the document using backend file-handling libraries such as python-docx and PyMuPDF. Once the text is extracted, it is preprocessed and passed into a dual-model scoring pipeline. This pipeline consists of a BERT model for assessing contextual understanding and semantics, and an LSTM model for evaluating the structural flow and sequential coherence of the essay. These models generate a series of scores based on predefined grading rubrics, focusing on critical essay elements like grammar, clarity, structure, and argument strength.

Once the grading is complete, the scores—along with key excerpts from the essay and the original rubric—are passed into a natural language generation (NLG) module. This module uses GPT-2, a transformer-based model, to generate narrative feedback in a format that mimics human evaluators. To manage GPT-2's token limits, a prompt engineering framework was developed to intelligently compress and format the input. The result is a clear, concise, and student-friendly feedback report, which might include comments like “You scored 80% on your essay, but should focus on supporting your arguments with more specific examples and improving grammar consistency.”

Finally, the application displays the results to the user through a web interface built using Flask and HTML templates. This interface presents the grading scores and GPT-generated feedback side by side, making it easy for students to understand how their work was assessed and what they can do to improve. From an educator’s perspective, the tool serves as a rapid evaluation assistant, offering a scalable and consistent way to evaluate multiple essays without sacrificing feedback quality. In doing so, AEGAF promotes deeper student engagement and helps mitigate the growing workload challenges in higher education environments.

## 3.5. Conclusions

The software design phase of this project played a crucial role in shaping the overall structure and functionality of the Automatic Essay Grader with AI Feedback (AEGAF) system. Through careful planning and selection of suitable technologies, I was able to design a modular and scalable system that integrates multiple machine learning models, file processing capabilities, and a user-friendly web interface. Each component—from document ingestion to model orchestration and front-end delivery—was designed with clarity, maintainability, and future expansion in mind.

The decision to use the Waterfall methodology provided a structured framework for development, allowing me to approach each phase of the system sequentially with clearly defined goals and deliverables. It also suited the independent nature of the project, where iteration cycles could be managed efficiently without the complexities of team-based collaboration. Requirements gathering, early-stage feasibility studies, and system architecture planning laid a strong foundation for the successful implementation of the application in later stages.

The integration of transformer-based and sequential models for grading, combined with GPT-2 for feedback generation, allowed for a hybrid approach that is both technically robust and pedagogically meaningful. The design ensures that AI-generated feedback is explainable, actionable, and relevant to students' learning outcomes. Additionally, the front-end interface was designed to prioritize usability, accessibility, and real-time interaction, ensuring the tool is practical for students and lecturers alike.

Overall, the software design process has established a solid and adaptable blueprint for development. It ensures that the system is capable of delivering meaningful feedback and reliable grading, while leaving room for future enhancements such as additional language support, broader rubric integration, or deployment as a standalone platform within educational institutions. With the design phase complete, the project is well-positioned to move into full-scale development and testing.

# 4. Experiment / Software Development

## 4.1. Introduction

## The software development section outlines the practical implementation of the project’s architecture, moving from planned designs to functional systems. It breaks down the development process into core modules including backend infrastructure, machine learning model integration, feedback generation, and web application components. The section documents how the system evolved from a basic input/output interface into a responsive web application capable of interpreting and evaluating student essays using advanced AI models.

## It also highlights key challenges encountered during development and the strategies used to address them. These include handling file formats (.docx and .pdf), ensuring compatibility between different machine learning models (BERT, LSTM, GPT-2), managing token limits during feedback generation, and creating a clean user interface. Each development phase is supported by sample code snippets, logic explanations, and insights gained through iterative testing. This section serves to demonstrate the technical complexity and ingenuity required to build an AI-enhanced grading tool from scratch.

## 4.2. Software Development

## 4.X. Oher Sections

## 4.X. Conclusions

# 5. Testing and Evaluation

## 5.1. Introduction

Testing and evaluation play a critical role in ensuring the reliability, accuracy, and user-friendliness of the AI grading system. This section presents the approach used to validate each component of the software, from model performance and error rates to front-end interactions and user feedback loops. It outlines the testing phases—unit tests, integration tests, system tests—and the metrics used to evaluate the grading models, such as mean absolute error (MAE) and response consistency. The goal of this testing phase is not just to ensure functionality, but to confirm that the tool can reliably simulate human-like grading behaviour.

Furthermore, the section addresses user evaluation, where feedback from test users (students and educators) was gathered to assess the practicality of the tool in real-world scenarios. Their responses helped validate the system’s ease of use, quality of feedback, and perceived grading accuracy. These insights were crucial for refining the user experience and setting priorities for future improvements. By combining quantitative testing with qualitative evaluation, this section illustrates a well-rounded approach to validating the AI grading system’s effectiveness.

## 5.2. System Testing

The methodology for the development of the project is a Waterfall approach. This involves performing development cycles over a development period, for example get a component done in one week. I aim to get units of the system, for example document input, essay analysis, rubric application, and feedback generation, programmed and then validation tested. I will then do system testing to validate the integration of the components. Automated test cases will be written to verify functional requirements, and simulated user scenarios will assess the system.

## 5.3. System Evaluation

Evaluation will focus on two key aspects:

1. Performance Metrics: The system’s speed, accuracy of grading, and relevance of feedback will be compared against manual grading benchmarks.
2. UserFeedback: I hope to get students and lecturers to provide input and performance evaluation on the system’s usability and the value of its feedback. This input will guide iterative improvements.

The system evaluation is further discussed in more detail in section 4 of this report.

## 5.4. Testing grading models

There are several tests, both primitive and automated, in my demo model. Some of them are primitive print statements, like the print("hello World") in the first section of my Model.ipynb file to test the interactive kernel or the print("Model trained and saved successfully!") are used to see if the model training is finished. Other tests are automated like the test for the mean absolute error used to test the accuracy of my model. These tests are important to the maintenance, usability, and functionality of my demonstration model.

## 5.5. Testing frontend View page of the website

There are several tests, both primitive and automated,

## 5.6. Testing Controller file of the website

There are several tests, both primitive and automated,

## 5.7. Testing backend Model file of the website

There are several tests, both primitive and automated,

## 5.8. Conclusions

# 6. Conclusions and Future Work

## 6.1. Introduction

The final section of this report reflects on the development and outcomes of the AI essay grading system, summarizing its achievements and limitations. It revisits the project goals and evaluates the extent to which they were met, particularly the successful integration of multiple machine learning models and the creation of personalized feedback through natural language generation. It also highlights the significance of the project in the broader context of educational technology, particularly its role in easing lecturer workloads and supporting student learning through timely feedback.

Looking ahead, the section outlines potential areas for expansion and refinement. These include training the models on more diverse datasets, improving feedback precision through fine-tuning or model replacement, integrating the system into third-party learning platforms, and exploring multilingual support. Future work could also involve creating purpose-built transformer models specifically optimized for essay evaluation tasks. This section provides a roadmap for continued research and development, building on the current project to create an even more robust and adaptable tool for the educational sector.

## 6.2. Future Work

* Development of a specialised artificial intelligence: For this project I used an API of an existing artificial intelligence. A potential future addition to the project could be the development of an open-source artificial intelligence trained for just this project.
* Expansion of database of exam paper models: I have only trained the models for 3 exam paper. The training of models for every past paper in TUD could be a potential future for this project.
* Development of mock papers: I made an example of a 2025 exam paper for Advanced Security 1 for demonstration for my Interim report in December. It is still used as an example in my project. An area of future development could be the creation of model specific mock papers for students to practice with.
* Creation of Generative Pre-trained Transformer machine learning models: My project currently only uses LSTM and BERT models for grading, but GPT models could also be made to aid in the grading process.

## 6.3. Conclusions

# Bibliography

Azad, S., Chen, B., Fowler, M., West, M., Zilles, C., 2020. Strategies for Deploying Unreliable AI Graders in High-Transparency High-Stakes Exams, in: Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millán, E. (Eds.), Artificial Intelligence in Education, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 16–28. https://doi.org/10.1007/978-3-030-52237-7\_2

Flodén, J., 2025. Grading exams using large language models: A comparison between human and AI grading of exams in higher education using ChatGPT. Br. Educ. Res. J. 51, 201–224. https://doi.org/10.1002/berj.4069

Gobrecht, A., Tuma, F., Möller, M., Zöller, T., Zakhvatkin, M., Wuttig, A., Sommerfeldt, H., Schütt, S., 2024. Beyond human subjectivity and error: a novel AI grading system. https://doi.org/10.48550/arXiv.2405.04323

Kortemeyer, G., Nöhl, J., 2024. Assessing Confidence in AI-Assisted Grading of Physics Exams through Psychometrics: An Exploratory Study. https://doi.org/10.48550/arXiv.2410.19409

Kundu, A., Barbosa, D., 2024. Are Large Language Models Good Essay Graders? https://doi.org/10.48550/arXiv.2409.13120

Messer, M., Brown, N.C.C., Kölling, M., Shi, M., 2023. Machine Learning-Based Automated Grading and Feedback Tools for Programming: A Meta-Analysis, in: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. Presented at the ITiCSE 2023: Innovation and Technology in Computer Science Education, ACM, Turku Finland, pp. 491–497. https://doi.org/10.1145/3587102.3588822

Saborido-Fernández, P., Fernández-Pichel, M., Losada, D.E., 2025. ChatGPT as a Solver and Grader of Programming Exams written in Spanish. https://doi.org/10.48550/arXiv.2409.15112

# Appendices