**insert front page here**

**Abstract**

Text-generating applications like chatGPT have been seeing more usage as time goes by. Not only that, but there are also no doubts that much better text-generators await us in the future, capable of delivering far better results. Because of that, it is only a matter of time before much of the text we see on the internet starts to be authored by such software applications, raising concerns about originality. As such, many would like to have a guarantee that the text they are reading had a human hand behind it.

This report describes the development process of an application capable of distinguishing between Artificial Intelligence (AI) and human text whilst also being user-friendly, easy to use and effective at accomplishing its goal.

**Acknowledgements**

# Alexandre, Ricardo, Daniel.

**Contents**

Table of Contents

**1 Introduction**

* 1. Context
  2. Motivation
  3. Goals
  4. Document Structure

1. **State of the Art**
   1. Related Projects
      1. GPT-Zero
      2. Originality.AI
      3. Open AI’s AI Text Classifier
   2. Technology
   3. Conclusions

**Table of Figures**

Table of Contents

1. **Introduction**
   1. System Architecture
2. **State of the Art**

**3. System Architecture**

3.1 Use Case Model

**Abbreviations**

**AI** Artificial Intelligence

**LM** Language Models

Chapter 1

**Introduction**

With the advent and proliferation of text-generating AIs comes many concerns about originality and search-result quality. Users of search engines prefer to read human-authored articles about the topic they are searching rather than AI-generated articles that are often considered to be regurgitated, low quality and spammy when compared to articles written by humans.

Websites that rely on AI to generate their articles know this, so they try to disguise their robotic articles as human-authored articles, normally by attaching a (normally fake) name and face to its preface. This makes it hard for any normal person to reliably make the distinction.

Not only, but teachers whose jobs is to evaluate work in the form of written assignments also face the problem of having to differentiate between students who actually wrote their assignments and students who generated their assignments with AI.

* 1. **Context**

This is a software development project that focuses on AI, specifically the branch of machine learning and neural networks.

* 1. **Motivation**

As programmers, it can be hard to get a good answer for a bug or a code question as a lot of AI-generated pages clutter most search engines. Being able to steer clear of AI-using websites would greatly help us improve our productivity.

* 1. **Goals**

The goal of this project is to create a functional, top-of-the-line AI capable of correctly distinguishing between human and AI-generated text.

Chapter 2

**State of the Art**

* 1. **Related Projects**

The goal of this section is to present other projects or applications whose domain intersects with this project.

* + 1. **GPT-Zero**

GPT-Zero is a website where you paste text or upload documents to see if it was AI-generated. Has the best detection rate of all. Has a minimum of 250 characters, it only shows the first 5000 characters in the free version, and it does not give a percentage of confidence, it only shows the perplexity and burstiness scores alongside a general message of the likelihood of your text being written by an AI.

* + 1. **Originality.AI**

Is a website and an extension used to detect both AI written content and plagiarism. Has a poor AI detection rate. Costs credits to scan documents, with credits costing money. Claims to have a full website scan coming soon.

* + 1. **OpenAI’s AI Text Classifier**

Website very similar to GPT-Zero except it has an awful detection rate, as flipping a coin has a better chance of detecting if a text was written by an AI or not.

* 1. **Technology**

For now, we are limited by our language model’s and neural networks, but the application will be built in a way to allow for introduction of more complex models capable of differing results.

**2.3 Conclusions**

This can only be done by the end of the project, so this is just filler text.

Chapter 3

**System Requirements and Architecture**

* 1. **System Requirements**

This section presents the system requirements specification, as a result of the first phase of the prototype development.

* + 1. **Requirements Elicitation**

After a lengthy reunion with the stakeholders and the team of developers, various requirements were brainstormed from the perspectives of the end-user, external developers and administrators.

* + 1. **Context Description**

This section contains concise descriptions on how the application is expected to be used by different users.

An end user, such as a teacher evaluating a student’s body of work, will open a website/pdf file and either will, automatically or manually (depending on their preferences), have their text highlighted, depicting which sentences the application notices as being written by an AI.

To an external developer, the system’s architecture was designed in such a way that implementing new languages is very simple.

* + 1. **Actors**

**End User:** Represents the user, who can take on several roles when utilizing the application. Accesses it through the extension and is able to get all the AI verification needed.

**Developer:** Capable of adding Language Models as needed to the system.

* + 1. **Use Cases**

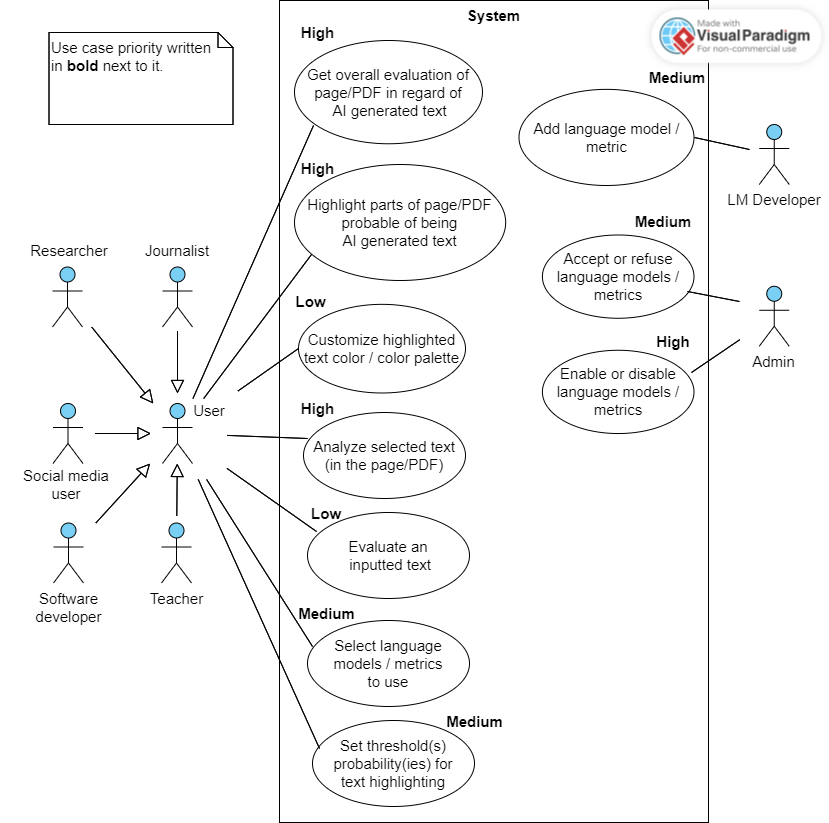
****

Figure 3.1: Use Case Model