Detecting Fake Apps Through User Review Analysis on the Google Play Store

By
Kazi Khalid Hasan Tamim
212-15-4129

Md. Razib Hossain
212-15-4132

FINAL YEAR DESIGN PROJECT REPORT

This Report Presented in Partial Fulfillment of the Requirements forthe **Degree of Bachelor of Science in Computer Science and Engineering**

Supervised by

Ms. Faria Nishat Khan

Lecturer
Department of Computer Science and EngineeringDaffodil International University

Co-Supervised by

Mr. Md. Ashaf Uddaula

Lecturer
Department of Computer Science and EngineeringDaffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

May 14, 2025

APPROVAL

This Project titled **"Detecting Fake Apps Through User Review Analysis on the Google Play Store"**, submitted by Kazi Khalid Hasan Tamim and Md. Razib Hossain, ID No: **212-15-4129 and 212-15-4132** to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on **14 May, 2025**.

BOARD OF EXAMINERS

Dr. S.M Aminul Haque (SMAH) — **Chairman**
Professor & Associate Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Mohammad Jahangir Alam (MJA) — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

14/5/25

Mr. Md Mohammad Masum Bakaul (MB) — **Internal Examiner**
Sr. Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

14.05.25

Dr. Md. Arshad Ali (DAA) — **External Examiner**
Professor
Department of Computer Science and Engineering
Hajee Mohammad Danesh Science & Technology University

DECLARATION

We hereby declare that this project has been done by us under the supervision Ms. Faria Nishat Khan, Lecturer, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this projectnor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

14.5.25

Ms. Faria Nishat Khan
Lecturer
Department of Computer Science and EngineeringDaffodil International University

Co-Supervised by:

Mr. Md. Ashaf Uddaula
Lecturer
Department of Computer Science and EngineeringDaffodil International University

Submitted by:

Tamim 14-05-2025

Kazi Khalid Hasan Tamim
Student ID: 212-15-4129
Department of Computer Science and EngineeringDaffodil International University

14.05.2025

Md. Razib Hossain
Student ID: 212-15-4132
Department of Computer Science and EngineeringDaffodil University

ACKNOWLEDGEMENTS

This work would not have been possible without the support and contributions of many individuals over the past two semesters. We are deeply grateful to everyone who has as- sisted us in one way or another.

First, we express our heartfelt thanks and gratefulness to the almighty for His divine blessing making it possible for us to complete the **Final Year Design Project(FYDP)** successfully.

We are grateful and wish our profound indebtedness to **Ms. Faria Nishat Khan**, **Lecturer**, Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh. Deep knowledge and keen interest of our supervisor in the field of **Natural Language Processing (NLP)** to carry out this project. His endless pa- tience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting themat all stages have made it possible to complete this project.

We would like to express our heartfelt gratitude to the Head of the Department of Com- puter Science and Engineering, for his kind help in finishing our project and also to other faculty members and the staff of the Department of Computer Science and Engineering, Daffodil International University.

We would like to thank our entire course-mates at Daffodil International University, who took part in this discussion while completing the coursework.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

ABSTRACT

As mobile applications rapidly grow in popularity on platforms like the Google Play Store, users face an increasing threat from fake or malicious apps. Numerous applications take advantage of user trust by mimicking well-known apps or by manipulating the review system to seem more trustworthy. This research presents a deep learning-based approach to detect fake apps by examining user reviews using advanced Natural Language Processing (NLP) techniques. The review data was gathered from various apps on the Google Play Store and underwent necessary preprocessing. Sentiment and contextual clues were extracted utilizing transformer-based models. Three prominent pre-trained models, RoBERTa, DistilBERT, and BART-large, were fine-tuned to categorize apps as authentic or fake based on their reviews. These models were assessed with metrics such as accuracy, loss, ROC curve, and precision-recall scores. RoBERTa attained the highest accuracy at 99.12%, followed by DistilBERT at 98.42% and BART-large at 97.96%. A comprehensive system that integrates web scraping, model inference, and a user-friendly interface was developed. This platform enables users to enter an app link and receive instant predictions regarding its authenticity.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The digital revolution has radically changed the way consumers access products, services, and information. Efficient digital marketplaces like app stores emerged as robust platforms connecting developers to millions of users globally. One such multitude is the Google Play Store which is one of the biggest app stores that provides applications in hundreds of categories ranging across education, productivity, gaming, entertainment, and more. Although this convenience and accessibility is beneficial for users to a large extent, they equally expose them to certain risks, especially fake applications.

Fake apps are the "mimic" of the real applications and these fake applications exploit the users by acting for stealing users or property information, satisfy them with malware or scam. These applications not only put the security and privacy of their users at risk, but also undermine trust in digital ecosystems. However, the increasing sophistication of malicious developers has rendered traditional detection methods less effective, despite attempts by platforms like the Google Play Store to detect and eliminate fake apps on the platform. It requires new, data-driven solutions to solve the problem.

Machine Learning and Natural language processing to analyze the user-generated content (app reviews) is one of the most promising solutions. Reviews provide a treasure trove of information about how users experience these tricks, and they highlight suspicious behaviors or inconsistencies often associated with fake apps. A significant drawback of traditional detection methods is that they use compre¬hensive details, such as metadata or app structure, to avoid false positives and false negatives, while review-based analysis scrutinizes user feedback, providing a more flexible and adjustable method for identifying fraudulent apps.

This paper aims to create a strong framework to detect fake applications by processing user reviews from the Google Play Store. This framework utilizes state-of-the-art deep learning models including RoBERTa and DistilBERT to predict whether the app reviews indicate whether it is a genuine one or fake application. The models were trained and evaluated on a curated dataset comprising more than 30,000 reviews across all app categories, proving effective with an overall high accuracy in classification tasks.

The framework was implemented as an interactive web-based application deployed using Hugging Face Transformers and Streamlit to guarantee practical usability and scalability. The app takes a Google Play Store app permalink as input, scrapes reviews in real-time, and estimates whether the app is real based on model predictions. The deployment of the model enables real-world applications and offers a simple tool through which developers, users, and marketplace admins can

seamlessly detect fake apps.
Therefore, this research will enhance the security and performance of user and developer in-app transactions by providing cutting-edge latent vector construction in-app fraud detection.

1.2 Motivation

The increasing number of fake apps and the ineffectiveness of traditional detection methods demand a smarter solution. Fake apps steal data, spread malware, and harm user trust, while outdated techniques fail to catch them.

This project develops an innovative detection system that combines deep learning and NLP techniques to analyze user reviews for more accurate identification of fake apps. By using deep learning (RoBERTa, DistilBERT), we create a fast, scalable, and adaptive system that improves security and protects users.

Solving this problem not only enhances Google play store safety but also advances my skills in NLP and deep learning, contributing to a real-world issue with meaningful impact.

1.3 Objectives

1. Process user reviews to train a deep learning model using NLP to recognize the sentiment and rate them for app authenticity.
2. Explore applying different techniques for advanced text preprocessing to better preparetext data such as, tokenization, lemmatization, stop-word removal and sentiment analysis to boost results.
3. Train deep learning models using transformers for effective classification of fake and genuine apps based on user reviews.
4. Create a system that allows users to input app details (name, link, package) for real-time analysis and prediction
5. To find the best model based on hyperparameter optimization and whether the accuracy, precision, recall, and F1 score
6. Create an easy-to-use web app that uses the trained model.
7. Ensure scalability and efficiency of the system for handling large-scale app review datasets.

1.4 Methodology

The current study proposes a systematic technique to identify fake applications by analyzing the user reviews posted in Google Play Store. The structure of the method is divided into several stages: data collection, preprocessing, sentiment analysis, classification, model training, evaluation, and deployment. These steps are all important in building an accurate and reliable detection system.
The sample applications, user reviews, and related metadata like ratings and timestamps were collected using the google_play_scraper library. Reviews across different apps categories were mixed together into a more diverse dataset featuring similar amount of true and fake apps.

Tokenizing and removing of special characters, stop words, unnecessary columns, and other pre-processing were done. To ensure the quality of the data to perform the analysis, reviews were transformed into lowercase, tokenized and normalized. This helped to make sure there was more consistency and performance in model.
For sentiment analysis, the cardiffnlp/twitter-roberta-base-sentiment-latest model was used to classify data as negative, neutral, or positive based on softmax scores. Sentiment scores and user ratings were then used to categorise reviews as either fake or genuine.

Deep learning models such as Distilbert & BERT were investigated for more accurate classification. 80% of the dataset was used for training and 20% for testing, and models were trained with the AdamW optimizer, with hyperparameter tuning to boost performance.

Performance metrics of the model were measured and interpreted using accuracy, precision, recall, F1-score confusion matrix and ROC-AUC score. Training multiple models and comparing the results of some deep learning approaches first to understand the best approach.
The last step is to deploy this trained model as a web-based system, where someone can enter an app name or link and obtain an authenticity prediction in real-time. The next steps will include data expansion, classification model refinement, and the addition of more app metadata to improve fraud detection.

1.5 Project Outcome

Improved Detection Framework:
A novel deep learning system combining RoBERTa and DistilBERT to analyze user reviews with higher accuracy than traditional methods. Demonstrated effectiveness through testing on 30,000+ real-world app reviews across diverse categories.

Real-World Deployment:
A scalable web application (built with Hugging Face and Streamlit) for real-time fake app detection by users, developers, and platform moderators. Enables proactive identification of suspicious apps before they cause harm.

Dataset & Benchmark Contributions:
A publicly available labeled dataset of app reviews annotated for fake app indicators to support future research. Performance benchmarks comparing transformer models against rule-based and classical ML approaches.

Cybersecurity & User Protection Impact:
Reduces risks of data theft, financial scams, and malware infections by improving fake app detection rates. Strengthens trust in app marketplaces by providing transparent, AI-driven fraud analysis.

Research Advancements:
Validates the effectiveness of NLP-based review analysis for security applications.
Provides a foundation for extending this approach to other fraud detection domains (e.g., phishing, fake websites). By bridging the gap between cutting-edge AI and practical cybersecurity needs, this project delivers both technical innovations and measurable societal benefits.

1.6 Organization of the Report

The report is divided into six chapters, each of which offers an in-depth look at the process and findings of the project.

Chapter 1: Introduction is the first chapter which presents the project by explaining its motivation, objectives and area of operation. It stresses the significance of the study and presents the major research questions that the study addresses. The chapter ends with a summary of what the analysis is expected to accomplish and an overview of the content of the report.

Chapter 2: The relevant background, such as important terminology and context, are presented in Section 2. It analyses the literature and contrasts studies to uncover the extent of the problem and the complexity thereof. Moreover, this chapter performs a gap analysis to determine where the prior research is inadequate.

Chapter 3: Research Methodology describes the proposed research methodology, containing data collection, data set description, and applied preprocessing method. This chapter also delivers a more thorough description of the model used in this study.

Chapter 4: Experimental Results and Discussion presents and discusses the experimental results, offering an in-depth analysis of the findings and their implications.

Chapter 5: Societal, environmental and sustainability Impact directs attention to the wider societal implications of the research, specifically its societal impact, environmental issues, ethical considerations and sustainability.

Chapter 6: Conclusion and Future Scope highlights that RoBERTa outperformed other models in fake app detection using user reviews, proving the effectiveness of NLP and deep learning. Future work may include larger datasets, and support for more platforms.

At the end, the report includes a list of references, which appears in the body of the paper.

Chapter 2

Background

2.1 Introduction

The proliferation of mobile applications has significantly transformed user engagement, with app reviews serving as a pivotal source of feedback. However, the authenticity of these reviews is often compromised by fake or manipulated content, leading to misinformation and potential harm to users and developers alike. Detecting such fraudulent reviews necessitates sophisticated techniques that can effectively analyze and interpret user sentiments. This chapter delves into the existing literature on fake review detection, highlighting the methodologies employed, key findings, and the existing gaps that our proposed model aims to address.

2.2 Literature Review

Table 2.2.1: Literature review.

Author(s)	Year	Title/Topic	Methodology	Key Findings
Martens & Maalej	2017	Fake Reviews in App Stores	Behavioral analysis, NLP	Detected fake reviews using linguistic and behavioral patterns.
Gupta & Kamthania	2021	Sentiment on Google Play Apps	Sentiment Analysis, NLP	Identified trends in app reviews showing user satisfaction.
Panigrahi et al.	2017	Sentiment Analysis on Play Store Reviews	Text preprocessing, Sentiment Scoring	Analyzed polarity to improve app evaluation insights.
Gaur & Sharma	2017	Sentiment Analysis in NLP	NLP techniques, TF-IDF	Highlighted effectiveness of syntactic models in sentiment classification.
Sagar & Kiran	2020	Toxic Comment Classification	CNN, LSTM, NLP	Achieved high accuracy in toxic comment detection using deep learning.

Aldabbas et al.	2019	Google Play Scraping with NLP	NLP, Knowledge Engineering	Extracted review insights and patterns from Google Play data.
Dudhankar et al.	2020	Google Play Sentiment Analysis	SVM, Naive Bayes, TF-IDF	SVM achieved higher accuracy than Naive Bayes in sentiment classification.
Ahmad et al.	2020	Fake News Detection	Ensemble ML (XGBoost, RF)	Ensemble methods improved fake news detection significantly.
Salminen et al.	2020	Fake Review Generation and Detection	GPT, ML classifiers	GPT-based systems outperform traditional models in fake review detection.
Kavitha et al.	2021	YouTube Comment Classification	Decision Tree, SVM	Identified spam, offensive, and irrelevant comments effectively.
Haque & Islam	2021	Airline Tweet Sentiment Analysis	NLP, ML Techniques	Compared several ML algorithms for Twitter sentiment classification.
Panthaplackel et al.	2020	Code Comment Updates	NLP, Code Change Modeling	ML improves accuracy in updating code comments based on code changes.
Dang et al.	2020	Deep Learning for Sentiment Analysis	CNN, RNN, Hybrid Models	Hybrid DL models outperformed single architecture in sentiment tasks.
Panthaplackel et al.	2020	Comment-Code Entity Linking	CRF, Source Code Embedding	Efficient association of comments with source code via semantic links.
Chang et al.	2020	Fake Comment Detection via Sentiment	Lexicon, Sentiment Analysis	Sentiment-based methods efficiently separate fake and real reviews.

Gaurav et al.	2021	Fake Review Detection in OSNs	NLP, Deep Learning	Proposed robust framework for fake review detection on social platforms.
Tran et al.	2016	Attitude Detection via Facebook Comments	Real-time/Batched Sentiment Analysis	Detected user attitudes from real-time Facebook comments.
Ahmad et al.	2020	Fake News Detection via ML Ensembles	SVM, RF, XGBoost	Ensemble learning enhances fake content classification.
Shcherbakov et al.	2016	Social Media Sentiment for Fake News	NLP, Sentiment Scoring	Social media sentiment patterns help spot fake news trends.

2.2.1 Similar Applications

Several research studies and applications have explored fake review detection and sentiment analysis using NLP and machine learning. Martens & Maalej (2017) analyzed app store reviews to detect fake patterns, while Salminen et al. (2020) developed systems to generate and detect fake product reviews using deep learning. Mobile and web apps such as Fakespot and ReviewMeta also analyze user reviews to assess authenticity. Methodologies commonly include clustering, sentiment scoring, and classification models like SVM, LSTM, and ensemble methods, providing a foundation for our system to detect fake apps based on user reviews.

2.2.2 Related Research

They suggest a batched process of performing data on sentiment analysis in this paper. It provides details of data collection from Facebook comments, analysis and clustering by utilization of NLTK Sentiment analysis, k-means and MB-means algorithms. The study emphasises the importance of clustering to detect patterns and normal behaviour of sentiment scores [1].

In their work, they proposed applications of six different machine learning classifiers to fake news detection with tokenization and stop phrases as the first step. A pre-assembled set of fake news data from Kaggle was used. Removing of stop-words at preprocessing. Data was vectorized using counter vectorization, and tested with different machine learning models using statistical methods. By varying the amount of labeled data used in training, the results demonstrated increased classification accuracy as measured by accuracy, recall, precision, and F1-score [2].

The textual data used in this article comes from Amazon 1930 data on commodities and food. They deal with the problem of identifying phony online remarks. It presents a novel approach based on the CFSFDP clustering algorithm and sentiment analysis. Fake comments are detected using the CFSFDP clustering method, which chooses clustering centers based on their high density and great separation from other centers. It offered a more efficient way to identify phony comments [3].

Code summaries, commit messages, and API comments are all discussed in this article as essential forms of developer communication. Sequence labeling and binary classification are the two classification strategies that are employed. Four fully-connected layers make up the binary classifier, a feedforward neural network that independently determines whether code tokens are related to the noun phrase or not. Taking into account the relationships between nearby tokens, the sequence labeling model is a conditional random field (CRF) that simultaneously labels code tokens according to their association with the noun phrase [4].

Their research on public opinion, especially through social network sentiment analysis, yields useful data. They used deep learning models for sentiment analysis tasks, such as word embedding techniques and TF-IDF. Lexicon-based techniques, machine-learning-based techniques, and hybrid approaches that mix lexicon and machine-learning-based approaches are used in deep learning, CNN, RNN, and DNN, and sentiment analysis [5].

This research proposes a model that outperforms baselines in updating code comments based on code changes. It enhances developer consistency and communication by automatically updating code comments in response to code changes. While the edit model focuses on modeling edit sequences, the generation model is trained on GitHub method/comment pairings. METEOR, BLEU-4, SARI, GLEU, and exact match are evaluation metrics. To supplement the automated measures, human review is also carried out [6].

The significance of customer satisfaction in the cutthroat business environment of today is covered in this essay. It draws attention to the difficulties businesses have in comprehending and meeting the wants of their clients, which can result in a decline in client loyalty and higher marketing expenses. In order to address these issues, the study suggests using Sentiment Analysis, which combines Natural Language Processing (NLP) and Machine Learning (ML). With a 77% accuracy rate, the study presents two natural language processing (NLP) methods, Bag-of-Words and TF-IDF, as well as a number of machine learning (ML) classification algorithms, such as Support Vector Machine, Logistic Regression, Multinomial Naive Bayes, and Random Forest [7].

Using the criteria of relevance to the video content, the study "Analysis and Classification of User Comments on YouTube Videos" analyzes and classifies user comments on YouTube videos. Four classes—relevant, good, negative, and irrelevant—were manually created using comments taken from YouTube videos. Precision (P), recall (R), and accuracy (A) metrics were used to evaluate the classified remarks. These metrics can be calculated using the given formulas. utilizing nine distinct learning classifiers, such as function-based, Bayesian, and decision tree classifiers. To identify spam comments on YouTube, a number of common machine learning techniques have been used, including Random Forest, Support Vector Machine (SVM), Naïve Bayes, N-grams, K-Nearest Neighbors classifiers, and Logistic Regression [8].

The study focuses on identifying fraudulent reviews, which compromise the validity of product reviews found online. An Amazon dataset is utilized to create phony reviews using two language models, ULMFiT and GPT-2. The superior model turns out to be GPT-2. For the classification task of identifying fraudulent reviews, a dataset is generated. They make use of the OpenAI false detection model and the support vector machine (SVM) technique. A popular baseline method for NLP jobs, SVM is renowned for its reliable performance. NBSVM, which blends SVM with Naïve Bayes features, is the particular variant of SVM that is employed [9].

They talk about the difficulties presented by fake news as well as the effects of social media and the Internet on the spread of news. It emphasizes how computer methods, such as natural language processing (NLP), are used to identify and categorize fake news using textual material. Random Forest, Boosting Classifiers (XGBoost), Perez-LSVM, Linear SVM, Multilayer Perceptron, Bagging Classifiers, Wang-CNN, and Wang-Bi-LST [10].

They talk on the significance of sentiment analysis and opinion mining, especially when it comes to mobile applications. It draws attention to how difficult it is to keep up with the ever expanding number of apps available. Sentiment analysis of Google Play Store application evaluations is the goal of the suggested design. The purpose of machine learning, a branch of artificial intelligence (AI), is to comprehend data structures and incorporate them into models that humans can use. The significance of machine learning algorithms in everyday life is also emphasized in the text, especially when it comes to solving classification and regression issues. The

performance of regression and classification algorithms is evaluated using metrics like mean squared error (MSE), precision, recall, and F-score [11].

In this work, machine learning methods like Multinomial Naïve Bayes, Random Forest Classifier, and Logistic Regression were used to assess user reviews and ratings from the Google Play Store. The best algorithm for review analysis was determined to be logistic regression, which had a high classification accuracy for reviews that were neutral, negative, or favorable. Google Play Store reviews are scraped, language analysis is applied, and different categorization algorithms are used to statistically examine and assess the reviews. Factors including precision, recall, accuracy, and F1 score are used to identify the optimal algorithm [12].

This work employs Deep Learning techniques including tokenizing, stemming, and embedding to classify text in order to identify hazardous remarks. Other models such as LSTM, Naive Bayes Support Vector Machine, Fasttext, and Convolutional Neural Network are contrasted with the suggested neural network model. According to reports, the suggested model's accuracy is 98.15% [13].

This paper suggests methods to improve the efficacy and upkeep of natural language components in software repositories, including to-do and API comments. It suggests a number of methods, such as automatically identifying dangling comments, responding inquiry comments, making trigger-action comments executable, synthesizing code from task comments, and enhancing comment quality. In order to automatically identify hanging to-do comments in source code, the authors of this work are creating a machine learning-based method. In addition to samples of deleted comments, they are using a sizable dataset of more than 700k todo comments. They seek to extract context and examine the co-evolution of code and comments by utilizing commit messages, static code analysis, and software histories [14].

The purpose of this study was to examine weight-loss and weight-tracking applications available on the Google Play Store in order to gather user opinions and preferences. To evaluate the connection between app complexity and popularity metrics, 179 mobile apps were examined utilizing content analysis and correlational analyses. Fifteen weight-tracking applications were chosen from this selection, and framework and theme analysis were used to look at 569 user reviews. In order to collect weight loss apps from the Google Play Store, the researchers used a sampling approach. By utilizing the search phrases "weight loss" and "weight track*," 249 mobile applications were found. After screening 120 apps for the term "weight loss" and 110 apps for the term "weight track*," saturation was achieved. Over the course of two days, the apps were sampled until no theoretically novel apps were discovered [15].

NLP and sentiment analysis are used in this paper. Sentiment analysis, sometimes referred to as opinion mining, is a method for recognizing and categorizing sentiments in text by using natural language processing. The NLTK libraries in NLP are also discussed. NLTK is a top platform for natural language processing (NLP) Python programming. It offers interfaces to text processing libraries, lexical resources, and corpora. NLTK is extensively utilized in both industry and research and is free and open source. It is commended for its adaptability and vibrant community and provides the Naive Bayes algorithm for classification [16].

The sentiment analysis of Google Play Store app reviews was the main focus of this study. For suppliers and other interested parties, examining these reviews might yield insightful information. Both supervised and unsupervised machine learning techniques are used in the suggested system for user review sentiment analysis. There are two primary methods used: the corpus-based method and the dictionary-based

method for sentiment analysis. An essential step in sentiment analysis is feature extraction, which includes stop-word removal, tokenization, sentence splitting, parts-of-speech tagging, and named entity recognition (NER) [17].

Using a Kaggle dataset, this research suggests a sentiment analysis method for Google Play Store apps. The dataset used in the study was sourced from Kaggle.com. Ten thousand apps that are available on the Android Play Store are included in this collection. Using sklearn-extracted bag-of-words features, a logistic regression model was used for classification. A tri-polar classifier was added to the model, enabling the classification of positive, neutral, and negative sentiment. The dataset was divided into 70% for training and 30% for testing, and sentiment analysis was carried out at the document level [18].

The growing amount of user reviews and ratings in app stores is the main topic of this article. To learn more about the tactics and products of forty-three fictitious review providers, the researchers administered disguised questionnaires to them. They discovered notable variations in a number of areas, such as the related apps, reviewers, rating distribution, and frequency, when they contrasted 60,000 fraudulent evaluations with 62 million authentic reviews from the Apple App Store. The creation of a basic classifier to automatically identify phony app store reviews is presented in the study. On a labeled and unbalanced dataset, which represents a tenth of the fraudulent reviews, the classifier obtained a 91% recall and an AUC/ROC value of 98% [19].

2.3 Gap Analysis

Table 2.3.1: Gap Analysis.

Identified Gap	Existing Work	How Our Project Addresses It
Limited focus on app reviews for fake detection	Most studies focus on news articles (e.g., Arjun Roy et al., 2018) or generic reviews, not specifically Play Store app reviews.	We directly target Google Play Store app reviews to identify fake patterns using review text.
Lack of usage of transformer-based models like RoBERTa or DeBERTa	Studies primarily use SVM, Naive Bayes, or CNN/LSTM (e.g., Ahmed et al., 2020; Goyal Dev et al., 2024).	Our model uses advanced transformer models (RoBERTa, DistilBERT, DeBERTa) for better contextual understanding and classification.
Absence of a real-time detection system	Existing systems are mostly batch-processed and offline.	We propose a real-time fake review classification system with a user interface for dynamic input and prediction.

2.4 Summary

This chapter has provided a comprehensive review of existing literature on fake review detection, sentiment analysis, and the application of Natural Language Processing (NLP) and Machine Learning (ML) in mobile app ecosystems. Various approaches such as clustering techniques, deep learning models, traditional classifiers (e.g., SVM, Naive Bayes), and ensemble methods have been explored across different studies. The review highlights the effectiveness of hybrid models and sentiment-based feature extraction in enhancing classification performance. Despite significant progress in this field, most existing studies focus either on general product reviews or lack domain-specific customization for app marketplaces like the Google Play Store. This reveals a notable gap in context-aware, automated systems for fake app detection based on user reviews. The insights gained from this review form the foundation for the development of a more robust and targeted fake app detection system in the subsequent chapter.

Chapter 3

Research Methodology

3.1 Methodology

The approach that we followed in this project aims to create a stable, generalizable, and precise solution to identify spoof applications relying on the review text data provided by the users. The complete procedure can be divided into several main stages: data obtaining, data processing, sentiment analysis, model construction and evaluation, and system implementation. Every step is specifically designed to help the building of the detection framework successfully. Leveraging the power of natural language processing with deep learning models, the system can correctly identify whether an app is authentic or fake, using user reviews.

3.1.1 Overview

The research work starts with examining the user reviews of Google Play Store through web scraping. This raw data must be cleaned and pre-processed to filter out the noise, normalize the text and make it ready for analysis.

After preprocessing, the comments undergo sentiment analysis utilizing pre-trained RoBERTa model to deduce the sentiment score, thereby being able to extract more detailed information of the users' opinion.

The processed reviews, along with their sentiment labels, are then used to train deep learning classification models (such as RoBERTa and DistilBERT) to distinguish between genuine and fake apps. The effectiveness of models are measured using different performance metrics including accuracy, precision, recall, and F1-score.

Lastly, the most successful model is deployed with a simple interface, made in Streamlit and Hugging Face Transformers, able to detect a fake app in real time as it recieves input from the user.

3.1.2 Proposed Methodology

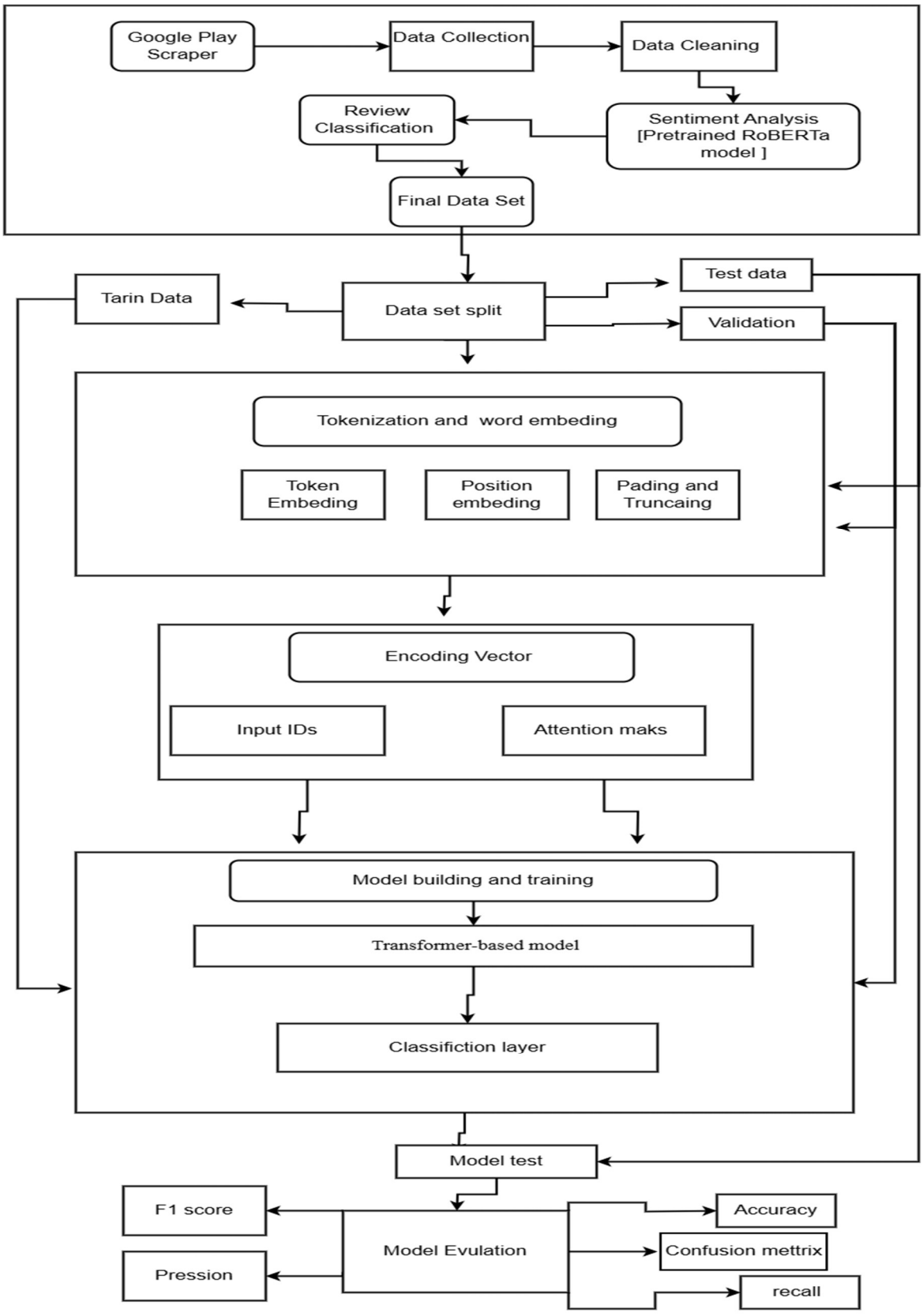

Figure 3.1.1: Proposed Methodology.

3.1.3 Functional and Nonfunctional Requirements

Functional Requirements:

- Accept a single input field in which users could insert a valid GooglePlay Store app link.
- Scrape the first 1000 user reviews associated with the app.
- Clean and preprocess the acquired reviews by removing special characters, converting the text to lowercase, removal of stopwords and tokenizing.
- Apply the model on the review data and classify the app into "Genuine" and "Fake".
- Display the result ("Genuine" or "Fake") to the user in a clear and concise manner.
- Handling: If the link is not a valid link, show some sort of error message without crashing.

Nonfunctional Requirements:

- Usability: It needs to be a simple UI, with nothing more than an input box and an output area for results.
- Performance: The system needs heavy scraping and predicting in seconds to have a better customer experience.
- Scalability: The burst model, as well as the scraping pipeline, need to cope with concurrent requests without much waiting.
- Reliability: This approach cannot persistently scrape, analyze, and predict well for various app links.
- Security: Input should be validated to avoid injecting attacks or malformed requests causing damaged systems.
- Maintainability: Hugging Face deployment should support one-click model or scraping logic updates.
- Portability: The system implemented should be available for use using web and mobile browsers.

3.1.4 Context Diagram

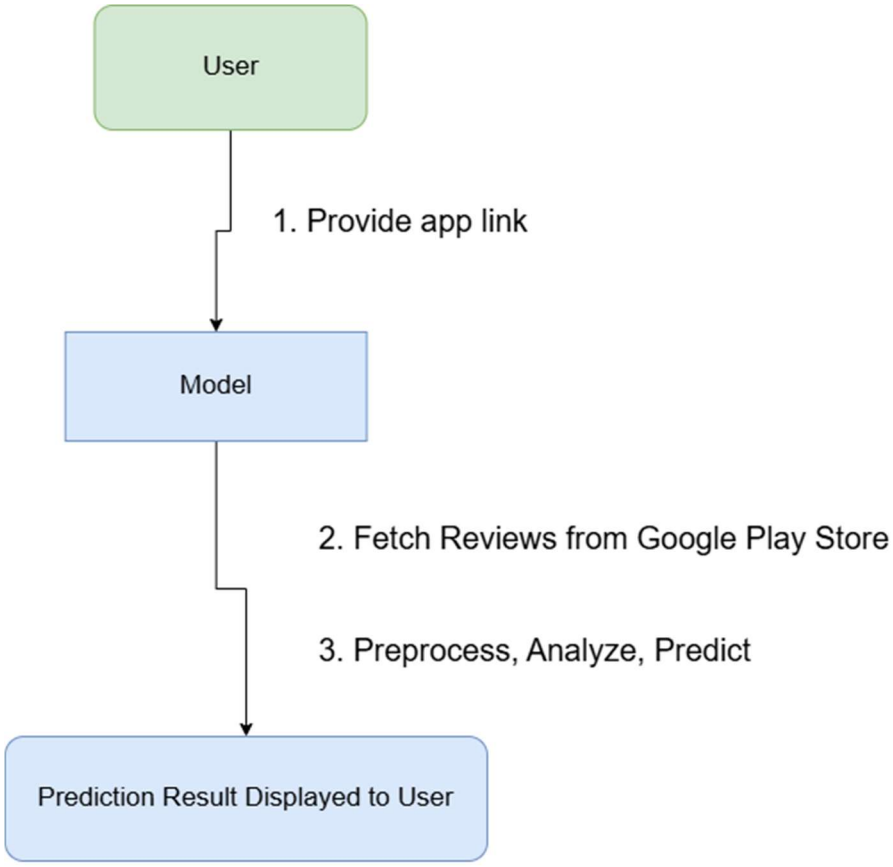

Figure 3.1.2: Context Diagram.

3.1.5 Data Flow Diagram Level 1

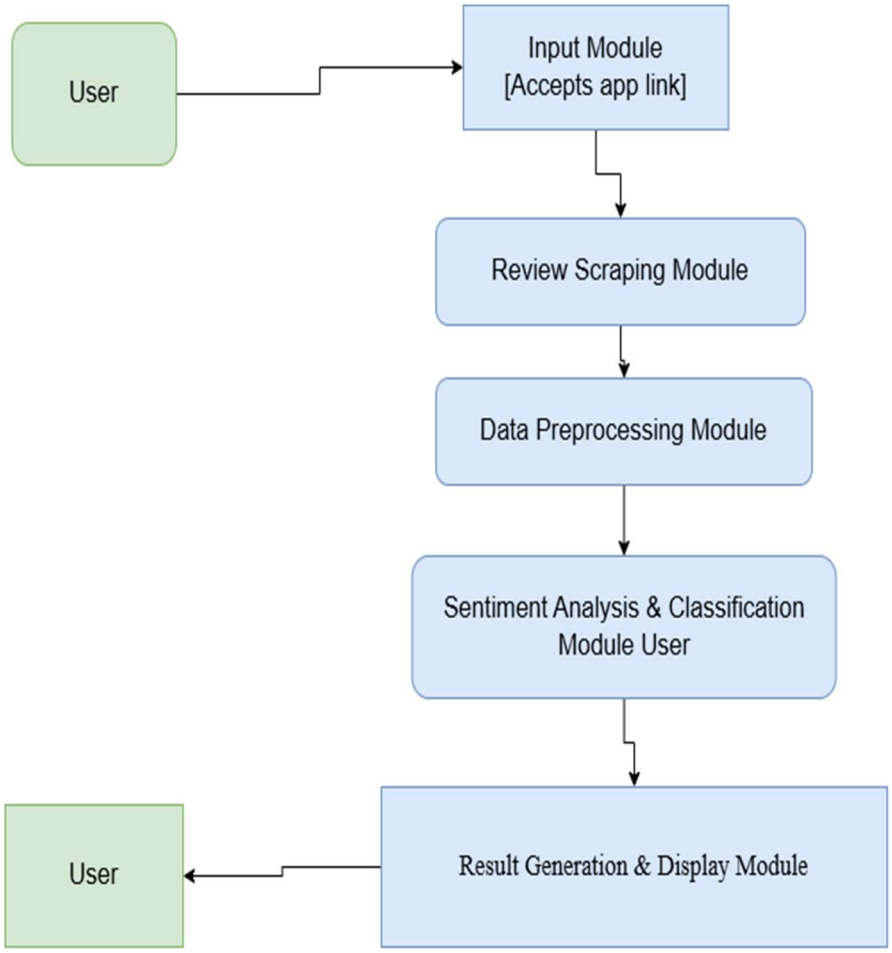

Figure 3.1.3: Data Flow Diagram Level 1.

3.1.6 UI Design

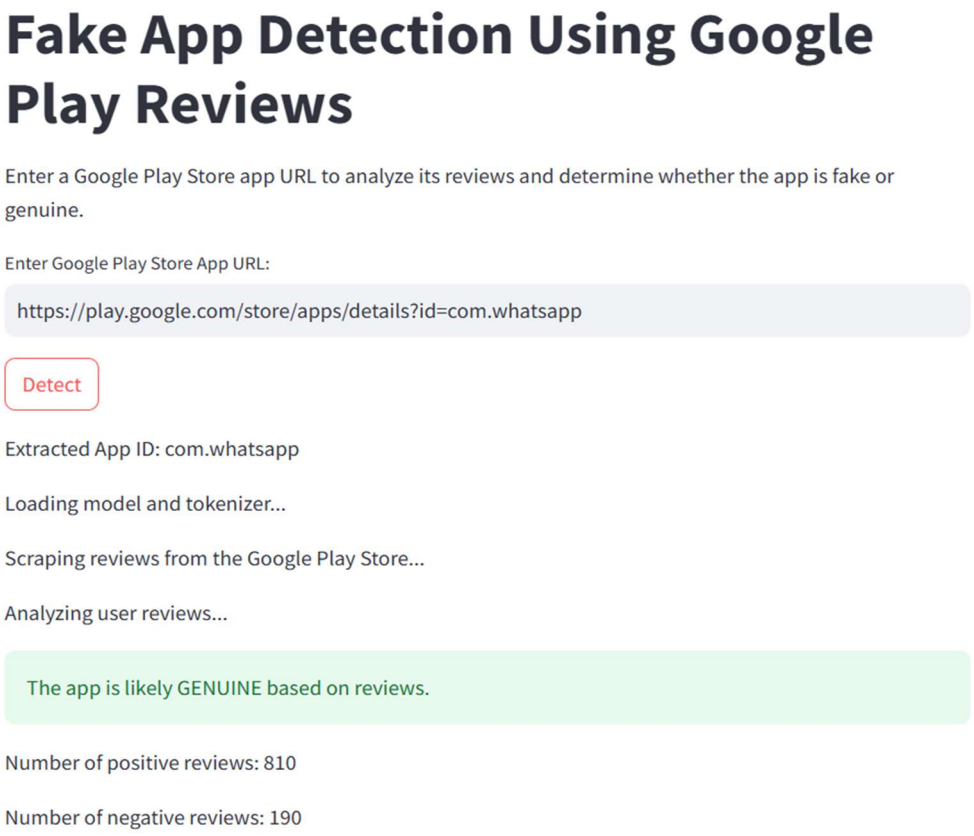

Figure 3.1.4: UI For Genuine Application.

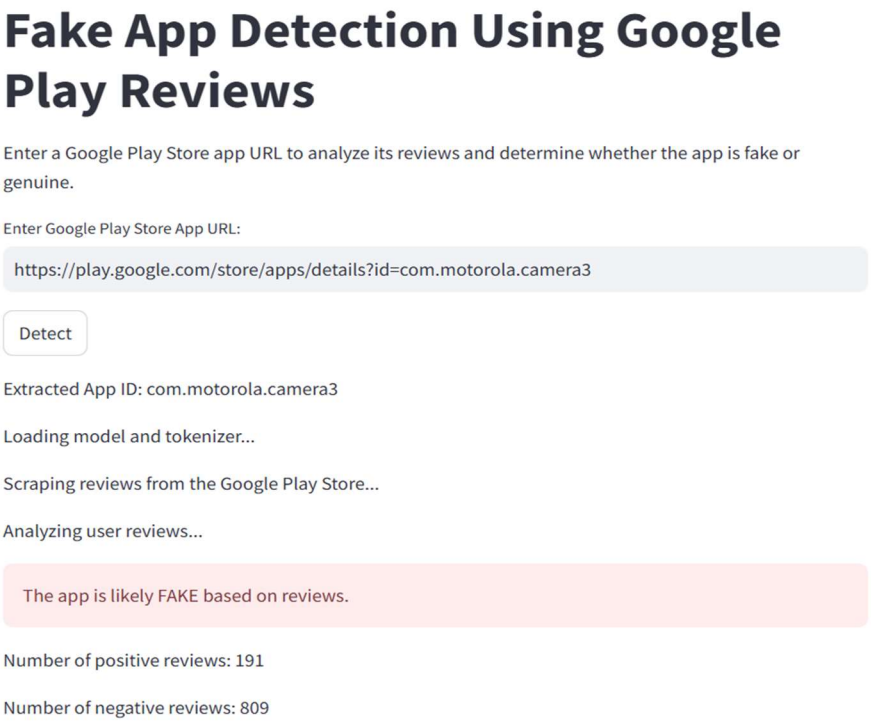

Figure 3.1.5: UI For Fake Application.

3.2 Detailed Methodology and Design

3.2.1 Alternative Approaches Considered:

Metadata Analysis:

Predicting the legitimacy of an app based on app meta-information, such as developer name, download frequency, version history and permissions. Many fake apps today mimic trusted metadata, so this is harder to trust.

User Review Sentiment Analysis (Selected Approach):

Users reviews analysis on your services by NLP and Deep Learning.

Reason for Selection: Easier to access publicly available reviews. The ratio of publicly available reviews you can easily check. Reviews frequently are reports of unhappy users, scams or indication of malware. Easily automated and scaled.

Therefore, we chose the user review analysis as the most promising and scalable option.

3.2.2 Data Collection:

For this project, user reviews from different apps located in Google Play store were used to form useful dataset in identifying fake application. We handle the data collection process through utilizing the google_play_scraper Python library which provides an easyto-use way to extract all the user comments without unnecessary complex APIs. Reviews were also crawled from some popular apps, e.g., MyFitness, AppLocker, Candy Crush, Finger Scanner, WiFi Scanner and X-Ray.

3.2.3 Dataset Description:

There are more than 30,000 user reviews in the final dataset. Reviews were tagged according to the general consensus expressed in the reviews for that app. Apps having predominantly positive and similar reviews were defined as positive and those that had a high frequency of negative reviews, or were behaving suspiciously, or were reported as scams, were defined as negative. This data is an organized data in this project to carry out the subsequent analysis, preprocessing as well as model training.

Dataset Sample:

Table 3.2.1: Dataset Sample

Rating	Comment	Label
5	excellent app for tracking nutrition outstanding with an easy learning curve	positive
4	app works quite well for my needs the intruder self feature is great	positive
5	candy crush saga is a fun game i love it	positive
1	one thing is very bad this version is cannot hide from screen	negative
2	this app has too many long adverts one after the other and are disruptive and find difficult to immediately exit or find exit button	negative
1	it suddenly delete all the photos inside the vault without my permission	negative

3.2.4 Analysis Technique:

The review process consisted of two stages. Sentiment analysis First we performed sentiment analysis using the cardiffnlp/twitter-roberta-base-sentiment-latest model, a RoBERTa-based transformer that was fine-tuned on the sentiment classification data. Each review was input into the model, which outputs probabilities for the three sentiment classes (negative, neutral and positive) using the softmax function. Each review was assigned one of the three sentiment categories depending on the maximum likelihood score.

After sentiment classification, a rule-based strategy was adopted to further classify fake vs. genuine apps. A review is considered to be fake if the negative sentiment score (roberta_neg) is larger than 0.5 and the original app rating corresponding to the review is 3 or fewer stars. On the other hand, a review was considered authentic if its positive sentiment score (roberta_pos) was higher than 0.5 and if the rating was at least 3 stars. The two-condition method made it possible to exploit both sentiment and user rating to enhance the effectiveness of fake app detection.

3.2.5 Data preprocessing:

Data preprocessing used to prepare user reviews for sentiment analysis and to classify fake apps was a crucial point in this work. In the beginning, we cleaned the raw reviews collected from the Google Play Store by changing all the text to lower-case letters to remove case sensitivity of the text. Punctuation, emojis and special characters and numerical digits were removed to rule out noise. After the cleaning, the usual stopwords were eliminated, so as to maintain the core of each review. The sentences were subsequently tokenized to words and broken down into individual words by sentences, that is the input format of the RoBERTa model. Whereas, the blank or very short reviews have been eliminated, in a way it allows to pass only informative and data-rich reviews to the stage of sentiment analysis. This systematic pre-processing steps contributed to the improved quality and the robustness of the final classification results.

3.2.6 Data Analysis and Visualization:

Loading the data and plot a count plot to see the distribution of the app ratings. This helped understanding how users rated various apps, which was mostly bunched around the upper being the higher scores.

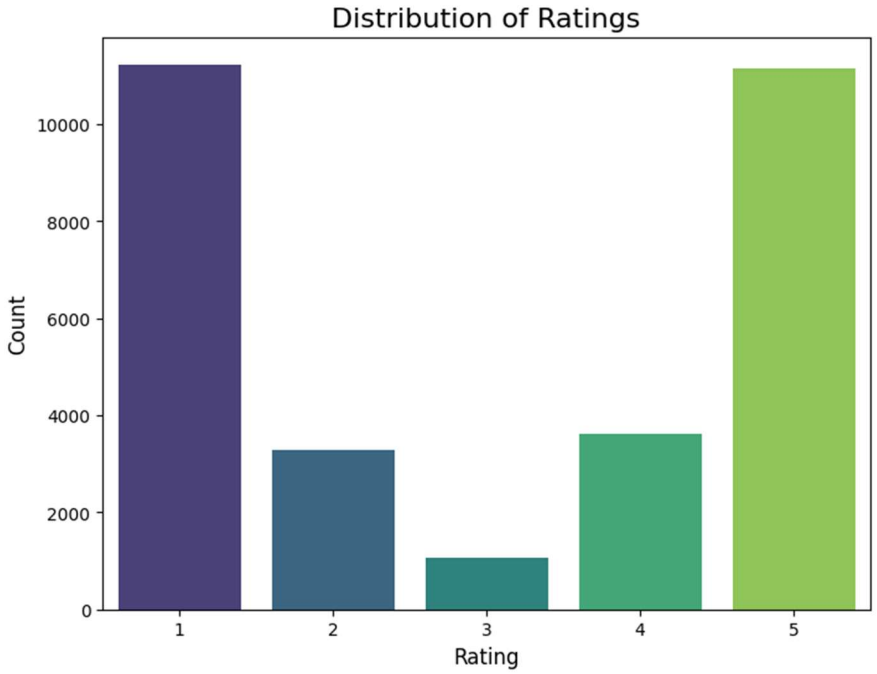

Figure 3.2.1: Rating Distribution.

Subsequently, the distribution of the user comment lengths was studied. This Spread was examined with a histogram and KDE curve graph. This analysis helped bring out if users usually rated and left short reviews or longer ones.

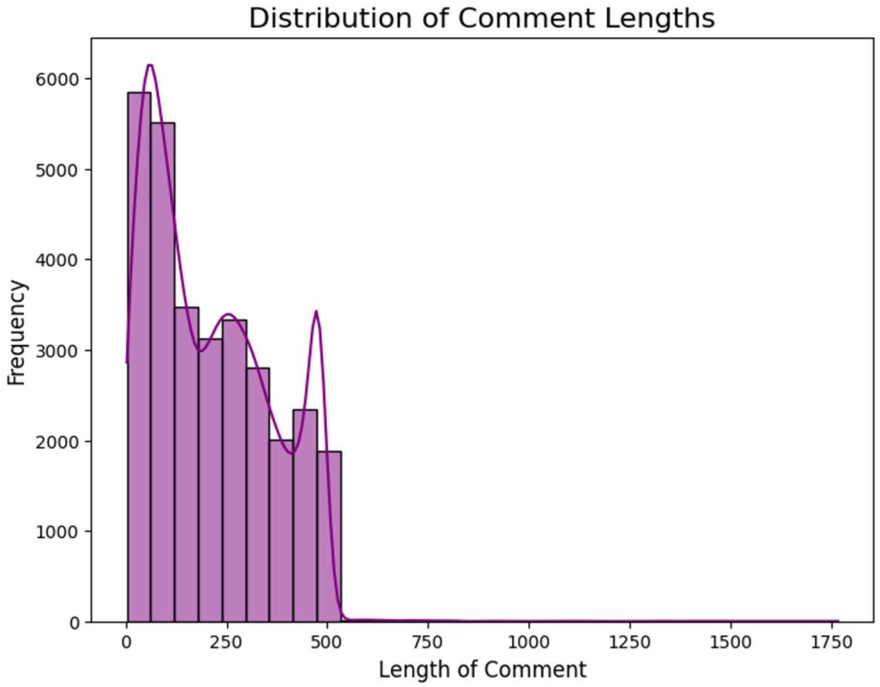

Figure 3.2.2: Comment Length Distribution.

A pie chart was used to visualize the performance of labels (i.e, positive and negative) assigned to apps. This visualization also immediately revealed the balance of the dataset among the two classes.

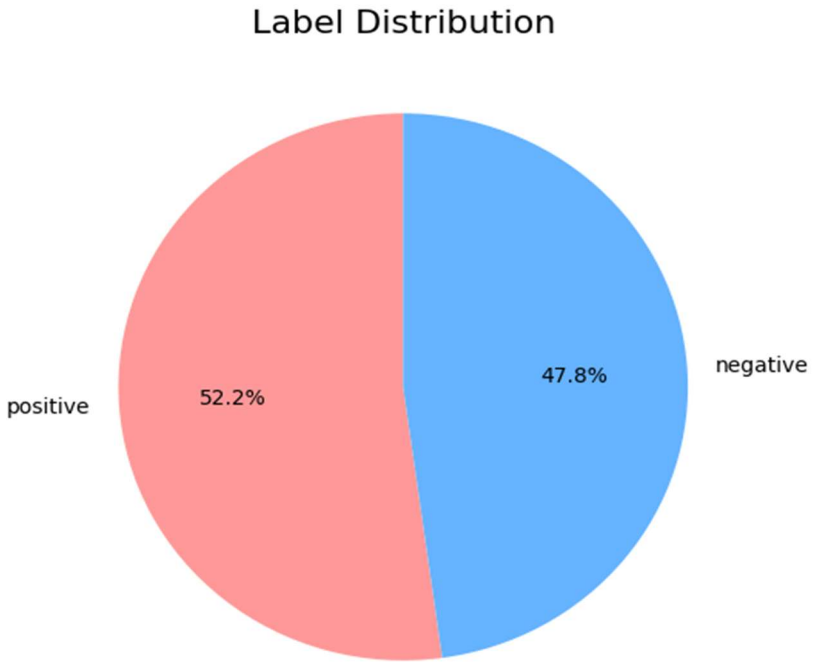

Figure 3.2.3: Label Distribution.

A word cloud was constructed to summarize the what people frequently discuss about the apps. This allowed to be aware of the context and the sentiment of the reviews, before the application of the model.

Figure 3.2.4: Word Cloud of Comments.

Lastly, we created a grouped bar chart to present ratings across "positive" and "negative".

This examination enabled scrutiny of the correlation of the user ratings with the legitimacy of apps.

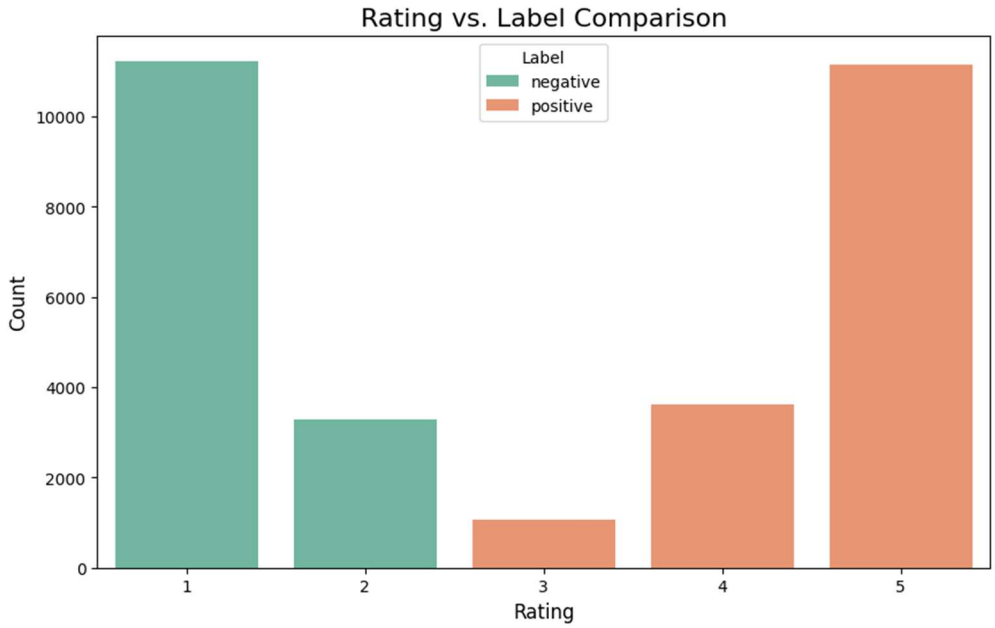

Figure 3.2.5: Rating vs. Label Comparison.

Proposed Model:

We also adopted one of the pretrained transformer models to classify apps to "Genuine" or "Fake" based on user reviews. In particular, RoBERTa, DistilBERT, and BART-large were chosen because they have been shown to be effective in many Natural Language Processing (NLP) tasks.

RoBERTa:

Given that RoBERTa is a deeply optimized model over BERT, the deeper interpretation of review sentiment is possible by considering not only the context of comments but also grasping the subtleness of user comments.

DistilBERT:

DistilBERT was also used as a lightweight version of BERT to provide an incremental approach to begin testing a lighter and speedier architecture. Even though it has a smaller number of parameters, DistilBERT achieves 95% of BERT performance and is considered a faster, smaller and free alternative to BERT for classification and other tasks, which perform preferentially on TPU or TPU-Nodes.

BART-large:

We also employed the BART-large model, which combines the strengths of bidirectional and autoregressive transformers. Its encoder-decoder architecture allows it to generate high-quality representations of input text, making it especially effective for understanding complex user reviews. BART-large demonstrated strong performance in classifying reviews with high precision.

3.3 Project Plan

For systematic development and evaluation, the project was split into stages. Phase I dealt with data gathering and cleaning: we collected reviews from Google Play Store and pre-processed data for further analysis. Phase 2 specifically focused on model exploration using different models such as RoBERTa, DistilBERT, and BART-large for the classification of the reviews. Phase 3 involved developing the system, which involved creating a system, based on Hugging Face, for input processing, review scraping, classification and presentation of results. Finally, Phase 4 targeted the evaluation and deployment, and system performance was tested and the system was deployed for public use. Deliverables and timelines in each phase were well organized and executed in the whole the project.

3.4 Task Allocation

Two members took on tasking for this task. One team member was responsible for the data collection and pre-processing, the literature review and providing support. A second individual, was responsible for the model selection, training and system development. Both members worked on the final report together to cover all parts of the project.

3.5 Summary

In this section we have highlighted the methodology and design of the project, i.e., how a system was to be developed to detect fake applications using user reviews from Google Play Store. The chapter was based on processes such as data acquisition, model selection and system formulation. It also explored how responsibilities and jobs were divided between teammates to make the project process smoother. The project was divided into four stages: data collection and pre-processing, model exploration, system development and evaluation and deployment. We had carefully planned each stage and set a series of objectives carefully designed to guarantee the success and completion of the project on time.

Chapter 4

Implementation and Results

4.1 Environment Setup

For the realization of the fake app detection system, we set the environment, and used Google Colab as the main development environment to be collaborative by using GPU. Model training, data manipulation, and visualizations were done using python libraries such as transformers by Hugging Face, pandas, matplotlib and seaborn. We used the google_play_scraper library to gather user reviews from the Google Play Store. We used Jupyter notebooks for code authoring and for code testing because of their interactive behaviour. Model building and inference was done on Colab which helps in easy sharing and accessing in a team. MS Word and Google Docs were used whenever possible for documenting and writing reports about the project. The environment setup was designed to efficiently process the data, train the models, and present the results as a shared, resource-efficient platform.

4.2 Comparative Analysis

In this work, we used three transformer-based pre-trained language models BART-large, DistilBERT, and RoBERTa to classify Google Play Store app reviews as "Genuine" or "Fake." The models were tuned and evaluated in terms of their classification performance for the test dataset.

Among the three, RoBERTa performed the best with accuracy score of 99.12%, demonstrating good contextual learning capabilities, and redundant structure for classification tasks like sentiment and intent. Due to its good performance in learning subtleness in user-generated patterns, it was the most appropriate model for the job of detecting fake applications based on review patterns.

DistilBERT, a lighter, faster version of BERT, performed suprising results with accuracy 98.42%. Though computationally faster, their CROCS model achieved a remarkable performance and could be a good approach in applications of limited resources and real-time response time.

Even for a BART-large, which is a model designed specifically for sequence-to-sequence tasks such as summarization and translation, we were able to achieve competitive accuracy of 97.96 %. Despite being a little slower, its performance was comparable, which speaks to its overall performance at classification tasks when finetuned accordingly.

In general, the performance of all models was excellent, and substantially superior to random, with RoBERTa having the highest accuracy. The high performance achieved by all models again indicates how the transformer-based architectures can be powerful tools for processing user reviews to identify fake apps.

4.3 Results and Discussion

RoBERTa:

Post-Training Model Assessment:

Table 4.3.1: Post-Training Model Assessment For RoBERTa Model.

Matrix	Value
Loss	0.0281
Accuracy	0.9912

Classification Report:

Table 4.3.2: Classification Report For RoBERTa Model.

	precision	recall	f1-score	support
Fake	0.99	0.99	0.99	2156
Genuine	0.99	0.99	0.99	2401
accuracy			0.99	4557
macro avg	0.99	0.99	0.99	4557
weighted avg	0.99	0.99	0.99	4557

Confusion Matrix:

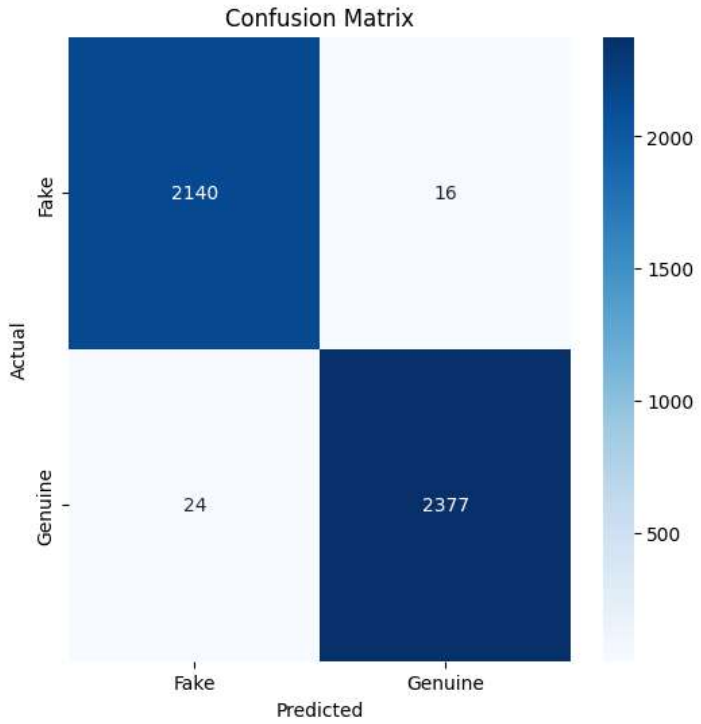

Figure 4.3.1: Confusion Matrix RoBERTa Model.

ROC Curve and Precision-Recall Curve:

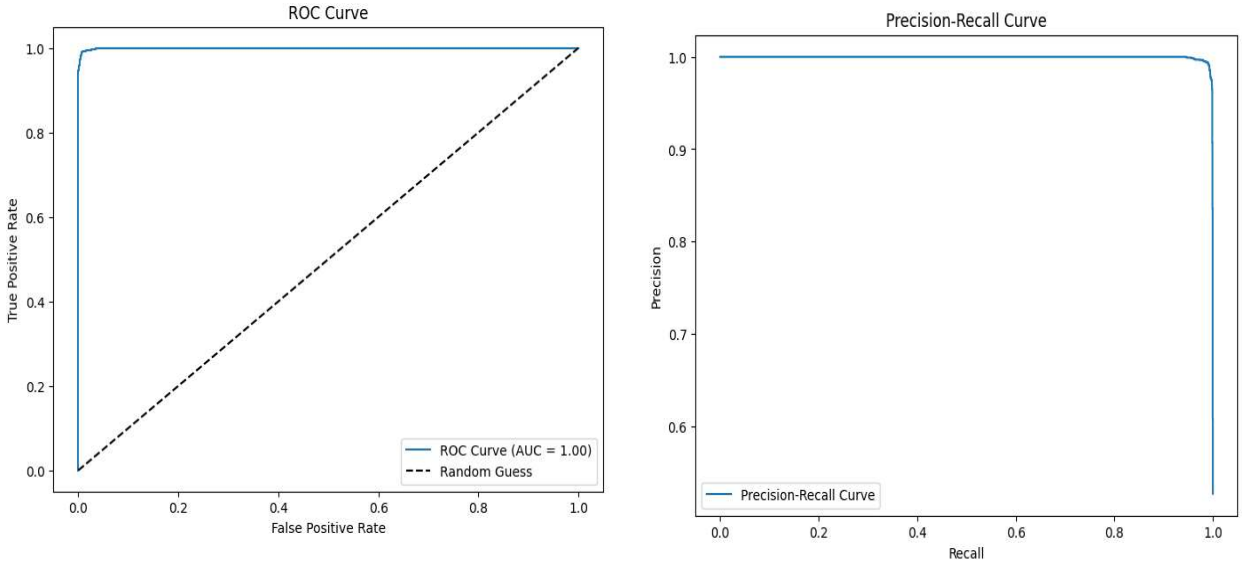

Figure 4.3.2: ROC Curve and Precision-Recall Curve For RoBERTa Model.

DistilBERT:

Post-Training Model Assessment:

Table 4.3.3: Post-Training Model Assessment For DistilBERT Model.

Matrix	Value
Loss	0.0920
Accuracy	0.9842

Classification Report

Table 4.3.4: Classification Report For DistilBERT Model.

	precision	recall	f1-score	support
Fake	0.99	0.97	0.98	2156
Genuine	0.98	0.99	0.99	2401
accuracy			0.98	4557
macro avg	0.98	0.98	0.98	4557
weighted avg	0.98	0.98	0.98	4557

Confusion Matrix:

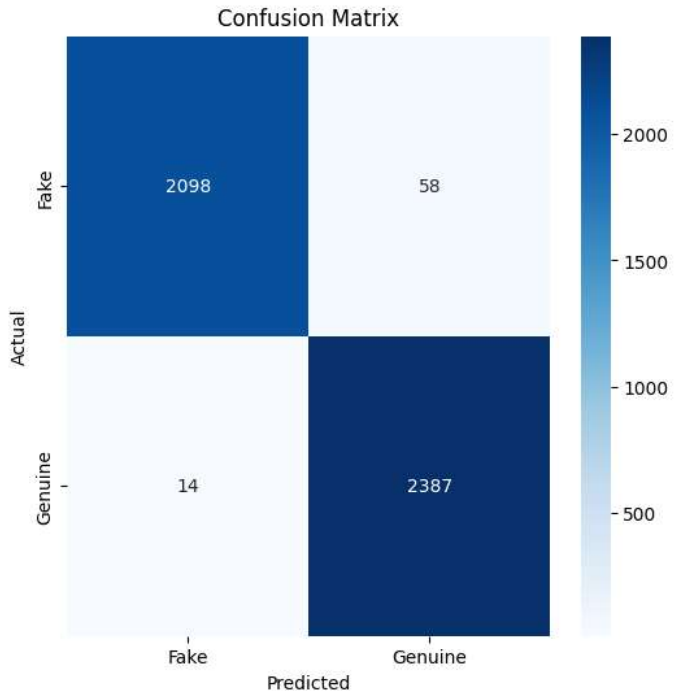

Figure 4.3.3: Confusion Matrix DistilBERT Model.

ROC Curve and Precision-Recall Curve:

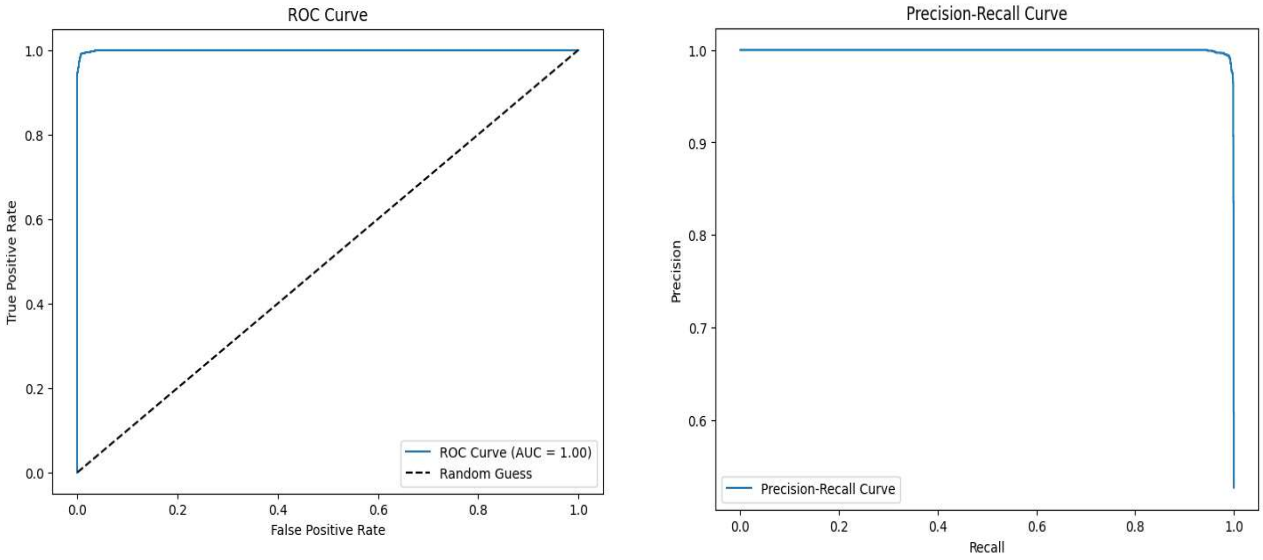

Figure 4.3.4: ROC Curve and Precision-Recall Curve For DistilBERT Model.

BART-large:

Post-Training Model Assessment:

Table 4.3.5: Post-Training Model Assessment For BART-large Model.

Matrix	Value
Loss	0.0610
Accuracy	0.9796

Classification Report:

Table 4.3.6: Classification Report For BART-large Model.

	precision	recall	f1-score	support
Fake	0.95	1.00	0.97	1554
Genuine	1.00	0.97	0.98	2411
accuracy			0.98	3965
macro avg	0.98	0.98	0.98	3965
weighted avg	0.98	0.98	0.98	3965

Confusion Matrix:

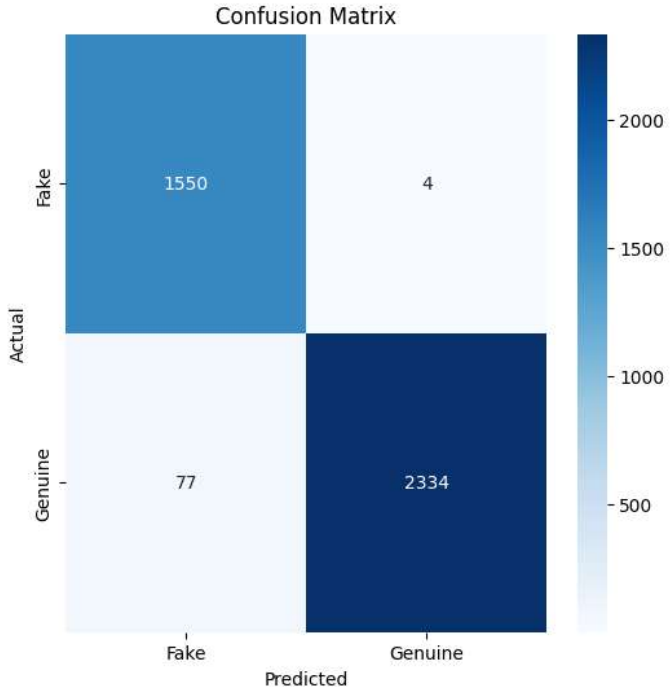

Figure 4.3.5: Confusion Matrix BART-large Model.

ROC Curve and Precision-Recall Curve:

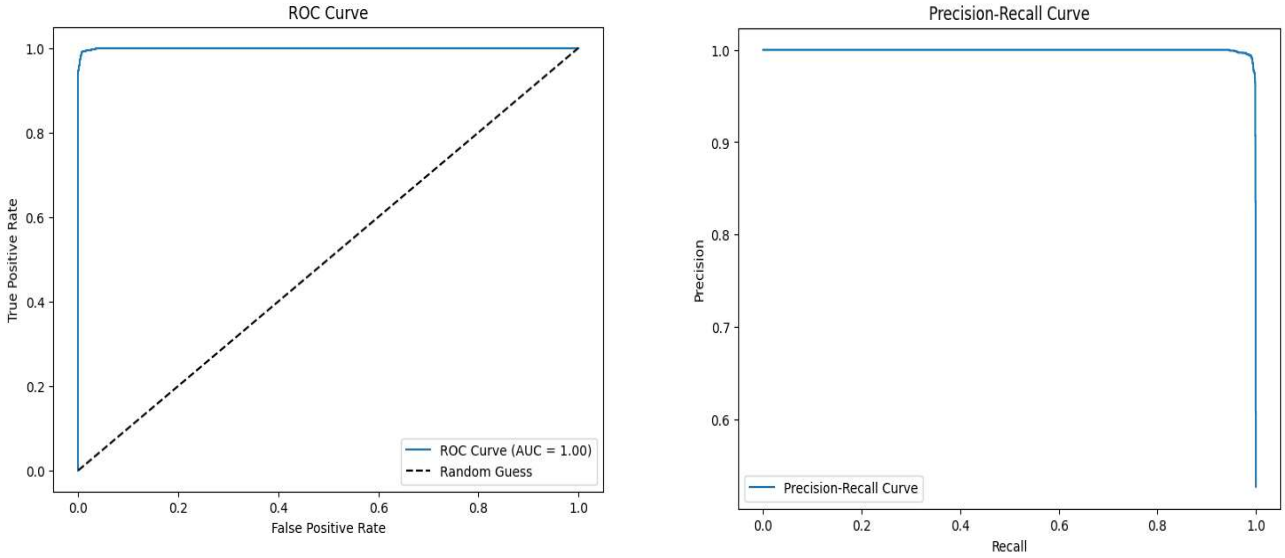

Figure 4.3.6: ROC Curve and Precision-Recall Curve For BART-large Model.

4.4 Summary

The “Fake App Detection using User Review Analysis with Transformer Based Models” is aimed at detecting fake applications in the Google Play Store based on user generated review data. The process starts with data gathering.ungi: Python libraries (google_play_scraper) are used to scrape reviews. This raw data is heavily preprocessed through text cleaning, normalization and tokenization prior to be used for sentiment analysis and classification. We then train and evaluate several transformer-based deep learning models (RoBERTa, DistilBERT and BART-Large) using this processed dataset.

We evaluate the performance of each of our models with different evaluation metrics-accuracy, test loss, confusion matrix, classification report,ROC score, precision-recall curves. RoBERTa obtained the best classification accuracy performance, and was hence selected for the task. The model was released using the Hugging Face Spaces which makes it possible to enter a Play Store link and receive a classification whether an app is likely fake or not. Our work is advancing the creation of smart, user-focused tools that improve online safety and trust via the recognition of potentially malicious apps based on automated review analysis.

Chapter 5

Engineering Standards and Design Challenges

5.1 Compliance with the Standards

In building the fake app detection system, it was essential to comply with relevant software standards to ensure model performance, data security, and system scalability. Since this project is entirely software-based, the focus is on software standards. However, basic hardware guidelines were also considered for model training and deployment.

5.1.1 Software Standards

The software standards for the Fake App Detection project leverage widely adopted platforms and tools to ensure consistency, reliability, and collaboration throughout the development process. Google Scholar and ResearchGate are utilized for sourcing relevant scientific literature to support the methodology and model development. The primary development environment is Google Colab, which is used for implementing and training deep learning models in Python, leveraging libraries such as Hugging Face Transformers, PyTorch, and TensorFlow. These frameworks provide the necessary support for training advanced NLP models like RoBERTa, DistilBERT, and DeBERTa. A personal laptop with GPU support is employed for testing and fine-tuning models locally. Documentation and reporting adhere to academic standards, using MS Word to ensure clarity, precision, and professionalism in the presentation of research findings.

5.1.2 Hardware Standards

Operating System: Windows 7 / 8 / 8.1 / Ubuntu 18.04+

Processor: Intel Core i3 (Latest Generation)

Memory (RAM): 4 GB

Graphics: 2 GB Dedicated GPU (e.g., NVIDIA GeForce MX Series)

Storage: 3 GB available space

Internet: Required for downloading models and scraping data

Recommended Requirements:

Operating System: Windows 10 / Ubuntu 20.04+

Processor: Intel Core i5 or higher

Memory (RAM): 8 GB or higher

Graphics: 4 GB or higher Dedicated GPU (e.g., NVIDIA GTX 1650 or better)

Storage: 5 GB available space

Internet: High-speed internet connection recommended for faster scraping and model downloads

5.1.3 Communication Standards

We conduct regular team meetings via Google Meet to present progress on the project, to discuss problems we are facing and to assign tasks, so that we stay in the know and work in harmony. The team uses google drive for document sharing and version tracking, leaving project wide docs accessible and making sure everyone has the most recent version. For rapid, daily communication, the team uses Telegram for expedited updates and notifications. All formal reports and documentation are template based in MS Word. Feedback loops are preserved enabling transparency, quick revisions and joint decision-making resulting in seamless and effective running of the project.

5.2 Impact on Society, Environment and Sustainability

5.2.1 Impact on Life

The fake app perception system has direct implications on human lives and society where it tackles a crucial problem of cyber world - fake applications. In a world where mobile apps are interwoven with everyday activities, users typically turn to app stores to explore new products and services. But there are serious risks behind the ease: fake apps that pretend to be the real thing, yet serve up malware and delve into your personal information. This system will guarantee users safety and privacy as it uses an advanced, precise AI to spot these fake apps. It is especially beneficial to keep sensitive personal information out of the hands of bad actors who prey on innocent users.

Also this mechanism prevent usersd from wasting their precious time, and from money of being spent on unreliable apps. In the end, people have fewer hassles, and the risk of running into apps that are resource hogs or security leaks is minimized. It breeds more

trust into digital platforms, and also confidence on the end-users in the tools they use daily.

5.2.2 Impact on Society & Environment

On a societal scale, the counterfeit app discovery algorithm is ensuring that millions of digital interactions are occurring in a safer manner across the globe. With an increasing number of people using smartphones and tablets for their everyday activities, secure app ecosystems are becoming ever more important. Many of these fraudulent apps can have a detrimental effect on real-life, such as by stealing sensitive data, spreading malware, or by running scams, which can create a financial loss and undermine people's trust in digital technology. Because we can detect these harmful apps, the system is a crucial part in mitigating such exposures and in promoting a safer digital space.

On a more societal level, there's the common good to think of, in terms of saving the world from the next big cybersecurity problem. This is a good project as it pushes developers to consider the security implications when developing apps, in an effort to help protect users from malicious rogue apps. It also means that it encourages a more solid public trust in digital marketplaces, which can then filter out to the wider tech industry and encourage platforms to put into place serious systems for ensuring that the apps they're making available are the real thing.

On an environmental level, the project will also decrease the burden that fake apps create for devices and networks. Harmful apps can consume battery resource, lead to excessive data usage, or tarnishing the performance of a device by which more energy may draw. By disallowing those apps from being installed, the system actively contributes to a more effective and long-lasting use of digital resources. In addition, by encouraging the identification and removal of counterfeit apps, it helps to protect the integrity of digital ecosystems and to render them safer and more reliable for the next generations of users.

5.2.3 Ethical Aspects

The ethical components of the fake app recognition system are also important since they are related to the user privacy, data protection, and the interpretability of the technology. One of the fundamental ethical issues that must be addressed is that the announced system only crawls publicly available data but respects the users ' privacy. From my scraping of reviews to detect fake apps, one of the most important rules is not to collect, store or misuse any personal information (PII). The system is compliant with data privacy laws such as GDPR and makes sure that the information about the users is anonymized and data protection is guaranteed, thus considering the ethical obligation of respecting user privacy.

And the project also must be clear, transparent about the data and methods it relies on. It's something where users and devs should know and these data could be inspected for fraud detection, they should have the possibility to opt out if they don't want to be analyzed. That transparency is so important in building trust between the technology and its users – something that is crucial for the ethical use of AI technologies.

In addition, there's a moral obligation to have nonjudgmental, evenhanded treatment. Both the models should not raise too many false alarms on genuine apps, because such a behavior could degrade a developer's reputation and the ecosystem as well due to false positives. The system needs regular updates and tests to work properly and not unfairly discriminate against type of app or type of developer.

Finally, the technology needs to advantageous to society. The aim is to make the digital world more secure by distinguishing between fraudulent apps that harm users from technologies that bad actors could leverage for surveillance or malicious purposes. This is in addition to ethical concerns relating to giving the same groups of users access to the system to wider group of users, including those in underprivileged areas, in order to uphold fairness in digital security.

5.2.4 Sustainability Plan

The design of this project is being done with a focus on sustainability. One of the main reasons that this system is sustainable is because it is flexible. The model used are trained with the possibility to retrain and fine tune the models with new data to guarantee the effectiveness of the system on new categories of fraudulent application. This enables the project to continually adapt to new trends in app development, and attackers' tactics – ensuring a long-term solution to the challenge of increasingly prevalent fake apps.

The project is additionally designed for scalability. It is capable of scaling up to analyse large volumes of reviews and analyzing thousands of app reviews in a fast and efficient manner. This scalability means that the solution is able to expand as the number of apps and users on digital channels grow. And with the solution widening to other app stores or review sites, the global influence might be more, eventually helping to make a more secure, more transparent digital environment.

Furthermore, cloud technologies, utilized by the system and generally designed for energy efficiency and cost effectiveness, will also help to assure long term sustainability. With companies like Google, Microsoft, and Apple leading the charge, cloud infrastructure providers are gradually shifting toward renewable energy sources to power their data centers, which in turn slashes the carbon footprint of digital services. The use of cloud-based services ensures the project has been maintained in keeping with worldwide initiatives to minimise the environmental impact of IT systems.

The open-source model of the project also supports sustainability. Since the code and implementation are open, the entire system can be iteratively developed and customized by the world-wide developer community. It is also a way of keeping the solution up to date for the long term while fostering collaboration and innovation in the app security space.

5.3 Project Management and Financial Analysis

Project Management:

Planning & Data Collection: Scoping sources for app review data, designing project objectives, and creating goals for data scraping, pre-processing, and model building.

Preprocessing & Model Development: Cleaning and utilizing user reviews from Google Play Store using google_play_scraper library and creating a DL model for fake app detection using sentiment analysis (RoBERTa-based model).

Training, Testing & Validation: Train the model with the pre-processed data and perform hyperparameter tuning to maximize the model and evaluate the model performance which is based on accuracy, precision, recall and F1-score.

Deployment & Evaluation: Deploying the model developed as an online system in which users can enter app names or links to it and check it's authenticity. Measuring the effectiveness of the system in the wild and guaranteeing its scalability and stability.

Documentation & Reporting: Create extensive documentation, performance report of the model, system architecture, and how it can be improved in the future. Report results and make suggestions about possible modifications or enhancements.

Financial Analysis:

Estimated Budget:

Table 5.3.1: Estimated Budget.

Components	Estimated Cost (BDT)
Laptop	70000
Wi-Fi	5000
Google Colab Pro	3000
Domain & Hosting	5000

Alternative Budget (Budget Optimization):

Table 5.3.2: Alternative Budget (Budget Optimization).

Components	Estimated Cost (BDT)
Laptop	70000
Wi-Fi	3000
Google Colab Pro	0 [Use free version]
Domain & Hosting	0 [Use free version]]

5.4 Complex Engineering Problem

5.4.1 Complex Problem Solving

In this section, provide a mapping with problem solving categories. For each mapping add subsections to put rationale (Use Table 5.3.3). For P1, you need to put another mapping with Knowledge profile and rational thereof.

Table 5.3.3: Mapping with complex problem solving.

EP1 Dept of Knowledge	EP2 Range Of Conflicting Requirements	EP3 Depth of Analysis	EP4 Familiarity of Issues	EP5 Extent of Applicable Codes	EP6 Extent Of Stake-holder Involvement	EP7 Interdependence
✓	✓	✓				✓

EP1 – Dept of Knowledge:

The project requires a deep understanding of Natural Language Processing (NLP), sentiment analysis techniques, and transformer-based models like RoBERTa. Additionally, it involves working knowledge of data scraping, deployment strategies, and integration of deep learning models into real-time web applications. Mastery in both theoretical ML concepts and practical engineering implementation is essential.

EP2 – Range of Conflicting Requirements:

A key challenge lies in balancing the accuracy of fake app prediction models with the performance needed for real-time deployment. While complex models like DistilBERT/RoBERTa provide high accuracy, they may be computationally expensive, slowing down inference time. Optimizing for speed while maintaining accuracy introduces conflicting design decisions in model selection, preprocessing, and system architecture.

EP3 – Depth of Analysis:

There is no obvious or straightforward formulation of the problem as a traditional machine learning task due to the diverse nature and variability of social media and user review data. A deep level of analytical thinking was necessary to explore and evaluate multiple algorithmic approaches—such as rule-based filtering, traditional ML classifiers, and advanced transformer-based models—and ultimately select the most appropriate model (RoBERTa) based on empirical performance, adaptability, and interpretability. This depth of analysis ensured the solution effectively captured deceptive behavior in fake app reviews.

EP7 – Interdependence:

The project is a multi-stage pipeline involving interdependent modules: Data scraping from Google Play Store, NLP-based preprocessing, transformer-based sentiment classification and labeling, review database creation, and model deployment via huggingface to serve real-time predictions. A failure in any of these components impacts the system's overall functionality, showcasing strong interconnectivity and interdependence.

Mapping with Knowledge Profile for EP1

This table 5.2) is designed to map the EP1 to the Knowledge Profile.

Table 5.3.4: Mapping with knowledge Profile.

K3 Engineering Fundamentals	K4 Specialist Knowledge	K5 Engineering Design	K6 Engineering Practice	K8 Research Literature
✓	✓	✓	✓	✓

K3 – The Engineering Fundamentals:

It is relying on several tools and methodologies native to engineering and computer science such as scraping, preprocessing, text analysis, model evaluation and performance tuning. It is systematic and uses specific methods to create a strong system for review classification of apps.

K4 – Specialist Knowledge:

Expertise in transformer models such as RoBERTa is necessary for applying pre-trained language models and performing fine-tuning for domain-specific tasks. The sentiment analysis component and review labeling strategy demand specialized understanding of contextual embeddings and classification pipelines.

K5 – Engineering Design:

Our project goes through a systematic design pipeline, consisting of data collection using web scraping, preprocessing, model selection, to gradually iterative training. It strikes a balance between model accuracy and interpretability, all while maintaining ethical considerations, such as privacy, in mind. The design details guarantee that the system remains scalable and could be applied in practice.

K6 – Engineering Practice:

Industry-relevant practices such as version control using Git, modular coding, Docker-based containerization, and CI/CD deployment workflows have been adopted. These practices ensure the maintainability, scalability, and quality of the engineering solution.

K8 – Research Literature:

The solution is informed by current state-of-the-art research in fake app detection, NLP sentiment classification, and the application of transformer models in cybersecurity. Review of peer-reviewed papers, benchmark datasets, and model comparison studies shaped the solution's design and direction.

5.4.2 Engineering Activities

In this section, provide a mapping with engineering activities. For each mapping add subsections to put rationale (Use Table 5.3.5).

Table 5.3.5: Mapping with complex engineering activities.

EA1 Range of re-sources	EA2 Level of Interaction	EA3 Innovation	EA4 Consequences for society and environment	EA5 Familiarity
✓	✓	✓	✓	✓

EA1 – Range of Resources:

The project utilizes diverse and modern resources, including Google Play store data via scraping python librarie, GPU-enabled cloud platforms (for RoBERTa inference and fine-tuning), pre-trained transformer models. It also uses libraries like transformers, TensorFlow, and scikit-learn for performing sentiment analysis and text processing.

EA2 – Level of Interaction:

The system was developed through team collaboration where roles were divided across data preprocessing, model training, and Web based application development. Additionally, users interacted with the app to provide feedback on the interface and predictions, and regular consultations were held with supervisors to refine the methodology.

EA3 – Innovation:

The project brings innovation by combining NLP-based sentiment analysis with user review behavior to detect fake apps. Creating a real-time web-based system that helps users determine app authenticity is an impactful approach not commonly deployed in commercial environments.

EA4 – Societal and Environmental Impacts:

The project addresses a critical issue in today's digital ecosystem—protecting users from malicious or deceptive applications. It enhances digital trust, contributes to safer app usage, and prevents potential data misuse, thus having a meaningful societal benefit with negligible environmental impact.

EA5 – Familiarity:

While the base technologies (Python, NLP, Deep learning) are familiar, integrating transformer models, building a real-time prediction pipeline, and designing a system for fake review detection introduced novel problem-solving scenarios that required adaptation and continuous learning.

5.5 Summary

The "Fake App Detection through User Review Analysis" is a highly difficult engineering challenge requiring in-depth expertise in NLP, machine learning, and sentiment analysis. It is also an illustration of a methodology used for problem solving in case of conflicting needs and demands and of the necessity to produce a superior high quality design. It draws on general engineering concepts in addition to specialized AI, ML, and NLP knowledge. Engineering practices, design principles, and the literature are effectively blended in and the result is a sound and ethically sound robot.

The project makes extensive use of technological tools and encourages inter-disciplinary work. It also serves a societal good by looking into the vulnerabilities of mobile app security and privacy. By taking user reviews, it blends well-known methods in an novel way in these present days and ages, doing this the technical and ethical right way

Chapter 6

Conclusion

6.1 Summary

This research project aimed to detect fake applications on the Google Play Store using user review analysis and transformer-based deep learning models. The process began with collecting over 30,000 reviews across various app categories using the google_play_scraper library. The dataset was preprocessed using natural language processing techniques such as text cleaning, tokenization, and sentiment scoring. Sentiment analysis was performed using the cardiffnlp/twitter-roberta-base-sentiment-latest model, and the results were used along with user ratings to classify reviews as fake or genuine.

Three transformer models—RoBERTa, DistilBERT, and BART-large—were trained and evaluated. Among them, RoBERTa achieved the highest accuracy of 99.12%, demonstrating its effectiveness in capturing the underlying patterns of fake reviews. A web-based system was developed and deployed using Hugging Face Spaces, allowing users to input app links or names and receive real-time predictions. The system provides an accessible and effective way to identify potentially harmful applications based on user feedback.

6.2 Limitation

Despite its high performance, the system has some limitations. Firstly, the classification depends heavily on user-generated content, which may be sparse or manipulated for less popular or newly launched apps. Secondly, the models do not consider app metadata (e.g., developer information, install count, or permissions), which could improve prediction accuracy. Additionally, while sentiment and rating thresholds work well in many cases, borderline reviews or sarcastic comments may be misclassified due to the limitations of sentiment models.

6.3 Future Work

Future work will focus on incorporating multimodal data, including app metadata, screenshots, and permission requests, for improved accuracy. Expanding the dataset to include more diverse apps and languages will also enhance model generalizability. Integrating explainable AI (XAI) techniques can help users understand why a particular app is classified as fake or genuine. Finally, deploying the system as a browser extension or integrating it with app store platforms could provide users real-time safety indicators while browsing or installing apps.

References

[1] Tran, H., Shcherbakov, M., & Maxim, S. (2016). Detection and Prediction of Users' Attitude Based on Real-Time and Batch Sentiment Analysis of Facebook Comments. International Conference on Intelligent Computing and Applications.

[2] Ahmad, I., Yousaf, M., Yousaf, S., & Ahmad, M. O. (2020). Fake News Detection using Machine Learning Ensemble Methods. International Journal of Computer Applications, 174(1).

[3] Chang, S., Zhenzhong, X., Xuan, G. (2020). Fake Comment Detection Based on Sentiment Analysis. Journal of Applied Computing and Informatics, 2020.

[4] Sagar, A. A., & Kiran, J. S. (n.d.). Toxic Comment Classification using Natural Language Processing. [Research Article].

[5] Gaurav, A., Gupta, B. B., Chui, K. T., Peraković, D., Chaurasia, P., & Hsu, C.-H. (2021). A novel approach for fake comments and reviews detection on online social networks. Journal of Artificial Intelligence and Soft Computing Research, 11(3), 2021.

[6] Panthaplackel, S., Gligoric, M., Mooney, R. J., & Li, J. J. (2020). Associating Natural Language Comment and Source Code Entities. Proceedings of the ACM International Conference on Software Engineering (ICSE), 79-89.

[7] Gaur, N., & Sharma, N. (n.d.). Sentiment Analysis in Natural Language Processing. Ganga Institute of Technology and Management, Jhajjar.

[8] Kavitha, K. M., Shetty, A., Abreo, B., D'Souza, A., & Kondana, A. (2021). Analysis and classification of user comments on YouTube videos. St Joseph Engineering College, Mangaluru, India.

[9] Salminen, J., Kandpal, C., Kamel, A. M., Jung, S.-G., & Jansen, B. J. (2020). Creating and detecting fake reviews of online products. ACM Transactions on the Web, 14(4), 1-24.

[10] Ahmad, Iftikhar, et al. "Fake News Detection Using Machine Learning Ensemble Methods." Complexity, vol. 2020, 17 Oct. 2020, pp. 1–11, https://doi.org/10.1155/2020/8885861. Accessed 3 May 2025.

[11] Gupta, A., & Kamthania, D. (2017). Study of sentiment on Google Play Store Applications. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2(2), ISSN: 2456-3307.

[12] Panigrahi, S., Momin, S., Patil, P., & Kshirsagar, P. (n.d.). Sentiment Analysis of Application Reviews on Google Play Store. Department of Information Technology, Mumbai University.

[13]Dang, N. C., Moreno-García, M. N., & De la Prieta, F. (2020). Sentiment Analysis Based on Deep Learning: A Comparative Study. Journal of Computational and Applied Mathematics, 389, 113295.

[14] Panthaplackel, S., Nie, P., Gligoric, M., Li, J. J., & Mooney, R. J. (2020). Learning to Update Natural Language Comments Based on Code Changes. Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), 1-9.

[15]Aldabbas, H., Bajahzar, A., Alruily, M., Qureshi, A. A., Amir, R. M., & Latif, M. F. (2019). Google Play content scraping and knowledge engineering using Natural Language Processing techniques with the analysis of user reviews. Journal of Intelligent Systems, 28(2), DOI: https://doi.org/10.1515/jisys-2019-0197

[16]Dudhankar, V., Sen, N., Langde, A., & Kupade, V. (2020). Google Play Sentiment Analysis. Bramhadevdada Mane Institute of Technology, Belati, Solapur, India.

[17] Panigrahi, Sangeeta, et al. "Sentiment Analysis of Application Reviews on Google Playstore." International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2017 IJSRCSEIT, vol. 2, no. 3, 2017, pp. 2456–3307. Accessed 3 May 2025.

[18] Gupta, Aayush, and Deepali Kamthania. "Study of Sentiment on Google Play Store Applications." SSRN Electronic Journal, 2021, https://doi.org/10.2139/ssrn.3833926. Accessed 3 May 2025.

[19]Martens, D., & Maalej, W. (2017). Towards understanding and detecting fake reviews in app stores. https://link.springer.com/article/10.1007/s10664-019-09706-9

[20]Sadiq, Saima, et al. "Discrepancy Detection between Actual User Reviews and Numeric Ratings of Google App Store Using Deep Learning." Expert Systems with Applications, vol. 181, Nov. 2021, p. 115111, https://doi.org/10.1016/j.eswa.2021.115111.

[21]Karunanayake, Naveen, et al. "A Multi-Modal Neural Embeddings Approach for Detecting Mobile Counterfeit Apps: A Case Study on Google Play Store." IEEE Transactions on Mobile Computing, 2020, pp. 1–1, https://doi.org/10.1109/tmc.2020.3007260. Accessed 18 Aug. 2020.

[22]Rahman, Mahmudur, et al. "FairPlay: Fraud and Malware Detection in Google Play." ArXiv (Cornell University), 30 June 2016, pp. 99–107, epubs.siam.org/doi/abs/10.1137/1.9781611974348.12, https://doi.org/10.1137/1.9781611974348.12. Accessed 4 May 2025.

[23]Syed, Mr, et al. "Decision Tree Analysis for Identifying False Apps in the Google Play Store." DogoRangsang Research Journal UGC Care Group I Journal, vol. 13, 2023, pp. 2347–7180, dogorangsangresearchjournal.com/admin/uploads/ijf%20dogo.org%20(current%20issue).pdf. Accessed 4 May 2025.

212-15-4129

ORIGINALITY REPORT

22%	16%	10%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	6%
2	Submitted to George Bush High School Student Paper	2%
3	Submitted to United International University Student Paper	2%
4	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
5	www.mdpi.com Internet Source	1%
6	Poonam Nandal, Mamta Dahiya, Meeta Singh, Arvind Dagur, Brijesh Kumar. "Progressive Computational Intelligence, Information Technology and Networking", CRC Press, 2025 Publication	<1%
7	Submitted to Liverpool John Moores University Student Paper	<1%
8	www.ncbi.nlm.nih.gov Internet Source	<1%
9	Submitted to Tilburg University Student Paper	<1%
10	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	<1%
	Submitted to The Robert Gordon University	