King Khalid University
College of Science and Arts
Department of Computer Science

جامعة الملك خالد
كلية العلوم والآداب
قسم الحاسب الآلي

Tarasd

Advanced AI Phishing Detection System for Enhanced Cybersecurity

Prepared by:

Student Name	Student ID	Signature
Lujain Saad Al Manea	443817081	
Haneen Hussain Alrubaie	442809210	
Marym Hussain Alasmari	443808216	
Bayan Awad Alqarni	442815531	
Wateen Ibrahim al Mubarak	443806288	
Asrar Saeed Alshahrani	443806047	

Supervised by:

Dr.Suha Abdullah Alhifthi

Academic Year 2024-2025
College of Science and Arts-KM Department of Computer Science King Khalid University

COMMITTEE REPORT

The committee has examined and approved the graduation project report submitted by the student for the B.Sc. in Computer Science. It is considered to be an adequate project document.

Supervisor	Examiner1	Examiner2
Name:	Name:	Name:
Signature:	Signature:	Signature:
Date: / / 2025	Date: / / 2025	Date: / / 2025

ACKNOWLEDGMENTS

First and foremost, we give thanks to God for His guidance and blessings. We would like to express our sincere gratitude to everyone who has supported us throughout this journey, beginning with our families and the dedicated members of our team. A special acknowledgment goes to **Dr. Suha Al Hafithi**, Lecturer at King Khalid University, for her unwavering belief in our abilities and for granting us the opportunity to undertake this project. Her invaluable guidance and insight have been instrumental in our success, providing us with the necessary resources and information to bring this project to fruition.

DECLARATIONS

We declare that everything written in this project is the result of our full effort and work. We did not copy from any sources except in cases where references or explicit acknowledgments are provided in the text. Furthermore, no part of this project was written by anyone else on our behalf.

Name	**Lujain Saad**	**Haneen Hussein**	**Bayan Awad**	**Mrym Alsmari**	**Wateen Ibrahim**	**Asrar Saeed**
Sign						
Date	/ / 2025	/ / 2025	/ / 2025	/ / 2025	/ / 2025	/ / 2025

ABSTRACT

As email communication becomes a primary tool for individuals and organizations, phishing attacks—where fraudulent emails attempt to steal personal information—are becoming increasingly sophisticated. Traditional detection methods are often insufficient, as attackers employ advanced tactics to make these emails appear legitimate. By leveraging cutting-edge AI technologies such as machine learning (ML) and natural language processing (NLP), this system is designed to continually enhance its ability to detect phishing attempts. Not only will the system improve security, but it will also contribute to Saudi Arabia's Vision 2030 by offering a locally developed, high- security solution tailored to the needs of institutions and businesses. Key features of the system include automatic detection, reduced reliance on human intervention, and seamless integration with Gmail platform.

Key Words: Artificial Intelligence, Phishing Attacks, Phishing Detection, Cybersecurity, Machine Learning, Natural Language Processing, Email Security.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLE

LIST OF ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligence
NLP	Natural Language Processing
SMEs	Smal Medium Enterprises
2FA	Two-factor authentication

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

The project we are developing is an advanced AI-driven system with a Saudi identity, designed to provide an innovative and effective solution to the growing challenges in cybersecurity, especially in combating email phishing attacks. Small and medium sized enterprises (SMEs) increasingly rely on email as their primary means of communication. However, the frequency and sophistication of phishing attacks targeting personal data and sensitive information have increased dramatically.

Phishing emails are designed to deceive and often mimic legitimate communications from trusted entities such as banks, online services, or employers. These messages can be extremely sophisticated, using personal details, logos, and formal-sounding language to convince recipients that they are authentic. As these attacks evolve, traditional detection methods such as spam filters or manually blacklisting known phishing domains have become less effective, as they rely on identifying previously known threats [1], [2].

This is where AI, especially machine learning (ML) and natural language processing (NLP), becomes indispensable. AI enables more sophisticated and adaptive email analysis, capable of spotting subtle patterns and anomalies that may be invisible to human users. Through natural language processing, the system can analyze the content of emails to identify phrases and patterns typically associated with phishing. For example, requests such as “verify your account” or “click here to update your details” may seem harmless but are strong indicators of phishing when analyzed in context. Using machine learning, the system can learn from past phishing attempts and continually improve its detection capabilities. The more data the system processes, the better it becomes at identifying new and evolving threats. AI can also improve user engagement by providing real-time alerts and flags when suspicious activity is detected. Rather than simply blocking or filtering emails, the system can notify users of why a particular message was flagged as a potential phishing attempt, helping to educate and raise awareness [3], [4].

1.2 PROBLEM STATEMENT

Saudi society, like global societies, suffers from the spread of email fraud. This phenomenon is getting worse day by day due to the rapid development of fraud technologies, making it difficult to distinguish between legitimate and fraudulent emails [5].

With the increasing reliance on email as a primary communication tool for businesses, phishing attacks have become one of the most prevalent and dangerous cyber threats. Phishing emails are designed to trick recipients into revealing sensitive information, such as passwords, financial details, and personal data, by imitating legitimate sources. As phishing evolves, traditional detection methods—such as spam filters and blacklists—are no longer sufficient to protect users from these sophisticated threats. The lack of advanced and adaptable systems to identify fraudulent emails poses significant cybersecurity risks, leading to data breaches, financial losses, and loss of user trust. Furthermore, many SMEs rely heavily on manual or outdated detection methods, which are time-consuming and error-prone. This increases the need for a more efficient and accurate solution that can automatically detect phishing attempts and adapt to new attack strategies [6], [7].

1.3 PROPOSED SYSTEM

Tarasd is a comprehensive AI-driven platform designed to tackle phishing threats in email communications while enhancing user cybersecurity awareness. It features a user-friendly website for accessing a Windows 10-compatible desktop application with advanced login options, including Gmail-based authentication and two-factor authentication (2FA). Leveraging machine learning (ML) and natural language processing (NLP), Tarasd detects and flags phishing emails in real-time, providing actionable safety insights. Additionally, an integrated AI assistant offers guidance and access to educational content, such as interactive quizzes, to empower users with knowledge about phishing techniques and cybersecurity threats.

1.4 AIM OF THE PROJECT

The aim of this project is to develop a Saudi-identity system using advanced artificial intelligence (AI) to detect phishing and suspicious emails. The system will leverage machine learning (ML) and natural language processing (NLP) to enhance cybersecurity and protect users from email fraud, contributing to the goals of Saudi Arabia's Vision 2030. The project seeks to provide a secure, locally developed solution designed for the needs of SMEs within the Kingdom.

1.5 OBJECTIVES OF THE PROJECT

1. Develop an Integrated User Platform:
    - Develop a professional website to introduce the "Tarasd" project, showcasing its vision and mission with a user-friendly interface.
    - Allow users to easily download the Windows compatible desktop application.
2. Build a Windows-Compatible Desktop Application:
    - Develop a desktop application for Windows that includes secure login mechanisms, featuring Gmail-based authentication and Google Authenticator for two-factor authentication (2FA), ensuring reliable and safe user experience.
    - Display occasional advertisements on the homepage, highlighting the latest and most popular cybersecurity threats through notifications.
    - Implement administrative features for efficient content and Quiz updates
3. Implement Advanced Phishing Detection and Real-Time Alerts:
    - Develop an AI-powered phishing detection system using machine learning
    - (ML) algorithms such as Support Vector Machine (SVM), along with natural language processing (NLP) techniques.
    - Analyze Gmail messages and classify them as either safe or phishing, ensuring continuous user protection.
    - Implement notifications to alert users of potential phishing emails in real time and prevent misclassification of important messages such as spam.

4. Users will be able to:

- Flag it: Mark the email as phishing.
- Keep it: Dismiss the alert without altering the email.

5. Enhance Usability with AI Assistant RAG and Educational Content:

- Integrate an intelligent AI assistant, designed as a task-oriented conversational agent (chatbot), to help users navigate the system and access educational content or quizzes.
- Enhance the chatbot with Retrieval-Augmented Generation (RAG) using the allMiniLM-L6-v2 model from Microsoft to efficiently retrieve relevant information from a knowledge base before generating responses, improving accuracy and relevance.
- Fine-tune the assistant with a tailored dataset to align it with the project's specific functionalities and ensure precise responses in the cybersecurity domain.

6. Enhancing Cybersecurity Awareness Through Interactive Learning

- Deliver educational content through interactive quizzes designed to evaluate and strengthen users' understanding of phishing techniques and cybersecurity threats, with a focus on improving learning outcomes and raising awareness of online security.

1.6 SCOPE OF THE PROJECT

Our project "Tarasd" focuses on meeting the needs of small and medium enterprises by providing an intelligent system to protect email from phishing and enhance cybersecurity awareness. We developed a Windows -compatible desktop application that relies on AI technologies such as Machine Learning (ML) and Natural Language Processing (NLP) to analyze Gmail messages and detect phishing with high accuracy, while sending instant alerts to users. We also created a website that allows users to easily download the application and view the project's goals and vision. We also added a smart assistant as an desktop app conversation that provides interactive educational tips to improve the handling and interaction with the system. Our goal is to enable organizations to enhance their cybersecurity in an effective and easy-to-use way.

In our future work, we aspire to become one of the top security companies in Saudi Arabia, based on the accuracy of our model in detecting phishing attempts. We intend to diversify the awareness methods we provide to users and develop more effective solutions to address phishing problems. By focusing on enhancing user education and engagement, we aim to empower organizations with the tools and knowledge needed to recognize and combat phishing threats, thereby strengthening overall cybersecurity in the region. Additionally, we plan to learn typical user behavior patterns to detect anomalies that may indicate phishing attempts using User and Entity Behavior Analytics (UEBA) systems. We also plan to support multiple email services, not just Gmail, to broaden the scope of our phishing detection capabilities and provide comprehensive protection across different platforms.

1.7 SIGNIFICANCE OF THE PROJECT

The importance of this project lies in its ability to revolutionize how phishing attacks are detected, especially in the context of the growing digital ecosystem in Saudi Arabia. By using artificial intelligence, specifically machine learning (ML) and natural language processing (NLP), the system enhances the detection of phishing emails with high accuracy. This proactive approach ensures that users are protected from evolving cyber threats, contributing to the overall cybersecurity resilience of the Kingdom.

1.8 PROJECT ORGANIZATION

The project is structured across six comprehensive chapters. Chapter 1 introduces the core problem, outlines the proposed solution, and details the project's objectives, scope, significance, and methodology. Chapter 2 provides a background and literature review, comparing existing systems and conducting feasibility analysis, which forms the basis for defining both functional and non-functional requirements. Chapter 3 focuses on system analysis and design, utilizing various UML models such as use case, class, and sequence diagrams, alongside data flow diagrams and database design. Chapter 4 covers the implementation phase, detailing the development process, coding, and interface design for each module. Chapter 5 presents the testing and evaluation phase, employing rigorous methodologies including unit, integration, system, and acceptance tests to ensure performance and reliability. Lastly, Chapter 6 concludes the project by Summarizing key achievements, addressing limitations, and outlining future enhancements to improve and expand the system's capabilities.

		Name	Duration	Start	Finish
1		Choose Ideas	6 days	8/15/24 8:00 AM	8/22/24 5:00 PM
2		INTRODUCTION	6 days	8/23/24 8:00 AM	8/30/24 5:00 PM
3		PROBLEM STATEMENT	3 days	8/30/24 8:00 AM	9/3/24 5:00 PM
4		PROPOSED SYSTEM	6 days	9/4/24 8:00 AM	9/11/24 5:00 PM
5		AIM OF THE PROJECT	3 days	9/11/24 8:00 AM	9/13/24 5:00 PM
6		OBJECTIVES	1 day	9/13/24 8:00 AM	9/13/24 5:00 PM
7		SIGNIFICANCE	2 days	9/13/24 8:00 AM	9/16/24 5:00 PM
8		METHODOLOGIES	6 days	9/16/24 8:00 AM	9/23/24 5:00 PM
9		PROJECT ORGANIZATION	2 days	9/23/24 8:00 AM	9/24/24 5:00 PM
10		LITERATURE REVIEW	5 days	9/24/24 8:00 AM	9/30/24 5:00 PM
11		COMPARISON TABLE	3 days	9/30/24 8:00 AM	10/2/24 5:00 PM
12		FEASIBILITY ANALYSIS	3 days	10/2/24 8:00 AM	10/4/24 5:00 PM
13		REQUIREMENTS ANALYSIS	3 days	10/4/24 8:00 AM	10/8/24 5:00 PM

Fig 1 Task schedule

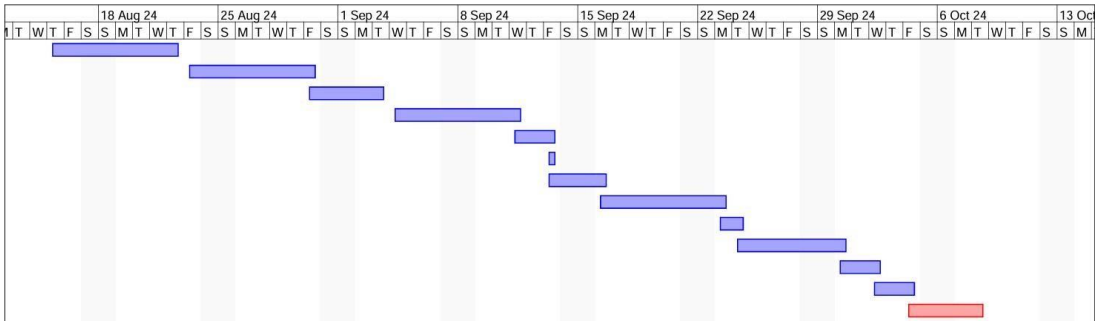


Fig 2 Gantt chart

1.9 SUMMARY

The project aims to Develop a desktop application an AI-driven phishing detection system to combat cybersecurity threats targeting email communications. The system leverages Machine Learning (ML) and Natural Language Processing (NLP) to improve phishing detection accuracy, automate detection processes, and protect sensitive user information. It reduces reliance on manual intervention and enhances trust in digital communication, making it particularly suitable for SMEs in Saudi Arabia. By integrating user-friendly features and real-time alerts, the project aligns with the goals of Vision 2030, ensuring a secure and efficient email security system

CHAPTER 2: BACKGROUND AND LITERATURE REVIEW

2.1 HISTORY

Origins of Phishing (1990s):

Phishing began in the mid-1990s, targeting users on early Internet platforms such as AOL (America Online). The term 'phishing' was coined as a metaphor for 'fishing' for sensitive information. Attackers would send fraudulent emails, pretending to be from AOL customer support, and trick users into providing their login credentials. The stolen accounts were then used to access paid services or resold on black markets. This early form of phishing relied heavily on social engineering, exploiting users' lack of awareness of online threats.

Attackers would create fake emails and instant messages that appeared to be from legitimate sources.

The Expansion of Phishing (Early 2000s):

By the early 2000s, phishing attacks began to spread globally, with scammers using email as their primary tool. Attackers would create emails that appeared to come from trusted organizations such as banks, ecommerce sites, or email providers.

Notable Attacks: Some of the most notorious early phishing attacks targeted PayPal and eBay users, with scammers sending emails asking for account verification.

Increased Sophistication: As phishing techniques advanced, emails became more convincing, using authentic logos, professional language, and sometimes personal details to trick recipients. The Rise of Anti-Phishing Technologies (Mid-2000s):

As phishing attacks increased, email providers and security companies began developing tools to combat them.

Spam filters, blacklists, and other email security measures were created to detect and block phishing attempts. However, attackers continued to evolve their tactics to bypass these defenses.

Introduction of Phishing Techniques: Fraudsters began using email spoofing, a technique in which they changed the “from” field of an email to make it appear as if it came from a legitimate source. This made it difficult for users to distinguish between real and fake emails.

Growth of Phishing and Cybercrime (2010s):

The first decade of the 21st century saw a significant increase in phishing attacks, both in frequency and sophistication. Attackers began using malware, ransomware, and keyloggers embedded in phishing emails to not only steal information but also compromise entire networks.

Business Email Compromise (BEC): A particularly damaging form of phishing targeting businesses became prominent during this time. In BEC attacks, fraudsters impersonate company executives or vendors, tricking employees into transferring large sums of money into fraudulent accounts.

Notable Incident: In 2016, a phishing attack on Democratic National Committee employees led to the breach of sensitive political data, which had devastating consequences during the US presidential election.

AI and Machine Learning in Phishing (Late 2000s – Present):

As phishing emails become more difficult to detect using traditional methods, security companies have turned to AI and machine learning to combat these threats. AI-powered systems can analyze emails in real-time, looking for subtle signs of phishing that might not be obvious to humans.

2.2 LITERATURE REVIEW

In the pursuit of progress and innovation in the field of phishing detection through the application of AI and ML techniques, it is essential to build on the existing body of knowledge and research. This section provides an overview of the relevant literature and previous works that have significantly contributed to shaping our current understanding of this critical cybersecurity issue by examining previous research and developments, gaining insight, and developing new solutions [8].

2.2.1 MAIL GUN

Mail gun is an email management and analysis system designed to combat phishing and other cyber threats. It utilizes artificial intelligence and machine learning to detect suspicious messages by analyzing content, assessing sender reputation, and providing real-time alerts. Mail gun adapts continuously to emerging threats, making it an effective tool for both individuals and businesses in enhancing their email security [11].

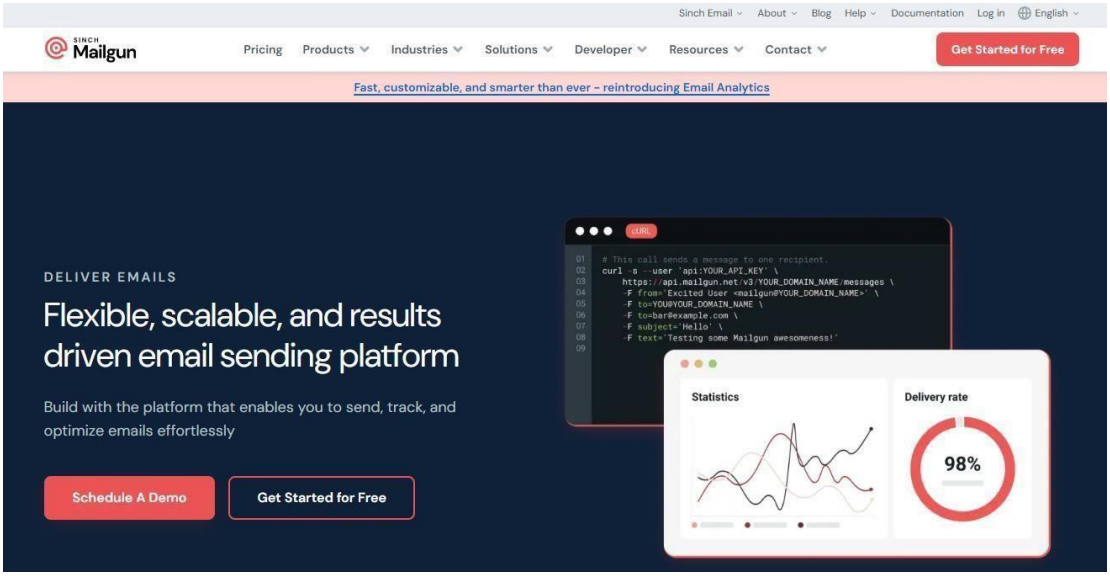


Fig 3 Mialgun Interface

Advantages:

- Email Verification: Provides robust email validation to ensure messages are sent to legitimate addresses, reducing the risk of phishing attacks.
- Integration with Security Tools: Easily integrates with existing security solutions to enhance phishing detection and prevention capabilities.

Disadvantages:

- Cost: Can be expensive for small businesses, especially as email volume increases.
- Support Limitations: Some users report slower response times from customer support, which can be challenging for urgent issues.
- Complexity: May have a steeper learning curve for new users, requiring time to fully utilize its features.

2.2.2 SPAM TITAN

Spam Titan is an email security solution that protects businesses from spam and cyber threats. It provides advanced spam filtering, malware protection, and detailed reporting, helping to minimize unwanted emails and enhance overall security.

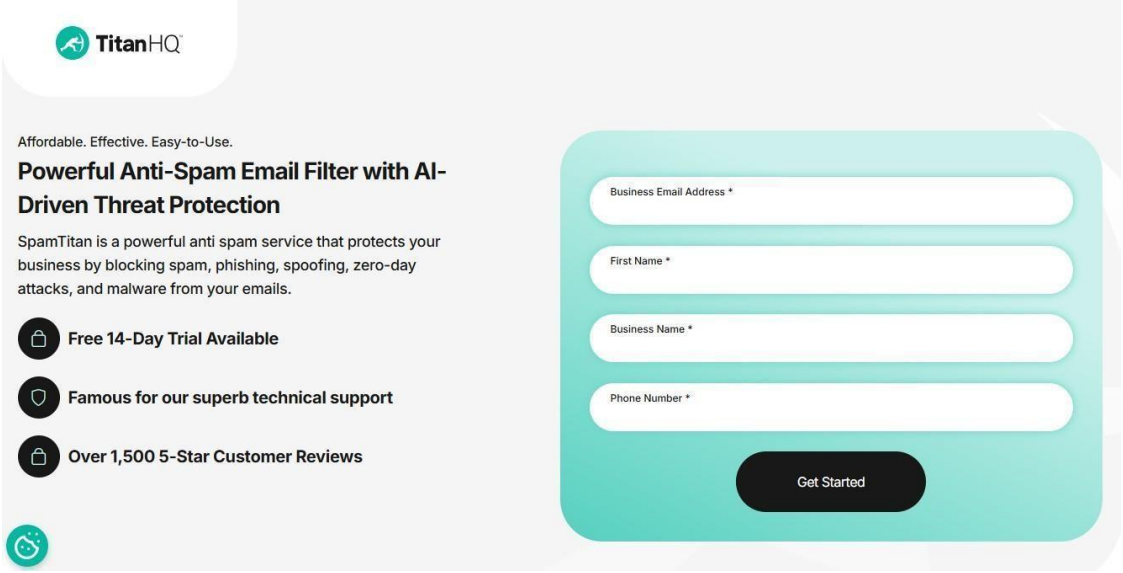


Fig 4 Spam Titan Interface

Advantages:

- Robust Phishing Protection: Effectively identifies and blocks phishing emails using advanced filtering techniques.
- Real-Time Threat Analysis: Continuously analyzes incoming emails for suspicious activity, providing immediate alerts and protection.

Disadvantages:

- Cost: May be relatively expensive for small businesses compared to other email security solutions.
- False Positives: Occasionally filters legitimate emails as phishing, which can disrupt communication.
- Dependency on Internet: Requires a stable internet connection for optimal performance.

2.2.3 ZIX

Zix is an email security solution that focuses on data protection and compliance. It offers strong email encryption to safeguard sensitive information, widely used in finance and

healthcare to meet regulatory standards like HIPAA and GDPR, while ensuring an easy-to-use interface and seamless integration [10].

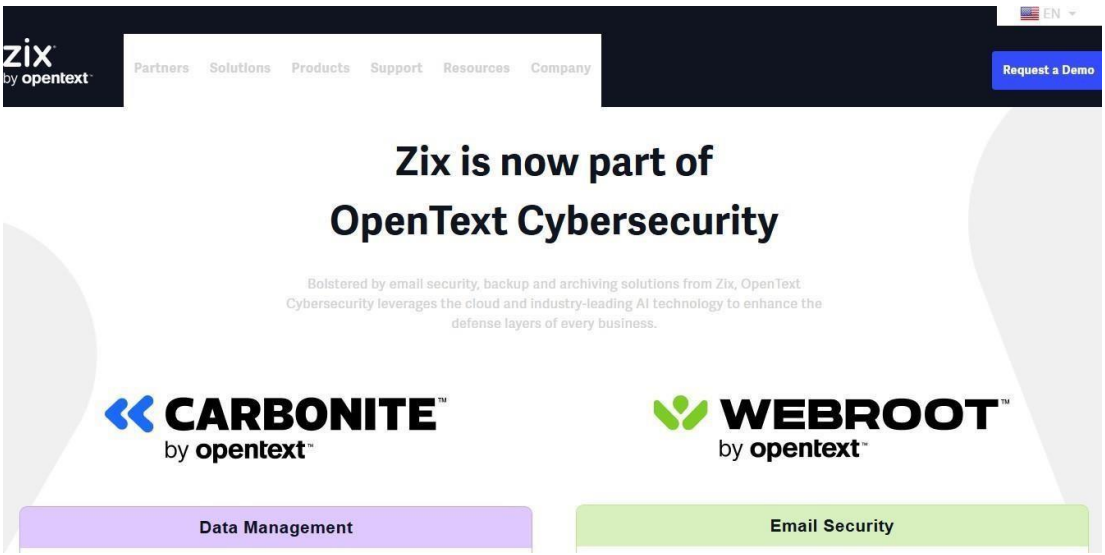


Fig 5 Zix Interface

Advantages:

- Advanced Threat Detection: Utilizes sophisticated algorithms to identify and filter phishing attempts before they reach users' inboxes.
- Real-Time Scanning: Continuously scans incoming emails for suspicious links and content, providing immediate protection against phishing threats.

Disadvantages:

- Cost: May be relatively expensive for small businesses compared to other email security solutions.
- Dependency on the Internet: Requires a stable internet connection for effective functionality.

2.3 COMPARISON TABLE

The following table summarizes the comparison between related platforms and our proposed platform:

Table 1Platforms Comparison

Feature	Our System	Mialgun	SpamTitan	Zix
Security Awareness Training	Yes	No	No	No
Detection Method (NLP/URL)	NLP	NLP	None	NLP
Real-Time Alert	Yes	Yes	Yes	Yes
AI Assistant Support	Yes	No	No	No
Cost	Free	Paid	Paid	Paid
Setup	Desktop	Web-based	Web-based	Web-based

2.4 FEASIBILITY ANALYSIS

2.4.1 TECHNICAL FEASIBILITY

Technical feasibility helps us evaluate the feasibility of our project and to ensure that the requirements are met effectively. We use several key points:

2.4.1.1 SOFTWARE REQUIREMENTS

Windows operating system: The desktop application must be compatible with Windows and newer versions to ensure broad compatibility across different environments and to leverage Windows-specific features.

Database:

- SQLite : will be used to store and manage lightweight local data such as user preferences, session states, and cached phishing detection results. It is a self-contained, serverless, and zero-configuration relational database, making it ideal

for desktop applications requiring fast and reliable storage without the need for a separate database server.

Development Tools:

- Python: Python will be the primary programming language for developing the machine learning model (for phishing detection) and handling other backend components of the desktop application. Programming Libraries:
- Scikit-learn: This library will implement machine learning algorithms like Support Vector Machine (SVM),

Natural Language Processing (NLP) Tools:

- TF-IDF (Term Frequency - Inverse Document Frequency) is a feature extraction technique used in Natural Language Processing (NLP) and text classification. It converts text into numerical vectors based on the importance of words in a document.

Machine Learning Frameworks:

- Google Collaboratory (Colab): Colab will be used for building and training machine learning models in a cloud-based environment, with access to free GPU resources. It is ideal for handling large datasets and enabling real-time phishing detection.

Website Framework:

- HTML and CSS: These technologies will be used to design and structure the website, providing a responsive, user-friendly interface for the project.
- JavaScript: Used to add interactivity and dynamic functionalities to the website.

Desktop Application Framework:

- Visual Studio with C# Language.

Real-Time Alerts:

- Notifications: The application will include notifications to alert users in real-time about phishing threats.

2.4.1.2 HARDWARE REQUIREMENTS

In this proposed system, we can use a personal computer with the following recommended specifications:

- Processor (CPU): Intel Core i5 or equivalent for better performance.
- Memory (RAM): 8 GB or more, especially if the app will analyze large datasets or run background processes.

2.5 ECONOMICAL FEASIBILITY

The economic feasibility of the project aims to assess whether the potential costs are commensurate with the expected benefits, taking into account the financial factors that may affect the success of the system. The project relies on the free SQLite database, which contributes significantly to reducing software costs. The project also requires a Windows operating system, in addition to the need for a desktop computer with certain specifications.

2.4.2.1 SOFTWARE REQUIREMENTS COST

Table 2 Software Requirements Cost

Resource	Cost
SQLite Database	Free
Visual Studio	Free
Google Collaboration	Free
Windows	$100

2.4.2.2 HARDWARE REQUIREMENTS COST

Table 3 Hardware Requirements Cost

Resource	Cost	Period
Desktop	1000–1000–4000	Forever

2.6 REQUIREMENTS ANALYSIS

2.6.1 FUNCTIONAL REQUIREMENTS:

1. Visit Website

- The system shall allow users to access the project's official website using a browser.
- The system shall display information about the project, its features, and usage.

2. Download Application

- The system shall provide a download link for the Windows compatible desktop application.

3. Login to Application

- The system shall allow users to log in using their Gmail account credentials via OAuth 2.0.
- The system shall trigger Google Authenticator-based 2-Factor Authentication after successful Gmail login.

4. Access Home Page

- After successful login, the system shall redirect the user to the home page.
- The home page shall display dynamic content including cybersecurity advertisements.

5. Interact with AI Assistant

- The system shall include an AI assistant accessible from the home page.
- The AI assistant shall provide help with navigation and cybersecurity-related questions.

6. Access Educational Content

- The system shall allow users to access interactive educational content such as quizzes.
- The system shall track user progress in quizzes.

7. Fetch Emails

- Upon user consent, the system shall securely connect to the user's Gmail account.
- The system shall fetch new emails for analysis

8. Analyze Emails

- The system shall process fetched emails using AI algorithms to detect phishing content.
- The system shall classify each email as "Safe" or "Phishing."

9. Receive Alerts

- The system shall display real-time alerts for phishing emails as desktop notifications.

10. Act on Alert

- The system shall allow users to take action on flagged emails.
- The available actions shall include “Flag Email” or “Keep Email.”

11. Flag Email

- When a user confirms an email as phishing, the system shall keep it in phishing email box

12. Keep Email

- If the user chooses to keep the email, the system shall remove it from phishing email box

13. Update Educational Content (Admin Only)

- Admins shall log into the website backend using admin credentials.
- Admins shall be able to add, edit, or remove quizzes and educational materials.
- Admin changes shall be reflected on the client application immediately or upon next sync

.2.6.2 NON-FUNCTIONAL REQUIREMENTS

1. Security

- The system must ensure user data confidentiality and comply with cybersecurity best practices.
- Password authentication must be securely implemented.

2. Performance

- The system must quickly process emails and deliver phishing detection results within a short time frame.

3. Scalability

- The system should be scalable to accommodate increased email volume and support future email provider integrations.

4. Reliability

- The system must maintain high availability with minimal downtime.
- Phishing detection should continue even during temporary email service interruptions.

5. Usability

- The user interface must be intuitive and accessible to users with limited technical expertise.
- Visual indicators and navigation elements must be clear and easy to follow.

6. Maintainability

- The system should be easy to maintain and update with minimal disruption.

2.7 SUMMARY

In this chapter, we outlined the background and review of existing phishing detection systems. We identified gaps like complex usability and limited self-learning. A comparison shows the proposed system's advantages: faster performance, continuous learning, and multi-platform support. The feasibility analysis covers technical and economic aspects, emphasizing cost-efficiency. Requirements analysis details core features such as real-time detection, user-friendly interfaces, and security. This chapter lays the foundation for an advanced, scalable, and efficient phishing detection system

CHAPTER 3: DESCRIPTIONS OF THE PROJECT AND SYSTEM

3.1 INTRODUCTION

This chapter presents UML diagrams to illustrate the architecture of the system. We will discuss the basic components that make up the system and explain how they interact with each other. We will also highlight the database used to store the information necessary for the system, in addition to explaining the basic interfaces that enable users to interact within the system. In concluding this chapter, we will review the basic methods that we plan to implement in the proposed system.

3.2 INTERACTION ANALYSIS

Interaction analysis is a part of the dynamic behavior of a system, where diagrams are used to illustrate how different components of a system interact with each other. This interactive behavior is depicted using UML diagrams that illustrate the interactive activities of the system. Due to the complexity of the connections between components, multiple types of models are used to illustrate different aspects of interaction.

3.2.1. UML MODELS AND DIAGRAMS

Unified Modeling Language (UML) is a visual tool used in systems modeling that facilitates the presentation of the stages of system development in a uniform manner. Although UML is not a programming language, it allows the structure and behavior of a system to be visually illustrated, helping architects, business owners, and developers to design and analyze the system effectively.

3.2.1.1 USE CASE DIAGRAMS

A use case diagram visually represents how a system interacts with its users. It is often used in conjunction with other types of diagrams to show the different interactions of users and the range of possible use cases within a system. Each use case is depicted by an oval or circle, which illustrates a specific scenario or function.

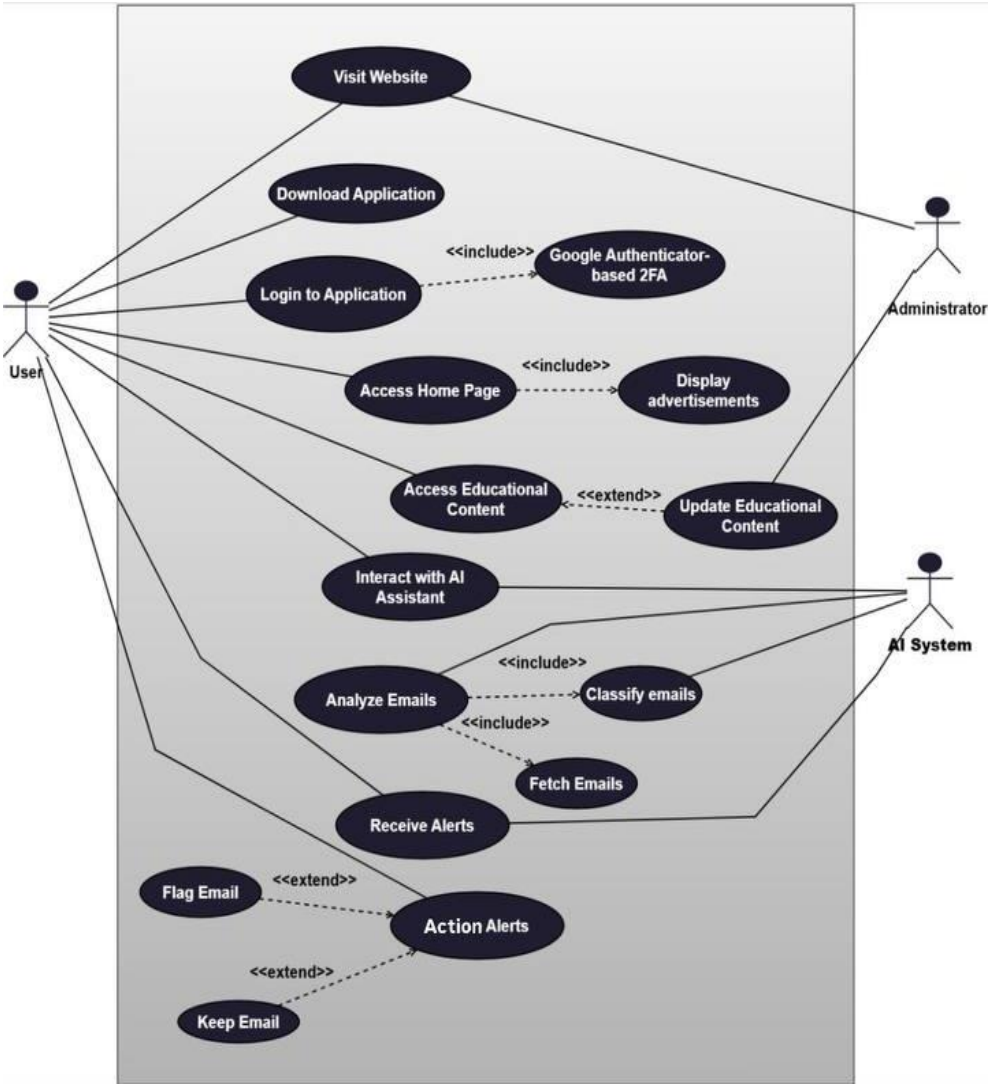


Fig 6Use case

3.2.1.2 CLASS DIAGRAM

A class diagram is a type of UML (Unified Modeling Language) diagram used to depict the structure of a system by showing classes and the relationships between them. This diagram focuses on identifying the objects that make up the system, their properties, behaviors, and the relationships between them.

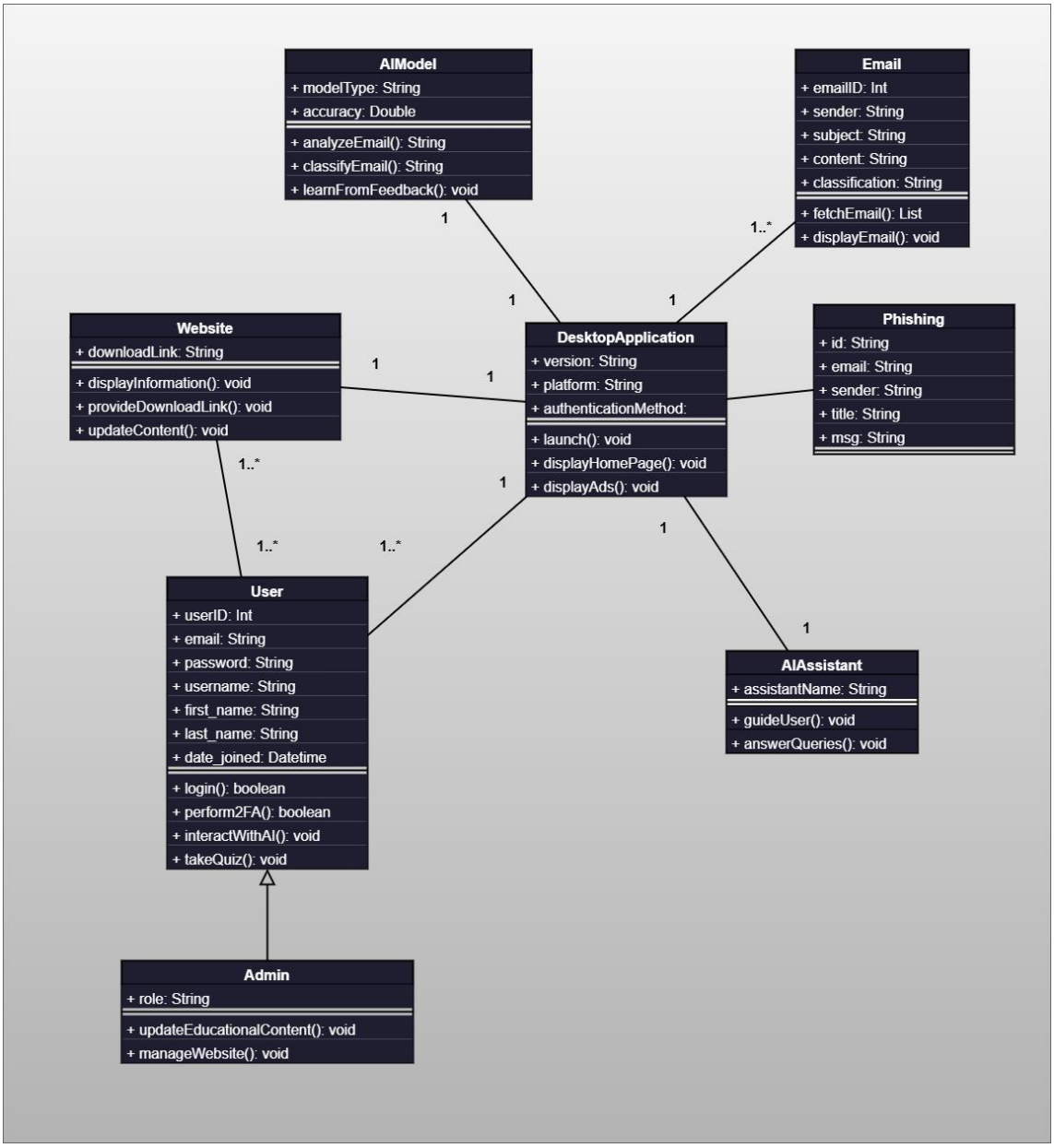


Fig 7 Class Diagram

3.2.1.3 SEQUENCE DIAGRAM

A sequence diagram is a tool used to display the sequence of interactions between objects within a system. It shows how objects interact, and the steps required to implement a specific scenario.

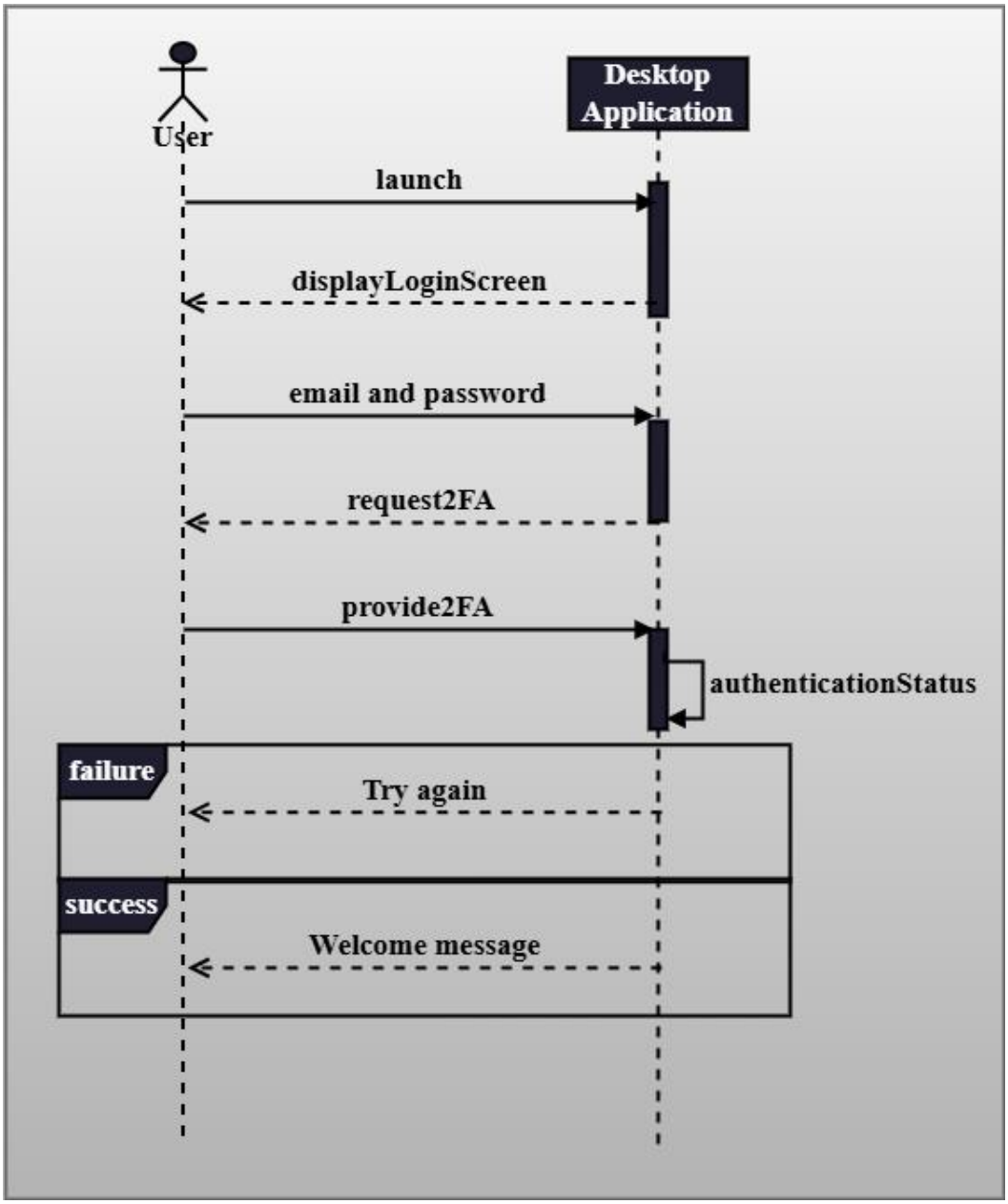


Fig 8Sequence Diagram 1

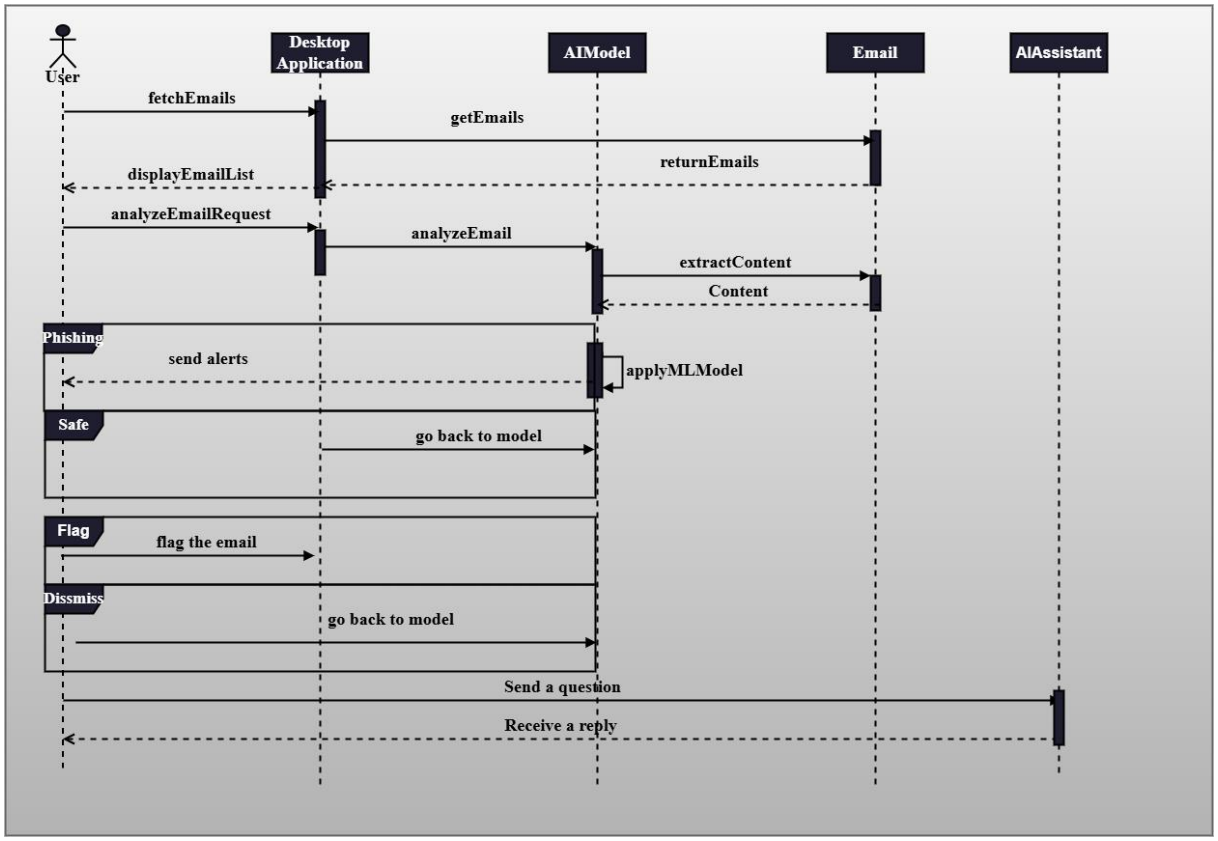


Fig 9 Sequence Diagram 2

3.2.1.4 ACTIVITY DIAGRAM

An activity diagram is a flowchart that focuses on the sequence of activities and actions and shows how objects interact with each other.

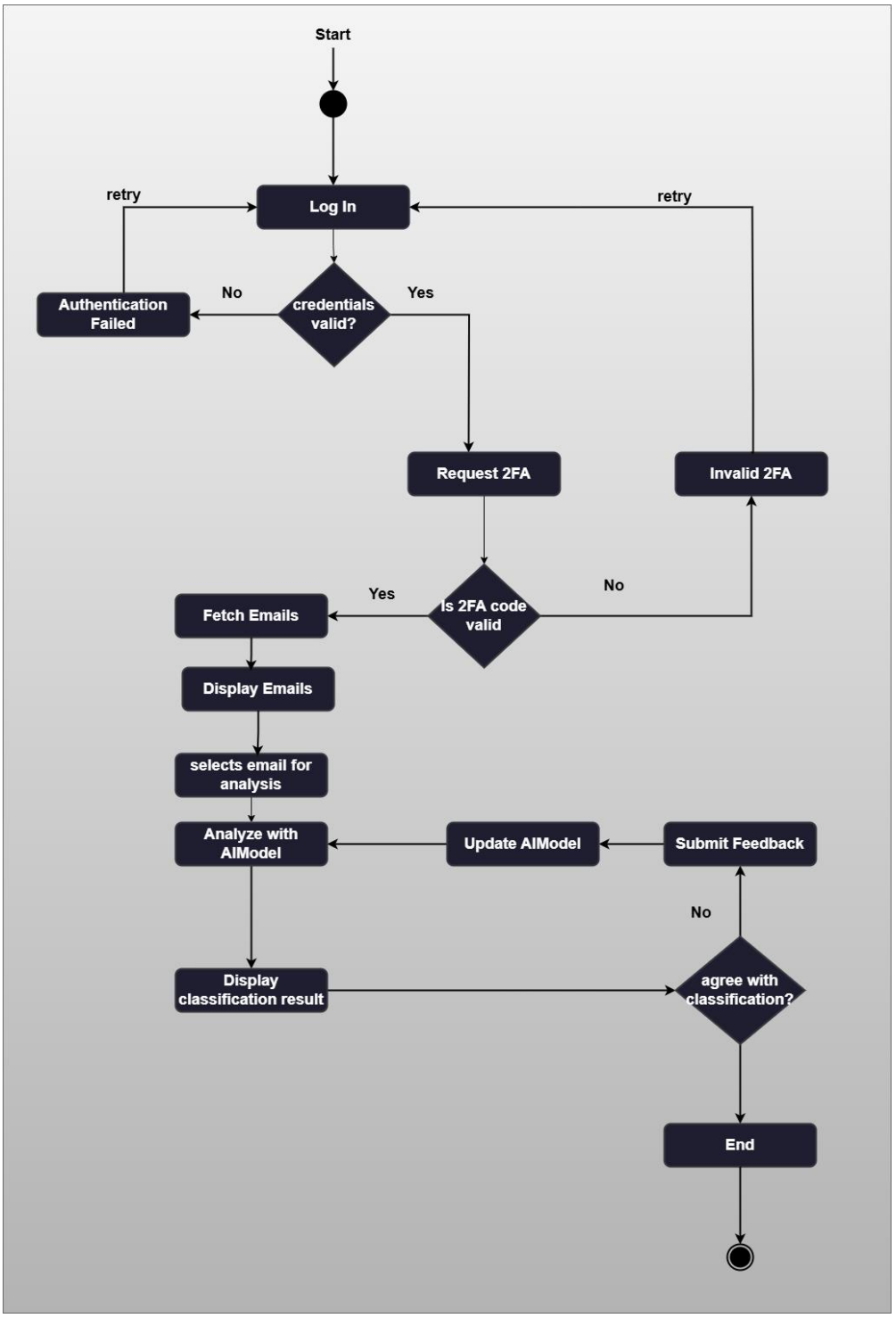


Fig 10 Activity Diagram

3.3 DESIGN CONCEPT

The design concept, which forms the foundation for the system's architecture and its components, requires careful consideration of various factors for developing a desktop application aimed at detecting phishing attempts in email messages using AI. This design concept explores the principles and techniques necessary to create an effective, user-friendly, and secure system capable of identifying and alerting users to potential phishing threats in real-time.

The proposed design prioritizes intuitive interaction, with a straightforward layout for managing and analyzing email security risks, instilling confidence in users regarding their email security. By integrating intelligent analysis capabilities, the system becomes a proactive tool that assists users in identifying and avoiding phishing threats while providing clear insights into identified risks.

3.4 CONTEXT FLOW DIAGRAMS

The Context Flow Diagram (CFD) is a high-level overview of the system that illustrates how it interacts with its external environment. It is used to identify the system's inputs, outputs, and main processes. A Context Flow Diagram is a type of Data Flow Diagram (DFD) specifically used to model the system's context and illustrate how it interacts with external entities, such as users, connected email servers, and security databases. The Context Flow Diagram is typically created early in the system development process and helps inform the creation of more detailed data flow diagrams and system components.

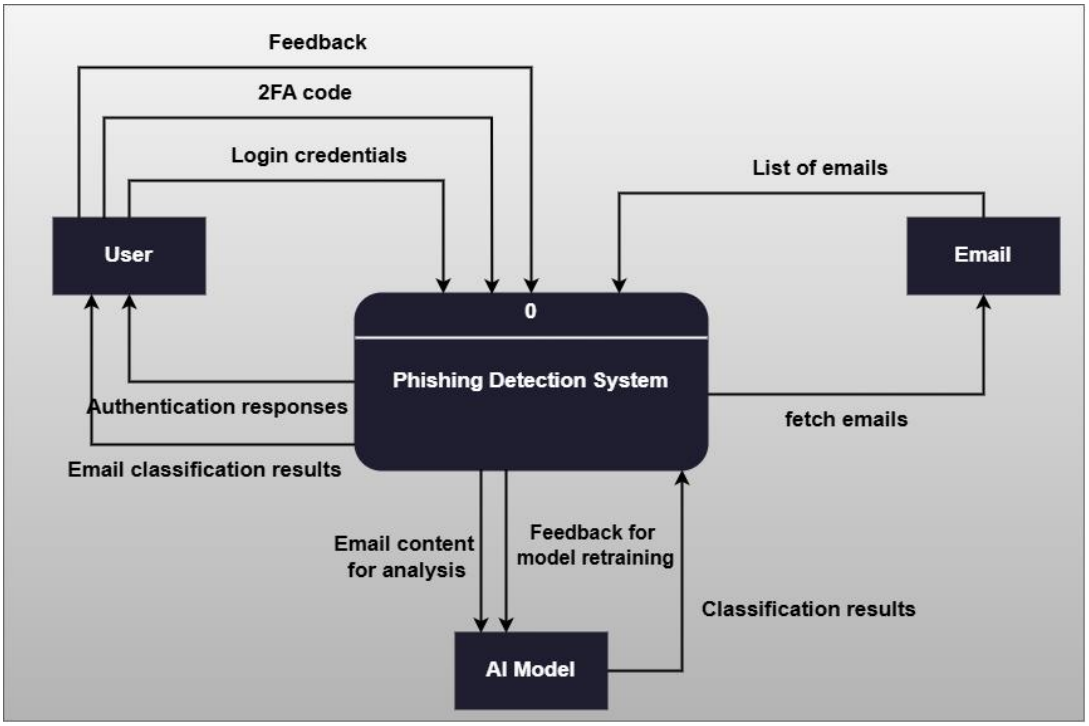


Fig 11 Context Flow Diagram

3.5 DATA FLOW DIAGRAM

The Data Flow Diagram (DFD) provides a structured and detailed view of data movement across the AI-powered phishing detection system, starting from the initial user interaction to backend processing by the phishing detection engine. This DFD illustrates how data flows between different processes within the system, data storage locations, and interactions with external entities.

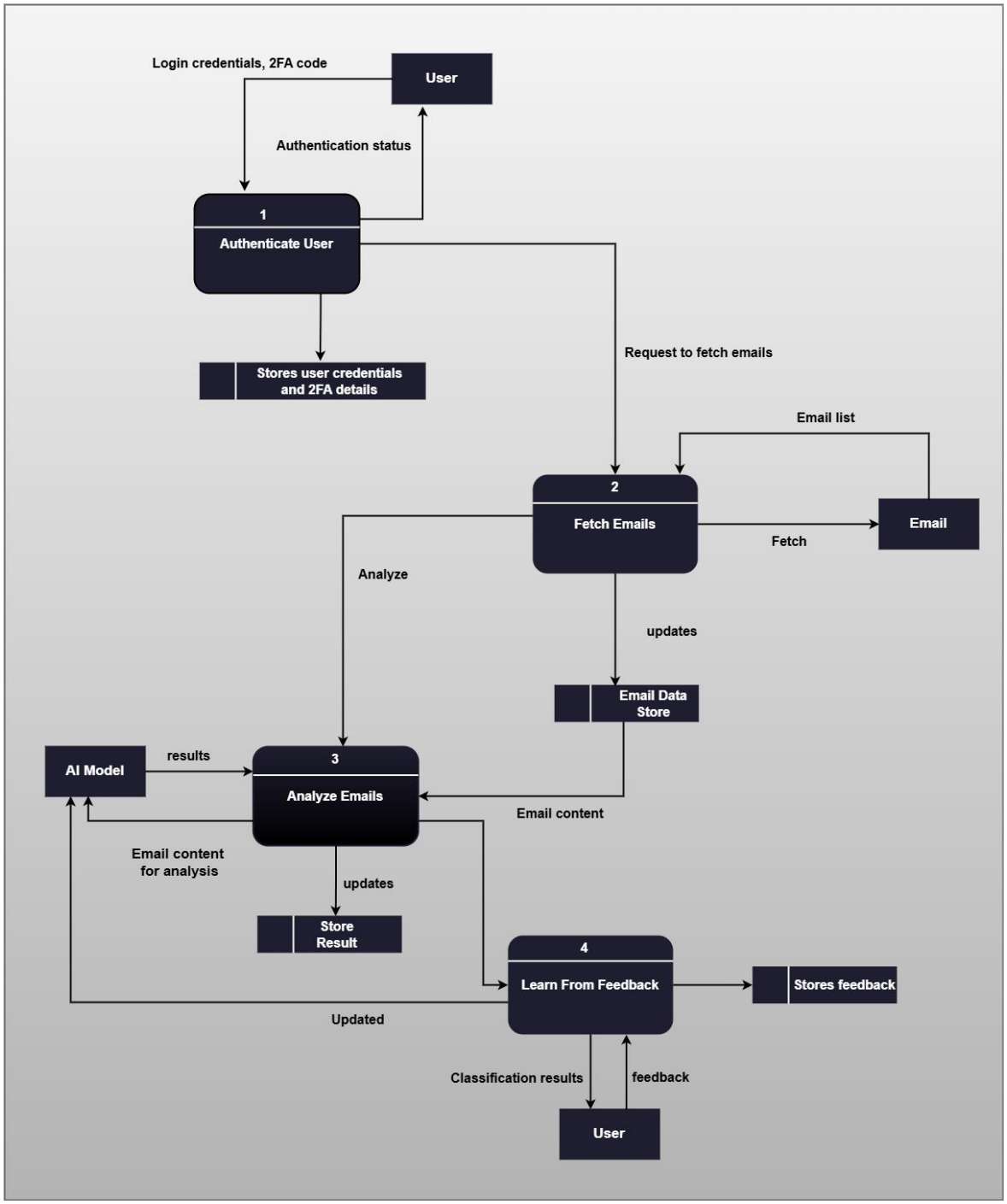


Fig 12 Data Flow Diagram

3.6 DATA BASE DESIGN

The database is organized into multiple tables, each containing rows and columns. In a relational database, each row is uniquely identified by a primary key (PK), which may consist of a single column or a combination of columns. This ensures that each row is unique, and no two rows share the same primary key value. Additionally, some columns

act as foreign keys (FK), which create relationships between tables by referencing primary keys in other tables. Each column in a table has a specific purpose, defined by a unique name and data type. The design and structure of the tables, along with their primary and foreign keys, are outlined below.

3.6.1 ERD Diagram

Entity-relationship diagrams are used to clearly illustrate the relationships between objects or entities in a system.

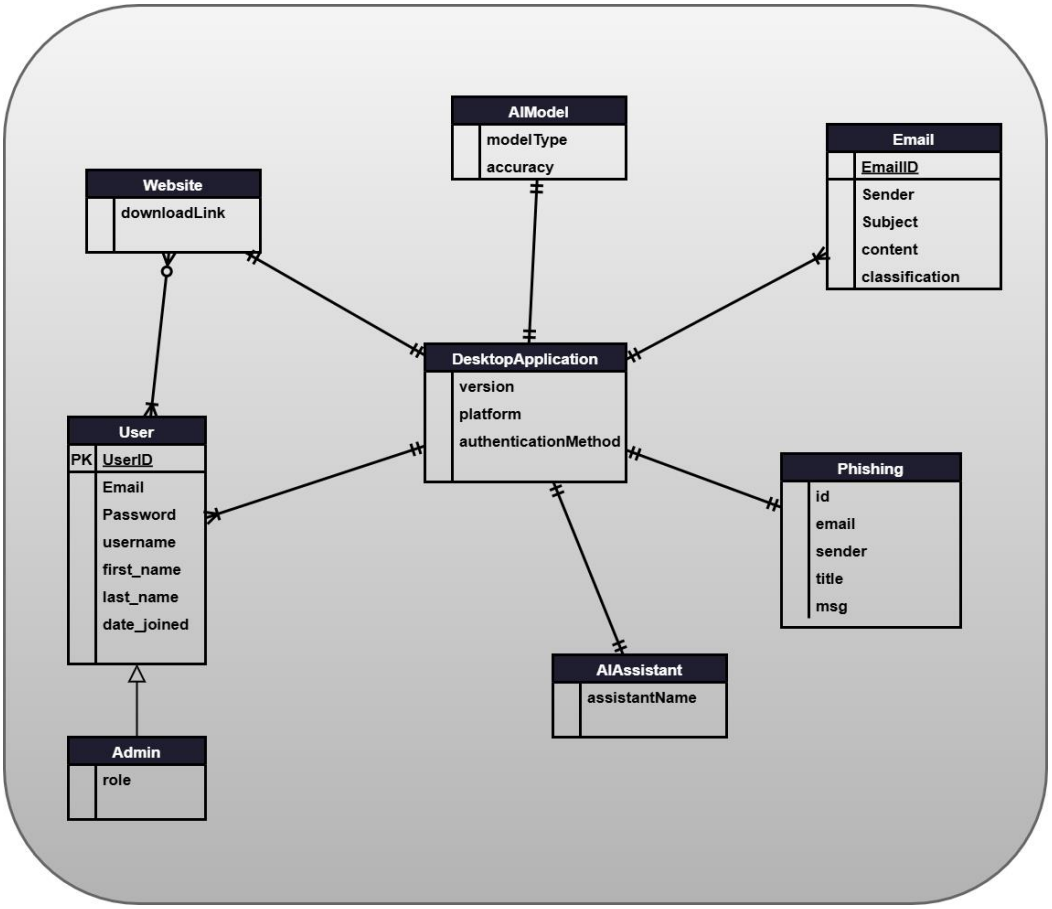


Fig 13 ERD Diagram

3.6.2 DATABASE TABLES

Table 4Chat

Column Name	Data Type	Description
id	BIGINT (Auto)	Unique identifier for each chat.
email	VARCHAR (255)	Email of the user.
sender	VARCHAR (255)	Sender of the message (default Bot).
msg	TEXT	Message content.

Table 5Phishing

Column Name	Data Type	Description
id	BIGINT (Auto)	Unique identifier for each phishing case.
id_message	VARCHAR(255)	Identifier for the phishing message.
email	VARCHAR(255)	Email of the sender.
title	VARCHAR(1000)	Title or subject of the message.
msg	TEXT	Body of the message.
sender	VARCHAR(255)	The name or email of the sender.
status	VARCHAR(255)	Status of the phishing message (e.g., flagged).

Table 6 Account

Column Name	Data Type	Description
id	BIGINT (Auto)	Unique identifier for each account.
email	VARCHAR(255)	User's email address (must be unique).
secret_key	VARCHAR(255)	Secret key used for authentication (TOTP).

Table 7 Question

Column Name	Data Type	Description
id	BIGINT (Auto)	Unique identifier for each question.
question	VARCHAR(1000)	The question text.
choice1	VARCHAR(1000)	First answer choice.
choice2	VARCHAR(1000)	Second answer choice.
choice3	VARCHAR(1000)	Third answer choice.
answer	VARCHAR(1000)	Correct answer to the question.

3.7 SAMPLE INTERFACE DESIGN

In this section we will show the Tarasd system interfaces

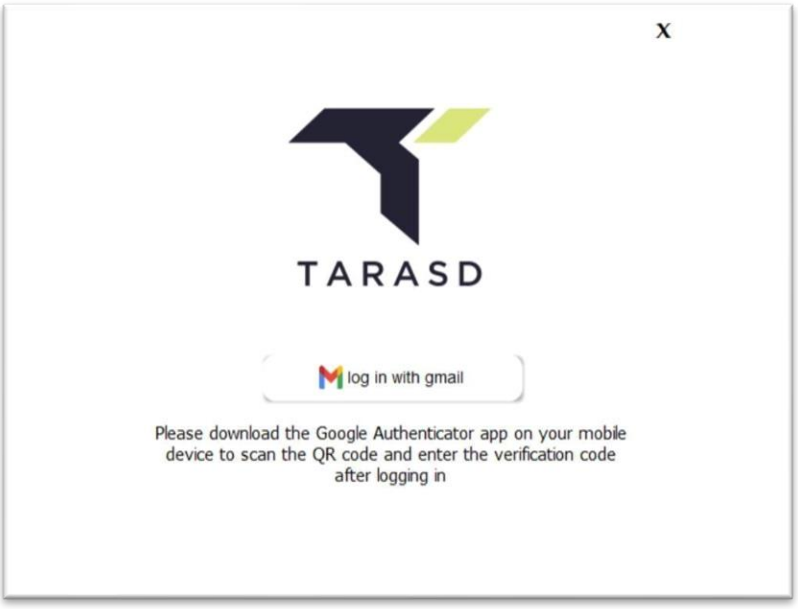


Fig 14 Tarasd Login

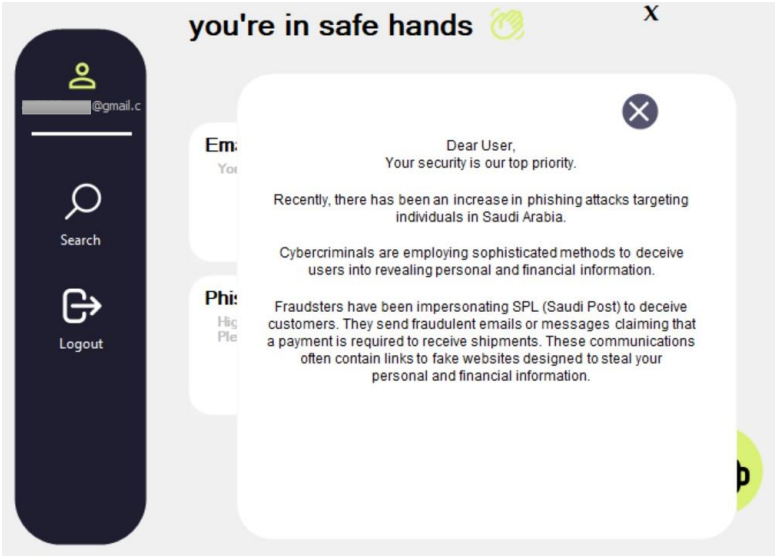


Fig 15Home page

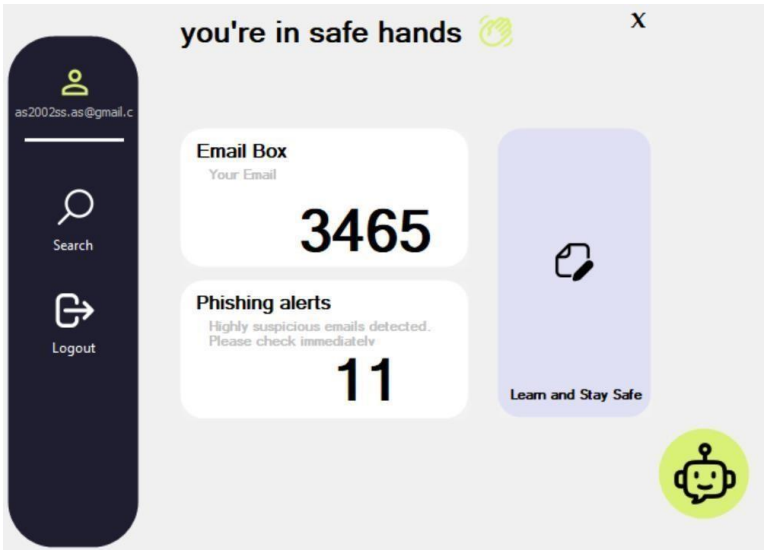


Fig 16 Main Page

Fig 17 Quiz page

3.8 UNDERLYING TECHNIQUES

Machine Learning (ML) Algorithms:

- The system uses advanced machine learning algorithms to build an AI model capable of detecting fraudulent emails by learning from historical data and continuously improving over time. We can use the following algorithms:

Support Vector Machine (SVM):

- It is highly effective in binary classification and processing high-dimensional data, with the ability to achieve ideal accuracy by identifying optimal boundaries for classification, but it requires intensive computations and is sensitive to noise and parameter tuning.

Multinomial Naive Bayes (MultinomialNB):

- A fast and simple algorithm ideal for text classification like spam detection. It performs well on highdimensional data but assumes feature independence, which can limit its accuracy in complex scenarios.

- Extreme Gradient Boosting (XGBoost - XGB)

  - A high-performance ensemble method that builds multiple trees to improve classification accuracy. It handles complex patterns and overfitting well but is computationally intensive and requires careful tuning.

Table 8 Comparison of algorithms

Algorithm	Core Equation	Advantages	Disadvantages	Metrics
***Support Vector* Machine (SVM)**	$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i(w \cdot x_i + b))$	Effective for imbalanced classes - Handles high dimensional data Kernel trick enables non linear classification	Sensitive to parameter tuning Struggles with noisy datasets	Results for SVM: Confusion Matrix: [[2146 33] [39 1219]] Accuracy: 0.9791 Recall: 0.9690 Precision: 0.9736 F1 Score: 0.9713
XGBoost	$f(x) = \sum_{k=1}^{K} f_k(x), \quad f_k \in \mathcal{F}$	- High performance on structured data - Built-in regularization - Handles missing data	- Needs careful tuning - Computationally intensive	Results for XGBoost: Confusion Matrix: [[1202 56] [49 2130]] Accuracy: 0.9695 Recall: 0.9775 Precision: 0.9744 F1 Score: 0.9759

Multinomial Naive Bayes **(MultinomialNB)**	$P(y\|X) = \frac{P(X\|y)\cdot P(y)}{P(X)}$	- Lightweight and fast - Efficient on small datasets	Struggles with complex patterns Less effective on large datasets	Results for MultinomialNB: Confusion Matrix: [[1839 48] [128 2644]] Accuracy: 0.9622 Recall: 0.9538 Precision: 0.9822 F1 Score: 0.9678

Natural Language Processing (NLP)

- NLP forms the backbone of Tarasd's email analysis, enabling accurate phishing detection through:
- Term Frequency – Inverse Document Frequency (TF-IDF)

Gmail API Integration:

- Fetches email content in real-time for analysis using ML and NLP techniques. Emails are classified as either "safe" or "phishing."

Real-Time Notifications:

- Users are immediately alerted via desktop notifications about suspicious emails. Notifications offer actionable options:
- Flag as phishing OR KEEP IT

AI Assistant:

Task-Oriented Conversational Agent:

- Designed to guide users, answer system-related queries, and direct them to educational resources.

RAG-Based Intelligence (Retrieval-Augmented Generation):

- The assistant leverages RAG, combining document retrieval with generative models to provide accurate, context-based responses.

Powered by Microsoft all-MiniLM-L6-v2:

- Built on a lightweight, transformer-based model pre-trained for conversational tasks, ensuring fast and contextaware dialogue generation.

Educational Content:

- Quizzes:

- Interactive phishing knowledge assessments to enhance user awareness and learning.

- Agile Development Methodology:
    - The system is developed using the Agile methodology to ensure:
    - Continuous updates based on user feedback
    - Seamless integration of advanced features
    - Adaptability to new cybersecurity threats

Programming Tools and Infrastructure:

Languages and Frameworks:

- Python: Core language for ML and NLP components
- Visual Studio: Used for developing the desktop application

Development Tools:

- Google Collaboratory: For model training and experimentation in the cloud
- SQLite: Lightweight database for storing user sessions, preferences, and detection outcomes

3.9 SUMMARY

Chapter 3 covers system analysis and system design, detailing key methodologies and tools for system design. Interaction Analysis uses UML diagrams (Use Case, Class, Sequence, Activity) to map system functionality and workflows. The design concept section defines guiding principles, while context and data flow diagrams illustrate data interactions and movement. Database design includes an ER diagram and database tables to show entity relationships and structure. Sample interface design showcases interface mockups, and underlying techniques detail algorithms and methods applied in the system.

CHAPTER 4 SYSTEM IMPLEMENTATION

4.1 INTRODUCTION

This chapter outlines the practical realization of the Tarasd system, focusing on how theoretical designs and planned functionalities were brought to life through development and integration. It highlights the steps taken to build the desktop application and accompanying website, implement the user interface, and connect the system with artificial intelligence models for real-time phishing detection. Additionally, this phase includes database setup, authentication mechanisms, and the deployment of interactive educational features aimed at enhancing cybersecurity awareness.

4.2 INTERFACE DESIGN AND CODING

Tarasd website home page

This is the landing interface of the Tarasd system's official website. It provides visitors with an overview of the project’s vision, mission, and features. From here, users can access general information, explore cybersecurity awareness content, and download the desktop application.

Fig 18 Home page website

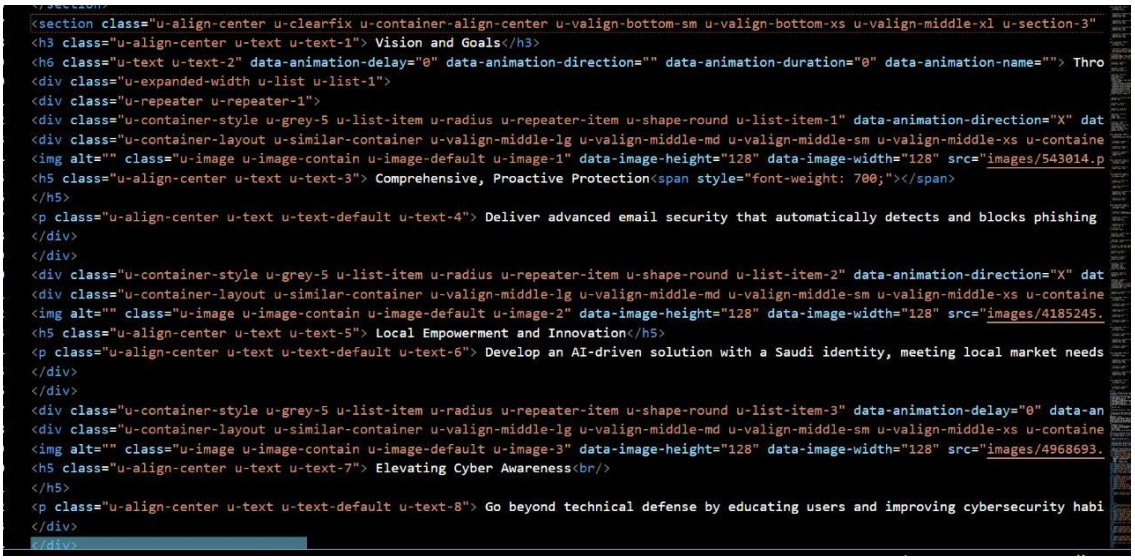


Fig 19 code Home page website

Download link in website

The download link is implemented to directs users to the installer file for the Windows-compatible Tarasd desktop application.

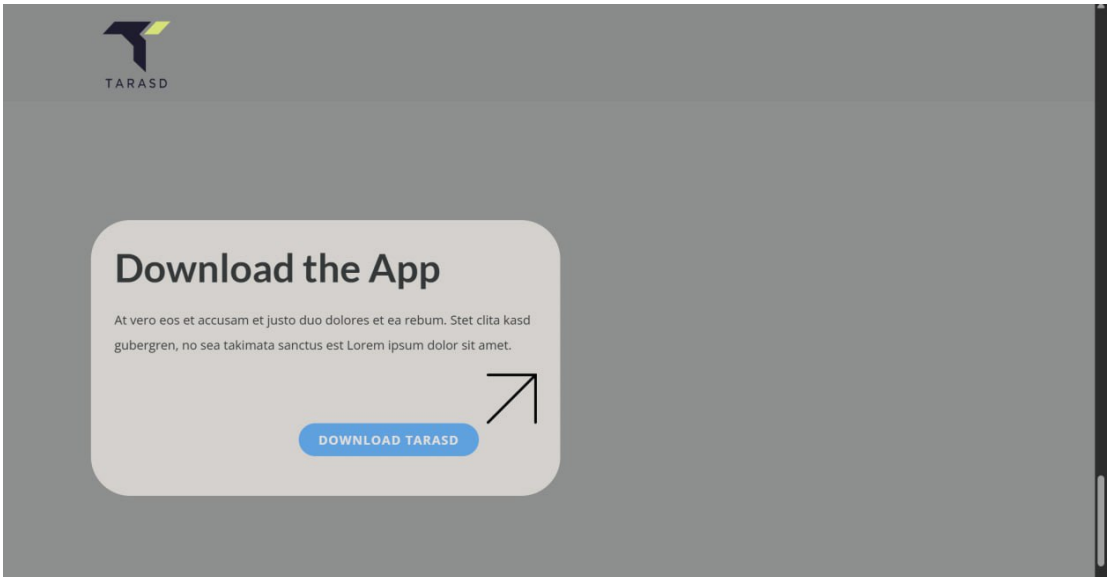


Fig 20 Download link in website

```
</div>
</section>
<section class="skrollable u-clearfix u-image u-parallax u-shading u-section-7" data-image-height="1736" data-image-width="2700" id="sec-
<div class="u-clearfix u-sheet u-sheet-1">
<div class="u-container-style u-gradient u-group u-radius u-shape-round u-group-1" data-animation-delay="0" data-animation-duration="2000
<div class="u-container-layout u-valign-top-lg u-valign-top-md u-valign-top-sm u-valign-top-xs u-container-layout-1">
<h3 class="u-text u-text-default u-text-2"> Download the App</h3>
<p class="u-text u-text-default u-text-3"> At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea tak
<a class="u-border-none u-btn u-button-style u-hover-grey-30 u-palette-3-base u-btn-1" href="https://drive.google.com/drive/folders/1B4RG
</div>
</div>
</div>
```

Fig 21code Download link in website

login page of the Tarasd desktop application

The login page of the Tarasd desktop application is developed using C# providing a smooth and secure user interface for authentication.

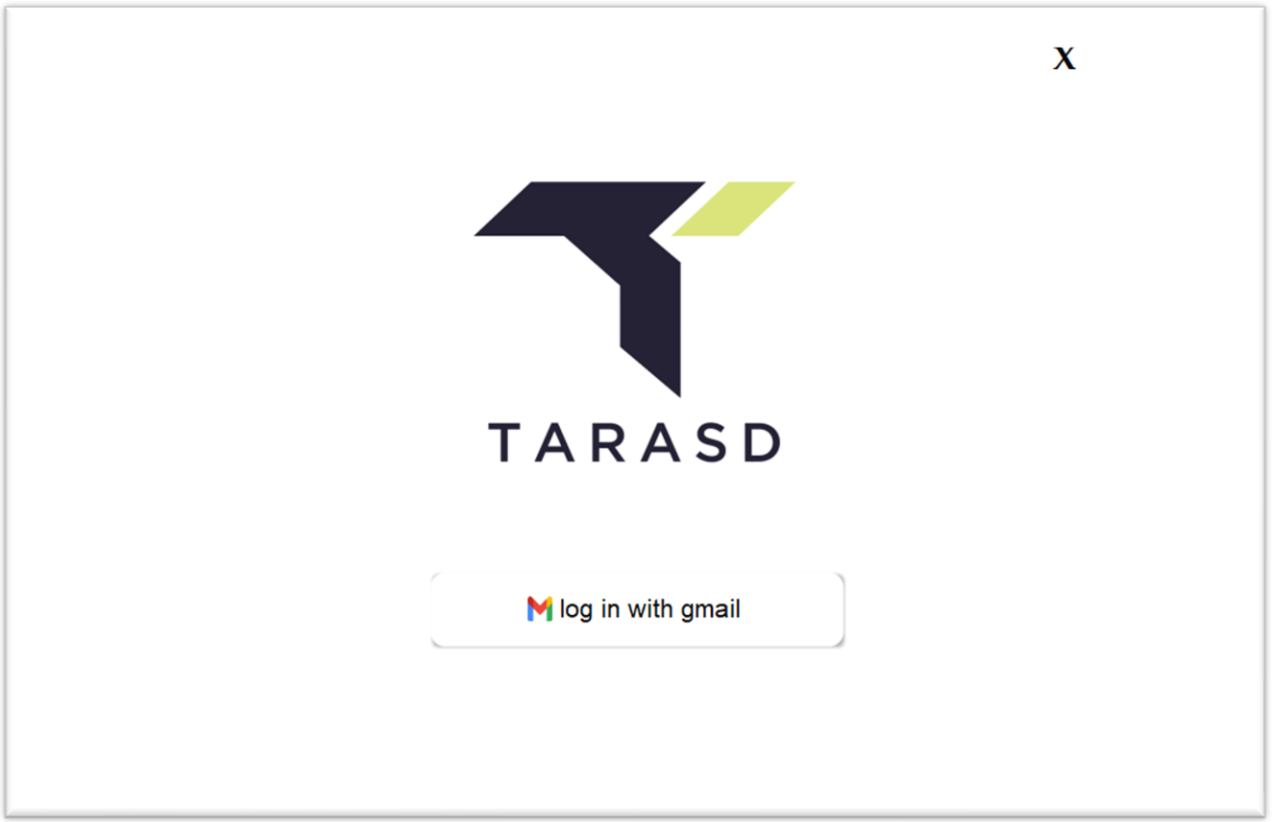


Fig 22 login page of the Tarasd desktop application

```
public static GmailService AuthenticateGmail()
{
    UserCredential credential = null;
    if (File.Exists(tokenPath))
    {
        Properties.Settings.Default.isLogged = true;
        Properties.Settings.Default.Save();
        try
        {
            credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
                new ClientSecrets { ClientId = "your_client_id", ClientSecret = "your_client_secr
                Scopes,
                "user",
                CancellationToken.None,
                new FileDataStore(tokenPath, true)).Result;
        }
        catch (Exception ex)
        {
            Properties.Settings.Default.isLogged = false;
            Properties.Settings.Default.Save();
            MessageBox.Show(ex.Message.ToString());
        }
    }
    else
    {

        using (var stream = new FileStream("credentials.json", FileMode.Open, FileAccess.Read))
        {
            credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
                GoogleClientSecrets.Load(stream).Secrets,
                Scopes,
                "user",
                CancellationToken.None,
```

Fig 23 login page of the Tarasd desktop application

Verification Page

The verification page is triggered after the user logs in with their Gmail account. It is built using C# in the desktop application and is responsible for handling the Two-Factor Authentication (2FA) process

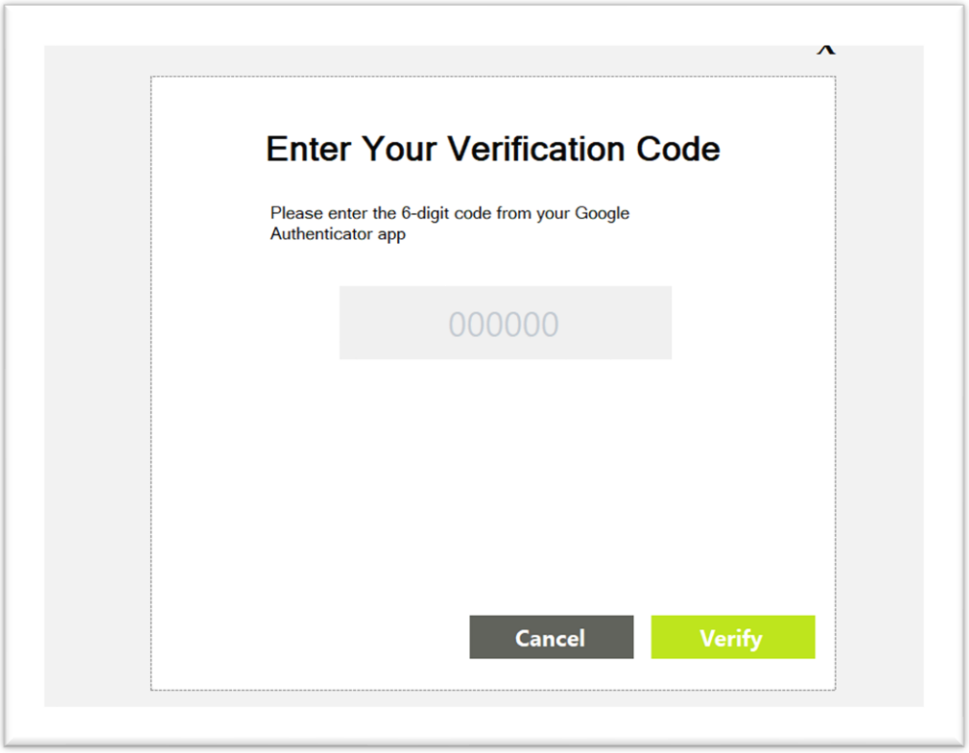


Fig 24 Verification Page

```
void Verify()
{
    //check otp verification
    byte[] secretKey = Base32Encoding.ToBytes(GlobalData.SecretKey.Replace("=", ""));
    var totp = new Totp(secretKey);
    bool isValid = totp.VerifyTotp(txtOTP.Text, out long timeStepMatched, new VerificationWindow(previous
    if (isValid)
    {
        this.Hide();
        if (form1 == null || form1.IsDisposed)
        {
            form1 = new Form1();
        }
        form1.Show();
    }
    else
    {
        MessageBox.Show("❌ Invalid OTP.");
    }
}
```

Fig 25 code Verification Page

Tarasd Home Page

The home page of the Tarasd desktop application is serves as the main interface after successful login. It displays a cybersecurity awareness advertisement and includes all main system component.

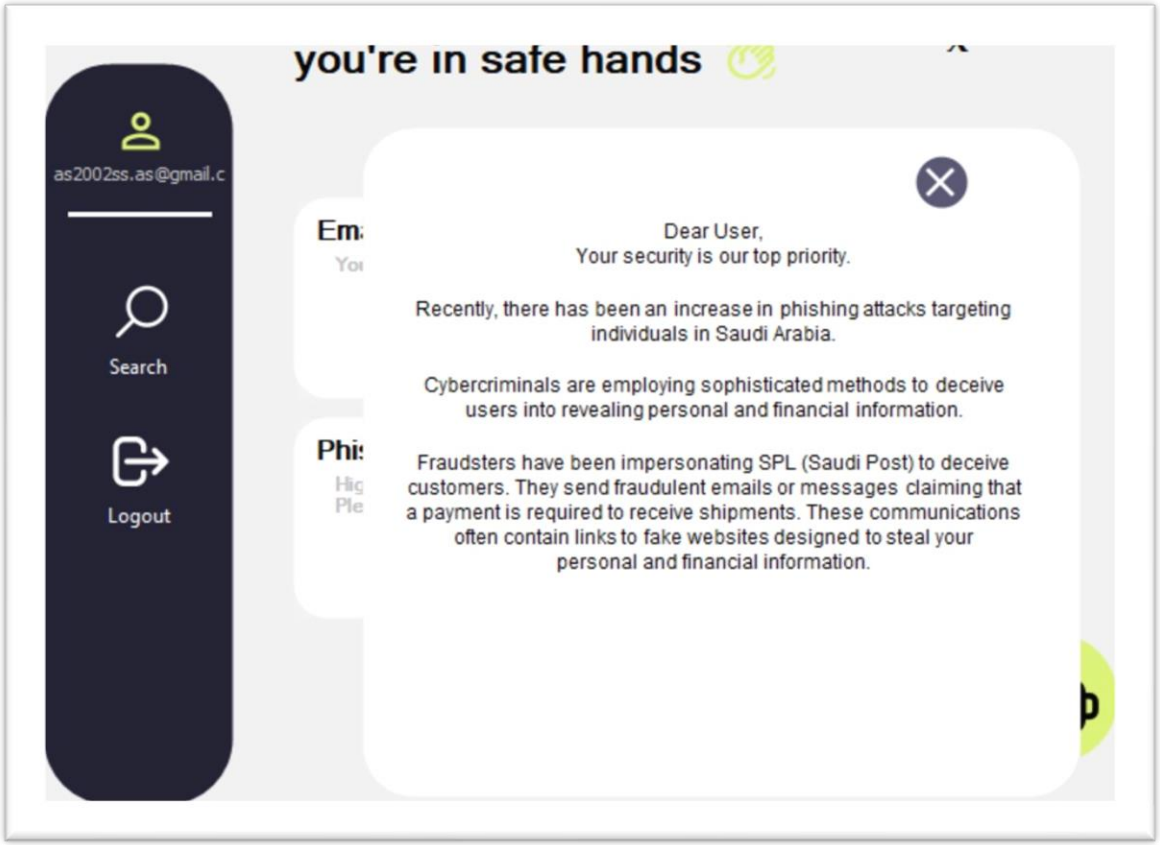


Fig 26 Tarasd Home Page

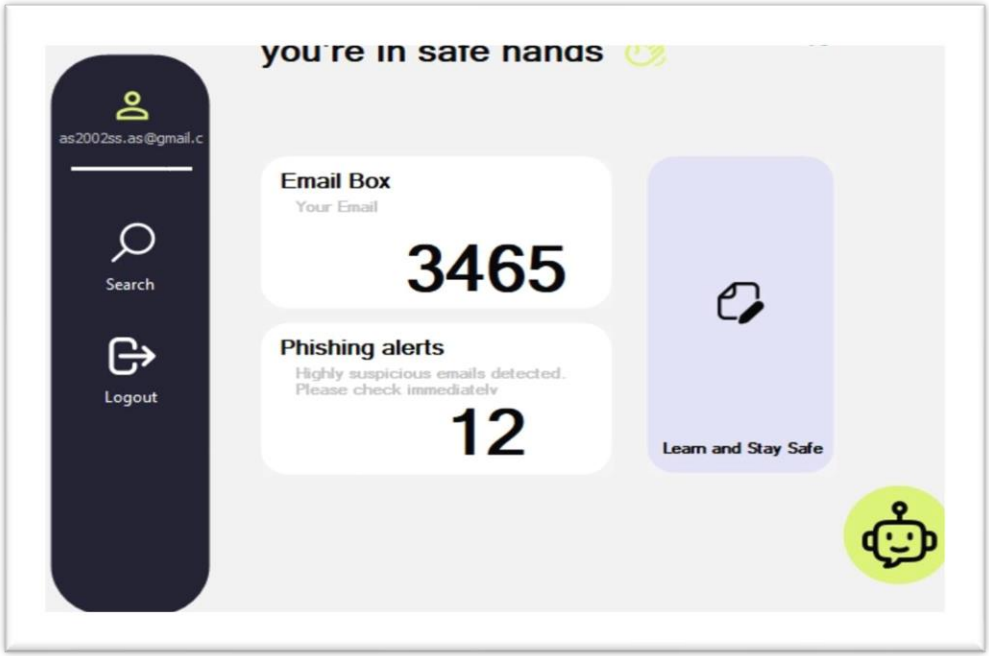


Fig 27 Tarasd Home Page2

```
private static string DecodeBase64String(string encodedData)
{
    if (string.IsNullOrEmpty(encodedData)) return string.Empty;

    // Gmail API uses Base64Url encoding
    var base64 = encodedData.Replace('-', '+').Replace('_', '/');
    switch (base64.Length % 4)
    {
        case 2: base64 += "=="; break;
        case 3: base64 += "="; break;
    }

    var bytes = Convert.FromBase64String(base64);
    return Encoding.UTF8.GetString(bytes);
}


1 reference
private void Form1_Load(object sender, EventArgs e)
{
    this.Width = 680;
    this.Height = 500;
    bool msgStatus = Properties.Settings.Default.msgStatus;

    if (msgStatus)
    {
        panelMsg.Visible = false;
    }
    else
    {
        panelMsg.Visible = true;
    }

    if (panelMsg.Visible == true)
    {
        lblMsg.Width = 400;
        panelMsg.Location = new Point(209, 74);
    }
```

Fig 28 code Tarasd Home Page

Phishing alerts box

The Phishing Alerts Box inside the Tarasd desktop application displays real-time notifications about detected phishing emails. It helps users stay aware of potential threats and enhances their email security and allow them to flag it or keep it

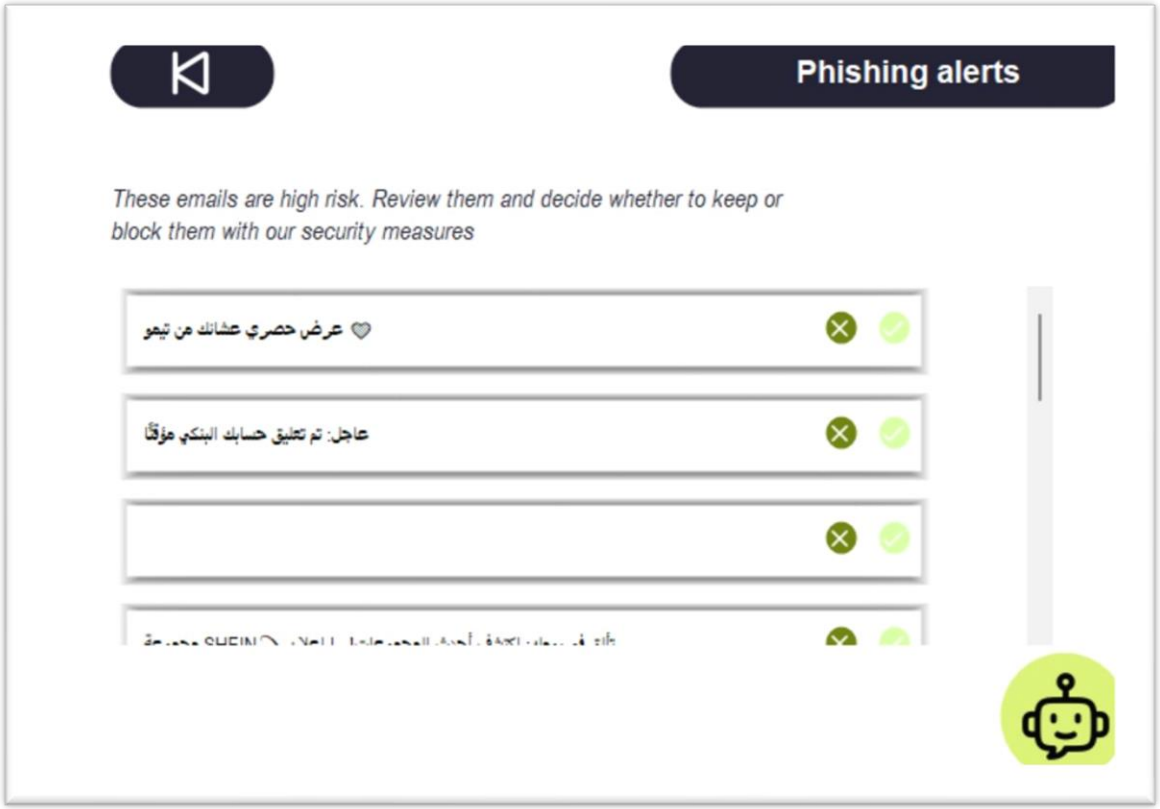


Fig 29 Phishing alerts box

```
private void PhishingForm_Load(object sender, EventArgs e)
{
    this.Width = 680;
    this.Height = 500;
    yOffset = 0;
    var phishingData = GlobalFunctions.phishingMessages.Reverse().ToList();
    foreach (var kvp in phishingData)
    {
        var pk = kvp.Key;
        var value = kvp.Value;
        AlertsBox item = new AlertsBox
        {
            IdMessage = pk,
            TextContent = value[0],
            BodyContent = value[1],
        };
        item.CloseClicked += (s, args) =>
        {
            if (panelBoxes.Controls.Contains(item))
            {
                panelBoxes.Controls.Remove(item);
                item.Dispose();
            }
        };
        item.CheckClicked += (s, args) => MessageBox.Show($"Checked: {item.TextContent}");
        item.Location = new Point(5, yOffset);
        yOffset += item.Height + 10;
        panelBoxes.Controls.Add(item);
    }
    HistoryChats();

}
```

Fig 30 code Phishing alerts box

AI assistant chatbot

The AI Assistant Chatbot is integrated into the Tarasd desktop application to help users navigate the system, answer questions, and provide guidance on using all app features efficiently.

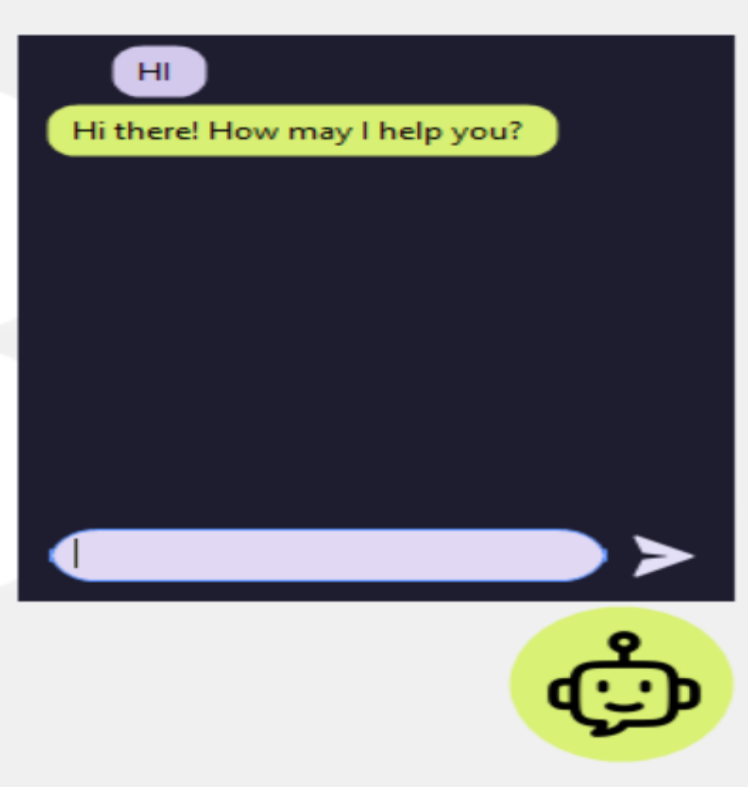


Fig 31 AI assistant chatbot

```
3 references
public static string ChatBot(string msg)
{
    byte[] Bytes = new System.Text.UTF8Encoding().GetBytes($"msg={msg}&email={GlobalData.UserEmail}&access_token={GlobalData.AccessToken}");
    HttpWebRequest HttpWebRequest = (HttpWebRequest)WebRequest.Create("https://mb-cheats-released-outsourcing.trycloudflare.com/chat/");
    HttpWebRequest.Method = "POST";
    HttpWebRequest.ContentType = "application/x-www-form-urlencoded";
    HttpWebRequest.ContentLength = Bytes.Length;
    System.IO.Stream Stream = HttpWebRequest.GetRequestStream();
    Stream.Write(Bytes, 0, Bytes.Length);
    Stream.Dispose();
    Stream.Close();
    HttpWebResponse Response;
    try
    {
        Response = (HttpWebResponse)HttpWebRequest.GetResponse();
    }
    catch (WebException ex)
    {
        Response = (HttpWebResponse)ex.Response;
    }
    System.IO.StreamReader StreamReader = new System.IO.StreamReader(Response.GetResponseStream());
    string resp = StreamReader.ReadToEnd().ToString();
    StreamReader.Dispose();
    StreamReader.Close();
    return resp;
}
1 reference
public static string RegisterUser(string secret_key)
{
    byte[] Bytes = new System.Text.UTF8Encoding().GetBytes($"email={GlobalData.UserEmail}&secret_key={secret_key}");
    HttpWebRequest HttpWebRequest = (HttpWebRequest)WebRequest.Create("https://mb-cheats-released-outsourcing.trycloudflare.com/register/");
    HttpWebRequest.Method = "POST";
    HttpWebRequest.ContentType = "application/x-www-form-urlencoded";
    HttpWebRequest.ContentLength = Bytes.Length;
    System.IO.Stream Stream = HttpWebRequest.GetRequestStream();
    Stream.Write(Bytes, 0, Bytes.Length);
    Stream.Dispose();
    Stream.Close();
    HttpWebResponse Response;
    try
```

Fig 32 code AI assistant chatbot

Cybersecurity Awareness Quiz

The Quiz feature in the Tarasd desktop application tests the user's cybersecurity awareness through interactive questions, helping reinforce safe online practices.

Fig 33Cybersecurity Awareness Quiz1

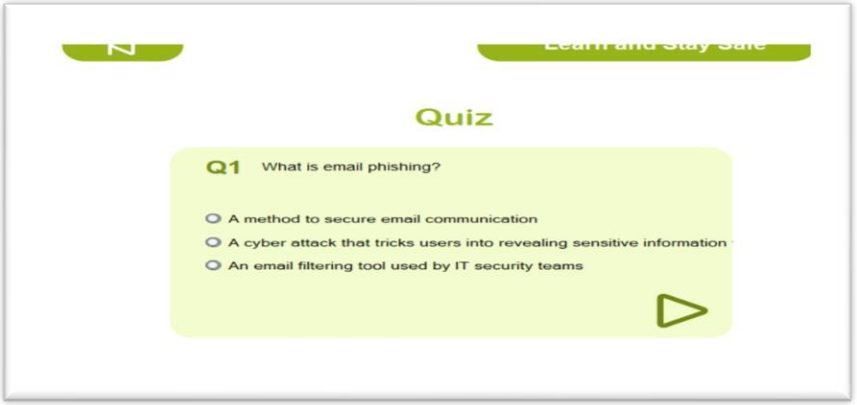


Fig 34 Cybersecurity Awareness Quiz 2

```
private void btnNext_Click(object sender, EventArgs e)
{
    if(currentQuestion == 5)
    {
        panelHome.Visible = false;
        panelQuizs.Visible = false;
        panelResult.Visible = true;
        panelResult.Location = new Point(34, 68);
    }
    else
    {
        if (radioFirstAnswer.Checked)
        {
            answered = radioFirstAnswer.Text;

        }
        else if (radioSecondAnswer.Checked)
        {
            answered = radioSecondAnswer.Text;

        }
        else if (radioThirdAnswer.Checked)
        {
            answered = radioThirdAnswer.Text;
        }
        else
        {
            answered = "";
        }
        if(answered == randomQuestions[currentQuestion].answer)
        {
            score += 2;
            lblScore.Text = $"{score.ToString()}\\10";
        }

        currentQuestion += 1;
        if(currentQuestion == 5)
        {
            panelHome.Visible = false;
            panelQuizs.Visible = false;
            panelResult.Visible = true;
            panelResult.Location = new Point(34, 68);
        }
        else
        {
            SetQuestions();
        }
```

Fig 35code Cybersecurity Awareness Quiz

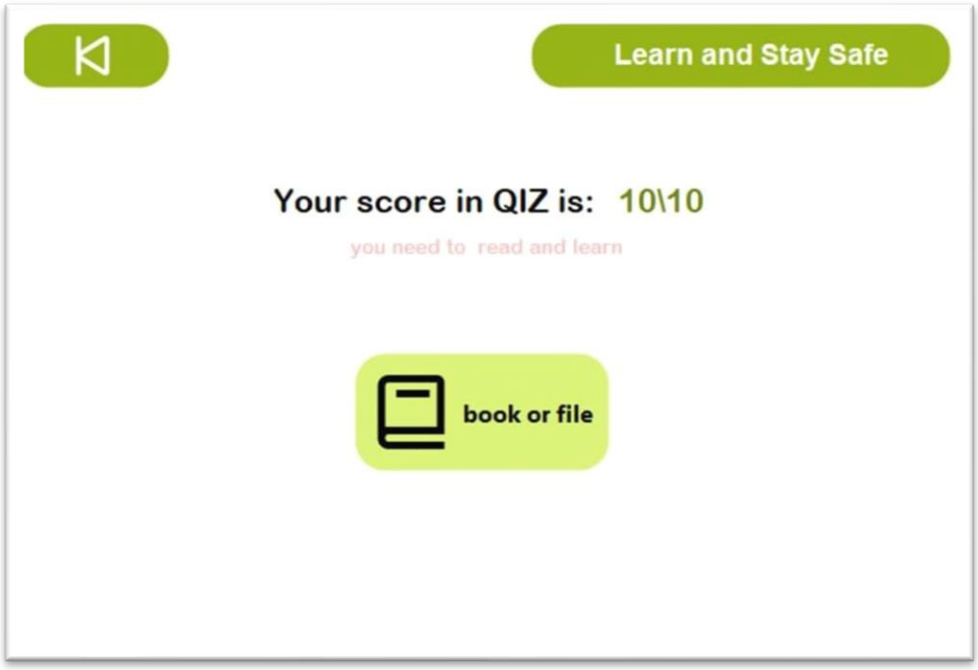
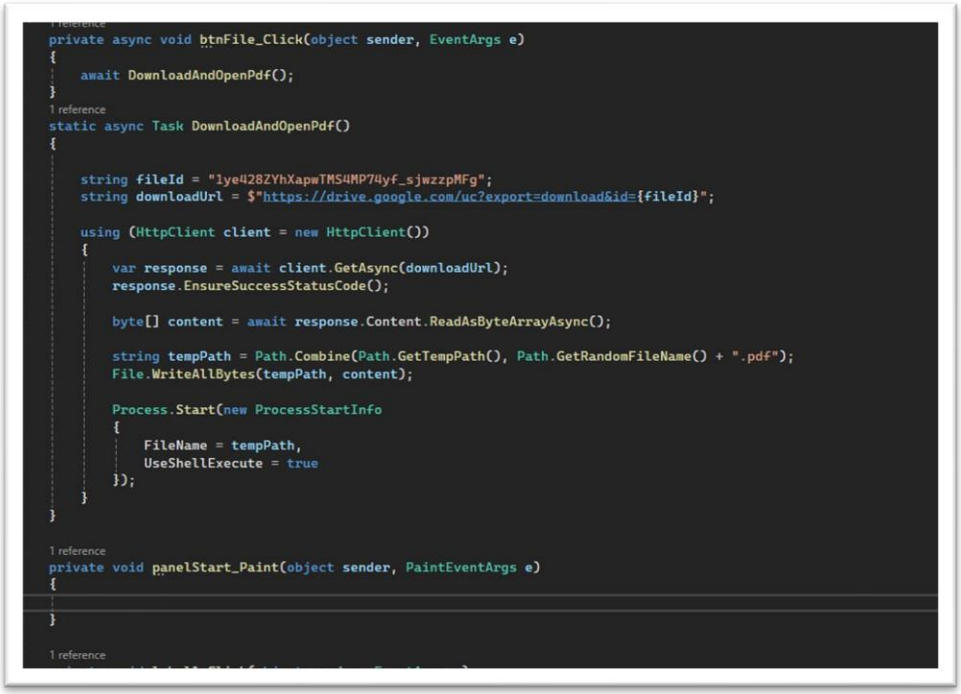


Fig 36 Result Quiz

Educational content

The Educational Content section provides users with informative materials on phishing threats and cybersecurity best practices to enhance their knowledge and awareness.

Educational content Fig 37

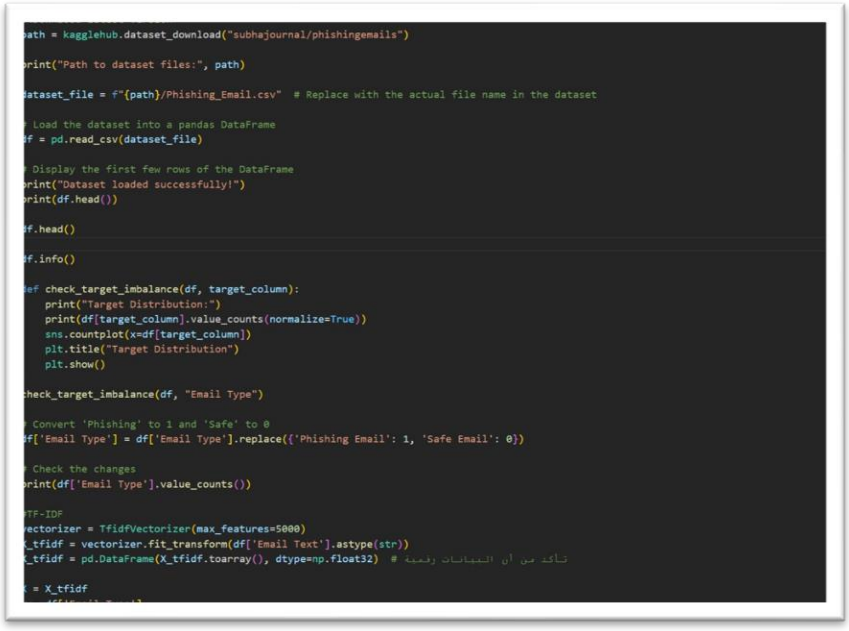


```
private async void btnFile_Click(object sender, EventArgs e)
{
    await DownloadAndOpenPdf();
}
1 reference
static async Task DownloadAndOpenPdf()
{

    string fileId = "1ye428ZYhXapwTMS4MP74yf_sjwzzpMFg";
    string downloadUrl = $"https://drive.google.com/uc?export=download&id={fileId}";

    using (HttpClient client = new HttpClient())
    {
        var response = await client.GetAsync(downloadUrl);
        response.EnsureSuccessStatusCode();

        byte[] content = await response.Content.ReadAsByteArrayAsync();

        string tempPath = Path.Combine(Path.GetTempPath(), Path.GetRandomFileName() + ".pdf");
        File.WriteAllBytes(tempPath, content);

        Process.Start(new ProcessStartInfo
        {
            FileName = tempPath,
            UseShellExecute = true
        });
    }
}

1 reference
private void panelStart_Paint(object sender, PaintEventArgs e)
{

}

1 reference
```

Fig 38 code Educational content

SVM Model Algorithm

The SVM (Support Vector Machine) model algorithm is used in the Tarasd application to accurately detect phishing emails by analyzing patterns and classifying messages based on their content.



```
ath = kagglehub.dataset_download("subhajournal/phishingemails")

rint("Path to dataset files:", path)

ataset_file = f"{path}/Phishing_Email.csv"  # Replace with the actual file name in the dataset

# Load the dataset into a pandas DataFrame
f = pd.read_csv(dataset_file)

# Display the first few rows of the DataFrame
rint("Dataset loaded successfully!")
rint(df.head())

f.head()

f.info()

ef check_target_imbalance(df, target_column):
    print("Target Distribution:")
    print(df[target_column].value_counts(normalize=True))
    sns.countplot(x=df[target_column])
    plt.title("Target Distribution")
    plt.show()

heck_target_imbalance(df, "Email Type")

# Convert 'Phishing' to 1 and 'Safe' to 0
f['Email Type'] = df['Email Type'].replace({'Phishing Email': 1, 'Safe Email': 0})

# Check the changes
rint(df['Email Type'].value_counts())

TF-IDF
ectorizer = TfidfVectorizer(max_features=5000)
_tfidf = vectorizer.fit_transform(df['Email Text'].astype(str))
_tfidf = pd.DataFrame(X_tfidf.toarray(), dtype=np.float32)  # تأكد من أن البيانات رقمية

 = X_tfidf
```

Fig 39 SVM Model Algorithm1

```python
#oversampling
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X, y)

X_resampled = pd.DataFrame(X_resampled, dtype=np.float32)
balanced_df = pd.concat([X_resampled, pd.DataFrame(y_resampled, columns=['Email Type'])], axis=1)
print(balanced_df['Email Type'].value_counts())

check_target_imbalance(balanced_df, "Email Type")

balanced_df.dropna(inplace=True) #CLEEN

balanced_df.drop_duplicates(inplace=True)

X = balanced_df.drop('Email Type', axis=1)
y = balanced_df['Email Type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

from sklearn.linear_model import SGDClassifier

svm_model = SGDClassifier(loss="hinge")  # محاكاة SVM خطي
svm_model.fit(X_train, y_train)

predictions_svm = svm_model.predict(X_test)

models = {'SVM': predictions_svm}

for model_name, predictions in models.items():
    cm = confusion_matrix(y_test, predictions)
    accuracy = accuracy_score(y_test, predictions)
    recall = recall_score(y_test, predictions, pos_label=1)
    precision = precision_score(y_test, predictions, pos_label=1)
    f1 = f1_score(y_test, predictions, pos_label=1)

    print(f"\nResults for {model_name}:")
    print("Confusion Matrix:\n", cm)
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"F1 Score: {f1:.4f}")
```

Fig 40 SVM Model Algorithm 2

```python
import pandas as pd
from sentence_transformers import SentenceTransformer
import faiss
import numpy as np
import pickle

file_path = "/content/Tarasd_FAQ_Expanded_Final.xlsx"
df = pd.read_excel(file_path)
model = SentenceTransformer("all-MiniLM-L6-v2")
texts = df['Question'] + " " + df['Answer']
embeddings = model.encode(texts.tolist(), convert_to_numpy=True)
index = faiss.IndexFlatL2(embeddings.shape[1])
index.add(embeddings)
faiss.write_index(index, "tarasad_faiss.index")
with open("tarasad_metadata.pkl", "wb") as f:
    pickle.dump(df.to_dict(), f)

print(" FAISS database has been created!")

def search_faq(query, top_k=3):
    query_embedding = model.encode([query], convert_to_numpy=True)
    distances, indices = index.search(query_embedding, top_k)

    results = []
    for idx in indices[0]:
        results.append(df.iloc[idx][['Question', 'Answer']].to_dict())

    return results


def search_query(user_input):
    question_list = df["Question"].astype(str).tolist()
    best_match, score = process.extractOne(user_input, question_list)

    if score > 86:
        return df[df["Question"] == best_match]["Answer"].values[0]
    else:
        return "I can't answer, I'm just here for help Tarasd"
```

Fig 41 AI Assistant RAG model

Database coding

```python
from django.db import models
You, 2 weeks ago | 1 author (You)
class Chat(models.Model):
    email = models.CharField(max_length=255,default='',blank=True)
    sender = models.CharField(max_length=255,default='Bot')
    msg = models.TextField(default='')
    def __str__(self):
        return self.email
You, 1 second ago | 1 author (You)
class Phishing(models.Model):
    id_message = models.CharField(max_length=255,default='')
    email = models.CharField(max_length=255,default='',blank=True)
    title = models.CharField(max_length=1000,default='')
    msg = models.TextField(default='')
    sender = models.CharField(max_length=255,default='')
    status = models.CharField(max_length=255,default='')
    def __str__(self):
        return self.title
You, 1 second ago | 1 author (You)
class Account(models.Model):
    email = models.CharField(max_length=255,unique=True)
    secret_key = models.CharField(max_length=255)

    def __str__(self):
        return self.email
You, 1 second ago | 1 author (You)
class Question(models.Model):
    question = models.CharField(max_length=1000)
    choice1 = models.CharField(max_length=1000)
    choice2 = models.CharField(max_length=1000)
    choice3 = models.CharField(max_length=1000)
    answer = models.CharField(max_length=1000)
    def __str__(self):
        return self.question
```

```python
class RegisterUser(APIView):
    def post(self, request):
        email = request.data.get('email')
        secret_key = request.data.get('secret_key')
        account, created = Account.objects.get_or_create(
            email=email,
            defaults={'secret_key': secret_key}
        )
        return Response({
            'account_id': account.id,
            'email': account.email,
            'secret_key': account.secret_key,
            'created': created  # True if new, False if it already existed
        })
```

```python
class AskChat(APIView):
    def post(self, request):
        query = request.data.get('msg')
        email = request.data.get('email')
        access_token = request.data.get('access_token')  # just in case you need to handle this later
        results = search_query(query, model_store.model, model_store.index, model_store.metadata)
        resp = ''
        for i, res in enumerate(results, 1):
            resp = res
        Chat.objects.create(
            email=email,
            sender=query,
            msg=resp
        )
        return Response({"message": resp})
```

```python
class GetChats(APIView):
    def post(self, request):
        email = request.data.get('email')
        access_token = request.data.get('access_token')  # Not used yet
        if not check_email(email,access_token):
            return Response({"error": "Not Authenticated"}, status=400)
        if not email:
            return Response({"error": "Email is required"}, status=400)

        chats = Chat.objects.filter(email=email).values('sender', 'msg')

        return Response({"chats": list(chats)})
```

```python
class StatusMessage(APIView):
    def post(self, request):
        id_message = request.data.get('id_message')
        status = request.data.get('status')

        message = Phishing.objects.get(id_message=id_message)
        message.status = status
        message.save()

        return Response({"status": "done"})
```

```python
class QuestionListView(APIView):
    You, last week | 1 author (You)
    class InlineQuestionSerializer(serializers.ModelSerializer):
        choices = serializers.SerializerMethodField()

        You, last week | 1 author (You)
        class Meta:
            model = Question
            fields = ['question', 'choices', 'answer']

        def get_choices(self, obj):
            return [obj.choice1, obj.choice2, obj.choice3]

    def get(self, request):
        questions = Question.objects.all()
        serializer = self.InlineQuestionSerializer(questions, many=True)
        return Response({"QUESTIONS": serializer.data})
```

4.5 SUMMARY

This chapter outlined the hands-on development and programming of the Tarasd system. It features a secure and intuitive interface supported by a strong database and AI-driven backend, enabling efficient phishing detection and boosting cybersecurity awareness for users.

CHAPTER 5 TESTING AND EVALUATION

5.1 INTRODUCTION

Testing plays a vital role in software development by verifying that the system fulfills its functional requirements and operates reliably across various conditions. This chapter presents the testing approaches used for the Tarasd phishing detection system and assesses the results to ensure the application's stability, precision, and deployment readiness.

5.2 TEST PLAN AND OBJECTIVES

The primary objective of testing is to ensure the system operates as intended and delivers the expected features.

The key goals of the testing plan include:

- Verifying that the system fulfills all functional and non-functional requirements.
- Ensuring users experience the anticipated performance and ease of use.
- Detecting and resolving bugs, logical errors, or inconsistencies prior to deployment.
- Verifying that the AI model generates accurate phishing predictions.

Testing also focuses on:

- User satisfaction with the interface and alert system.
- Accuracy of phishing detection results.
- Effective handling of invalid inputs and edge cases.

5.3 STAGES OF TESTING

Different types of testing should be applied to the system, each serving distinct purposes and employing various approaches. The two primary stages are white box testing, which focuses on the internal components of the system, and black box testing, which aims to evaluate the final product from an end-user perspective.

5.3.1 WHITE BOX TESTING

White Box Testing is a type of software testing that focuses on examining the internal components of a system. It involves analyzing the system's source code, internal structures, and mechanisms that control how the system operates

5.3.1.1 UNIT TESTING

Unit Testing is a software testing technique that involves testing individual components or units of a software system in isolation to ensure that they function as expected. A unit refers to the smallest testable part of the application

Example: The login component was tested using incorrect credentials to ensure the system displays an appropriate error message: "Invalid username or password."

In signup page, error message of missing data shows up when a user miss filling a field in the form. Each field is specified with different message, clarifying the restriction on the input data.

Email error Fig 42

In signup page, error message of missing data shows up when a user miss filling a field in the form. Each field is specified with different messages, clarifying the restriction on the input data.

Password error Fig 43

The following figure shows the error message that appears when the user inputs an incorrect code during the verification step.

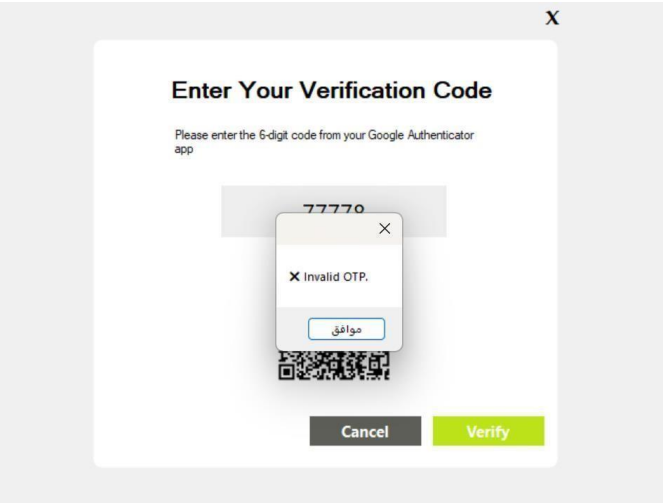


Fig 44 Invalid OTP Message

The following figure shows the chatbot's response when the user asks a question that is outside the scope of the Tarasd system's functionalities.

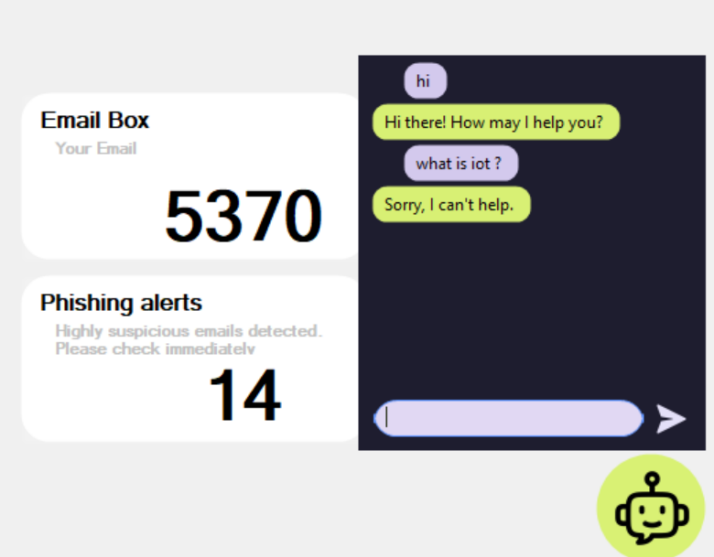


Fig 45error chatbot

5.3.2 BLACK BOX TESTING

Black box testing focused on the system's behavior from the end user's perspective, without examining internal code. Inputs such as login attempts, phishing email samples, and invalid verification codes were tested.

5.3.2.1 SYSTEM TESTING

System testing ensured that the full application worked as intended. This includes interaction between modules:

- Login → Verification → Email scan → Alert
- User interface flows across the website and desktop app
- Chatbot responses based on AI knowledge base

5.4 TEST CASES AND TEST RESULTS

5.4.1 TEST CASE: PHISHING ALERT TRIGGERED ON NEW SUSPICIOUS

Table 9Phishing Alert Triggered on New Suspicious Emai

Test Case No	Description	Input	Expected Output	Actual Output	Passed
1	The system should trigger an alert when a new phishing email is received	A phishing email is sent to the connected Gmail inbox	The Phishing alerts count increases	The alert count increased and a sound notification is played	Yes

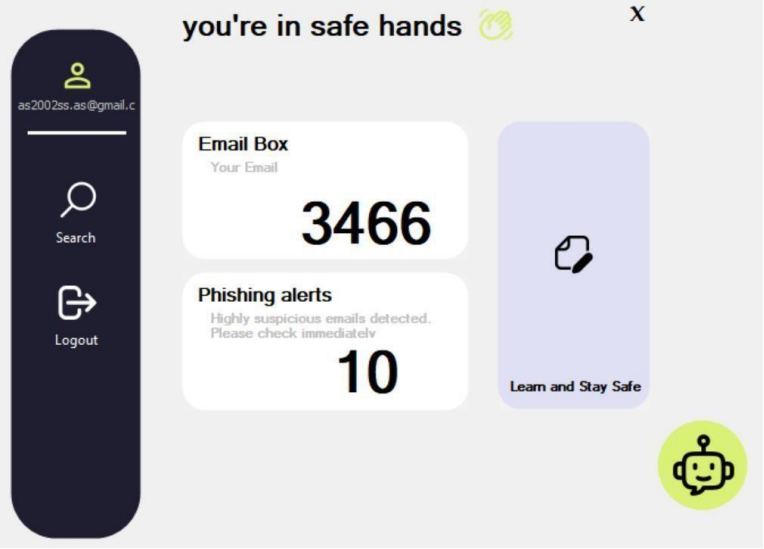


Fig 46Dashboard before new phishing alert

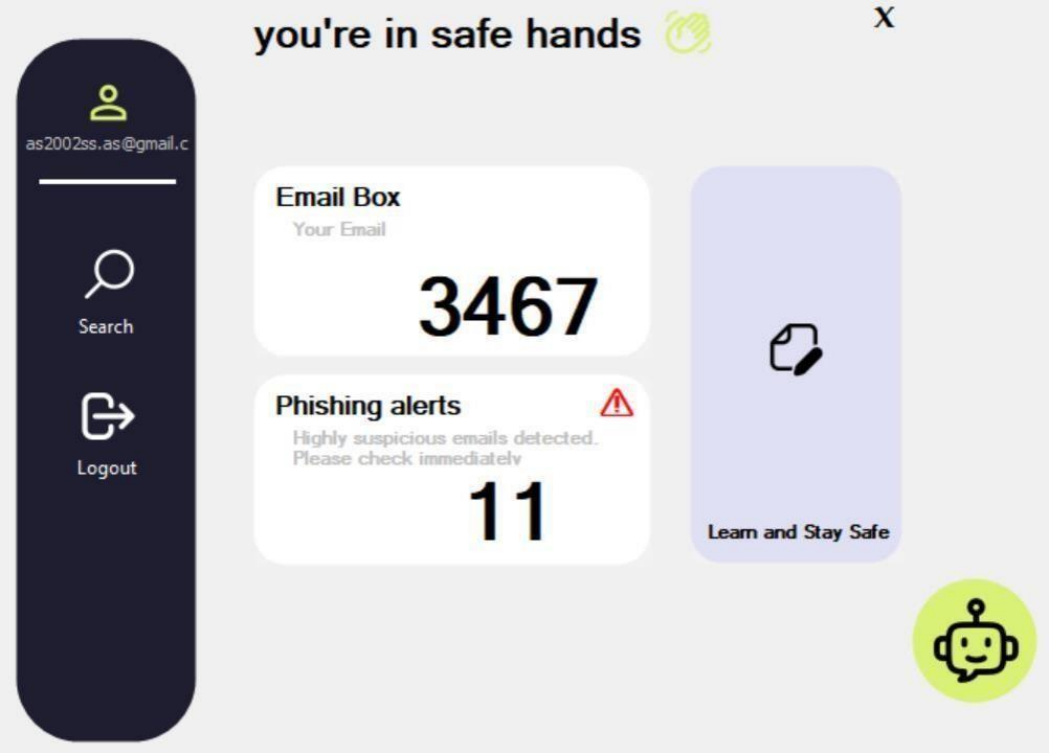


Fig 47 Dashboard after phishing alert increase

5.4.2 TEST CASE: SEARCH FOR EMAILS BY SENDER KEYWORD

Table 10Search for Emails by Sender Keyword

Test Case No	Description	Input	Expected Output	Actual Output	Passed
2	The user should be able to search all emails using a keyword to filter by sender or subject	User types * ********** in the search bar	System displays all emails where * ********** appears in the s ender's email or subject	Results appear in a scrollable list All matching emails were displayed correctly Sender and subject lines matched	Yes

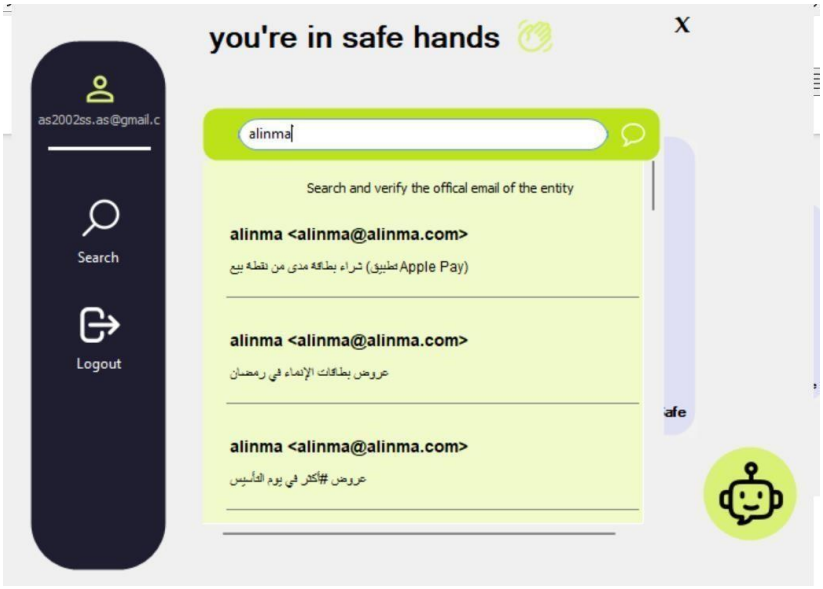


Fig 48Email search

5.4.3 TEST CASE: CHATBOT RESPONDS BASED ON TRAINED

Table 11 Chatbot Responds Based on Trained Dataset

Test Case No	Description	Input	Expected Output	Actual Output	Passed
3	The chatbot should respond accurately to user queries using the internal trained dataset	User enters questions like: • "What does the Tarasd system do?" • "How does Tarasd detect phishing emails?"	Chatbot provides responses that match information in the dataset	Answers are clear, structured, and context - a ware	Yes

Chatbot response Fig 49

5.5 TEST RESULTS

Overall, the testing process confirmed that the Tarasd system performs reliably and meets its requirements. All critical features — including login, Gmail authentication, 2FA, phishing detection, and alert generation — passed testing scenarios. Minor styling adjustments were made to improve user experience.

5.6 SUMMARY

This chapter highlighted the various testing techniques applied to the Tarasd system. Both white-box and black-box testing approaches were used to validate the software's reliability, security, and usability. Results confirmed the system's readiness for deployment, achieving the desired goals of accuracy, efficiency, and ease of use.

CHAPTER 6 CONCLUSION AND FUTURE WORKS

6.1 CONCLUSION

This project successfully introduced Tarasd, an AI-powered system designed to address a critical challenge in today's digital landscape—phishing emails and the broader scope of cybersecurity threats. By leveraging advanced machine learning and natural language processing (NLP), the system automates the process of detecting and classifying phishing emails, effectively reducing the reliance on users to manually assess potential threats. This functionality is especially valuable for small and medium-sized enterprises (SMEs), which often lack the resources for comprehensive cybersecurity infrastructure, enabling them to mitigate the risk of falling victim to phishing attacks.

The Tarasd system was developed using a diverse set of technologies, including C# for the desktop application, Python for backend processing and AI integration, and HTML, CSS, and JavaScript for the web interface. This multi-technology approach facilitated the creation of a seamless, intuitive, and secure user experience across different platforms. The integration of core features, such as Gmail login and two-factor authentication (2FA), ensures a high level of security and helps protect users' accounts from unauthorized access. Additionally, the push notifications feature provides timely alerts, enabling users to respond quickly to potential threats. Moreover, the inclusion of an intelligent AI assistant not only enhances the user experience but also offers personalized assistance, further increasing the overall usability of the system.

One of the most notable aspects of Tarasd is its alignment with the strategic goals of Saudi Arabia's Vision 2030, particularly in its focus on enhancing cybersecurity infrastructure and promoting technological innovation. By equipping SMEs with an affordable, effective tool to combat phishing and other email-based threats, Tarasd contributes to building a more secure digital environment for businesses in the region. Furthermore, the system supports the Vision's emphasis on creating a knowledge-based economy by improving the technical literacy of businesses and individuals, helping them make informed decisions regarding email security.

Throughout the development process, the project adhered to best practices in system design, implementation, and testing. Rigorous testing methods were employed to ensure the accuracy and reliability of the phishing detection algorithms, resulting in a robust solution that delivers both precision and usability. The system was tested under various

conditions to ensure its ability to detect phishing emails in real-time, a critical feature in combating the constantly evolving nature of cyber threats. The integration of machine learning models ensured that Tarasd could continually improve its phishing detection capabilities by learning from new data, making it adaptable to future threats.

6.2 LIMITATIONS OF THE PROJECT

While the Tarasd system has demonstrated strong potential in detecting phishing emails and enhancing cybersecurity awareness, it is important to acknowledge several limitations that may affect its current performance and scalability. Identifying these limitations provides valuable insight for future development and refinement of the system

The limitations include:

Dependency on Gmail API

The system is currently optimized for Gmail accounts through the use of the Gmail API. While this provides a focused and secure integration, it limits the application's use for individuals or organizations using alternative email providers such as Outlook, Yahoo, or corporate mail servers. Expanding compatibility would enhance the system's reach and impact.

Basic AI Assistant Capabilities

Although the project includes an AI assistant to support user interaction and cybersecurity guidance, its conversational abilities are still limited compared to advanced chatbot systems. It currently focuses on predefined topics related to phishing and system navigation, with minimal natural language understanding beyond the training data.

Limited Dataset Scope

The accuracy of the AI model heavily depends on the quality and diversity of the training dataset. In this version of the project, the phishing detection model was trained using a dataset that may not fully cover all types of phishing techniques,

Platform Dependency

The Tarasd desktop application is currently developed and available only for Windows operating systems. This restricts accessibility for users on other platforms such as macOS or Linux, which may be commonly used in certain organizations or by cybersecurity professionals.

6.3FUTURE WORKS

To enhance the functionality, scalability, and impact of the Tarasd system, several areas have been identified for future improvement and development. These enhancements aim to address current limitations, support a broader range of users, and strengthen the system's adaptability to evolving cybersecurity threats.

- **Multi-Email Provider Support**

To overcome the current dependency on Gmail, future versions of Tarasd will integrate with a broader range of email platforms such as Outlook, Yahoo, and custom enterprise email servers. This will enhance the system's applicability across different environments and ensure that users are not restricted to a single provider.

- **Cross-Platform Compatibility**

Development efforts will focus on creating cross-platform versions of the desktop application to support macOS and Linux systems in addition to Windows. This will provide greater flexibility for organizations with diverse IT infrastructures and promote wider adoption.

- **Enhanced AI Assistant Capabilities**

The AI assistant will be improved to provide more natural and dynamic conversations, expanding beyond predefined scripts. By incorporating more advanced natural language understanding (NLU) and context awareness, the assistant will be better equipped to handle a wider range of user queries and offer more personalized guidance.

- **Expansion of the Training Dataset**

The AI model's accuracy will be enhanced by incorporating larger and more diverse datasets, including samples of sophisticated, multilingual, and context-specific phishing attempts. This will improve the system's ability to detect newer and more complex phishing threats with greater precision.

- **Multilingual Phishing Detection**

  The system will expand to support phishing detection in multiple languages beyond English, such as Arabic, French, and others. This will be especially important for users in multilingual regions or organizations with global operations.

REFERENCES

Research Papers**:**

[1.]Anti-Phishing Working Group (APWG), Phishing Activity Trends Report 2023. Analysis of global phishing activity. Available. (https://apwg.org/)

[2.]Verizon, 2023 Data Breach Investigations Report. Analyzing cyberattack trends and phishing incidents globally. Available. (https://enterprise.verizon.com/resources/reports/dbir/)

[3.]M. Khan et al., "Phishing Website Detection Using Machine Learning," 2023 IEEE International Conference on Artificial Intelligence and Applications (AIA), pp. 450–456. DOI: 10.1109/AIA.2023.9824801 (https://ieeexplore.ieee.org/document/9824801)

[4.]S. Ali et al., "Next Generation Phishing Detection and Prevention System Using Machine Learning," 2024 IEEE Advanced Cybersecurity Conference, pp. 110–115. DOI: 10.1109/ACSC.2024.10085529 (https://ieeexplore.ieee.org/document/10085529)

[5.]B. Krebs, Krebs on Security: Investigating Cyber Threats. Personal blog, ongoing since 2009. Insights on cybercrime and cybersecurity. Available. (https://krebsonsecurity.com/)

[6.]M. Jakobsson and S. Myers, Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. Hoboken, NJ: Wiley-Interscience, 2006, pp. 1–700. Available. (https://archive.org/details/isbn_9780471782452

[7.]IEEE, "Phishing Attacks and Detection Techniques: A Systematic Review," 2024 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 302–309. DOI: 10.1109/CSR.2024.10630203 (https://ieeexplore.ieee.org/document/10630203) .

Websites:

[8.]R. Zieni et al., “Phishing or Not Phishing? A Survey on the Detection of Phishing Websites,” IEEE Access 2023, vol. 11, pp. 18,501–18,515. DOI: 10.1109/ACCESS.2023.3255214 (https://ieeexplore.ieee.org/document/10435211)

[9.]SpamTitan, “Advanced Email Filtering for Phishing Detection,” Official Website, 2024. Available. (https://www.spamtitan.com/)

[10.] Zix, “Secure Email Encryption and Phishing Protection,” Official Website, 2024. Available. (https://www.zix.com/)

[11.] Mialgun, “Real-Time Email Analysis for Cybersecurity,” 2024. Available. (https://www.mailgun.com/)