UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BSD2513 ARTIFICIAL INTELLIGENCE

GROUP PROJECT

GROUP B

TITLE:

SCAMMER DETECTOR

LECTURER:

DR. KU MUHAMMAD NA'IM KU KHALIF

PREPARED BY:

MATRIC ID	NAME	SECTION
SD22012	PUTRI BALQIS BATRISYIA BINTI MOHD RIZAL	02G
SD22055	AMIRAH AISYAH BINTI ABDULLAH	01G
SD22009	HASNIZA BINTI MOHD SOMAN	02G
SD22045	NURUL ALIS BINTI YUSRI	02G
SD22050	MUHAMMAD BIN ABDUL HADI	01G

TABLE OF CONTENTS

1.0 EXECUTIVE SUMMARY

1.1 Description of the selected project

In an era dominated by digital connectivity, online scams have become an omnipresent threat, causing substantial financial losses and emotional distress to victims worldwide. It's a scenario we've all encountered: receiving an enticing message or email promising extravagant prizes, exclusive deals, or financial windfalls. These messages often come from unfamiliar numbers or dubious sources, enticing users with the allure of easy gains. Yet, behind these seemingly lucrative opportunities lurk elaborate schemes designed to exploit our trust and compromise our personal information.

The ubiquity of online scams poses a significant challenge, especially for those less versed in the intricacies of digital communication. While younger generations may navigate online platforms with relative ease, older individuals and less tech-savvy users may find themselves unwittingly falling prey to fraudulent tactics. The lack of awareness about common scam techniques, coupled with the innate desire for quick rewards, creates fertile ground for scammers to operate with impunity.

Moreover, the advent of social media has further complicated the landscape of online scams. Fake accounts proliferate across platforms like Instagram and Shopee, peddling counterfeit goods or misleading users with false promises. This proliferation not only exacerbates the prevalence of scams but also undermines the trust and reliability of online marketplaces.

In response to these challenges, the "Scammer Detector" project emerges as a beacon of digital vigilance and security. This multifaceted application stands as a bulwark against the tide of online scams, leveraging advanced technologies and innovative methodologies to identify and thwart fraudulent activities across various digital channels.

At its core, the Scammer Detector integrates a diverse array of functionalities, including real-time detection of scam URLs, messages, phone numbers, and QR codes. Harnessing the power of machine learning algorithms and natural language processing techniques, the application analyzes textual content to discern between genuine communications and fraudulent solicitations. Additionally, the project maintains a

repository of known scam phone numbers, enabling users to identify and block potential threats with confidence.

Through its user-friendly interface and intuitive design, the Scammer Detector empowers users of all ages and technological backgrounds to safeguard themselves against online scams. By fostering awareness, promoting digital literacy, and providing robust security measures, the project strives to create a safer and more secure digital ecosystem for all. As we navigate the complexities of the digital age, the Scammer Detector stands as a steadfast guardian, protecting users from the perils of online deception and ensuring their peace of mind in an increasingly interconnected world.

1.2 Problem to be solved

Online scams erode trust and confidence in digital transactions and communication channels. When users fall victim to scams, they may lose faith in the integrity and security of online platforms, hindering the growth of e-commerce and digital innovation. The prevalence of scams creates a pervasive atmosphere of suspicion and uncertainty, making users hesitant to engage in online interactions or share personal information.

Online scams have a global reach and can target individuals and organizations across geographical boundaries. With the increasing interconnectedness of the digital world, scammers exploit vulnerabilities in communication channels to reach a wide audience and perpetrate fraudulent activities on a massive scale. The borderless nature of online scams makes them a pervasive and persistent threat that requires international cooperation and collaborative efforts to combat effectively.

The dynamic nature of technology and communication channels presents an ever-evolving threat landscape, where scammers continually adapt their tactics to evade detection and exploit new vulnerabilities. As technology evolves, so do the methods and strategies employed by scammers, necessitating proactive measures and innovative solutions to stay ahead of emerging threats. The rapid proliferation of new scams and fraudulent schemes underscores the need for adaptive and resilient cybersecurity measures to protect users from evolving risks.

Beyond the financial losses incurred, falling victim to online scams can have profound social and psychological consequences for individuals. Victims may experience feelings of embarrassment, shame, and betrayal, leading to a loss of trust in online interactions and reluctance to seek help or report incidents. The psychological toll of scams can extend beyond the immediate financial repercussions, affecting victims' mental well-being and sense of security in the digital realm.

In light of these challenges, the Scammer Detector project aims to provide a comprehensive solution to detect and prevent various types of online scams. By integrating multiple detection methods into a single tool, users can benefit from enhanced security measures and proactive safeguards to mitigate the risks posed by online scams. Through education, awareness, and technological innovation, the project seeks to empower users to protect themselves from falling victim to fraudulent activities and safeguard their digital security and privacy.

1.3 Basic Description of the data selected

The Scammer Detector application is designed to help users identify and avoid scams by analyzing text messages, URLs, phone numbers, and QR codes. It aims to provide a reliable tool for detecting fraudulent activities and ensuring user safety. Here's a basic description of the selected data used in this project:

1. Phone Numbers:

This dataset includes several phone numbers that are known to be associated with scams. These numbers have been collected from various sources around the world.
Example phone numbers:
'9001234567'
'60169494016' (Malaysia)
'6285314112053' (Indonesia)

2. Scam Messages:

This dataset includes examples of typical scam messages that the application is trained to identify. These messages often contain keywords like ‘kaya’, ‘iphone’, ‘account’, ‘money’, and others.

Examples of scam messages:

"Click this link to win a free iPhone!"

"Invest in this amazing opportunity for guaranteed returns!"

"Get rich quick by working from home!"

"Your account has been compromised. Click here to reset your password."

"Join our network and earn money by recruiting others."

3. QR Codes:

The application scans QR codes from a camera feed to check if they link to legitimate URLs or potential scams.

The data includes URLs embedded in QR codes that are verified for legitimacy.

This data forms the backbone of the Scammer Detector application, enabling it to perform accurate and reliable scam detection across various types of inputs. By leveraging these datasets, the application can help users avoid falling victim to fraudulent activities.

4. URL link:

The applications will scan a legitime link if it starts with 🔒https://"): # Check for 'https://' or '🔒 https://'

2.0 SUMMARY OF THE PROJECT CONTEXT AND OBJECTIVES

2.1 Summary of the Project Context

In recent years, the number of scams targeting individuals especially students through various platforms such as websites, phone calls, QR codes, and messages has increased dramatically. These scams can lead to major financial losses, privacy violations, and emotional distress. As students, we are not immune to these threats. We frequently use many internet platforms for educational resources, communication, and transactions, making us perfect candidates for scammers. Therefore, developing an effective AI-powered scammer detector is essential to protecting students and others from these malicious activities.

Our project aims to create an AI-based system that can efficiently identify and expose potential scams across multiple mediums. By utilising advanced machine learning techniques, natural language processing (NLP), and data analysis, we intend to build an efficient tool that enhances digital security and promotes awareness among students. This project will not only contribute to our learning and application of AI technologies but also provide a valuable service to our peers and the wider community.

2.2 Objectives

- To develop AI system to identify scams on various platforms
- To enhance student awareness and safety
- To provide real-time alerts when potential scams are detected
- To create a user-friendly interface for easy interaction and accessibility for students

3.0 METHODOLOGY

3.1 Data Collection & Preparation

In order to create a reliable scam detection system, we gathered a wide range of datasets that included different types of potential scam content. Our team actively searched for real-life examples from text messages, websites, phone numbers, and QR codes to compile a comprehensive dataset.

The dataset consists of various types of data such as text, URLs, phone numbers, and QR codes. This allows the system to handle different kinds of information and provide strong analytical capabilities. The textual data we collected included different types of messages, some of which were identified as scams and others as legitimate. This helped us cover a wide range of scenarios in scam detection. For example, a message offering a free prize was labelled as a scam, while a genuine birthday greeting was labelled as legitimate. By using these diverse texts, we trained the system to improve its ability to identify scams in written content.

In addition to textual data, we also collected URLs to simulate real-life phishing situations. For instance, we included a URL that leads to a fake banking website and another URL that belongs to a legitimate e-commerce site. This allows the system to analyze scam attempts in dynamic environments and test its capability to detect fraudulent websites in real-time.

To make the system even more effective, we developed a real-time URL scanning feature. This feature enables the system to monitor and analyze URLs in real-time, allowing us to capture and analyze new scam URLs as they emerge. This real-time data stream provides valuable input for training and testing the system's ability to analyze URLs in real-time.

Furthermore, we collected phone numbers and categorized them into known scam numbers and legitimate numbers. For example, we included a phone number frequently used in robocalls as a known scam number, along with a legitimate customer service number. These phone numbers help the system learn to differentiate between scam calls and legitimate calls.

In addition, the dataset contains QR codes that imitate modern scam tactics using QR codes to lead people to fake websites or phishing pages. For example, one QR code directs users to a counterfeit banking website, while another leads to a legitimate restaurant menu. These QR codes assist the system in learning how to recognize and verify the authenticity of QR codes in different situations.

The information gathered, which includes text messages, URLs, phone numbers, and QR codes, was meticulously prepared and utilized to train algorithms for detecting scams. Annotation and labelling methods were employed to identify and mark instances of scams and legitimate data within the dataset. This annotation process enables the model to effectively learn and differentiate between scams and legitimate information.

By utilizing this carefully curated dataset, the project aims to train models that are strong, accurate, and capable of detecting scams in text messages, URLs, phone numbers, and QR codes. The dataset's diversity in terms of scam variations, contexts, and data formats contributes to the system's ability to handle different scenarios and generalize effectively to new data.

3.2 Coding without GUI

The project imports various libraries necessary for text processing, machine learning, and GUI creation.

OpenCV : OpenCV (Open Source Computer Vision Library) is a powerful library used for computer vision tasks.

Pyzbar : library used for decoding barcodes and QR codes.

NLTK : The Natural Language Toolkit (NLTK) is a suite of libraries and programs for natural language processing (NLP).

- word_tokenize: Splits text into words (tokens).
- stopwords: Provides common words to filter out from text.
- WordNetLemmatizer: Reduces words to their base form.

Regular Expressions: The re library is used for working with regular expressions, which are patterns for matching text.

Request: requests is a simple and elegant HTTP library for making requests to web servers.

BeautifulSoup : used for parsing HTML and XML documents.

Scikit-learn: machine learning library for Python.

Tkinter : The standard Python interface to the Tk GUI toolkit.

PIL (PIL): Python Imaging Library (PIL), now maintained under the name Pillow, is used for opening, manipulating, and saving many different image file formats.

Threading : Standard Python library for concurrent execution of code.

```
import cv2
from pyzbar.pyzbar import decode
import nltk
import re
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
import tkinter as tk
from tkinter import messagebox, Tk, Label, Toplevel, Message
from PIL import Image, ImageTk, ImageEnhance
import threading



# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

ScamDetector class contains the core functionality for detecting scams. It uses a TF-IDF vectorizer and a Naive Bayes classifier for text classification. The class includes methods for text preprocessing, training the model, and predicting scam messages.

```
class ScamDetector:
    def __init__(self):
        self.vectorizer = TfidfVectorizer(max_features=1000)
        self.classifier = MultinomialNB()
        self.stop_words = set(stopwords.words("english"))
        self.lemmatizer = WordNetLemmatizer()
        self.known_scam_numbers = [
            '1234567890',
            '5555555555',
            '8005551212',
            '9001234567',
            '8761234567',
            '8001234567',
            '8441234567',
            '8551234567',
            '8661234567',
            '8771234567',
            '8881234567',
            '8091234567',
            '9002345678',
            '8009876543',
            '8449876543',
            '8559876543',
            '8669876543',
            '8779876543',
            '8889876543',
            '8099876543',
```

```
            '60169494016',
            '60105687366',
            '60199848088',
            '601137788921',
            '60107150445',
            '6285314112053',
            '60164307673',
            '6282126866465',
            '60163489426',
            '60182252043',
        ]
```

TF-IDF Vectorizer: Transforms text data into numerical vectors based on word importance.

Naive Bayes Classifier: A probabilistic classifier suitable for text classification.

Stop Words: Common words that are filtered out to focus on significant words.

Lemmatizer: Reduces words to their root form for normalisation.

Known Scam Numbers: A predefined list of phone numbers identified as scams.

```
def preprocess_text(self, text):
    tokens = word_tokenize(text.lower())
    tokens = [self.lemmatizer.lemmatize(token) for token in tokens if token.isalnum()]
    tokens = [token for token in tokens if token not in self.stop_words]
    return " ".join(tokens)
```

Tokenization: Splits text into individual words.

Lemmatization: Normalizes words to their base form.

Stop Words Removal: Filters out common words to focus on meaningful content

```
def train(self, data):
    X = [self.preprocess_text(text) for text, label in data]
    y = [label for text, label in data]
    X_tfidf = self.vectorizer.fit_transform(X)
    self.classifier.fit(X_tfidf, y)
```

Training Data: Preprocesses text data and converts it into TF-IDF vectors.

Model Fitting: Trains the Naive Bayes classifier on the processed data

```
def extract_text_from_url(self, url):
    try:
        response = requests.get(url, timeout=5)
        soup = BeautifulSoup(response.content, 'html.parser')
        text = ' '.join([p.get_text() for p in soup.find_all('p')])
        return text
    except requests.exceptions.RequestException as e:
        return str(e)
```

Web Scraping: Fetches and extracts text content from a given URL using BeautifulSoup.

Error Handling: Manages request exceptions gracefully

```
def detect_scam(self, message):
    # Check for the specific word 'kaya'
    if 'kaya' in message.lower():
        return 'scam'

    processed_message = self.preprocess_text(message)
    vectorized_message = self.vectorizer.transform([processed_message])
    prediction = self.classifier.predict(vectorized_message)
    return prediction[0]
```

Specific Word Check: Directly classifies messages containing the word 'kaya' as scams.

Text Classification: Preprocesses and classifies the message using the trained model.

```
def detect_scam_from_website(self, url):
    if 'kaya' in url.lower():
        return 'scam'
    text = self.extract_text_from_url(url)
    if text:
        return self.detect_scam(text)
    return 'legitimate'
```

URL Content Analysis: Extracts and classifies text content from a given URL.

Specific Word Check: Directly classifies URLs containing the word 'kaya' as scams.

```
def detect_phone_number_scam(self, text):
    phone_numbers = re.findall(r'\b\d{10,12}\b', text)  # Simple regex for finding 10-12 digit phone numbers
    for number in phone_numbers:
        if number in self.known_scam_numbers:
            return 'scam'
```

```
return 'legitimate'
```

Regex for Phone Numbers: Finds phone numbers in the text using regex.

Known Scam Numbers: Checks if found numbers are in the list of known scam numbers

```
def is_legitimate_url(self, url):
    if url.startswith("https://") or url.startswith("🔒https://"):  # Check for 'https://' or '🔒https://'
        return True
    else:
        return False
```

HTTPS Check: Verifies if the URL uses HTTPS (indicating a more secure connection).

```
def decode_and_verify_qr_codes(self, frame):
    # Convert the frame to grayscale
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Decode QR codes in the frame
    decoded_objects = decode(gray_frame)

    # Iterate over all detected QR codes and verify against the trusted criteria
    for obj in decoded_objects:
        # Extract QR code data
        qr_data = obj.data.decode('utf-8')

        print("Detected QR code:", qr_data)  # Print the detected QR code for debugging

        # Check if the QR code content is a legitimate URL
        if self.is_legitimate_url(qr_data):
            result = "Legitimate"
            color = (0, 255, 0)  # Green for legitimate
        else:
            result = "Scammer"
```

```
            color = (0, 0, 255)  # Red for scammer

        # Draw a rectangle around the QR code
        (x, y, w, h) = obj.rect
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

        # Put the result on the frame
        cv2.putText(frame, result, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    return frame
```

QR Code Scanning: Uses OpenCV to decode QR codes from a camera frame.

URL Verification: Checks if the decoded QR code data is a legitimate URL.

Visual Feedback: Draws rectangles and labels around detected QR codes, indicating if they are legitimate or scams.

```
# Initialize the ScamDetector and train it with sample data
detector = ScamDetector()
scam_data = [
    ("Click this link to win a free iPhone!", "scam"),
    ("Invest in this amazing opportunity for guaranteed returns!", "scam"),
    ("Get rich quick by working from home!", "scam"),
    ("Your account has been compromised. Click here to reset your password.", "scam"),
    ("Join our network and earn money by recruiting others.", "scam"),
    ("Check out this cute cat video!", "legitimate"),
    ("Happy birthday to our dear friend!", "legitimate"),
    ("Congratulations on your new job!", "legitimate"),
    ("Here's a discount code for your next purchase.", "legitimate"),
    ("Reminder: Don't forget to backup your data regularly.", "legitimate")
]
detector.train(scam_data)
```

Initialization: An instance of the ScamDetector class is created.

Training Data: A list of sample messages labeled as 'scam' or 'legitimate' is defined.

Training: The train methodf is called to train the scam detector model on the provided data

```
def detect_link_scam():
    url = input_field.get()
    if url.startswith('http'):
        result = detector.detect_scam_from_website(url)
        if result == 'scam':
            messagebox.showinfo("Link Scam Detection", "This URL appears to be a scam.")
        else:
            messagebox.showinfo("Link Scam Detection", "This URL appears to be legitimate. Please be careful.")
    else:
        messagebox.showwarning("Invalid Input", "Please enter a valid URL.")
    input_field.delete(0, tk.END)  # Clear the input field
```

URL Input: Retrieves the URL from the input field.

Validation: Checks if the URL starts with 'http'.

Detection: Uses detect_scam_from_website method to check if the URL is a scam.

User Feedback: Displays the detection result in a message box.

Input Field: Clears the input field after processing.

```
def detect_message_scam():
    message = input_field.get()
    result = detector.detect_scam(message)
    if result == 'scam':
        messagebox.showinfo("Message Scam Detection", "This message appears to be a scam.")
    else:
        messagebox.showinfo("Message Scam Detection", "This message appears to be legitimate. Please be careful.")
    input_field.delete(0, tk.END)  # Clear the input field
```

Message Input: Retrieves the message from the input field.

Detection: Uses detect_scam method to check if the message is a scam.

User Feedback: Displays the detection result in a message box.

Input Field: Clears the input field after processing

```
def detect_phone_number():
    message = input_field.get()
    # Show a message about the phone number format
    messagebox.showinfo("Phone Number Format",
                        "For Malaysian numbers, please include the country code '60'. Example: 60123456789")

    # Find all numbers with at least 11 digits starting with '60'
    phone_numbers = re.findall(r'\b60\d{9,11}\b', message)
    print("Detected phone numbers:", phone_numbers)

    if not phone_numbers:
        messagebox.showwarning("Invalid Input", "No valid phone numbers detected. Ensure numbers start with '60' and are at least 11 digits long.")
        return

    # Check for invalid phone numbers
    for number in phone_numbers:
        if len(number) < 11:
            print("Invalid Malaysian Phone Number:", number)
            messagebox.showwarning("Invalid Malaysian Phone Number",
                                   "Invalid Malaysian phone number. It must be at least 11 digits long.")
            input_field.delete(0, tk.END)
            return

    result = detector.detect_phone_number_scam(message)
    if result == 'scam':
        messagebox.showinfo("Phone Number Scam Detection",
                            "This phone number appears to be associated with a scam. Please be careful.")
    else:
        messagebox.showinfo("Phone Number Scam Detection",
                            "This phone number appears to be legitimate.")
    input_field.delete(0, tk.END)  # Clear the input field
```

Message Input: Retrieves the message from the input field.

Phone Number Format: Displays information about the expected phone number format.

Phone Number Extraction: Uses regex to find phone numbers that start with '60' and are 11 to 12 digits long.

Validation: Checks if any phone numbers are found and ensures they meet the length criteria.

Detection: Uses detect_phone_number_scam method to check if the phone number is a scam.

User Feedback: Displays the detection result in a message box.

Input Field: Clears the input field after processing

```
def exit_application():
    # Function to exit the application
    root.destroy()
```

Exit: Terminates the application by closing the main window.

```
def run_qr_scanner():
    # Access the default camera (0)
    cap = cv2.VideoCapture(0)

    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()

        # Check if the frame was captured successfully
        if not ret:
            print("Error: Failed to capture frame")
            break

        # Decode and verify QR codes in the frame
        frame_with_verification_result = detector.decode_and_verify_qr_codes(frame)

        # Add instructions to exit the QR scanner
        cv2.putText(frame_with_verification_result, "Press 'q' to exit QR Scanner", (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                    0.7, (0, 255, 255), 2)

        # Show the frame with the exit instructions
        cv2.imshow('QR Code Scanner', frame_with_verification_result)

        # Check for key press, if 'q' is pressed, exit the loop
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

```
    # Release the camera and close OpenCV windows
    cap.release()
    cv2.destroyAllWindows()
```

Camera Access: Accesses the default camera.

Frame Capture: Captures frames in a loop.

QR Code Detection: Uses decode_and_verify_qr_codes to detect and verify QR codes in each frame.

User Instructions: Adds instructions to the frame on how to exit the QR scanner.

Display: Shows the processed frame in a window.

Exit: Allows the user to exit the QR scanner by pressing 'q'.

```
def start_qr_scanner():
    qr_thread = threading.Thread(target=run_qr_scanner)
    qr_thread.start()
```

Threading: Runs the QR scanner in a separate thread to keep the main application responsive.

```
def show_main_frame():
    welcome_frame.pack_forget()
    main_frame.pack(fill='both', expand=True)
```

Frame Management: Hides the welcome frame and displays the main frame

3.3 Coding with GUI

All the figures below show the coding with GUI in scanner scammer application.

```
# Create the main window
root = tk.Tk()
root.title("Scammer Detector")
root.geometry("800x600")

# Load and configure the background image
background_image_path = R"C:\Users\User\Downloads\phone-Photoroom (2).jpg"
background_image = Image.open(background_image_path)
background_image = background_image.resize( size: (root.winfo_screenwidth(), root.winfo_screenheight()), Image.LANCZOS)
background_photo = ImageTk.PhotoImage(background_image)

# Enhance the sharpness of the image
enhancer = ImageEnhance.Sharpness(background_image)
background_image = enhancer.enhance(5.0)  # Increase the factor for more sharpness, e.g., 2.0 or 3.0

# Resize the image to fit the screen
background_image = background_image.resize( size: (root.winfo_screenwidth(), root.winfo_screenheight()), Image.LANCZOS)

# Convert the image to a PhotoImage object for Tkinter
background_photo = ImageTk.PhotoImage(background_image)

# Create a label widget to display the background image
background_label = Label(root, image=background_photo)
background_label.place(x=0, y=0, relwidth=1, relheight=1)
```

The graphical user interface (GUI) for this scammer detector project is implemented as a Tkinter window, which provides a clear and user-friendly interface, allowing the users to easily interact with the application and access its features. This window is customised by setting the background image and specifying the title "Imposter Detector" as well as the image dimensions, which improves the overall user experience and visual appeal for the user.

We improved the sharpness of the background image and resized the image that we added to this window to make the interface more attractive and high-quality.

```
# Create the welcome frame
welcome_frame = tk.Frame(root)
welcome_frame.pack(fill='both', expand=True)

# Display the background image in the welcome frame
background_label = tk.Label(welcome_frame, image=background_photo)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

# Create the main application frame but do not pack it yet
main_frame = tk.Frame(root)

# Display the background image in the main frame
background_label_main = tk.Label(main_frame, image=background_photo)
background_label_main.place(x=0, y=0, relwidth=1, relheight=1)

# Load the icon image
icon_image = Image.open("C:\\Users\\User\\Desktop\\UMPSA 23_24\\artificial intelligence 23_24\\GROUP ASSIGNMENT AI\\SCAM ALERT.jpg")
icon_image = icon_image.resize( size: (400, 200), Image.LANCZOS)
icon_photo = ImageTk.PhotoImage(icon_image)

# Create a label for the icon
icon_label = tk.Label(welcome_frame, image=icon_photo, bg="#800000")  # Maroon background color
icon_label.pack(pady=150)

# Create a welcome message label with custom font
welcome_font = ("Times New Roman", 35, "bold")
welcome_label = tk.Label(welcome_frame, text="Welcome to Scammer Detector", font=welcome_font, bg="#800020",
                         fg="white")
welcome_label.pack(pady=10)
```

We added the icon image of a scam alert in the window to enhance the user awareness and recognition of potential scams within the application. The text "Welcome to

Scammer Detector" has also been included to welcome the user to use this application. Also, we customised each feature with the appropriate font style, text size, icon size, background colour, foreground colour, and spacing between the widget.

```
# Create a container frame for the continue button to provide better alignment
button_frame = tk.Frame(welcome_frame, bg="#800000")  # Maroon background color
button_frame.pack()

# Create a button to proceed to the main application
continue_button = tk.Button(welcome_frame, text="Continue", command=show_main_frame,
                            font=("Times New Roman", 17, "bold"), bg="maroon", fg="white", bd=5, relief=tk.RAISED,
                            height=1, width=20)
continue_button.pack(pady=10)

# Pack the button frame to the bottom of the welcome frame
button_frame.pack(side=tk.BOTTOM, pady=(0, 30))
```

Then, the frame for the continue button that is located at the bottom is created. This button enables the user to go to the next page of this application for detecting the scammer.

```
# Create a label for the title with custom font
title_font = ("Times New Roman", 40, "bold")
title_label = tk.Label(main_frame, text="Scammer Detector", font=title_font, bg="maroon", fg="white")
title_label.pack(pady=150)

# Create an entry field for user input
input_field = tk.Entry(main_frame, width=60, font=("Times New Roman", 14), bd=5, relief=tk.GROOVE)
input_field.pack(pady=5)
```

The main heading page shows the title of “Scammer Detector” with the specific type, size, and colour of font and also shows the label of title with the suitable size and colour.

```
# Create buttons for detecting link URL, message scammers, phone scammer, & QR scammer with custom font and size
button_font = ("Times New Roman", 16, "bold")
button_style = {"font": button_font, "bg": "#800020", "fg": "white", "bd": 5, "relief": tk.RAISED, "height": 1,
                "width": 20}

link_button = tk.Button(main_frame, text="Detect Link URL", command=detect_link_scam, **button_style)
link_button.pack(pady=10)

message_button = tk.Button(main_frame, text="Detect Message", command=detect_message_scam, **button_style)
message_button.pack(pady=10)

phone_button = tk.Button(main_frame, text="Detect Phone Number", command=detect_phone_number, **button_style)
phone_button.pack(pady=10)

qr_button = tk.Button(main_frame, text="QR Scanner", command=start_qr_scanner, **button_style)
qr_button.pack(pady=10)
```

Different frames of buttons are created to link with the different types of scammers. This enables users to choose the type of scammer they want to detect, like detecting link URLs, text messages, phone numbers and even QR codes. Each button has been customised with the same font style format, background colour, foreground colour, font colour and font size.

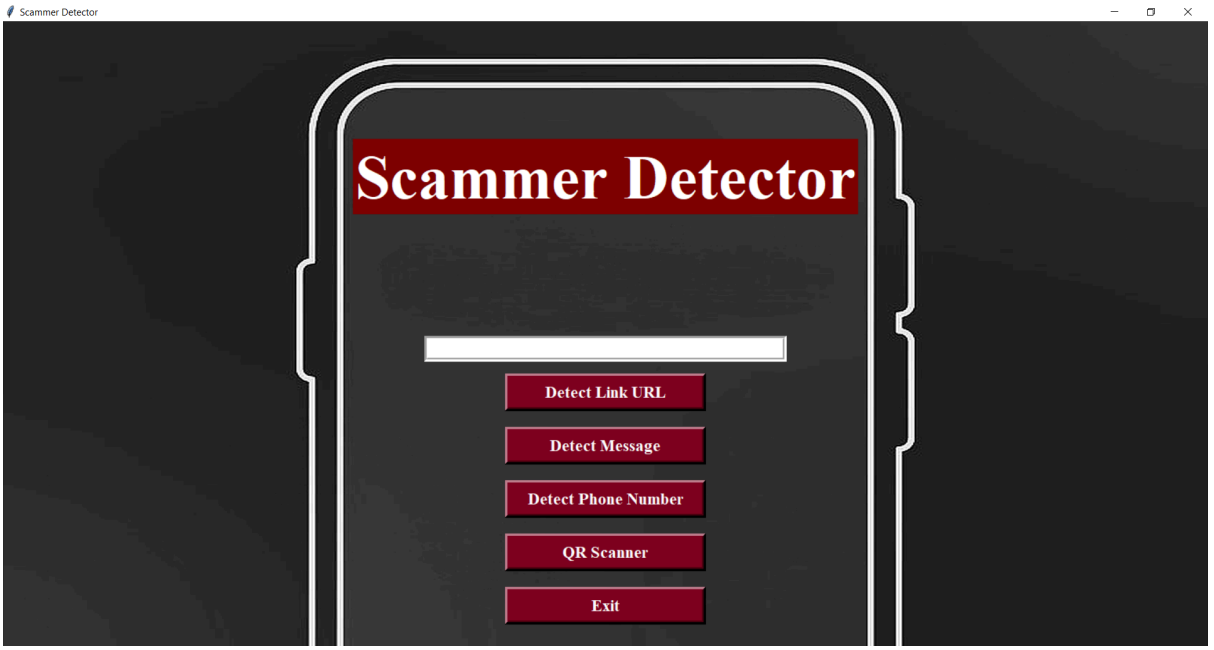
```
# Create an "Exit" button
exit_button = tk.Button(main_frame, text="Exit", command=exit_application, **button_style)
exit_button.pack(pady=10)

# Run the application
root.mainloop()
```

An exit button has been added to help in user interaction. This button allows the user to exit from this application smoothly through the exit_application function.

Output of GUI :

Main Page

This page shows the front welcome page of the scammer detector application.

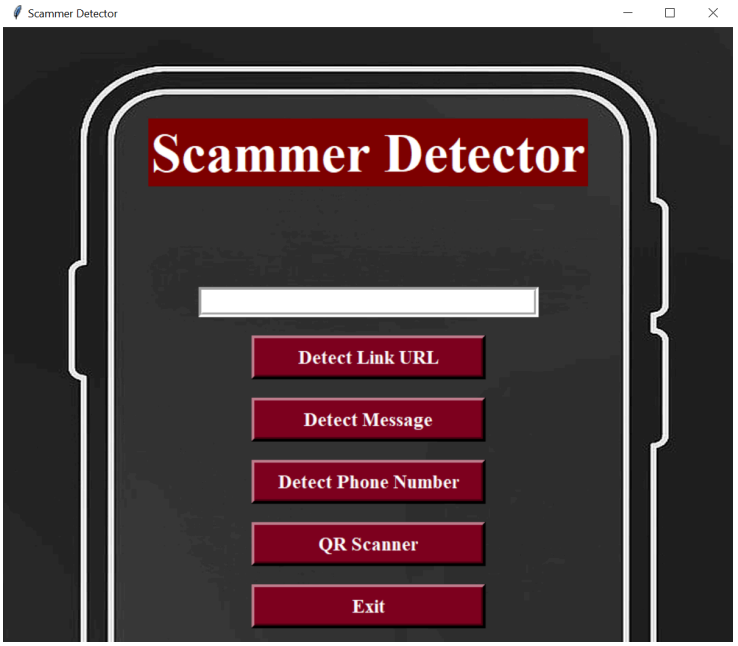


This page shows the main page for the user to check or detect the information that they get.

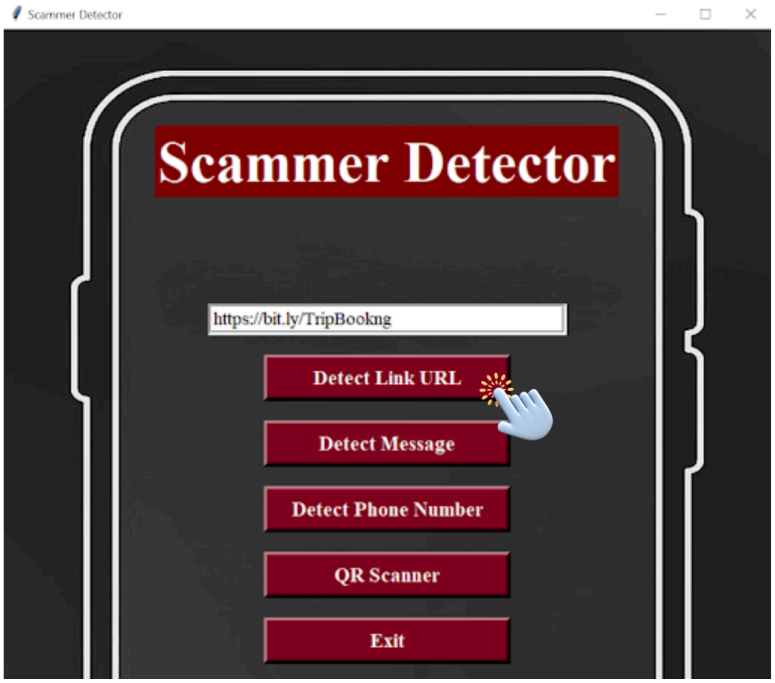
4.0 RESULT AND DISCUSSION

At the beginning of the code execution, users are greeted with the GUI homepage featuring the "SCAM ALERT" icon and the title "Welcome to Scammer Detector." Users have the option to proceed by selecting the "Continue" button to access the main page.

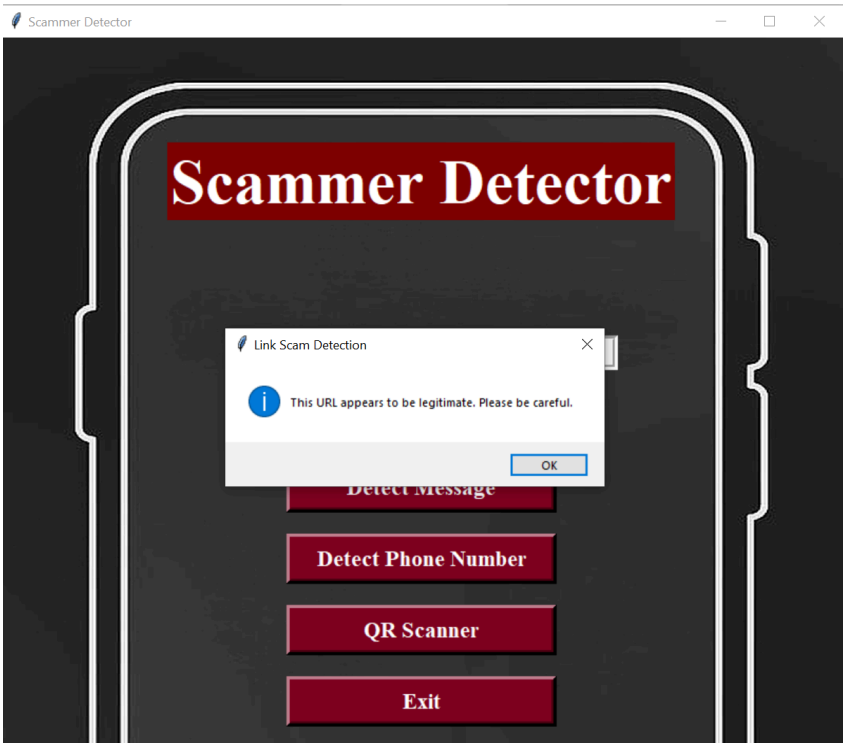
Upon clicking the "Continue" button, users are directed to the main page of the system, which includes the title "Scammer Detector," an input box, and five buttons. Users have the option to test the system by selecting any of these buttons. The input is entered into the input box, and the system is tested by utilizing the buttons below.



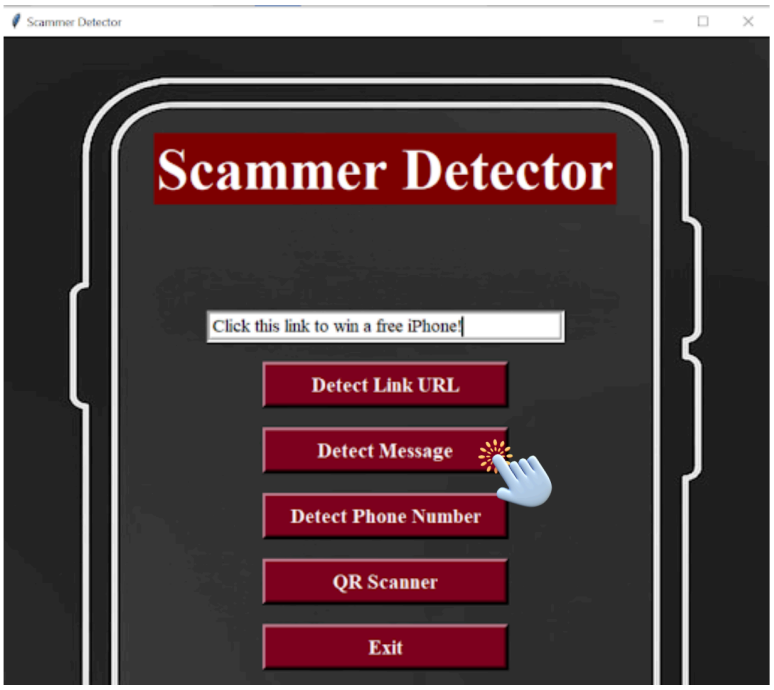
When the user clicks the first button, labelled "Detect Link URL," the system will assess whether the URL entered by the user in the input box is genuine or potentially a scam. The diagram provided below displays an example of a user input URL that is legitimate.



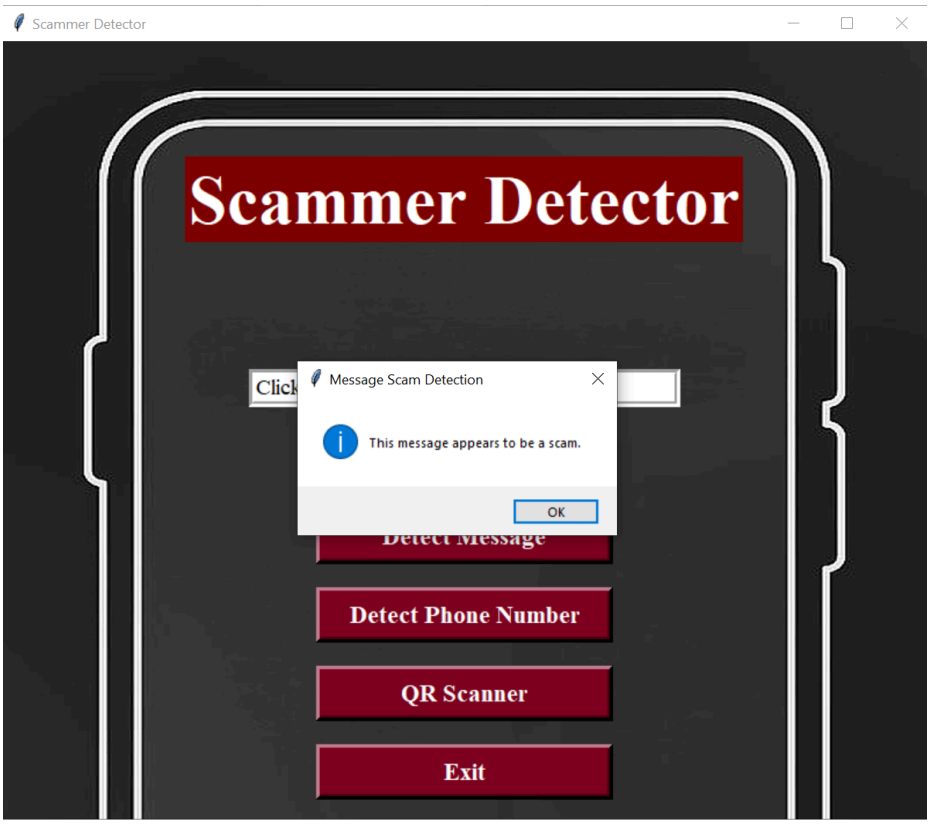
The output below is shown in the message box that the URL that user input is legitimate.



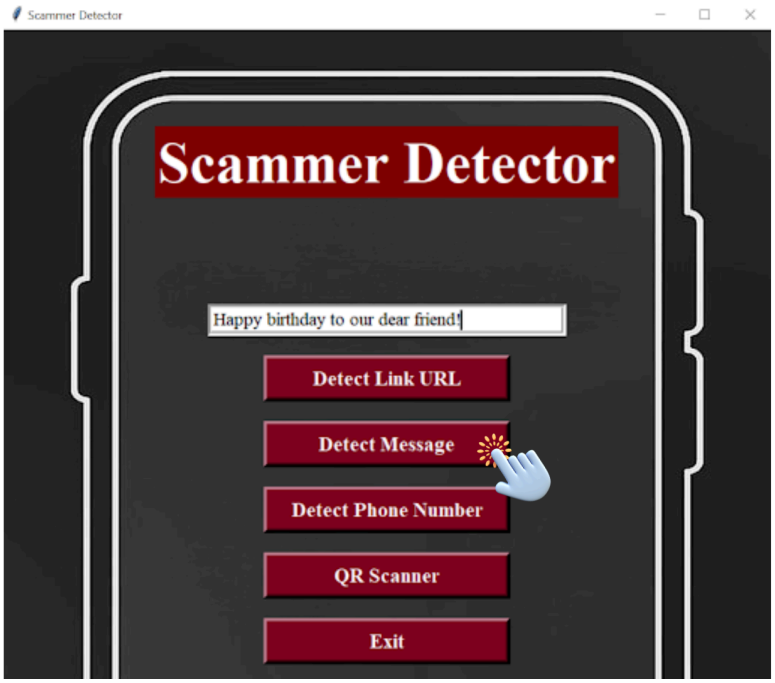
When the user selects the second button, labelled "Detect Message," the system will evaluate whether the message entered by the user is legitimate or a scam. The message provided by the user is sourced from files collected from various available sources.



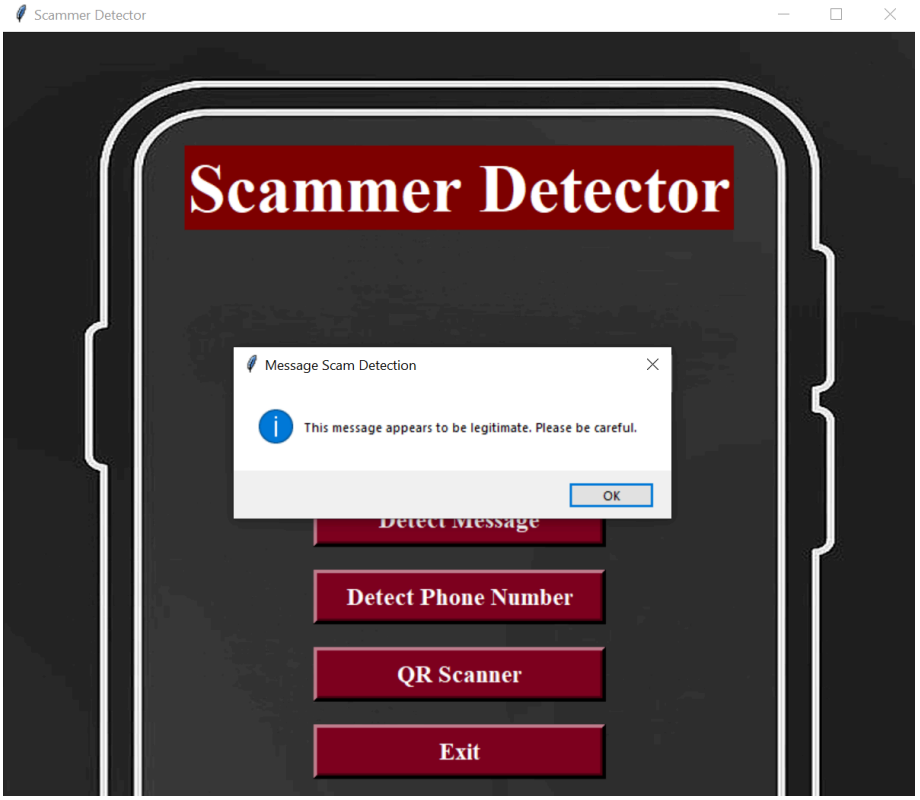
The output below shows the message box that the message that user input is a scammer.



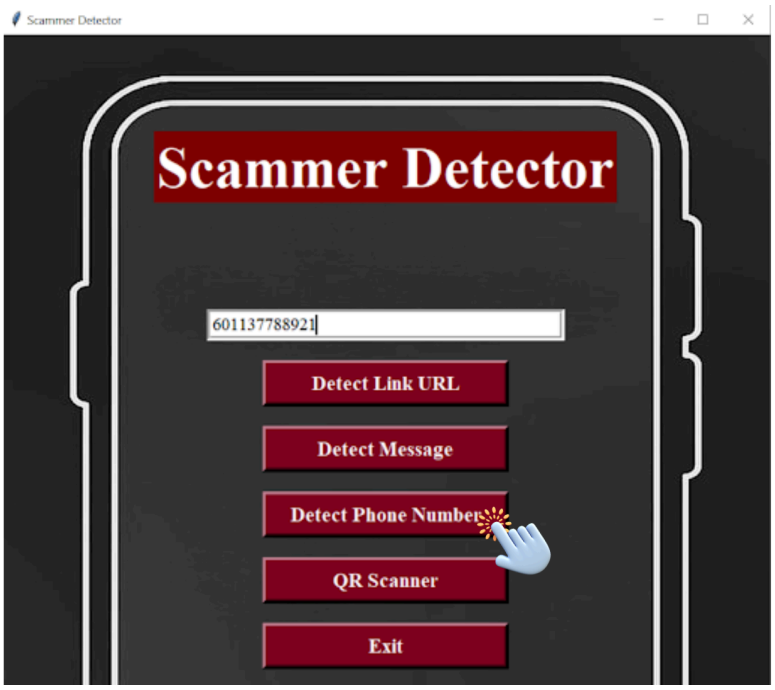
For confirmation, the message input by the user in the diagram below is legitimate. It will be evaluated by the system to determine whether it is correctly identified as legitimate or flagged as a scam.



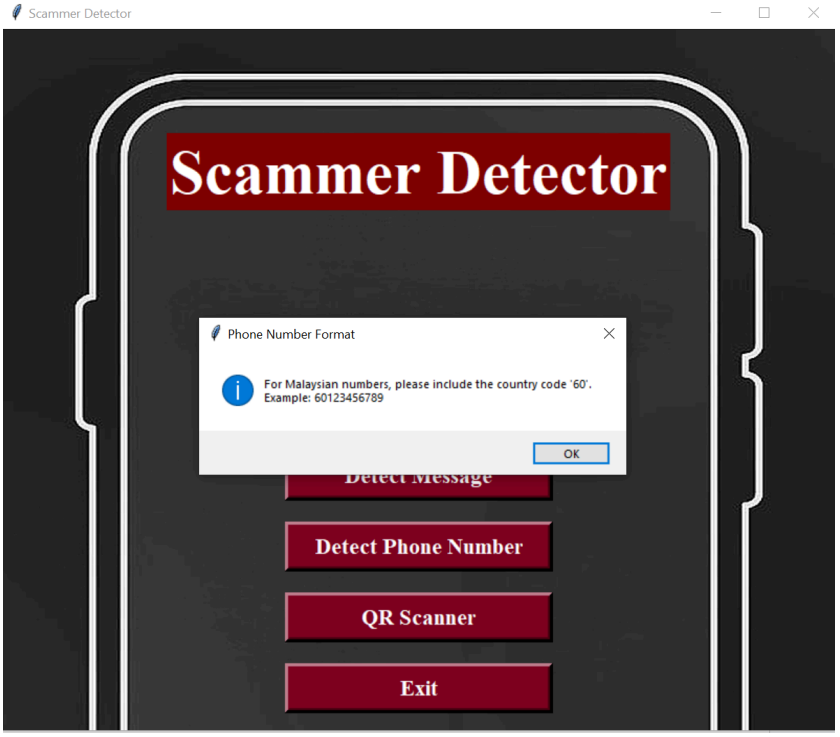
The output below shows that the system detects that the user input is legitimate.



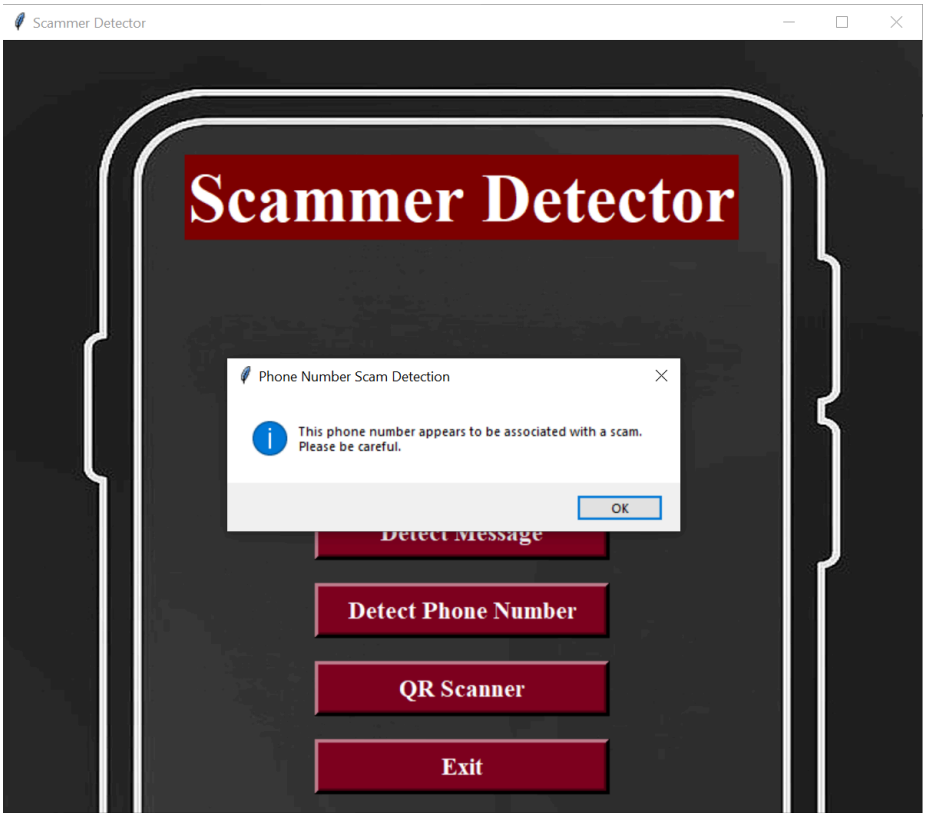
Upon selecting the third button, labelled "Detect Phone Number," the system will analyse the phone number entered by the user to ascertain its legitimacy or potential fraudulent nature. The phone numbers provided by the user are also sourced from files collected from various available sources.



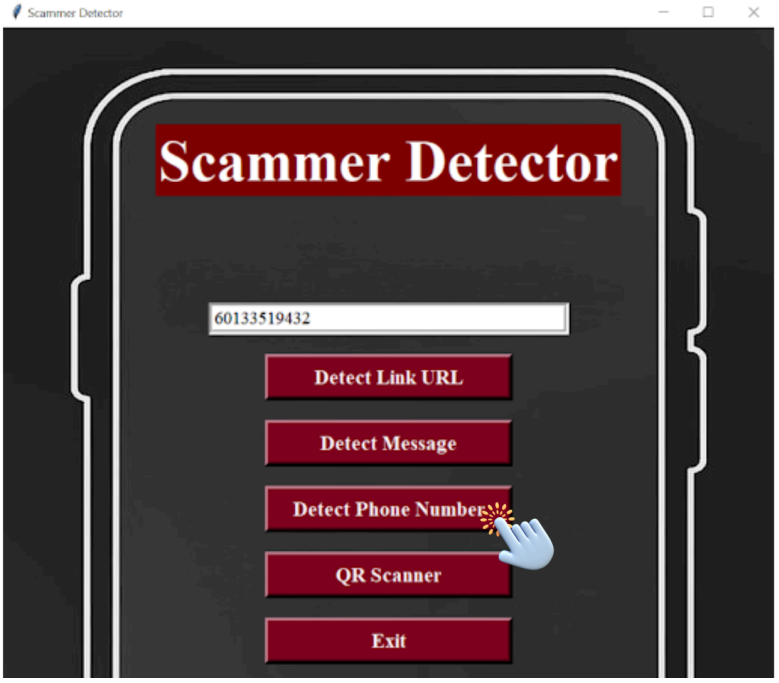
The output below shows that message box for users to input the country code ‘60’ for Malaysia numbers.



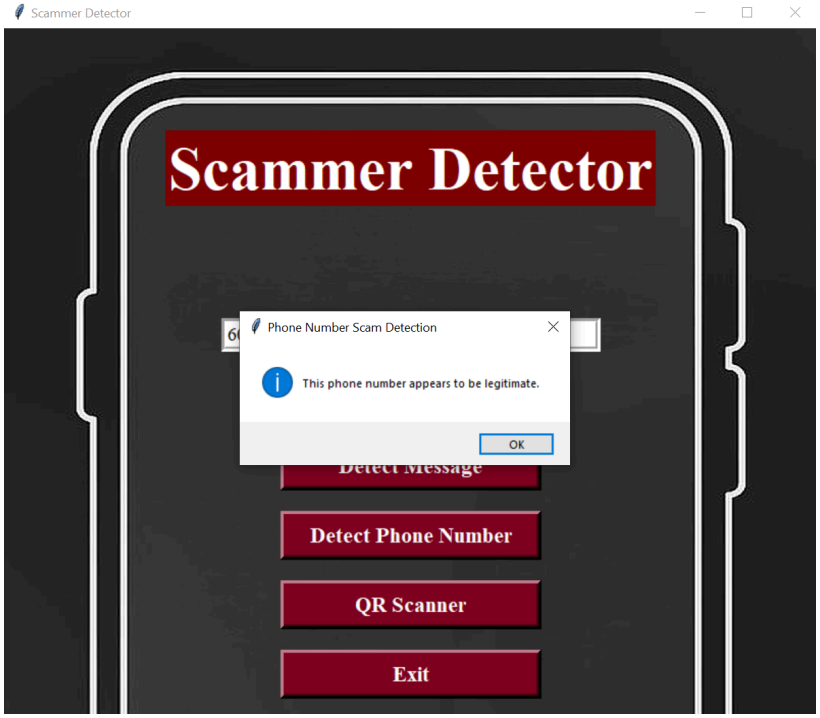
The output below shows the message box that the system detects that the user input of the phone number is a scammer.



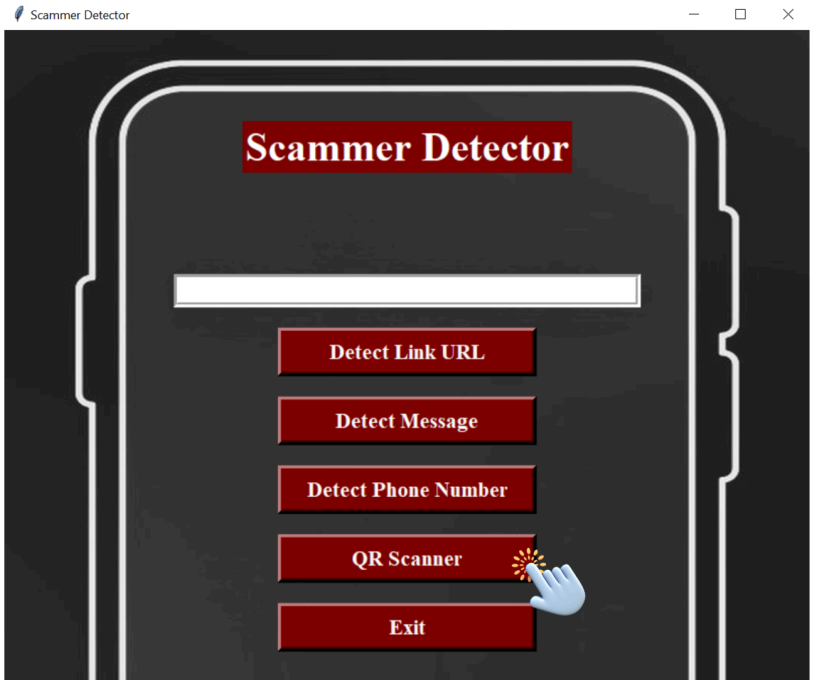
For confirmation, the user input the phone number that is legitimate to determine whether the system will detect the phone number is legitimate or scammer.



The output below shows the message box where the system successfully detects if the user input of phone number is legitimate.

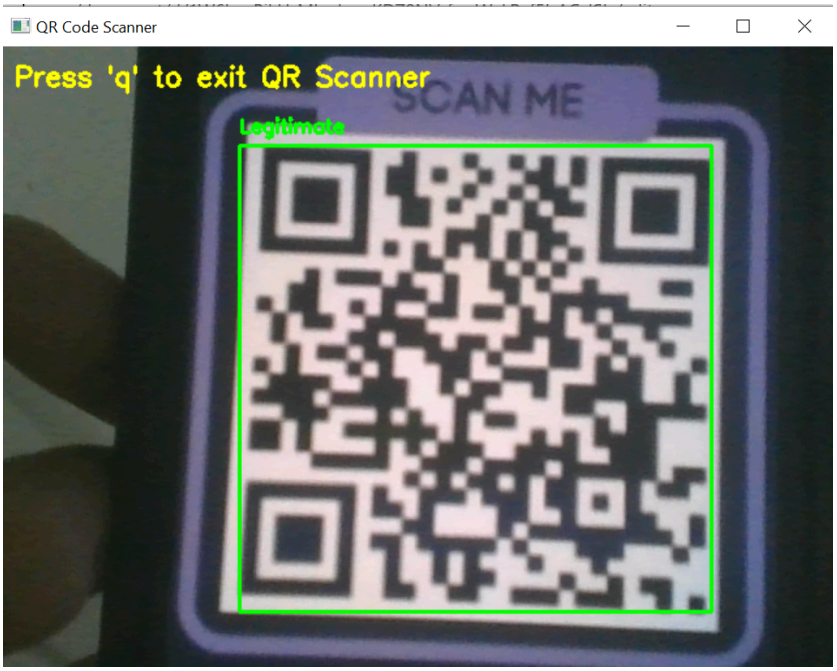


Upon clicking the fourth button, labelled "QR Scanner," the system activates the camera to capture live video input. It then processes this video feed in real-time, scanning for QR codes shown by the user. The system labels the QR codes as either legitimate or potentially fraudulent based on their content, providing real-time output to the user.

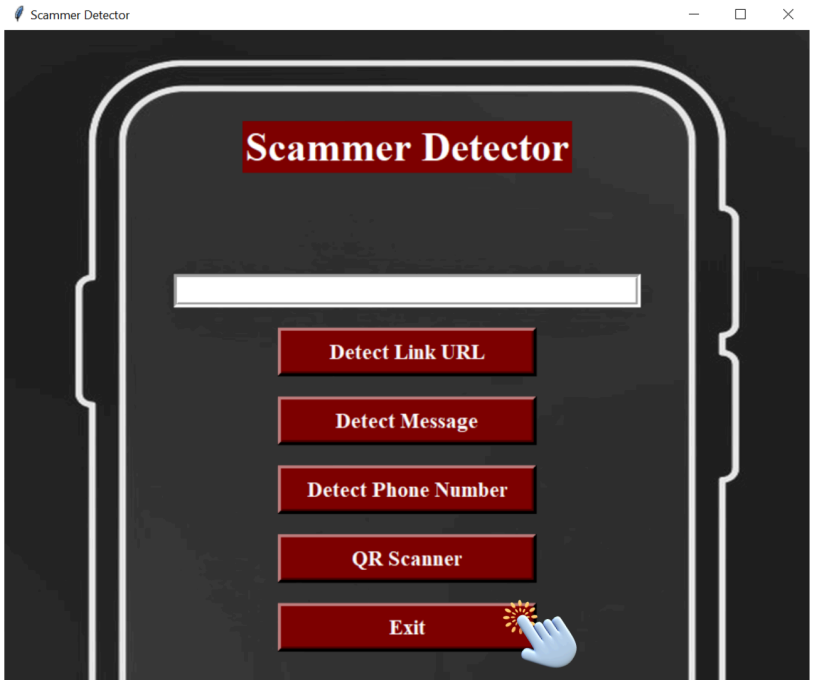


The output below shows that the algorithm quickly recognises the QR code provided by the user as being from a scam. The QR code comes from an online source, and the system is unable to verify its integrity, therefore it is classified as a possible scam.

For confirmation, the output displayed below indicates that the system effectively identifies the QR code presented by the user as legitimate. The QR code originates from the UMPSA organisation, demonstrating its authenticity without any doubt.



Exit is the last button on the main menu. If the user has utilised the Scammer Detector System, the user may quit.



5.0 CONCLUSION

To summarise, this project focuses on developing an AI-powered scammer detection system to identify link URL, message, phone number and QR code in real-time streaming. The system utilises face recognition algorithms, image processing techniques, natural languages processing and various libraries like Tkinter, NLTK, Numpy, PTL, OpenCV and Pyzbar to create a robust coding framework in Python. In addition, a user-friendly graphical user interface (GUI) with customised frames and buttons for seamless interaction is provided. Even if fraudulent operations in a variety of media have been successfully identified, difficulties still exist. For this study, we rely only on data obtained from a variety of sources. Moving forward, we hope that this technique will provide useful insights and aid in the creation of novel solutions. Furthermore, processing data streams in real time has issues in terms of computer resources, latency, and responsiveness. For efficient detection, the fraud detection system must be able to quickly analyse and respond to incoming data. Our attempt to create an AI-powered fraud detector shows great potential in addressing the growing wave of scams targeting individuals, particularly students, across several platforms. We hope to produce an efficient tool capable of detecting and revealing possible frauds in real time by combining powerful machine learning algorithms, natural language processing, and data analysis. This effort not only improves digital security for kids and the larger community, but it also raises awareness and education about online hazards. Furthermore, our initiative advances and applies AI technology to social concerns, enabling a safer and more secure digital world. By giving people the tools they need to defend themselves against harmful activity, we want to reduce money losses, privacy breaches, and emotional pain caused by scams, eventually increasing trust and confidence in online interactions.

6.0 REFERENCES

NLTK :: Natural Language Toolkit. (n.d.). https://www.nltk.org/

Build software better, together. (n.d.). GitHub. https://github.com/topics/phishing-detection

Kakarla, S. (2021, February 11). *Phishing URL Detection with Python and ML*. ActiveState. https://www.activestate.com/blog/phishing-url-detection-with-python-and-ml/

Lamberti, A. (2022, March 5). Phishing URL detection with Python - artificialis - medium. *Medium*. https://medium.com/artificialis/phishing-url-detection-with-python-a2c3dd3a87e8

7.0 APPENDIX

AI Coding :
https://drive.google.com/drive/folders/1QooroeRexqCSFXdG3BEqzStOpQTRaOim?usp=drive_link