SECURE SHEILD: A MULTI DOMAIN THREAT DETECTION SYSTEM

PROJECT THESIS

SUBMITTED

TO

AWH ENGINEERING COLLEGE
KUTTIKKATTOOR, CALICUT

IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE
OF

Master Of Computer Applications

BY

AKSHAY C K

DEPARTMENT OF COMPUTER APPLICATIONS
AWH ENGINEERING COLLEGE KUTTIKKATTOOR
CALICUT
MAY 2025

DEPARTMENT OF COMPUTER APPLICATIONS

AWH ENGINEERING COLLEGE
CALICUT.KERALA.INDIA

AWH ENGINEERING COLLEGE
CALICUT

CERTIFICATE

This is to certify that this thesis entitled "SECURE SHEILD: A MULTI DOMAIN THREAT DETECTION SYSTEM" submitted here with is an authentic record of the thesis work done by AKSHAY C K (AWH23MCA-2006) under our guidance in partial fulfillment of the requirements for the award of ***Master of Computer Applications*** *from APJ Abdul Kalam Technological University during the academic year 2025.*

Mrs. Sruti Sudevan	**Mr. Muhammed Muhsin. K**
Associate Professor	Assistant Professor
Dept. of Computer Applications	Dept. of Computer Applications
Head of the department	Project guide

External Examiner	**Internal Examiner**

ACKNOWLEDGEMENT

I express my sincere gratitude to our beloved principal **Dr. Sabeena M V** for providing me an opportunity with the required facilities for doing this project. I express my hearty thanks to **Mrs. Sruti Sudevan** Head of the department of Computer Applications and **Mr. Muhammed Muhsin. K** for their guidance. I am thankful to all other staff of the MCA department for their encouragement, timely guidance, valuable suggestions and inspiring ideas given throughout this project. I am grateful to my friends for the way they have cooperated, expected me to achieve success and have always stirred my ambition to dothe best. Above all, I am grateful to the almighty, who has showered His blessings on me throughout my life and throughout the project.

AKSHAY CK

ABSTRACT

The Multi-Domain Threat Detection System is a comprehensive solution designed to identify and mitigate various cyber threats, including phishing websites, spam emails, and image forgery. As cyberattacks become more sophisticated, there is a growing need for advanced, automated systems capable of detecting and preventing potential security breaches. This project leverages machine learning algorithms to analyze user-provided data and deliver accurate predictions across three critical domains: phishing detection, spam email classification, and image authenticity verification.

The system is divided into three primary modules. The Phishing Detection Module utilizes the Random Forest algorithm to distinguish between legitimate and malicious websites. The Spam Email Detection Module classifies emails as spam or legitimate by training on email datasets, while the Image Forgery Detection employs advanced image analysis techniques to identify manipulated images. Each module operates independently but integrates seamlessly to provide a unified threat detection framework.

CONTENTS

Page No

INTRODUCTION

1. INTRODUCTION

In the digital age, cybersecurity threats are evolving rapidly, posing significant risks to individuals, businesses, and institutions. Phishing attacks, spam emails, and image forgery are among the most prevalent cyber threats that can lead to financial loss, data breaches, and misinformation. The Multi-Domain Threat Detection System is a comprehensive solution that leverages machine learning algorithms to identify and mitigate these threats effectively. This system is designed to function across three primary domains: phishing website detection, spam email classification, and image forgery analysis.

Phishing website detection aims to identify fraudulent websites that deceive users into disclosing sensitive information. By utilizing machine learning models such as Random Forest, the system analyzes website URLs to predict whether they are legitimate or phishing attempts. Similarly, the spam email classification module is designed to detect and filter out unsolicited or malicious emails. By training machine learning classifiers on email datasets, the system can distinguish between spam and legitimate messages, reducing the risk of users falling victim to scams or malware. Additionally, the image forgery detection feature ensures the authenticity of digital images by analyzing inconsistencies and alterations using CNN image classification and AI-driven image processing techniques.

The significance of this project lies in its ability to enhance digital security by proactively identifying potential threats before they can cause harm. By integrating phishing detection, spam filtering, and image forgery analysis into a unified system, users can safeguard their online activities and data. The system consists of two primary modules: the Admin Module, which manages users, monitors complaints, and oversees detection processes, and the User Module, which allows users to check URLs for phishing attempts, analyze emails for spam detection, and verify the authenticity of images.

This project is implemented using Python for machine learning and MySQL for database management. The user-friendly interface enables seamless interaction, while the robust backend ensures efficient data processing and security. The Multi-Domain Threat Detection System demonstrates the practical application of artificial intelligence in cybersecurity. By combining advanced machine learning techniques, this system provides an intelligent, automated, and efficient approach to detecting and preventing cyber threats. It offers a proactive solution for ensuring digital security, helping individuals and organizations stay protected from phishing attacks, spam emails, and image forgery.

SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

2.1 Existing system

The existing approaches to cyber threat detection are largely fragmented and specialized, requiring users to depend on multiple standalone tools to address different types of threats such as phishing attacks, spam emails, and image forgery. Moreover, many of these methods still rely heavily on manual detection processes, which are not only time-consuming but also prone to human error. As cyber threats become more sophisticated and dynamic, manual and rule-based detection systems struggle to keep pace. Traditional spam filters and phishing detectors, for example, often operate using static rules or blacklists, which are limited in their ability to detect new, evolving, or cleverly disguised attacks. These systems fail to generalize beyond known patterns and can be easily bypassed by attackers using slightly modified tactics.

In the case of image forgery detection, many conventional methods depend on visual inspection or basic forensic techniques, which are insufficient for identifying subtle digital manipulations such as cloning, splicing, or resampling. These types of alterations can easily evade detection by the human eye or by simple filters, especially when carried out using advanced editing tools. Due to these limitations, there is a clear need for a more intelligent, automated, and unified solution one that leverages modern machine learning algorithms to detect a wide range of threats with accuracy and adaptability. The Multi-Domain Threat Detection System addresses this gap by providing a centralized platform capable of identifying phishing websites, spam emails, and forged images, all through the use of robust machine learning models that can learn and evolve with the threat landscape.

2.2 Proposed system

The Multi-Domain Threat Detection System addresses the limitations of traditional cybersecurity methods by integrating three essential threat detection modules into a single, unified platform. This all-in-one solution is designed to enhance the accuracy, efficiency, and real-time responsiveness of cyber threat identification across multiple domains. Rather than relying on separate tools for different types of threats, the system streamlines the detection process, enabling users and administrators to manage and mitigate diverse cyber risks from a centralized interface.
The first component, the Phishing Detection Module, leverages machine learning algorithms specifically the Random Forest classifier to analyse and evaluate website URLs.

considers various URL features such as domain structure, presence of suspicious characters, and hosting details to accurately classify websites as either legitimate or phishing. This module is critical in preventing users from falling victim to fraudulent websites that attempt to steal personal or financial information. The second component, the Spam Email Detection Module, utilizes advanced machine learning models trained on large datasets of emails to distinguish between spam and non-spam messages. By analysing the content, metadata, and structural patterns of incoming messages, the system effectively filters out unwanted or malicious emails that could lead to phishing attacks, malware downloads, or other harmful activities.

The third component, the Image Forgery Detection Module, employs AI-based image analysis techniques, such as convolutional neural networks (CNNs), to detect manipulated or tampered images. This module plays a vital role in verifying the authenticity of digital images by identifying signs of editing or duplication that may not be visible to the human eye. It is especially useful in contexts where image integrity is crucial, such as legal documentation, media publications, or digital forensics.

Together, these modules form a comprehensive, automated detection system that not only improves the accuracy of threat detection but also ensures faster response times and reduced reliance on manual analysis. The system features a user-friendly interface developed using Flutter, which allows both administrators and regular users to easily interact with the platform, review detection results, and manage tasks efficiently. By unifying multiple cybersecurity functions into a single application, the Multi-Domain Threat Detection System represents a significant advancement in proactive, intelligent threat management.

2.3 Module Description

This project has 2 modules:

Admin: -

- Login
- View Users
- Change Password
- View App Review
- View Complaints
- Send Reply

Customer:

- Registration
- Login
- View Profile
- Edit Profile
- Send Complaint
- View Reply
- App Review
- Image Forgery
- Email Spamming
- URL phishing

2.4 Sprint

Sprint 1

Module	Task	Hours for completion	Expected date of completion	Actual date of completion
Installation	SQL Yog, Pycharm	6 hours	20-01-2025	20-01-2025
Basics	Django, Libraries	10 hours	21-01-2025	22-01-2025
Modules	Modules, Form Design	8 hours	23-01-2025	23-01-2025
Diagrams	Sequence, Use case	8 hours	24-01-2025	24-01-2025
Database	Database Setup	4 hours	29-01-2025	29-01-2025
Admin	Login	6 hours	29-01-2025	30-01-2025
	Change Password	4 hours	29-01-2025	29-01-2025
	View users	8 hours	30-01-2025	31-01-2025

Sprint 2

Module	Task	Hours for completion	Expected date of completion	Actual date of completion
Admin	View Complaint & replay	10 hours	03-02-2025	4-02-2025
	View Review	8 hours	05-02-2025	05-02-2025
	Home page	6 hours	05-02-2025	05-02-2025
	logout	6 hours	06-02-2025	06-0-2025
	Template	20 hours	07-02-2025	11-02-2025

Sprint 3

Module	Task	Hours for completion	Expected date of completion	Actual date of completion
Admin	Testing and bug fixes	10 hours	14-02-2025	14-02-2025
	Signup	8 hours	17-02-2025	17-02-2025
User	Send Review	8 hours	17-02-2025	19-02-2025

	Send Complaint	8 hours	19-02-2025	21-02-2025
	View Replay	8 hours	21-02-2025	21-02-2025
	Change Password	8 hours	24-02-2025	24-02-2025

Sprint 4

Module	Task	Hours for completion	Expected date of completion	Actual date of completion
User	View Profile	8 hours	25-02-2025	25-02-2025
	Update Profile	8 hours	27-02-2025	27-02-2025
	home	8 hours	28-02-2025	28-02-2025
	Change password	8 hours	03-03-2025	03-03-2025
	Template	18 hours	04-03-2025	06-03-2025

Sprint 5

Module	Task	Hours for completion	Expected date of completion	Actual date of completion
User	Data set & requirements collection	15 hours	07-03-2025	10-03-2025
	Cleaning Data	15 hours	10-03-2025	12-03-2025
	Choosing Model	8 hours	13-03-2025	13-03-2025
	Training Model	12 hours	14-03-2025	15-03-2025

Sprint 6

Module	Task	Hours for completion	Expected date of completion	Actual date of completion
User	Evaluating Model	20 hours	17-03-2025	19-03-2025
	Parameter Tuning	15 hours	19-03-2025	20-03-2025
	Make Prediction	4 hours	20-03-2025	24-03-2025

Sprint 7

Module	**Task**	**Hours for completion**	**Expected date of completion**	**Actual date of completion**
User	Image Forgey Detection	15 hours	26-03-2025	27-03-2025
	Email Classification	15 hours	28-03-2025	01-04-2025
	URL Phishing Detection	15 hours	02-03-2025	04-04-2025
	Testing	5 hours	04-03-2025	07-03-2025

Sprint 8

Module	**Task**	**Hours for completion**	**Expected date of completion**	**Actual date of completion**
User	Debug	15 hours	08-04-2025	10-04-2025
	Validation	4 hours	10-04-2025	10-04-2025

machine	Documentation	20 hours	15-04-2025	21-04-2025
	Integration Testing & debug	11 hours	21-04-2025	22-04-2025

2.1 User Stories

Admin

- As an admin, I want to view details of customers.
- As an admin, I want to view review from customer.
- As an admin, I want to view complaints and send replay.
- As an admin, I want to change password.

Customer

- As a customer, I want to view and edit profile.
- As a customer, I want to raise a complaint.
- As a customer, I want to view replay for complaint.
- As a customer, I want to make review of web app.
- As a customer, I want to check image forgery detection.
- As a customer, I want to check email content.
- As a customer, I want to check whether URL is phishing or not.

FEASIBILITY STUDY

3. FEASIBILITY STUDY

An analysis of the ability to complete a project successfully, taking into account legal, economic, technological, scheduling, and other factors is considered a feasibility study. Rather than just diving into a project and hoping for the best, feasibility study allows project managers to investigate the possible negative and positive outcomes of a project before investing too much money and time. The Multi-Domain Threat Detection System is a cybersecurity solution designed to detect phishing websites, spam emails, and image forgery using machine learning techniques. A feasibility study is essential to evaluate the practicality of developing and implementing this system, considering various factors such as technical, operational, economic, legal, and time feasibility.

3.1 Economic feasibility

Economic feasibility involves evaluating whether the benefits and savings generated by a proposed system justify the costs associated with its development, implementation, and maintenance. In the case of the Multi-Domain Threat Detection System, economic feasibility is strongly supported through a favourable balance of cost-effectiveness, operational efficiency, and long-term value. This system is designed using open-source technologies such as Python, Flutter, MySQL, and popular machine learning libraries like TensorFlow and scikit-learn, which significantly reduce software licensing expenses. By utilizing readily available and cost-efficient tools, the development costs are minimized without compromising on performance or reliability.

Furthermore, the system is time-effective, as it automates complex cybersecurity tasks such as phishing detection, spam filtering, and image forgery analysis. This automation not only reduces the need for manual review but also saves considerable time for users and organizations, leading to higher productivity and lower labour costs. The system also presents both tangible benefits, such as reduced incidents of cyber threats, minimized data loss, and operational cost savings, and intangible benefits, including improved user trust, enhanced digital security, and increased organizational reputation.

The process of cost-benefit analysis a widely accepted technique in assessing economic feasibility has been applied to compare the estimated costs of system development and maintenance with the potential benefits gained over time. The initial investment in development is offset by the long-term savings from enhanced security and reduced vulnerability to cyberattacks. Additionally, since the system is scalable and built on technologies that require minimal upkeep, future maintenance costs remain low. In conclusion, the Multi-Domain Threat Detection System proves to be economically feasible, offering a smart, secure, and sustainable solution that delivers value over time without incurring excessive financial burden.

3.2 Technical Feasibility

The technical feasibility of the Multi-Domain Threat Detection System is well-established, as the project is built upon a robust and proven technology stack capable of handling complex backend operations, including database transactions, user authentication, and secure communication. The backend of the system can be effectively developed using widely used technologies such as Python, PHP, or Java, all of which are highly suitable for building scalable and secure applications. For data storage and management, MySQL or PostgreSQL offer reliable and efficient database solutions that support structured data storage, user activity tracking, and system configuration management.

On the front-end, the system can leverage modern frameworks such as Flutter, React, or Angular to create a smooth, responsive, and user-friendly interface. These frameworks support cross-platform development, allowing the application to be accessible from various devices including desktops, tablets, and mobile phones. The system is further empowered by the integration of advanced machine learning algorithms like Random Forest, which provide accurate, real-time threat detection across multiple domains, including phishing websites, spam emails, and image forgery. The use of Python, along with its rich ecosystem of ML libraries such as scikit-learn, TensorFlow, and pandas, enables the implementation of efficient, high-performance models that can handle real-world data scenarios effectively.

Importantly, the system is designed to operate on commonly available hardware, requiring only a Pentium 4 or higher processor with at least 8 GB of RAM, ensuring broad accessibility and eliminating the need for specialized or high-cost equipment. Additionally, the modular and scalable nature of the technology stack ensures that the system can evolve over time to incorporate new features or handle increased user loads without requiring significant infrastructure changes. In summary, the system is technically feasible, as it utilizes mature, well-supported, and scalable technologies that collectively provide a strong foundation for reliable and efficient machine learning-based threat detection.

3.3 Operational Feasibility

The operational feasibility of the Multi-Domain Threat Detection System is highly robust, as the system is designed to function efficiently and meet the needs of all key stakeholders, including end-users, administrators, and organizational entities. The platform is built to detect and prevent cyber threats across multiple domains phishing websites, spam emails, and image forgeries making it a comprehensive cybersecurity solution. Its functionality is divided into user-friendly modules that streamline tasks for different roles. The user module allows individuals to easily access services such as phishing URL analysis, email spam detection, and image forgery verification through a clean and intuitive interface. The design prioritizes accessibility, enabling both technical and non-technical users to interact with the system without difficulty, thereby increasing overall usability and engagement.

Developed using HTML for the front-end interface, the system ensures smooth navigation and fast response times, which are essential for operational efficiency. The integration of machine learning algorithms automates threat detection processes, significantly reducing the need for manual review and allowing users to receive accurate results with minimal input. This level of automation enhances productivity while minimizing the margin of error. On the administrative side, the admin module provides centralized control for managing user interactions, responding to complaints, overseeing activity logs, and maintaining data integrity. This oversight mechanism ensures the system can operate smoothly even as the user base grows or security challenges evolve.

Furthermore, the system's back-end, built with Python and MySQL, is lightweight and scalable, requiring minimal maintenance and offering high performance across a variety of computing environments. With its clear workflows, low operational complexity, and ability to deliver real-time cybersecurity insights, the system not only addresses current user needs but also ensures sustained reliability and adaptability. Thus, the Multi-Domain Threat Detection System is operationally feasible, as it successfully combines ease of use, efficiency, and automation to deliver a dependable cybersecurity solution.

3.4 Behavioral Feasibility

The behavioural feasibility of the Multi-Domain Threat Detection System is highly promising, as the system is designed with user acceptance and practical usability in mind. It provides an intuitive interface that caters to both technical and non-technical users, allowing them to analyse suspicious emails, URLs, and images with ease. With simple input forms, one-click analysis, and clear, understandable results, the system ensures a minimal learning curve.

The system addresses critical and increasingly common cybersecurity concerns such as phishing attacks, spam emails, and image forgery. By offering real-time analysis and accurate predictions, it fulfils the essential need for personal and organizational digital security. This alignment with current digital behaviour and expectations enhances the likelihood of user engagement and consistent use. Additionally, its seamless integration with daily online activities such as browsing websites, checking emails, and uploading images means users can adopt it naturally without altering their workflow.

Furthermore, the system builds user trust through transparency, presenting detailed information behind its predictions (like domain age, IP presence, or image metadata), and by ensuring high accuracy, thereby minimizing false positives and negatives. Its design also encourages user feedback through features such as complaint submission, review sections, and admin responses, ensuring users feel involved in system improvements and reinforcing their trust and satisfaction. From an organizational standpoint, change management is simplified due to the system's straightforward functionality and the minimal training it requires. The low level of behavioural resistance and the system's ability to enhance users' sense of digital safety without adding complexity make it a viable and behaviourally feasible solution. In conclusion, the Multi-Domain Threat Detection System aligns well with user expectations, promotes safer digital behaviour, and offers a practical, user-friendly approach to combating online threats.

3.5 Software Feasibility

Software feasibility assesses the availability, suitability, and practicality of the software tools and technologies required for developing and operating a system. In the case of the Multi-Domain Threat Detection System, software feasibility is strong, as the system leverages widely supported, open-source, and easily accessible technologies that are well-established in the development and data science communities. The front-end is developed using Django and HTML, an open-source UI framework by Google that supports cross-platform development, allowing the application to run smoothly on devices with a single codebase. This significantly reduces development time and ensures consistent user experiences across various platforms.

The back-end of the system is built using Python, a versatile and powerful programming language that is particularly well-suited for machine learning and data analysis tasks. Python's extensive ecosystem includes libraries such as scikit-learn for classical machine learning algorithms, TensorFlow for deep learning and neural networks, and pandas for efficient data manipulation and preprocessing.

These libraries not only accelerate development but also enhance the accuracy and reliability of the system's threat detection capabilities. Additionally, Python's simplicity and readability make the codebase easier to maintain and extend, ensuring long-term sustainability of the system. The database management is handled through MySQL or SQLite, both of which are reliable and lightweight solutions suitable for handling user data, logs, and system configurations.

Moreover, other essential libraries such as matplotlib for visualizations and whois for domain information analysis further strengthen the system's technical foundation. Given the maturity, stability, and community support surrounding these tools, the Multi-Domain Threat Detection System is not only feasible from a software perspective but is also well-positioned for future upgrades and scalability. This solid technological stack ensures that the software infrastructure can fully support the system's functional requirements and performance expectations.

3.5 Hardware Feasibility

Hardware feasibility examines whether the existing physical infrastructure is adequate to support the development, deployment, and operation of a given system. For the Multi-Domain Threat Detection System, hardware feasibility is highly favourable, as the system is specifically designed to function efficiently on standard computing environments without requiring any specialized or high-end hardware components. This makes it accessible and deployable across a wide range of devices, including typical personal computers, laptops, and office systems used by individuals and organizations.

The system performs key operations such as phishing URL detection, spam email classification, and image forgery analysis using optimized machine learning models that are lightweight and resource-efficient. These models are built using libraries like scikit-learn and TensorFlow, which are capable of executing effectively even on moderate processing units without requiring GPU acceleration. The front-end, developed using Django, is similarly optimized for smooth operation devices, further ensuring broad compatibility.

The minimum hardware requirements for running the system include a Pentium 4 processor or above, 8 GB of RAM, and 320 GB of hard disk space, which are specifications commonly available in most standard desktop and laptop computers today. Additionally, the system can operate on a variety of monitors and input devices such as optical mice and standard keyboards, reinforcing its practicality in everyday computing environments Because the system does not impose heavy demands on CPU, memory, or storage resources, it can be adopted by individuals, small businesses, educational institutions, and large organizations alike without incurring additional hardware investment. This significantly lowers the barrier to adoption and supports the system's scalability and long-term usability. Therefore, the Multi-Domain Threat Detection System is fully feasible from a hardware perspective, ensuring smooth deployment and operation across conventional IT infrastructure.

SOFTWARE ENGINEERING PARADIGM

4. SOFTWARE ENGINEERING PARADIGM

The software engineering paradigm which is also referred to as a software processmodel or Software Development Life Cycle (SDLC) model is the development strategy thatencompasses the process, methods and tools. SDLC describes the period of time that startswith the software system being conceptualized.

4.1 Agile model

The Agile model is a modern software development approach that emphasizes iterative and incremental progress, prioritizing flexibility, collaboration, and continuous customer feedback. Unlike traditional linear models, Agile is designed to respond quickly to changing requirements, making it ideal for dynamic and fast-paced environments. Projects are divided into small, manageable units called iterations or sprints, typically lasting one to four weeks. At the end of each iteration, a working product increment, often a minimum viable product (MVP), is delivered, allowing for early user testing, feedback, and refinement.

One of the core strengths of Agile is its adaptability. Requirements are not fixed at the start of the project; instead, they evolve based on ongoing feedback and learning. Agile teams are typically cross-functional, bringing together developers, testers, designers, product owners, and sometimes even customers, to work collaboratively and ensure alignment at every step. Communication is continuous and transparent, often facilitated through daily stand-ups and regular reviews, which keeps everyone informed and engaged.

Agile also fosters a customer-centric approach, placing the end-user at the heart of development. Stakeholders are actively involved throughout the process, providing feedback and helping prioritize features based on business value. Additionally, Agile promotes a culture of continuous improvement. At the end of each sprint, teams hold retrospectives to reflect on what went well and what could be improved, encouraging iterative growth in both the product and the team's processes. Transparency is another key component of Agile, with tools like Kanban boards, burndown charts, and task boards helping teams and stakeholders visualize progress, identify bottlenecks, and maintain accountability. Several methodologies fall under the Agile umbrella, including Scrum, which organizes work into structured sprints with defined roles and ceremonies; Kanban, which focuses on visualizing and optimizing workflow without time-boxed iterations; and Extreme Programming (XP), which emphasizes engineering best practices like test-driven development and pair programming.

In essence, Agile is not just a set of practices but a mindset one that values individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. This mindset enables teams to deliver high-quality software efficiently while staying closely aligned with customer needs and expectations.

4.2 Scrum

Scrum is a widely-used Agile framework designed to manage and organize software development projects through an iterative and incremental approach. It structures the development process into fixed-length cycles known as sprints, which typically last between two to four weeks. Each sprint aims to deliver a potentially shippable product increment, allowing teams to continuously refine and enhance the product based on regular feedback. Scrum is built on principles of collaboration, transparency, and adaptability, enabling teams to respond swiftly and effectively to evolving requirements and customer needs.

At the core of Scrum are clearly defined roles, each with specific responsibilities. The Product Owner represents the stakeholders and is responsible for defining and prioritizing the product backlog to ensure that the most valuable features are developed first. The Scrum Master serves as a facilitator and coach for the team, ensuring that Scrum principles are followed, removing impediments, and helping the team stay focused and productive. The Development Team is a self-organizing group of professionals who collaborate to design, build, test, and deliver the product increment during each sprint.

Scrum also incorporates a set of structured ceremonies or events that support its process. Sprint Planning is held at the beginning of each sprint to define the sprint goal and select backlog items to be completed. Daily Standups (or Daily Scrums) are short, time-boxed meetings where team members share updates, discuss progress, and identify any roadblocks. At the end of each sprint, a Sprint Review is conducted to demonstrate the completed work to stakeholders and gather feedback, while the Sprint Retrospective offers an opportunity for the team to reflect on the sprint and identify ways to improve future performance. By breaking down complex projects into manageable, time-boxed iterations, Scrum allows for early and frequent delivery of value, reducing the risk of failure and promoting continuous improvement. Ultimately, Scrum provides a robust and flexible framework that supports a more efficient, adaptive, and responsive software development process, ensuring that the end product is aligned with both business goals and user expectations.

SYSTEM REQUIREMENT SPECIFICATION

5.SYSTEM REQUIREMENTS SPECIFICATION

5.1 Software requirements

• Operating system	: Microsoft Windows 10 or above
• Frontend	: HTML/CSS/BOOTSTRAP/JS
• Backend	: MySQL
• Language	: python
• IDE	: Pycharm
• Framework	: Django

5.2 Hardware requirements

• Processor	: Intel Core i3 or above
• Main memory	:4 GB or above
• Hard Dick Capacity	:320 GB or above

SYSTEM DESIGN

6. SYSTEM DESIGN

System design is the first in the development phase for many engineered product or system. It may define the process of applying various techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

6.1. Database Design

Database design is a crucial step in creating a structured and efficient database system. It involves developing a detailed data model that defines the logical structure of the database and includes both logical and physical design decisions. This process incorporates all necessary elements, such as tables, relationships, constraints, and storage parameters, to ensure data integrity and efficient retrieval. The output of database design is often represented using a Data Definition Language (DDL), which serves as the foundation for creating the actual database. One key concept in database design is Normalization, which is the process of organizing data to minimize redundancy and eliminate undesirable characteristics such as insertion, update, and deletion anomalies. Normalization aims to break down complex tables into smaller, more manageable ones, ensuring that the database structure is logical and reduces unnecessary duplication of data.

Normalization is achieved through Normal Forms, which are a set of rules that define how data should be structured to prevent redundancy and anomalies. The normal forms build on each other, with each subsequent normal form imposing stricter conditions to enhance data organization. Below are the primary normal forms in database normalization:

1. **First Normal Form (1NF)**: A relation (or table) is considered to be in First Normal Form if all the values in the table are **atomic**—that is, indivisible and singular for every attribute (column) in the relation. This means that each attribute must contain only one value for each record, ensuring that there are no repeating groups or arrays within a column. Achieving 1NF eliminates data redundancy related to repeating groups.
2. **Second Normal Form (2NF)**: A relation is said to be in Second Normal Form if it is already in First Normal Form and satisfies additional conditions. Specifically, a table is in 2NF if:

- It does not contain any partial dependencies, meaning all non-key attributes must be fully dependent on the entire primary key.
- If the table has a composite primary key (a primary key made up of multiple columns), each non-key attribute must depend on the **whole** primary key, not just a part of it. This ensures that data is fully normalized and reduces redundancy in tables with composite keys.

3. **Third Normal Form (3NF)**: A relation is in Third Normal Form if it is in Second Normal Form and there are no transitive dependencies between attributes. A transitive dependency occurs when one non-key attribute depends on another non-key attribute, which in turn depends on the primary key. In 3NF, each non-key attribute should only depend on the primary key and not on other non-key attributes, ensuring that there are no indirect dependencies that could lead to anomalies.

By applying these normal forms, database designers can create a structure that reduces redundancy, prevents data anomalies, and ensures that the data is logically consistent. Normalization is an ongoing process, and further normalization forms, such as Boyce-Codd Normal Form (BCNF), Fourth Normal Form (4NF), and Fifth Normal Form (5NF), are used to refine the design further based on specific needs and scenarios. Ultimately, the goal of normalization is to create a database that is both efficient and easy to maintain, ensuring data integrity and optimal performance.

6.2 Tables

LOGIN

Field name	**Type**	**Width**	**Constraints**
Login_Id	int	11	Primary key
Username	Varchar	100	Not null
Password	Varchar	100	Not null
Type	Varchar	40	Not null

CUSTOMER

Field name	**type**	**Width**	**Constraints**
Cust_Id	int	11	Primary key
Login_Id	int	11	Foreign key
Name	Varchar	100	Not null
Email	Varchar	100	Not null
Phone	Varchar	100	Not null

COMPLAINT

Field name	**Type**	**Width**	**Constraints**
Complaint_id	Int	11	Primary key
User_id	Int	11	Foreign key
Date	Varchar	100	Not null
Complaint	Varchar	100	Not null
Replay	Varchar	100	Not null
Status	Varchar	100	Not null

REVIEW

Field name	**Type**	**Width**	**Constraints**
Review_Id	Int	11	Primary key
User_Id	Int	11	Foreign key
Date	Varchar	100	Not null
Review	Varchar	100	Not null

6.3 UML Designs

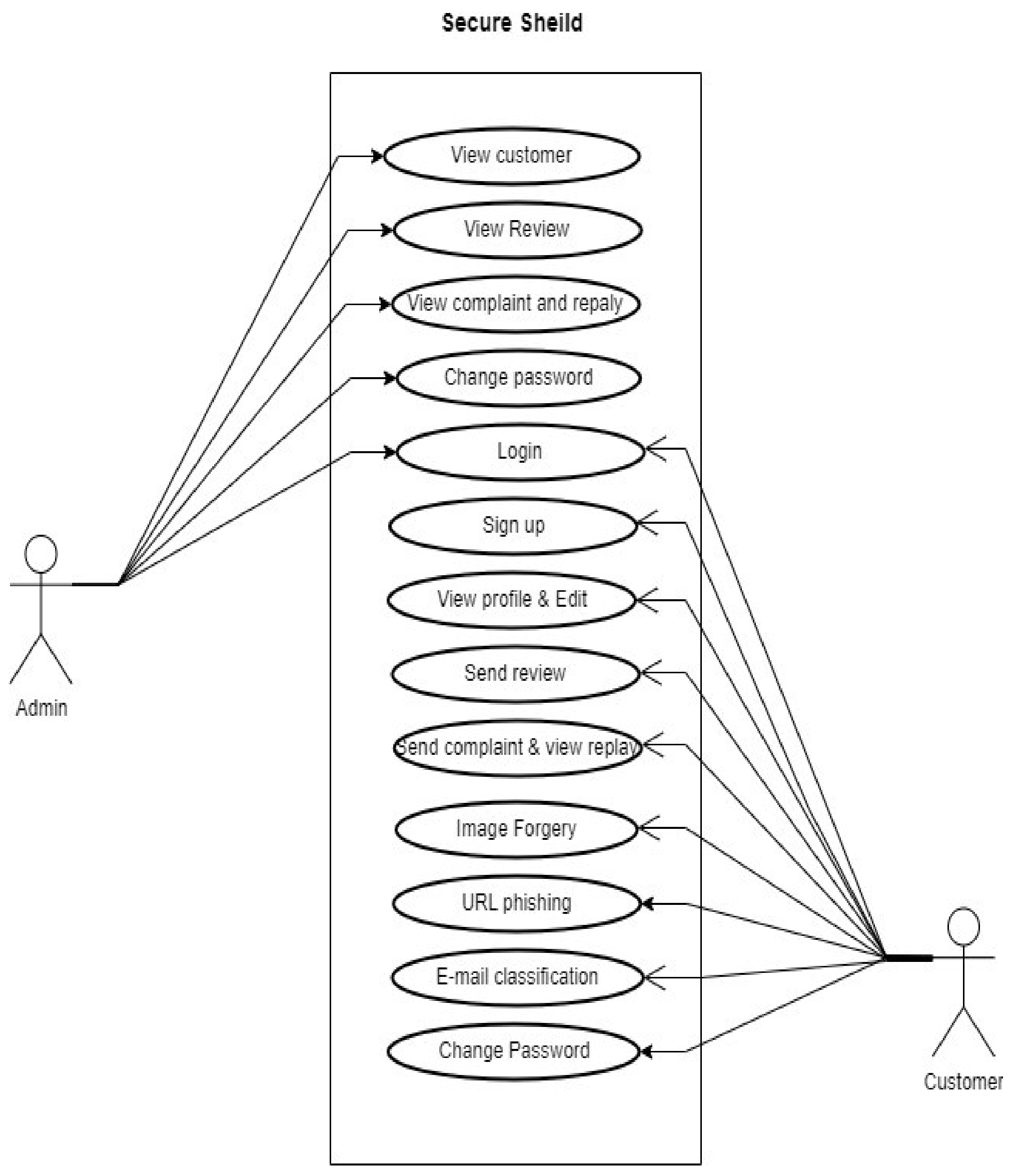
The Unified Modelling Language (UML) is a standardized language used for specifying, visualizing, constructing, and documenting the artifacts of software systems. It is also applicable in business modelling and non-software systems, making it a versatile tool for system designers. UML represents a collection of best practices for modelling large and complex systems and is widely adopted in software development. UML is essential in object-oriented software development, providing a common framework to understand system behaviours, structure, and interactions. Its use of graphical notations allows software teams to communicate effectively, explore potential designs, and validate the architectural design, ensuring all stakeholders have a clear understanding of the system.

Two important types of UML diagrams are sequence diagrams and use case diagrams, each serving distinct purposes in system modelling.

1. **Sequence Diagram**: A sequence diagram is a type of UML diagram that focuses on the interactions between objects or components within a system over time. It shows how objects communicate with each other by sending messages, and it also captures the sequence and order of those interactions. The sequence diagram is particularly useful for modelling the dynamic behaviours of the system, such as the flow of control between components during specific scenarios or use cases. It provides clarity on how objects collaborate and the timing of message exchanges, which helps developers understand and optimize the system's behaviour.
2. **Use Case Diagram**: A use case diagram represents the functionality of a system from the user's perspective. It provides a high-level view of the system’s behaviours by illustrating how users (actors) interact with the system to achieve specific goals or tasks. Use case diagrams are instrumental in capturing the functional requirements of a system, identifying the roles or actors involved, and showing which use cases or actions, each actor participates in. This diagram helps in understanding the system’s scope, the interactions between users and the system, and is valuable in the initial phases of system design to ensure that all user requirements are met.

Both sequence diagrams and use case diagrams play a crucial role in system modelling, helping developers, stakeholders, and project teams to communicate effectively, understand the system's dynamic and functional aspects, and ensure that the system meets its requirements.

6.4 Use case diagram:



6.5 Scenario

Admin:

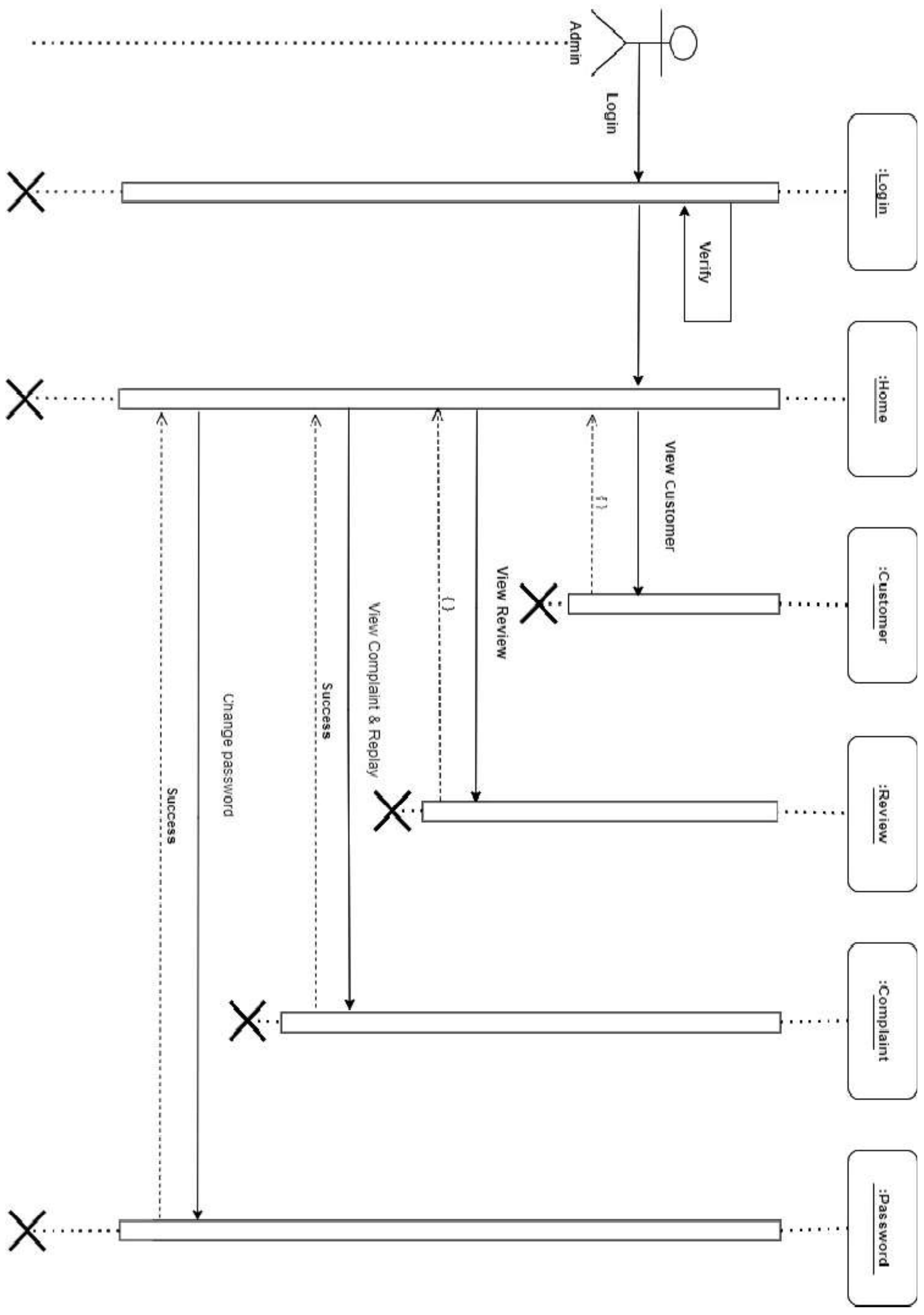
- Admin can view customers
- Admin can view review
- Admin can view complaints and make reply
- Admin can change password

Customer:

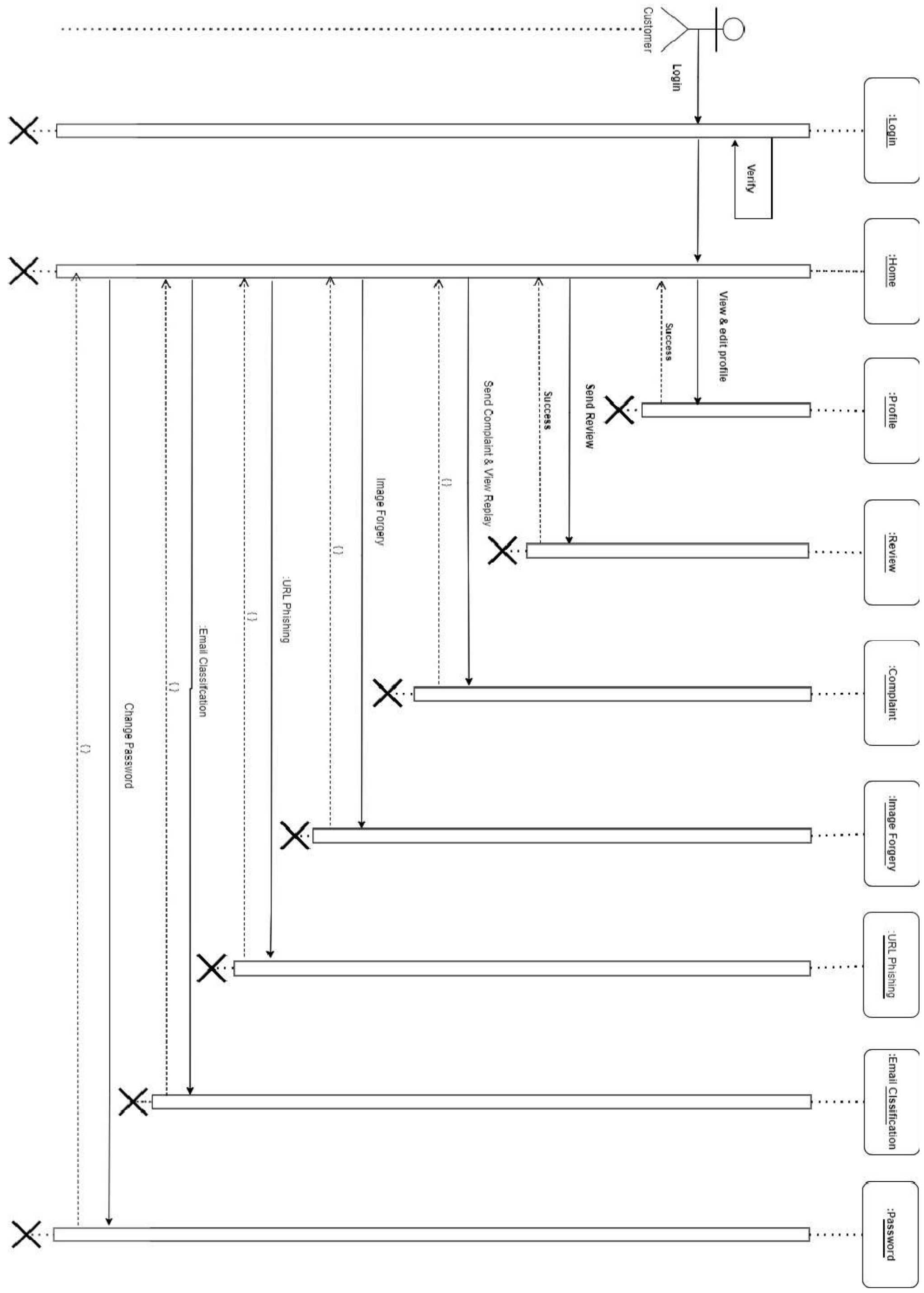
- Customer can check email content
- Customer can check image forgery
- Customer can check URL is phishing or not
- Customer can send complaint
- Customer can view replies
- Customer can send review
- Customer can view profile and update
- Customer can change password

6.6 Sequence Diagram

Admin:



Customer:


Customer
Login
:Login
Verify
:Home
View & edit profile
Success
:Profile
Send Review
Success
:Review
Send Complaint & View Replay
{}
:Complaint
Image Forgery
{}
:Image Forgery
:URL Phishing
{}
:URL Phishing
:Email Classification
{}
:Email Clssification
Change Password
{}
:Password

SYSTEM DEVELOPMENT

7. SYSTEM DEVELOPMENT

System development is a series of operations to manipulate data to produce output from a computer system. The principal activities performed during the development phase can be divided into two major related sequences.

- External system development
- Internal system development

The major external system activities are:

- Implementation
- Planning
- Equipment acquisition
- Installation

7.1 coding

The purpose of code is to facilitate the identification and retrieval of items of information. A code is an ordered collection of symbols designed to provide unique identification of an entity or an attribute. Code also shows interrelationship among different items.

Python

Python is a widely used high-level programming language designed for general-purpose programming. It was created by Guido van Rossum and first released in 1991. Over the years, it has gained immense popularity among developers, data scientists, educators, and researchers due to its simplicity and versatility.

As an interpreted language, Python executes code line by line, which makes debugging and testing easier. One of its most notable features is its clean and readable syntax, which emphasizes clarity and reduces the cost of program maintenance. Python's design philosophy emphasizes code readability, and it uses indentation (whitespace) to define code blocks, as opposed to the curly braces {} used in languages like C++ or Java. This distinctive feature encourages a uniform coding style and helps programmers produce visually uncluttered code. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It comes with a vast standard library that provides tools suited to many tasks, including file I/O, regular expressions, web development, and data manipulation. Additionally, the language has a rich ecosystem of third-party libraries and frameworks that extend its capabilities into areas such as web development (e.g., Django, Flask), data science (e.g., Pandas, NumPy, Scikit-learn), artificial intelligence, automation, scripting, and more.

TensorFlow

TensorFlow is a powerful open-source machine learning framework developed by the Google Brain team. Initially released in 2015, it has become one of the most widely used platforms for developing and deploying machine learning and deep learning models across a range of applications. Built on a flexible architecture, TensorFlow enables developers to deploy computation across a variety of platforms, including CPUs, GPUs, TPUs (Tensor Processing Units), and even mobile and edge devices.

One of TensorFlow's key strengths lies in its support for both low-level operations and high-level APIs. This allows both beginners and experienced researchers to build everything from simple models to highly complex neural networks. With its computational graph model and automatic differentiation capabilities, TensorFlow is well-suited for tasks involving large-scale numerical computations and backpropagation-based training algorithms. The framework supports a wide array of machine learning tasks, such as classification, regression, clustering, natural language processing, and especially deep learning. In the realm of computer vision, TensorFlow excels in handling complex tasks such as object detection, image segmentation, and image classification.

Scikit-learn

Scikit-learn is a powerful and user-friendly open-source machine learning library built on top of foundational Python libraries such as NumPy, SciPy, and matplotlib. It is designed to provide efficient and accessible tools for a wide range of machine learning tasks, making it an excellent choice for both beginners and experienced practitioners in the field.

Scikit-learn supports a variety of supervised and unsupervised learning algorithms, including classification, regression, clustering, dimensionality reduction, and model selection. Some of the widely used algorithms included in the library are Random Forest, Support Vector Machines (SVMs), k-Nearest Neighbours (k-NN), Naive Bayes, and Decision Trees. These algorithms are implemented with a consistent and intuitive API, which greatly simplifies the process of building, training, testing, and evaluating models. The library also offers tools for data preprocessing (e.g., scaling, normalization, encoding categorical features), model evaluation (and pipeline creation, which allows for cleaner and more reproducible machine learning workflows.

Matplotlib

It is a comprehensive and versatile data visualization library for Python that enables users to create static, interactive, and animated plots with ease. First developed by John D. Hunter in 2003, it has become one of the most widely adopted tools in the data science and scientific computing communities. Matplotlib supports a wide variety of plot types, including line charts, bar charts, scatter plots, histograms, pie charts, error bars, heatmap**s**, and more. It offers fine-grained control over every aspect of a plot such as colours, fonts, axes, labels, legends, and styles making it highly customizable and suitable for professional-quality visualizations. One of the key strengths of Matplotlib lies in its ability to integrate seamlessly with other Python libraries such as NumPy, Pandas, and SciPy, which makes it particularly effective for analysing and visualizing numerical and tabular data. Additionally, Matplotlib can be used in a variety of environments, including standalone scripts, Jupyter notebooks, and even graphical user interfaces (GUIs).

In practical applications, Matplotlib is frequently used to generate visual reports for system analytics. For example, it can be employed to plot trends in phishing website detections, allowing administrators to observe time-based spikes or patterns in malicious activity. Similarly, it can illustrate spam email frequency over time, or visualize image forgery detection patterns to help analysts and security personnel identify common techniques used by attackers. These visual insights are invaluable for administrative review, helping decision-makers understand system performance, monitor threats, and strategize more effectively.

Pandas

Pandas is a powerful and widely used open-source Python library designed for data manipulation and analysis. Built on top of NumPy, Pandas introduces two primary data structures **Series** (one-dimensional labelled arrays) and Data Frame (two-dimensional labelled tables) that provide a highly intuitive and flexible way to handle structured data. One of Pandas' key strengths lies in its ability to efficiently manage and process large datasets. It offers a rich set of functions for data cleaning, transformation, aggregation, filtering, merging, reshaping, and time-series analysis. These capabilities make it an indispensable tool in data preprocessing workflows across a wide range of domains, including finance, healthcare, marketing, and cybersecurity It allows users to clean missing or inconsistent values, normalize and scale features, encode categorical variables, and perform feature selection or extraction. These preprocessing steps are crucial to ensure that machine learning models receive high-quality and meaningful input data.

Whois

The whois Python package is a lightweight and convenient tool used to query domain registration information from WHOIS databases. WHOIS is a protocol that provides details about registered domain names, including information about the domain owner (registrant), registration dates, expiration dates, registrar identity, and more. The whois Python library allows developers to automate the retrieval and parsing of this data directly within their scripts or applications. By using whois, developers can extract crucial metadata such as the domain age, domain creation and expiration dates, registrar information, and name server details. This information is particularly valuable in cybersecurity applications, especially when dealing with phishing detection.

In phishing detection systems, the whois package is employed to gather and analyse domain-related information, which helps assess the legitimacy of a website. For example, newly registered domains or those with anonymized or suspicious registrar details are often red flags indicating potential phishing activity. By comparing these attributes against known patterns of malicious behaviour, the system can make more informed decisions about whether a domain should be classified as legitimate or phishing.

Django

Django is a high-level, open-source web framework written in Python that promotes rapid development, clean design, and robust security. Originally developed in 2003 and released publicly in 2005, Django was created by experienced developers to help build web applications quickly and efficiently without sacrificing scalability or maintainability.

Django also boasts an active and thriving community, with a wealth of tutorials, forums, and third-party packages that extend its functionality. The framework is accompanied by comprehensive documentation, making it beginner-friendly while still powerful enough for professional-grade applications. In addition to being completely free and open-source, Django benefits from both free community support and various paid support options for enterprise-level users. Because of its clear structure and modular approach, Django is particularly well-suited for building applications that require robust back-end logic, user management, and secure API integration. It's commonly used in projects that involve machine learning model deployment, data dashboards, e-commerce platforms, and cybersecurity tools.

MYSQL Client

MySQL client is a Python library that acts as a MySQL database client, enabling Python applications to interact with MySQL databases. It is a fork of the original MySQL-python library, which was widely used in the Python 2 era. The primary motivation behind MySQL client was to add support for Python 3 and to address numerous bugs and compatibility issues present in the older library.

This library is licensed under the GNU General Public License (GPL) and is OS-independent, meaning it can run on various operating systems such as Linux, Windows, and macOS without major platform-specific issues. MySQL client supports Python 2.7 as well as Python 3.4 and above, making it suitable for legacy systems as well as modern Python applications. One of the most notable endorsements of MySQL client comes from the Django web framework. According to the official Django documentation, MySQL client is the recommended MySQL driver for use with Django when connecting to a MySQL or MariaDB database. This recommendation is due to MySQL client's stability, performance, and compatibility with Django's ORM (Object-Relational Mapper).

HTML

HTML (Hypertext Markup Language) is the standard markup language used to create and structure content on the World Wide Web. It defines the building blocks of web pages, including elements such as headings, paragraphs, links, images, lists, tables, forms, and multimedia content. HTML provides the fundamental structure of a webpage, which web browsers interpret and render for users to view and interact with. Originally developed by Tim Berners-Lee in 1991, HTML has evolved through various versions, with HTML5 being the latest major revision. HTML5 introduces numerous enhancements, including support for audio and video playback, semantic elements (like <article>, <section>, and <nav>), canvas for drawing graphics, and APIs for offline storage, geolocation, and more making it a comprehensive platform for modern web development.

CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the visual presentation and formatting of documents written in markup languages such as HTML and XML (including its various dialects like SVG, MathML, and XHTML). CSS plays a vital role in the front-end development of websites and web applications by defining how elements are displayed on screen, in print, or on other media. CSS separates content (HTML) from design and layout, allowing developers to apply consistent styles across multiple pages or components while maintaining clean and maintainable code. It controls aspects such as:

- Fonts (typeface, size, weight)
- Colors (text, background, borders)
- Spacing (margins, padding, line height)
- Layout (flexbox, grid, positioning)
- Visibility and layering (z-index, display, visibility)
- Animation and transitions (hover effects, motion)
- Responsiveness (media queries for different screen sizes)

One of the key strengths of CSS is its cascading and inheritance system, where styles can be layered, overridden, or inherited from parent elements. This provides both flexibility and control over how styles are applied throughout a webpage.

JavaScript

JavaScript is primarily responsible for controlling the interactivity of a webpage. This includes handling tasks like form validation, animations, event handling (clicks, mouse movements, keystrokes), and AJAX requests (for fetching data without reloading the page). JavaScript also powers single-page applications (SPAs), where entire websites or applications run from a single HTML page, providing users with a smooth and responsive experience.

JavaScript is often used in combination with third-party libraries and frameworks like React, Angular, Vue.js, and jQuery to simplify development and enhance functionality. These libraries and frameworks provide pre-written code and tools that streamline the process of building interactive web interfaces and managing complex user interactions.

Bootstrap

Bootstrap is a widely used open-source front-end framework that simplifies the process of designing and developing responsive, mobile-first websites and web applications. Developed originally by Twitter, Bootstrap provides a comprehensive collection of HTML, CSS, and JavaScript components that help developers build visually appealing and user-friendly interfaces with minimal effort.

One of its core strengths lies in its responsive grid system, which allows web pages to adapt seamlessly to various screen sizes, enhancing the user experience on desktops, tablets, and mobile devices. It comes equipped with pre-built UI components such as navigation bars, buttons, modals, alerts, carousels, and forms that can be easily customized and integrated into any web project.

Bootstrap ensures cross-browser compatibility, making applications function consistently across different web platforms. Additionally, it includes JavaScript plugins powered by jQuery that adds interactivity to web elements without requiring custom scripts. Its high level of customization through Sass variables and strong community support makes it accessible to both beginners and experienced developers. In the context of the Multi-Domain Threat Detection System, Bootstrap can be effectively used to create a responsive and intuitive user interface for users and administrators, allowing smooth navigation for entering URLs, checking emails, uploading images, and viewing analysis results. Its flexibility and ease of use make it an ideal choice for building the front-end of security applications.

NLTK

NLTK (Natural Language Toolkit) is a powerful open-source Python library used for natural language processing (NLP) and computational linguistics. It provides easy-to-use interfaces and a wide range of text-processing libraries that enable developers and researchers to work with human language data efficiently. NLTK supports tasks such as classification, tokenization, stemming, lemmatization, parsing, tagging, and semantic reasoning, making it a comprehensive toolkit for linguistic analysis.

NLTK includes a variety of corpora and lexical resources, such as WordNet, and also offers wrappers for industrial-strength NLP tools. It is particularly useful for building applications that require the ability to analyse, understand, or generate human language such as sentiment analysis, spam detection, and chatbot development.

In the context of the Multi-Domain Threat Detection System, NLTK plays a crucial role in the Spam Email Detection module. It can be used to process raw email texts by removing stop words, performing tokenization, stemming, and converting textual data into a format suitable for machine learning algorithms. By leveraging NLTK's text processing capabilities, the system can more accurately classify emails as spam or non-spam based on linguistic features. Its combination of educational resources and practical tools makes NLTK a valuable asset in the development of intelligent and language-aware security systems.

NumPy

NumPy (Numerical Python) is a powerful open-source library in Python that provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays. It is a foundational package for scientific computing in Python and is widely used in data analysis, machine learning, and other computational applications.

NumPy offers a fast and efficient way to perform array operations, such as element-wise calculations, linear algebra, statistical analysis, and Fourier transforms. Its core data structure is the ndarray, which enables efficient storage and manipulation of numerical data. NumPy also integrates well with other libraries like Pandas, Scikit-learn, TensorFlow, and Matplotlib, making it an essential tool for data scientists and machine learning engineers.

In the Multi-Domain Threat Detection System, NumPy is primarily used for handling and processing numerical data during the implementation of machine learning models. For example, in the Phishing URL Detection and Spam Email Classification modules, NumPy helps in feature extraction, data normalization, and matrix operations required for model training and prediction. It ensures that the underlying data manipulations are both accurate and computationally efficient, contributing to the overall performance and reliability of the system.

Datetime

The datetime module in Python is a built-in library used for manipulating dates and times. It provides classes for working with both date and time in both simple and complex ways. This includes retrieving the current date and time, formatting date-time strings, performing arithmetic operations like adding or subtracting days, and comparing different date-time values.
Some commonly used classes and functions in the datetime module include:

- datetime.datetime: Combines date and time into one object.
- datetime.date: Handles year, month, and day without the time component.
- datetime.time: Represents time (hours, minutes, seconds) without the date.
- datetime.timedelta: Used for performing operations like date differences or future/past date calculations.
- datetime.now(): Retrieves the current local date and time.
- datetime.strftime() and datetime.strptime(): Used for formatting and parsing date-time strings.

In the context of the Multi-Domain Threat Detection System, the datetime module is useful for logging events, timestamping user activity, tracking complaint submissions, and recording system responses. For instance, when a user checks a URL or uploads an image for forgery detection, the exact date and time of the action can be stored using datetime, helping in maintaining logs for audit purposes, generating reports, and managing time-based alerts or data retention policies. Its precision and ease of use make it a fundamental utility for time-sensitive operations in software systems.

Re

The **re** module in Python is used for working with regular expressions, which are powerful tools for pattern matching and text manipulation. This module allows developers to search, match, and manipulate strings based on specific patterns defined using regular expression syntax.
Key functions provided by the re module include:

- re.search(): Searches a string for the first match of a pattern.
- re.match(): Checks for a match only at the beginning of a string.
- re.findall(): Returns all non-overlapping matches of a pattern in a string.
- re.sub(): Replaces parts of a string that match a pattern with another string.
- re.split(): Splits a string by the occurrences of a pattern.
- re.compile(): Compiles a regular expression into a regex object for reuse.

In the context of the Multi-Domain Threat Detection System, the re module is especially useful in the Phishing URL Detection and Spam Email Classification modules. It can be used to extract and clean URLs, email addresses, or specific keywords from user input. For example:

- Extracting domain names from URLs.
- Identifying suspicious patterns in email content (e.g., phishing phrases or encoded links).
- Preprocessing text data by removing unwanted characters or HTML tags.

By leveraging the re module, the system can efficiently parse and process unstructured text, which enhances the accuracy of machine learning models used for threat detection. Its ability to handle complex string operations makes it a vital component in cybersecurity and data preprocessing tasks.

Pickle

The pickle module in Python is used for serializing and deserializing Python objects, also known as *pickling* and *unpickling*. Serialization refers to converting a Python object into a byte stream, which can be saved to a file or transferred over a network. Deserialization is the reverse process—reconstructing the original Python object from the byte stream.
Key Functions of the pickle Module:

- pickle.dump(obj, file): Serializes obj and writes it to file.
- pickle.load(file): Reads a pickled object from file and returns the original Python object.
- pickle.dumps(obj): Returns the pickled representation of obj as a byte string.

- pickle.loads(bytes_obj): Converts a pickled byte string back into a Python object.

Use in Multi-Domain Threat Detection System:
In the Multi-Domain Threat Detection System, the pickle module is commonly used to save and load trained machine learning models. For example:

- After training a Random Forest model for phishing URL detection or spam email classification, the model is serialized using pickle.dump() and saved to disk.
- When the system needs to make predictions (e.g., when a user submits a URL or email), the saved model is loaded back into memory using pickle.load().

This allows the system to reuse trained models without retraining them every time it runs, improving both performance and efficiency. pickle is especially useful for deploying ML models in production environments, where fast and consistent access to models is required. Since pickle can execute arbitrary code during deserialization, it's important to only unpickle data from trusted sources to avoid security risks.

Urlparse

The urlparse module, found in Python's urllib.parse package, is used to break down URLs into components such as scheme, netloc, path, query, and fragment. This is especially useful in web-related applications like phishing detection, where analysing the structure of a URL can reveal suspicious or malicious patterns.

Key Function: urllib.parse.urlparse(url)

This function parses a URL string into six components:

- **scheme**: Protocol used (e.g., http, https)
- **netloc**: Network location part (e.g., www.example.com)
- **path**: Hierarchical path (e.g., /path/page.html)
- **params**: Parameters for last path element (rarely used)
- **query**: Query component (e.g., ?id=123)
- **fragment**: Fragment identifier (e.g., #section)

Use in Multi-Domain Threat Detection System:

In the Phishing Detection Module, urlparse is used to:

- Extract the domain (netloc) and compare it against known phishing domains.
- Analyse the path and query for suspicious strings, such as long encoded sequences, hidden redirects, or login pages mimicking legitimate websites.

- Identify shortened URLs or domains that deviate slightly from well-known brands (a common phishing tactic).

nltk.corpus

The nltk**.**corpus module is part of the Natural Language Toolkit (NLTK) library in Python and provides access to a wide range of **corpora**—large and structured sets of texts that are used for natural language processing tasks. These corpora include things like collections of movie reviews, names, stopwords, word lists, tagged text, and more. A **corpus** (plural: corpora) is a body of text used to train or evaluate NLP models. It can consist of books, articles, speech transcripts, tweets, or any textual content used for analysis.

Commonly Used Corpora in nltk.corpus:

- **stopwords**: A list of common words (like "is", "the", "and") that are usually filtered out during text processing.
- from nltk.corpus import stopwords
- print(stopwords.words('english'))
- **names**: Contains lists of male and female first names, useful in gender classification tasks.
- **movie_reviews**: A collection of movie reviews labelled as positive or negative—often used for sentiment analysis.
- **wordnet**: A lexical database of English. It groups words into sets of synonyms and provides short definitions and usage examples.
- **punkt (via nltk.tokenize)**: Used for tokenizing text into words or sentences.

Use in Multi-Domain Threat Detection System:

In the Spam Email Detection Module, nltk.corpus is particularly helpful for:

- **Filtering out stopwords** to focus on meaningful words that might indicate spam.
- Using datasets for training and testing spam classifiers.
- Enhancing text preprocessing before feeding data into machine learning models.

By using nltk.corpus, the system becomes more linguistically aware, allowing it to better analyse and classify emails based on natural language content.

OS

PIL (Python Imaging Library) is a powerful library in Python for opening, manipulating, and saving many different image file formats. PIL supports a wide range of image formats such as PNG, JPEG, BMP, TIFF, and others. It also provides many image processing features, including opening, converting, resizing, cropping, rotating, and applying various effects like blur and sharpen. As of now, PIL is no longer maintained, Pillow provides the same features as PIL, along with additional improvements and fixes.

Common Features of PIL/Pillow:

- **Opening and Saving Images**: You can open and save images in various formats.
- **Image Manipulation**: Resize, crop, rotate, and transform images.
- **Drawing on Images**: Add text, shapes, and other graphics to images.
- **Filters**: Apply different image filters such as blur, sharpen, etc.
- **Image Enhancement**: Adjust brightness, contrast, and other parameters.
- **Image Conversion**: Convert images between different formats (e.g., from PNG to JPEG).

Use in Multi-Domain Threat Detection System:

The Image Forgery Detection Module in your Multi-Domain Threat Detection System can benefit from PIL/Pillow for:

- **Image Preprocessing**: Before running machine learning models for image forgery detection, you may need to preprocess images (resize, convert to grayscale, or enhance features).
- **Image Analysis**: Apply filters or transformations to detect anomalies or alterations in the images, helping to identify forgery.
- **Extracting Image Features**: Manipulate or visualize certain features of the images that could help distinguish original content from manipulated content.

Using **Pillow** in combination with machine learning models can significantly enhance the system's ability to detect tampered or manipulated images by ensuring that the images are pre-processed and analysed accurately.

Keras

Keras is an open-source, high-level neural network API written in Python. It runs on top of powerful backends like TensorFlow, Theano, or CNTK, and is designed to enable fast experimentation with deep neural networks. Keras provides a simple, user-friendly interface to create and train machine learning and deep learning models, making it ideal for both beginners and professionals.

Key Features of Keras:

- **User-Friendly**: Clear and concise APIs make model building, training, and evaluation easy.
- **Modular and Extensible**: Models are made by connecting configurable modules (layers, optimizers, activation functions, etc.).

- **Pythonic**: Keras uses Python as its primary language, making it easy to write, debug, and deploy models.
- **Backend Engine**: Typically runs on TensorFlow as the backend (Keras is officially part of TensorFlow as tf.keras).

Use of Keras in Multi-Domain Threat Detection Systems:

Multi-Domain Threat Detection System, Keras can be particularly useful for building deep learning models, especially for the Image Forgery Detection Module, where Convolutional Neural Networks (CNNs) are commonly used for image classification tasks. It can also be used for advanced spam detection or phishing classification tasks when deeper models are needed.

Benefits:

- **Phishing Detection**: Build complex models for URL feature classification.
- **Spam Email Detection**: Use deep neural networks to classify emails more effectively.
- **Image Forgery Detection**: Implement CNNs to analyse image patterns for manipulation.

Keras provides an efficient and intuitive way to build, train, and deploy deep learning models, making it a great tool for implementing the AI-driven parts of your Multi-Domain Threat Detection System.

OS

The os module in Python is a built-in standard library that provides a way to interact with the operating system in a portable and system-independent manner. It allows you to perform tasks such as navigating the file system, handling directories, managing files, and accessing environment variables. It’s especially useful in applications that require file management, system commands, or platform-specific operations.

Use of os in Multi-Domain Threat Detection System:

- Automatically handle files uploaded for image forgery detection.
- Navigate folders containing datasets for spam detection or phishing URLs.
- Set and retrieve configuration settings using environment variables.
- Organize and maintain log files or model output files.

The os module is a fundamental part of writing cross-platform Python programs, especially those that interact with the file system or require automation of system-level tasks. It's an essential utility in any machine learning pipeline, including your Multi-Domain Threat Detection System.

pathlib

pathlib is a modern Python module introduced in Python 3.4 that provides an object-oriented interface for working with file system paths. It replaces older modules like os.path and offers a more intuitive and readable way to handle file and directory paths.

Multi-Domain Threat Detection System, where you manage files like datasets, models, logs, or uploaded images (for forgery detection), pathlib simplifies file operations and improves code clarity.

Benefits:

- **Image Forgery Detection**: Easily locate and read image files.
- **Model Management**: Save and load ML models using clear file paths.
- **Dataset Handling**: Organize training and testing datasets with clean code.
- **Cross-platform Compatibility**: Works consistently across Windows, macOS, and Linux.

Function	Purpose
Path()	Object-oriented path creation
.write_text()	Store data in files (e.g., logs, user inputs)
.read_text()	Load data from files (e.g., results, configs)
.glob('*.png')	Locate all images for forgery detection

pathlib modernizes file and path handling in Python, making your Multi-Domain Threat Detection System more maintainable, clean, and efficient.

sqlite3

sqlite3 is a built-in Python library that provides a lightweight and efficient way to work with SQLite databases. It allows developers to create, manage, and interact with databases directly from Python without the need for installing or configuring any external database server. This makes it particularly suitable for small to medium-sized applications such as the Multi-Domain Threat Detection System. Since SQLite stores data in a single file, it is highly portable and easy to manage, which is ideal for systems that need to log phishing attempts, spam detection results, image forgery analysis, and user interactions like complaints or reviews.

The library supports standard SQL queries, enabling developers to create tables, insert records, retrieve data, and perform updates and deletions using familiar syntax. In the context of the Multi-Domain Threat Detection System, sqlite3 can be used to maintain user credentials, feedback, system logs, and detection history in an organized and secure manner. The ease of use and minimal setup make it accessible to both beginner and advanced developers.

Additionally, using parameterized queries with sqlite3 helps in preventing SQL injection attacks, enhancing the security of the system. Overall, sqlite3 offers a reliable, integrated, and straightforward solution for local data storage, making it an essential component of the system's backend.

Shutil

shutil is a powerful built-in Python module used for high-level file operations, making it essential for managing files and directories efficiently within a system. It provides functionality to perform operations such as copying, moving, renaming, and deleting files and folders, which are often required in applications involving file management or processing. In the context of a system like the Multi-Domain Threat Detection System, shutil can be particularly useful in the Image Forgery Detection Module.

For instance, when users upload images for analysis, shutil can be used to move these images to specific directories for temporary storage, processing, or archiving. It can also help in organizing input and output files systematically and cleaning up old or unnecessary data to optimize storage. One of the major advantages of using shutil is its simplicity and ability to work across different operating systems, which ensures cross-platform compatibility. Functions like shutil.copy(), shutil.move(), and shutil.rmtree() simplify file handling and contribute to building a more robust and efficient backend. Therefore, shutil plays a crucial role in handling file-level operations seamlessly within Python applications.

BeautifulSoup

BeautifulSoup is a Python library used for parsing HTML and XML documents, making it an essential tool for web scraping tasks. In the context of the Multi-Domain Threat Detection System, BeautifulSoup plays a key role in phishing website detection. By parsing the HTML structure of a website, it helps identify suspicious elements like misleading links, hidden forms, or abnormal behaviours that are commonly found in phishing attacks.

The library enables the extraction of metadata, such as title tags and meta descriptions, which can be compared against known phishing patterns. Additionally, it allows for the analysis of links and form inputs on a page, helping to identify potentially malicious redirects or forms designed to steal sensitive information. BeautifulSoup is often used alongside other libraries like requests to retrieve the HTML content of a webpage, which can then be parsed for further analysis. It can also work in conjunction with whois to check the domain registration details, adding an extra layer of verification. With its simple interface and ability to handle poorly structured HTML, BeautifulSoup proves to be a valuable tool in the detection and analysis of phishing websites, contributing to the accuracy and efficiency of the Multi-Domain Threat Detection System.

PorterStemmer

The PorterStemmer in the Natural Language Toolkit (NLTK) is one of the most popular algorithms for stemming words, an essential step in text normalization for various natural language processing (NLP) tasks. Stemming is the process of reducing words to their base or root form by systematically removing prefixes, suffixes, and inflectional endings. This process helps standardize different grammatical forms or variations of a word, allowing them to be treated as a single entity during analysis. The PorterStemmer specifically implements the Porter Stemming Algorithm, developed by Martin Porter, which uses a series of linguistic rules and conditions to transform complex word forms into their stems. For example, it converts words such as "running," "runner," and "ran" to the simple root "run."

This normalization is crucial in tasks like text classification, sentiment analysis, information retrieval, and spam detection, where identifying semantic similarity between words improves system accuracy.

In the context of the Multi-Domain Threat Detection System, the PorterStemmer plays a vital role in enhancing the system's ability to process and interpret textual data from various sources like email contents, URLs, and image descriptions. By applying stemming to these textual elements, the system can better recognize patterns, detect anomalies, and group related threats more effectively. For instance, phishing emails often use slightly varied language to evade detection; stemming helps unify these variations under a common representation, making it easier to spot suspicious patterns. Similarly, in spam filtering, reducing different word forms to a single root allows for more robust and generalized learning models, thereby increasing classification accuracy. Overall, integrating the PorterStemmer into the Multi-Domain Threat Detection System streamlines text preprocessing, reduces data dimensionality, and significantly improves the system's precision and efficiency in identifying cyber threats across multiple domains.

SYSTEM TESTING AND IMPLEMENTATION

8. SYSTEM TESTING AND IMPLEMENTATION

Testing is the vital to the success of the system. It makes a logical assumption that ifall the parts of the system are correct, the goal will be successfully achieved in this project.It is the stage of implementation, which ensures that system works accurately and effectively before the live operation commences.

8.1 Types of Testing

Unit testing

In the Multi-Domain Threat Detection System, unit testing plays a crucial role in ensuring the accuracy, reliability, and robustness of each module independently. The system is divided into multiple components, such as the Phishing Detection Module, Spam Email Detection Module, and Image Forgery Detection Module, each of which contains its own set of functions and machine learning models. Unit testing is used to verify the correct behavior of these individual components, such as URL analysis functions, email content classification, and image tampering detection algorithms, by isolating them from the rest of the system.

Using frameworks like pytest in Python, developers can simulate specific inputs, such as phishing URLs or spam email samples, and confirm that the outputs match the expected classifications. This ensures that errors are caught early during development, improves code maintainability, and provides confidence that updates or model retraining efforts will not inadvertently break existing functionalities. Through systematic and continuous unit testing, the Multi-Domain Threat Detection System can deliver higher accuracy, faster debugging, and greater overall system stability, even as the threat landscape evolves.

Black box testing

Black box testing is a powerful software testing methodology that focuses on evaluating the functionality of an application without requiring knowledge of its internal code structure or implementation details. In this approach, the tester is concerned only with the inputs and the expected outputs not how the software arrives at those outputs. This makes black box testing an ideal method for assessing a system from the end-user's perspective. In black box testing, the software is treated as a "black box", meaning that the internal workings of the application are unknown to the tester. Instead, the focus is on verifying that the software behaves as expected under various conditions. The tester provides inputs, observes the outputs, and compares them against the expected results to identify discrepancies or defects. This technique is commonly used in several levels of software testing, including:

- **Functional testing**: Ensures that specific features work as defined by the requirements.
- **System testing**: Verifies the behaviour of the entire system as a whole.
- **Acceptance testing**: Validates that the software meets business requirements and is ready for delivery to the customer.

In a Multi-Domain Threat Detection System, black box testing is crucial for validating that the system accurately detects threats such as phishing websites, spam emails, and image forgery without delving into its internal code or logic. Testers evaluate the system solely based on the inputs (such as various URLs, email contents, and image files) and the outputs (whether threats are correctly identified or benign data is properly allowed), without any knowledge of how the detection algorithms work internally.

White box testing

White box testing, also referred to as clear box testing, glass box testing, or structural testing, is a software testing technique that focuses on examining and validating the internal structure, logic, design, and source code of a software application. Unlike black box testing, which evaluates the system's behaviour without any knowledge of its internal workings, white box testing provides the tester with full access to the implementation details. Testers design test cases based on the internal paths of the program, ensuring that each statement, condition, loop, and branch is exercised and functions correctly. This method helps identify logical errors, unreachable code, boundary issues, memory leaks, and security vulnerabilities that are not easily detected through external testing methods, Common techniques used in white box testing include statement coverage, branch coverage, path coverage, condition coverage, and loop testing. These approaches ensure comprehensive testing of the codebase by checking that all possible execution paths and logic flows are tested.

White box testing offers several advantages, such as early bug detection, improved code optimization, better performance, and enhanced security, particularly because it allows the identification of issues within the code itself. However, it also has some limitations, including the need for testers to have strong programming knowledge, the potential for time-consuming test creation for large and complex systems, and the maintenance burden when code changes necessitate updates to the test cases.

white box testing plays a crucial role in ensuring the accuracy, efficiency, and security of each module phishing detection, spam email detection, and image forgery detection. Since the system heavily relies on machine learning algorithms, deep white box testing allows developers to validate the logic within the classifiers, the data preprocessing functions, and the decision-making flows. For instance, during phishing detection, white box testing can be used to ensure that URL parsing functions correctly handle edge cases, such as malformed URLs or obfuscated domains.

8.3 Implementation

Implementation is a critical phase in the software development life cycle where the theoretical and conceptual design of a system is translated into a fully functional and operational application. It marks the transition from planning and design to actual usage, ensuring that the new system is correctly integrated into its intended environment. This stage plays a vital role in determining the success of the project, as it not only involves deploying the system but also add gradually confidence among users that the new solution will perform efficiently and reliably. A successful implementation requires detailed planning, thorough analysis of the existing manual or legacy systems, and meticulous execution to ensure a smooth transition.

Implementation activities typically include the installation of hardware and software, data migration, system testing, training of users, and the actual switchover from the old system to the new one. It may also involve parallel running of both systems for a certain period to ensure stability and accuracy. Adequate user training and documentation are essential at this stage to facilitate ease of use and promote acceptance of the system. Troubleshooting and post-implementation support are also part of this phase, addressing any issues that arise and ensuring the system operates as intended. Ultimately, implementation is more than just setting up the software it’s about transforming plans into real-world functionality, ensuring minimal disruption, and delivering a system that meets user expectations and business goals.

8.4 Algorithm

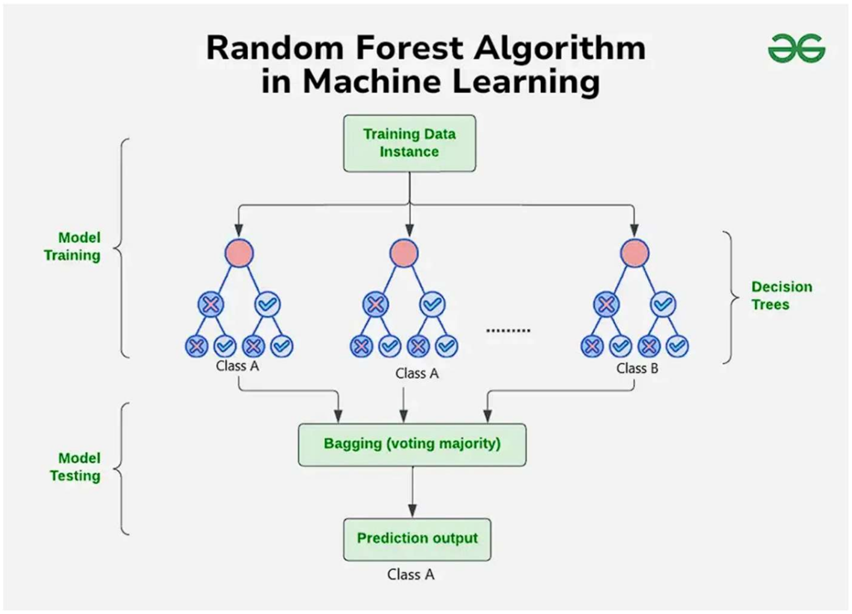
In the Multi-Domain Threat Detection System, both Convolutional Neural Networks (CNNs) and Random Forest algorithms play key roles in delivering accurate and efficient threat detection across different domains namely phishing websites, spam emails, and image forgery detection.

8.4.1 Random Forest algorithm

Random Forest algorithm is a powerful tree learning technique in Machine Learning to make predictions and then we do vote of all the tress to make prediction. They are widely used for classification and regression task.

- It is a type of classifier that uses many decision trees to make predictions.
- It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. This approach helps improve the accuracy of predictions. Random Forest is based on ensemble learning.

Imagine asking a group of friends for advice on where to go for vacation. Each friend gives their recommendation based on their unique perspective and preferences (decision trees trained on different subsets of data). You then make your final decision by considering the majority opinion or averaging their suggestions (ensemble prediction).



Process starts with a dataset with rows and their corresponding class labels (columns).

- Then - Multiple Decision Trees are created from the training data. Each tree is trained on a random subset of the data (with replacement) and a random subset of features. This process is known as bagging or bootstrap aggregating.
- Each Decision Tree in the ensemble learns to make predictions independently.
- When presented with a new, unseen instance, each Decision Tree in the ensemble makes a prediction.

The final prediction is made by combining the predictions of all the Decision Trees. This is typically done through a majority vote (for classification) or averaging (for regression).

Key Features of Random Forest:

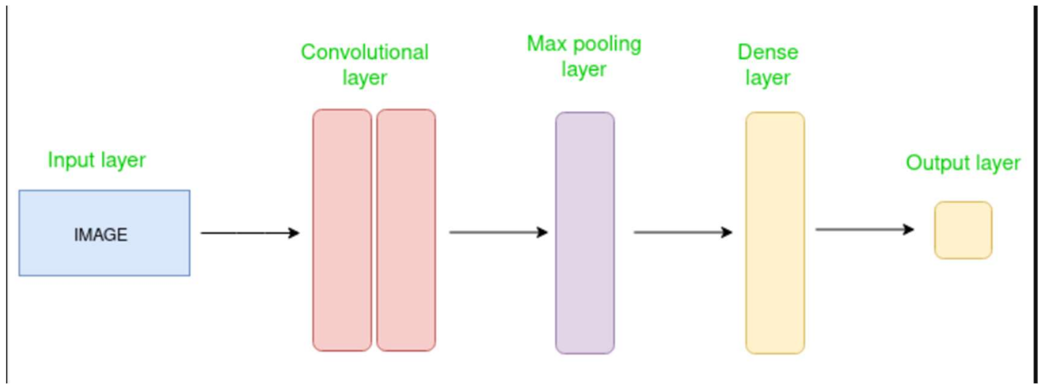
- **Handles Missing Data**: Automatically handles missing values during training, eliminating the need for manual imputation.
- Algorithm ranks features based on their importance in making predictions offering valuable insights for feature selection and interpretability.
- Scales Well with Large and Complex Data without significant performance degradation.
- Algorithm is versatile and can be applied to both classification tasks (e.g., predicting categories) and regression tasks (e.g., predicting continuous values).

The random Forest algorithm works in several steps:

- Random Forest builds multiple decision trees using random samples of the data. Each tree is trained on a different subset of the data which makes each tree unique.
- When creating each tree, the algorithm randomly selects a subset of features or variables to split the data rather than using all available features at a time. This adds diversity to the trees.
- Each decision tree in the forest makes a prediction based on the data it was trained on. When making final prediction random forest combines the results from all the trees.
- For classification tasks the final prediction is decided by a majority vote. This means that the category predicted by most trees is the final prediction. For regression tasks the final prediction is the average of the predictions from all the trees.
- The randomness in data samples and feature selection helps to prevent the model from overfitting making the predictions more accurate and reliable.

8.4.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is an advanced version of artificial neural networks (ANNs), primarily designed to extract features from grid-like matrix datasets. This is particularly useful for visual datasets such as images or videos, where data patterns play a crucial role. CNNs are widely used in computer vision applications due to their effectiveness in processing visual data. CNNs consist of multiple layers like the input layer, Convolutional layer, pooling layer, and fully connected layers. Let’s learn more about CNNs in detail.



Simple CNN architecture

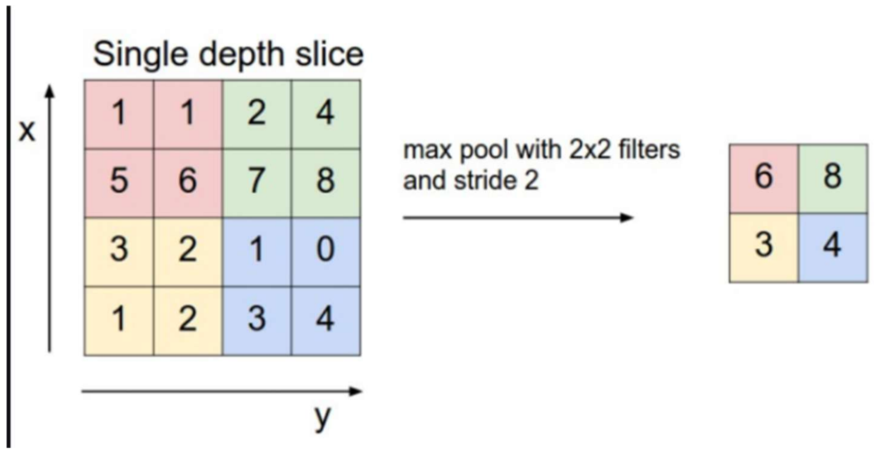
Layers Used to Build ConvNets

A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

Let’s take an example by running a covnets on of image of dimension 32 x 32 x 3.

- **Input Layers:** It’s the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.
- **Convolutional Layers:** This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we’ll get an output volume of dimension 32 x 32 x 12.

- **Activation Layer**: By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max (0, x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.
- **Pooling layer**: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.



- **Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
- **Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.
- **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

SYSTEM MAINTENANCE

9. SYSTEM MAINTENANCE

Maintenance is making adaptation of the software for external changes (requirements changes or enhancements) and internal changes (fixing bugs). When changes are made during the maintenance phase all preceding steps of the model must be revisited.
There are 3 types of maintenance:

- Corrective (Fixing bugs/errors)
- Adaptive (Updates due to environment changes)
- Perfective (Enhancements, requirements changes)

Maintenance is a crucial and inevitable phase of the software development lifecycle, often regarded as the ongoing process that ensures a system remains functional, efficient, and relevant after it has been deployed. It encompasses a set of activities that are performed post-release to address various issues and adapt to evolving requirements. One of the primary reasons for maintenance is the recognition that it is unrealistic to expect all defects to be identified and resolved during the initial testing phase, especially in large and complex systems. Even after thorough testing, certain bugs or inconsistencies may only surface during real-world usage, necessitating corrective maintenance.

Another significant driver of software maintenance is the rapid pace of change in technology. As hardware platforms, operating systems, network environments, and third-party libraries evolve, the software must be updated to remain compatible and functional. This form of maintenance, often referred to as adaptive maintenance, ensures that the system continues to operate smoothly in a changing technological landscape. In addition to fixing bugs and adapting to new environments, maintenance also involves enhancing the software’s capabilities. As users interact with the system, they may suggest new features, request modifications to existing functionalities, or highlight areas where performance can be improved. This leads to perfective maintenance, which focuses on refining and expanding the system’s operations based on user feedback and emerging needs.

The final aspect of maintenance is aimed at improving the software’s long-term maintainability and reliability. This preventive maintenance includes activities like code refactoring, documentation updates, and optimization, all of which contribute to making the system easier to manage, scale, and debug in the future.

FUTURE ENHANCEMENT

10. FUTURE ENHANCEMENT

The Multi-Domain Threat Detection System provides a strong foundation for identifying phishing websites, spam emails, and image forgery. However, to keep pace with evolving cyber threats and technological advancements, several future enhancements can improve the system's accuracy, efficiency, and user experience. Below are some potential areas for future development:

1. Enhanced Machine Learning Models
   - Integrate an ensemble learning approach that combines multiple algorithms to improve prediction accuracy and reduce false positives/negatives.
2. Real-Time Detection and Alerts
   - Develop a real-time monitoring system that continuously scans user inputs (URLs, emails, and images) and triggers instant alerts if a threat is detected.
   - Implement email and SMS notifications to alert users and administrators about potential threats immediately.
3. User Behaviour Analysis
   - Incorporate user behaviour analytics to identify suspicious patterns in user activity, which can help in detecting advanced persistent threats (APTs) and unusual user behaviour.
   - Use anomaly detection algorithms to flag uncommon patterns that might indicate new forms of phishing or spam.
4. Automated Model Updates
   - Implement an auto-update mechanism to retrain and improve machine learning models regularly using the latest threat intelligence data.
   - Allow integration with external threat databases to keep the system up-to-date with emerging phishing sites and email patterns.
5. Improved Image Forgery Detection
   - Incorporate metadata analysis and digital watermark detection to enhance image forgery identification.
   - Use blockchain technology for verifying and maintaining the authenticity of images over time.

6. Mobile Application
   - Develop a mobile version of the system using Flutter, allowing users to detect threats directly from their smartphones.

   - Enable users to scan QR codes to verify the legitimacy of URLs and detect phishing links on the go.
7. Advanced Reporting and Analytics
   - Provide detailed reports and visual dashboards for users and administrators to track detected threats and analyse system performance.
   - Implement exportable reports in formats like PDF or Excel for further analysis and record-keeping.
8. Integration with Cybersecurity Platforms
   - Enable API-based integration with third-party cybersecurity systems, such as firewalls, email gateways, and threat intelligence platforms for comprehensive protection.
   - Implement support for browser extensions to detect phishing attempts while users browse the internet.

By implementing these enhancements, the Multi-Domain Threat Detection System will become more robust, adaptable, and capable of addressing future cybersecurity challenges while delivering a seamless user experience.

CONCLUSION

11. CONCLUSION

The Multi-Domain Threat Detection System effectively demonstrates the application of machine learning algorithms in identifying and mitigating modern cyber threats across three critical domains: phishing detection, spam email classification, and image forgery detection. This system provides a comprehensive and automated solution to address the increasing risks associated with digital security breaches. By integrating advanced techniques such as the Random Forest algorithm for phishing detection and other machine learning models for spam filtering and image analysis, the system enhances the accuracy and efficiency of threat detection.

The system's dual-module approach allows for seamless interaction between users and administrators. Users can easily submit URLs, emails, and images for analysis, while administrators can manage user requests, monitor system performance, and respond to complaints. The use of Django for the front end ensures a user-friendly, cross-platform experience, while Python and MySQL provide a robust and scalable back-end infrastructure capable of handling real-time data analysis. Through this project, it is evident that machine learning is a powerful tool in cybersecurity, offering faster and more reliable detection of digital threats. The system’s ability to automate threat detection minimizes human error and increases response time, ultimately improving user safety. Furthermore, its scalability and adaptability ensure that it can accommodate future technological advancements and evolving cyber threats.

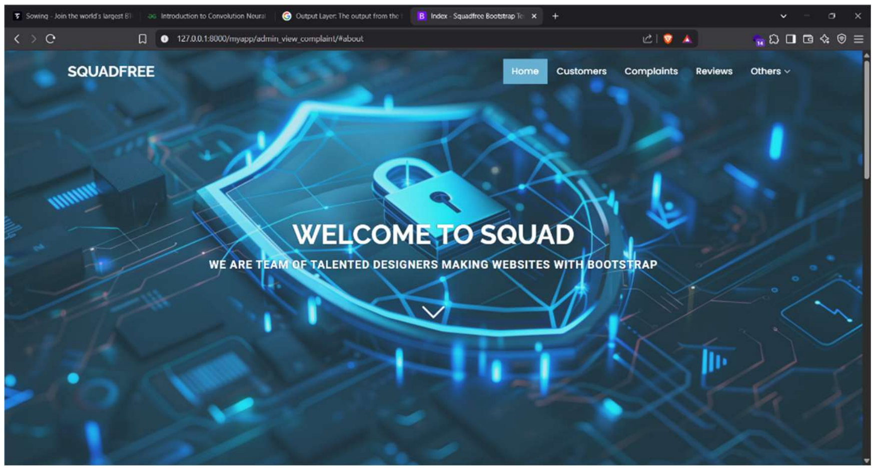
In conclusion, the Multi-Domain Threat Detection System is a cost-effective, scalable, and user-friendly solution for modern cybersecurity challenges. It plays a crucial role in safeguarding users from malicious websites, fraudulent emails, and tampered images. This system not only addresses present cybersecurity concerns but also lays the groundwork for future developments in automated threat detection and prevention.

APPENDIX

12.APPENDIX

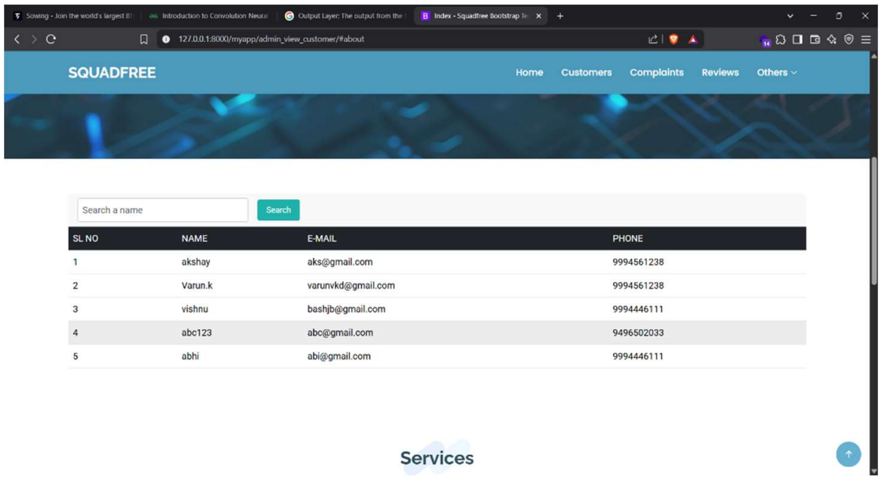
Admin Module:

Home page:


127.0.0.1:8000/myapp/admin_view_complaint/#about
SQUADFREE
Home
Customers
Complaints
Reviews
Others
WELCOME TO SQUAD
WE ARE TEAM OF TALENTED DESIGNERS MAKING WEBSITES WITH BOOTSTRAP

Home page

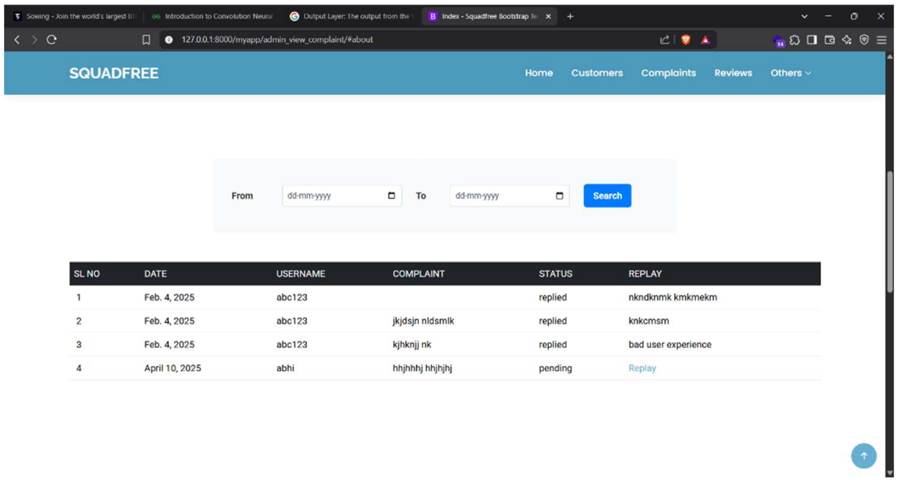
View User:



SL NO	NAME	E-MAIL	PHONE
1	akshay	aks@gmail.com	9994561238
2	Varun.k	varunvkd@gmail.com	9994561238
3	vishnu	bashjb@gmail.com	9994446111
4	abc123	abc@gmail.com	9496502033
5	abhi	abi@gmail.com	9994446111

View User

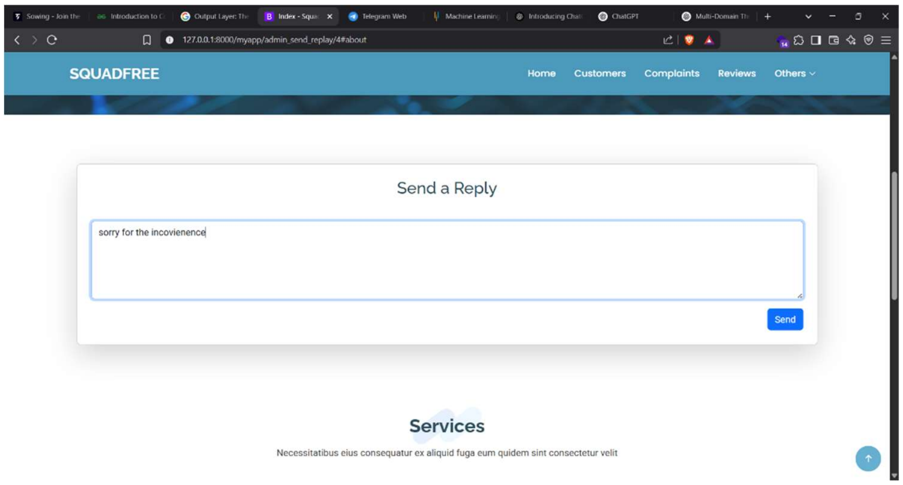
View Complaint:



SL NO	DATE	USERNAME	COMPLAINT	STATUS	REPLAY
1	Feb. 4, 2025	abc123		replied	nkndknmk kmkmekm
2	Feb. 4, 2025	abc123	jkjdsjn nldsmlk	replied	knkcmsm
3	Feb. 4, 2025	abc123	kjhknjj nk	replied	bad user experience
4	April 10, 2025	abhi	hhjhhhj hhjhjhj	pending	Replay

View Complaint

Send Replay:

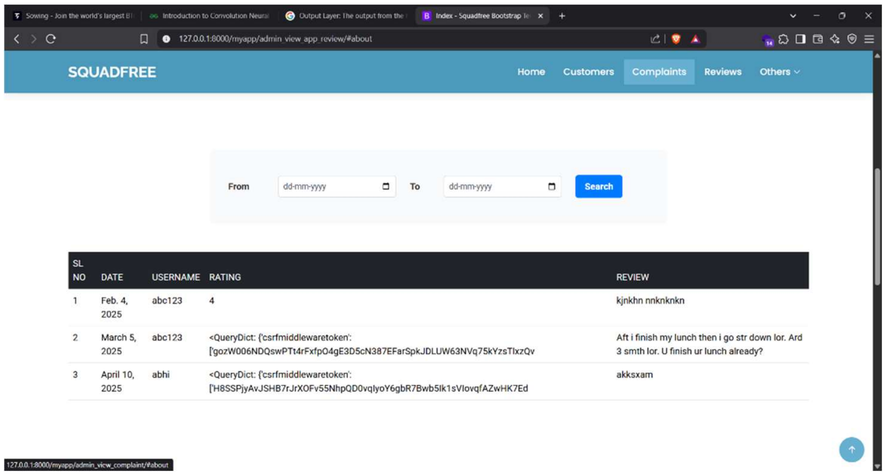


sorry for the incovienence

Necessitatibus eius consequatur ex aliquid fuga eum quidem sint consectetur velit

Send Replay

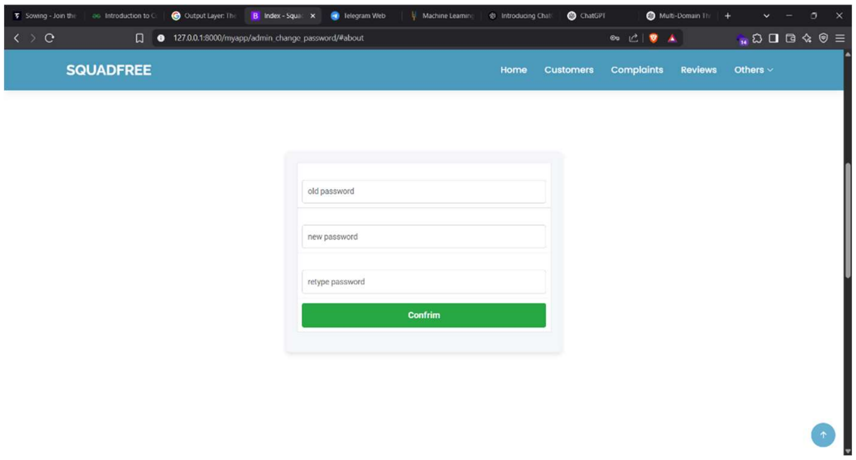
View Review:


127.0.0.1:8000/myapp/admin_view_app_review/#about
SQUADFREE
Home
Customers
Complaints
Reviews
Others
From
dd-mm-yyyy
To
dd-mm-yyyy
Search
127.0.0.1:8000/myapp/admin_view_complaint/#about

SL NO	DATE	USERNAME	RATING	REVIEW
1	Feb. 4, 2025	abc123	4	kjnkhn nnknknkn
2	March 5, 2025	abc123	<QueryDict: {'csrfmiddlewaretoken': ['gozW006NDQswPTt4rFxfpO4gE3D5cN387EFarSpkJDLUW63NVq75kYzsTlxzQv	Aft i finish my lunch then i go str down lor. Ard 3 smth lor. U finish ur lunch already?
3	April 10, 2025	abhi	<QueryDict: {'csrfmiddlewaretoken': ['H8SSPjyAvJSHB7rJrXOFv55NhpQD0vqIyoY6gbR7Bwb5Ik1sVlovqfAZwHK7Ed	akksxam

View Review

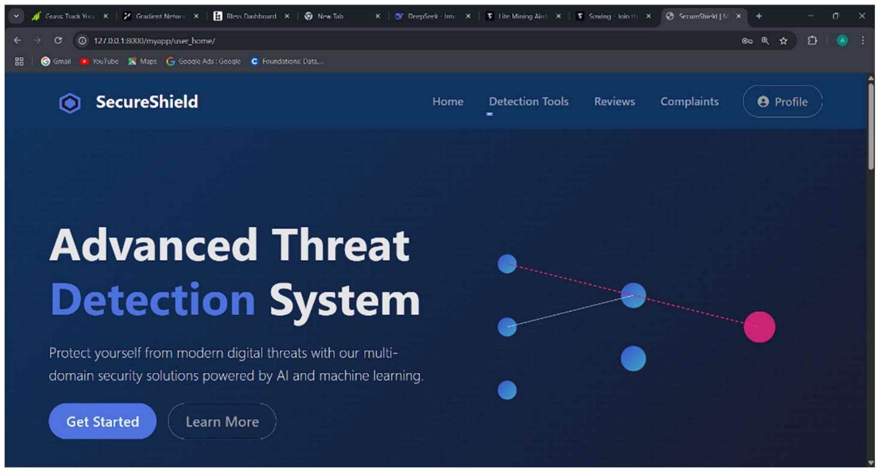
Change Password:



Change Password

User Module:

Home page:

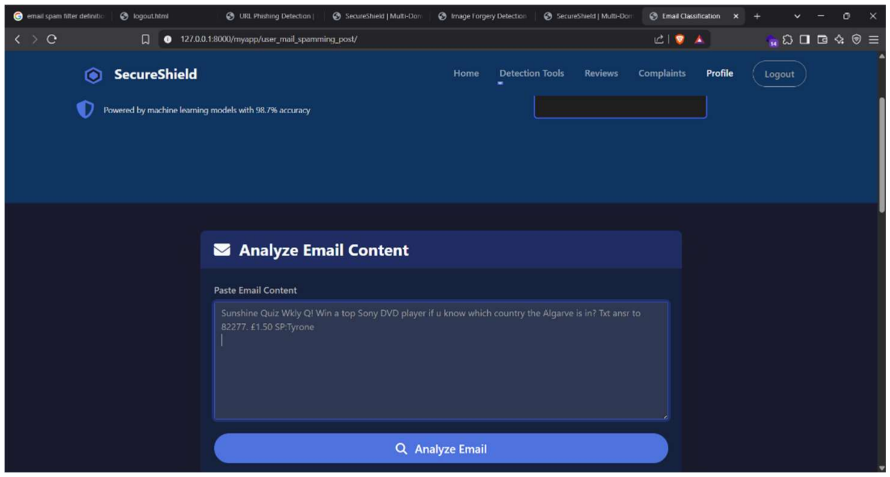


Advanced Threat Detection System

Protect yourself from modern digital threats with our multi-domain security solutions powered by AI and machine learning.

Home page

Email Classification:

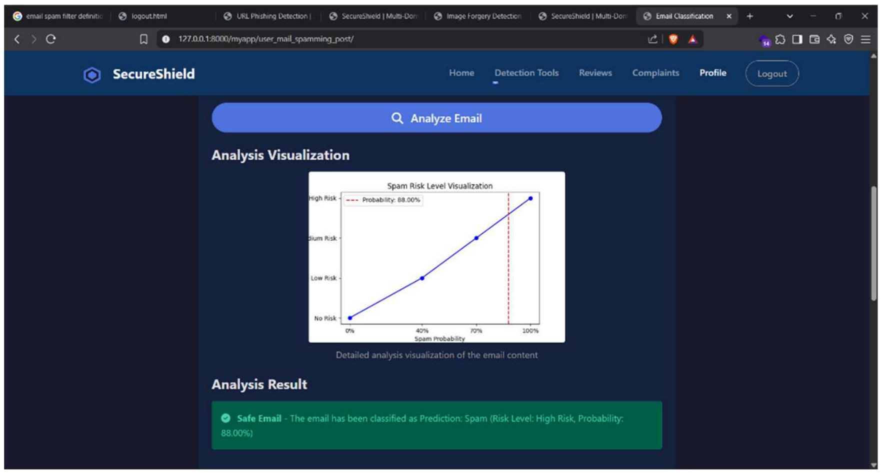


Powered by machine learning models with 98.7% accuracy

Paste Email Content

Sunshine Quiz Wkly Q! Win a top Sony DVD player if u know which country the Algarve is in? Txt ansr to 82277. £1.50 SP:Tyrone

Analyzing Email Content



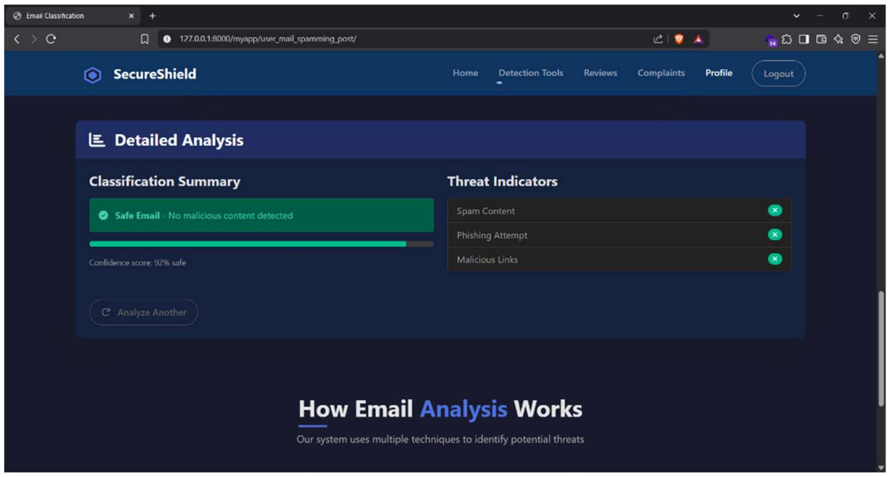
Analysis Visualization

Detailed analysis visualization of the email content

Analysis Result

Safe Email - The email has been classified as Prediction: Spam (Risk Level: High Risk, Probability: 88.00%)

Graphical representation



Detailed Analysis

Classification Summary

Safe Email - No malicious content detected

Confidence score: 92% safe

Threat Indicators

Spam Content

Phishing Attempt

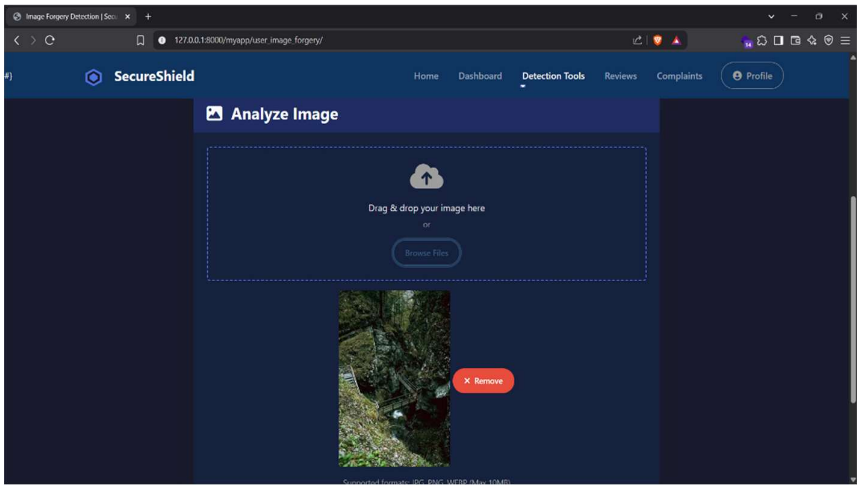
Malicious Links

How Email Analysis Works

Our system uses multiple techniques to identify potential threats

Analysis Result

Image Forgery Detection:



Analyze Image

Drag & drop your image here

or

Supported formats: JPG, PNG, WEBP (Max 10MB)

Uploading image

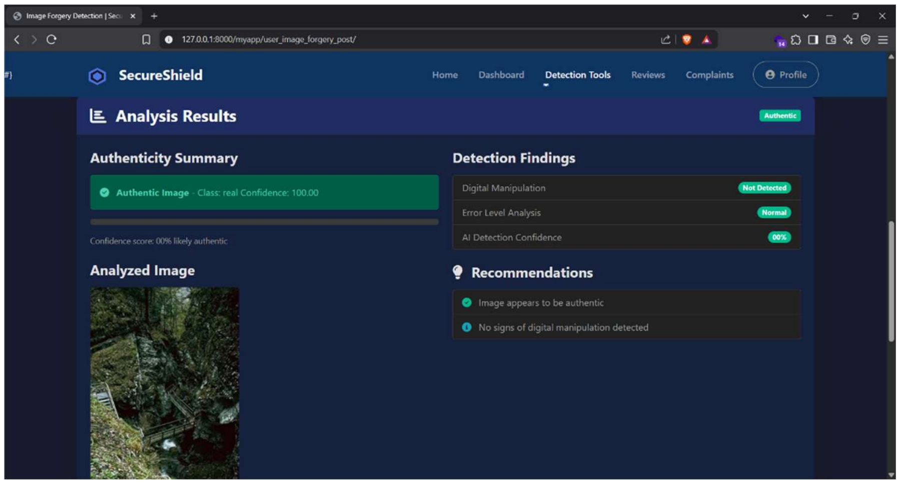


Image Forgery Detection | Secu
127.0.0.1:8000/myapp/user_image_forgery_post/
SecureShield
Home
Dashboard
Detection Tools
Reviews
Complaints
Profile

Analysis Results

Authentic

Authenticity Summary

Authentic Image - Class: real Confidence: 100.00

Confidence score: 00% likely authentic

Analyzed Image

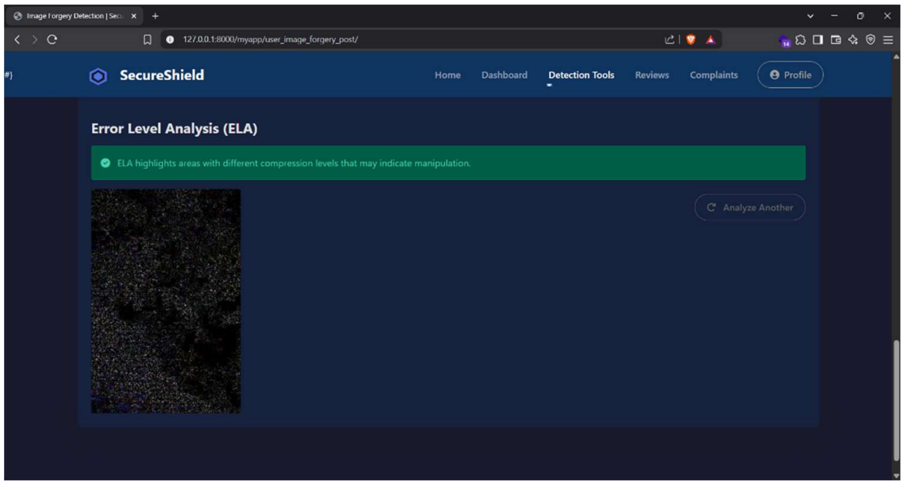
Detection Findings

Digital Manipulation	Not Detected
Error Level Analysis	Normal
AI Detection Confidence	00%

Recommendations

- Image appears to be authentic
- No signs of digital manipulation detected

Image Forgery Result

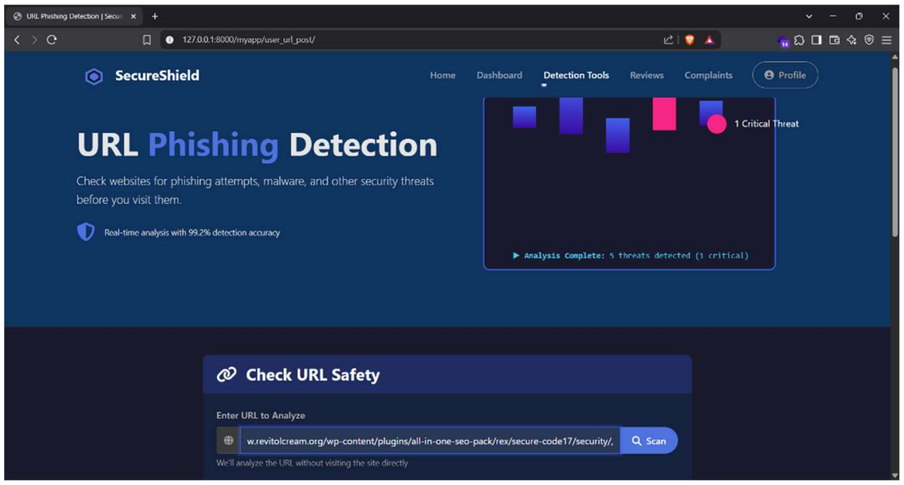


Error Level Analysis (ELA)

ELA highlights areas with different compression levels that may indicate manipulation.

Resulting Effected areas

URL Phishing Detection:



URL Phishing Detection

Check websites for phishing attempts, malware, and other security threats before you visit them.

Real-time analysis with 99.2% detection accuracy

```
▶ Analysis Complete: 5 threats detected (1 critical)
```

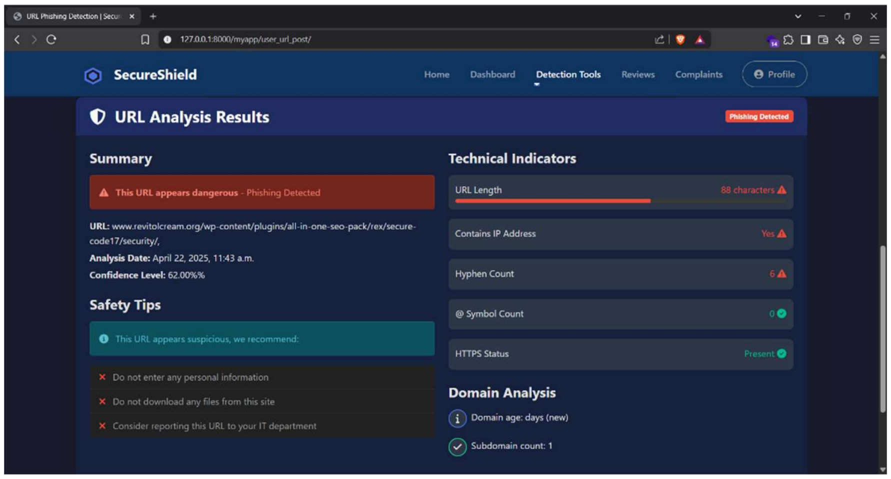
Check URL Safety

Enter URL to Analyze

w.revitolcream.org/wp-content/plugins/all-in-one-seo-pack/rex/secure-code17/security/,

We'll analyze the URL without visiting the site directly

URL Phishing Detection


URL Phishing Detection | Secur
127.0.0.1:8000/myapp/user_url_post/
SecureShield
Home
Dashboard
Detection Tools
Reviews
Complaints
Profile

URL Analysis Results Phishing Detected

Summary

This URL appears dangerous - Phishing Detected

URL: www.revitolcream.org/wp-content/plugins/all-in-one-seo-pack/rex/secure-code17/security/,
Analysis Date: April 22, 2025, 11:43 a.m.
Confidence Level: 62.00%%

Safety Tips

This URL appears suspicious, we recommend:

- Do not enter any personal information
- Do not download any files from this site
- Consider reporting this URL to your IT department

Technical Indicators

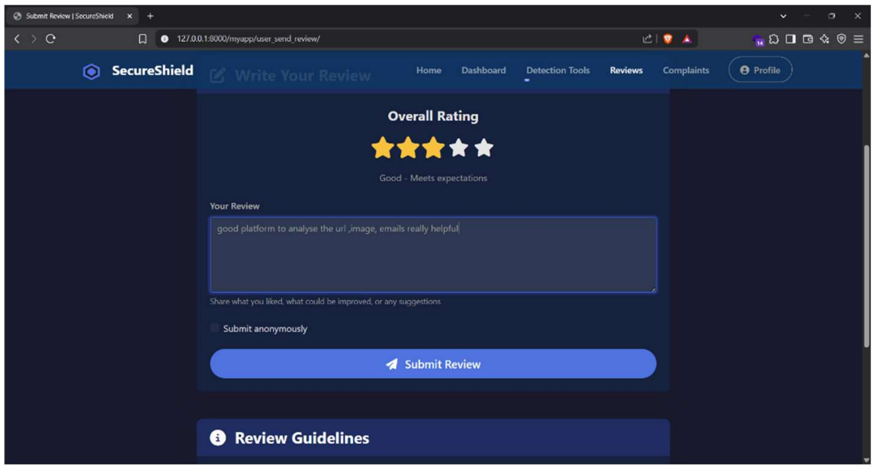
URL Length	88 characters
Contains IP Address	Yes
Hyphen Count	6
@ Symbol Count	0
HTTPS Status	Present

Domain Analysis

- Domain age: days (new)
- Subdomain count: 1

Phishing Detection result

Review:



Overall Rating

★★★☆☆

Good - Meets expectations

Your Review

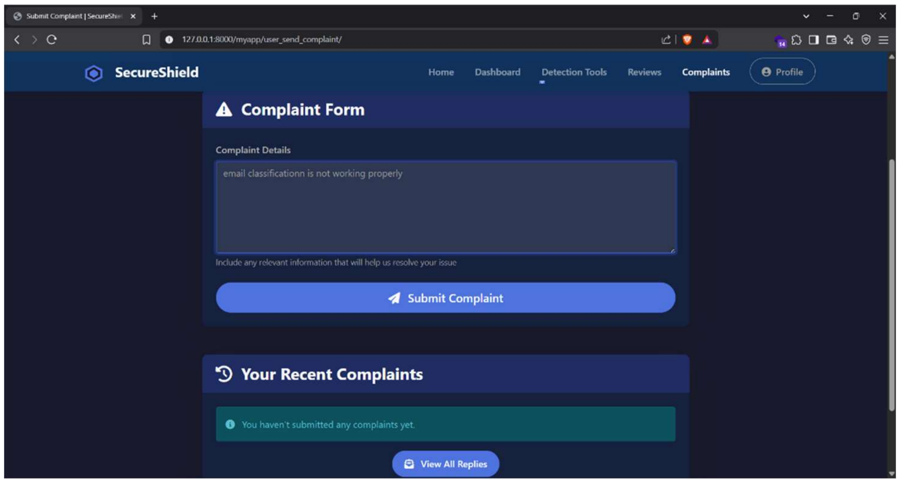
good platform to analyse the url ,image, emails really helpful

Share what you liked, what could be improved, or any suggestions

☐ Submit anonymously

Send Review

Complaint:



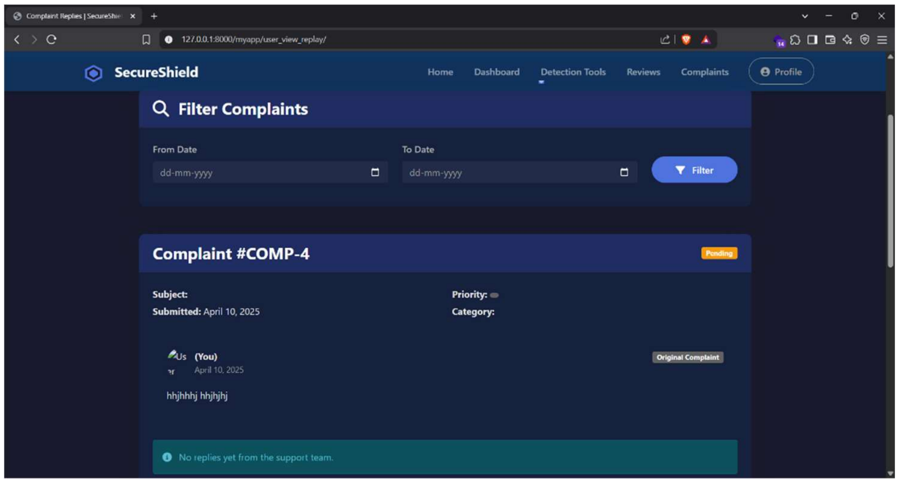
Complaint Details

email classificationn is not working properly

Include any relevant information that will help us resolve your issue

You haven't submitted any complaints yet.

Send Complaint


127.0.0.1:8000/myapp/user_view_replay/

SecureShield — Home | Dashboard | Detection Tools | Reviews | Complaints | Profile

Filter Complaints

From Date: dd-mm-yyyy — To Date: dd-mm-yyyy — Filter

Complaint #COMP-4 — Pending

Subject:
Submitted: April 10, 2025
Priority:
Category:

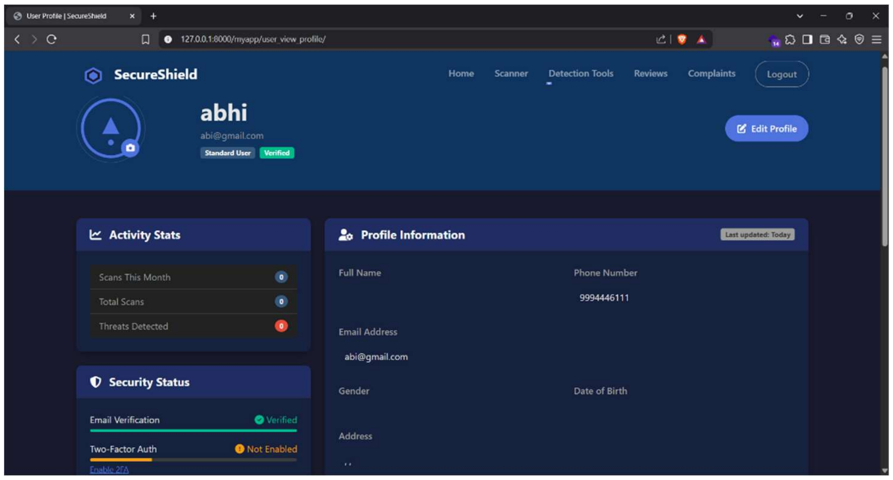
Us (You) — April 10, 2025 — Original Complaint

hhjhhhj hhjhjhj

No replies yet from the support team.

View Replay

Profile:


127.0.0.1:8000/myapp/user_view_profile/

SecureShield — Home | Scanner | Detection Tools | Reviews | Complaints | Logout

abhi
abi@gmail.com
Standard User — Verified
Edit Profile

Activity Stats

Scans This Month	0
Total Scans	0
Threats Detected	0

Security Status

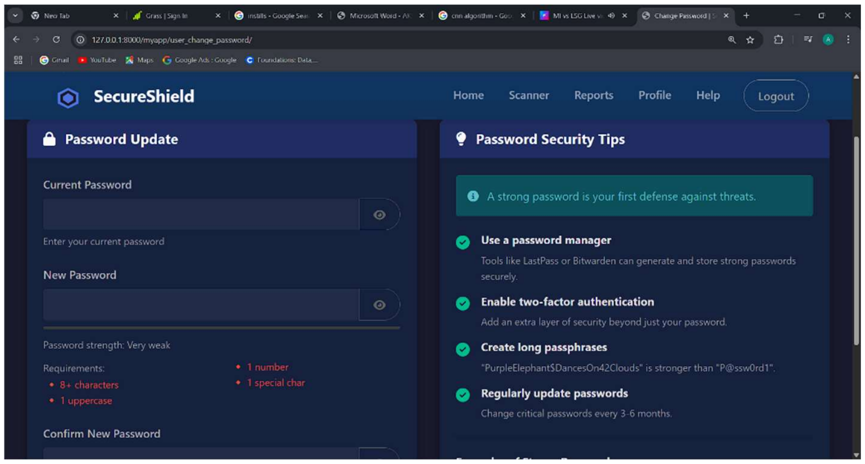
Email Verification — Verified
Two-Factor Auth — Not Enabled
Enable 2FA

Profile Information — Last updated: Today

Full Name	Phone Number
	9994446111
Email Address	
abi@gmail.com	
Gender	Date of Birth
Address	

View Profile

Change Password:


127.0.0.1:8000/myapp/user_change_password/

SecureShield — Home | Scanner | Reports | Profile | Help | Logout

Password Update

Current Password
Enter your current password

New Password
Password strength: Very weak

Requirements:
- 8+ characters
- 1 uppercase
- 1 number
- 1 special char

Confirm New Password

Password Security Tips

A strong password is your first defense against threats.

- **Use a password manager**
  Tools like LastPass or Bitwarden can generate and store strong passwords securely.
- **Enable two-factor authentication**
  Add an extra layer of security beyond just your password.
- **Create long passphrases**
  "PurpleElephant$DancesOn42Clouds" is stronger than "P@ssw0rd1".
- **Regularly update passwords**
  Change critical passwords every 3-6 months.

Change Password

BIBLIOGRAPHY

13. BIBLIOGRAPHY

Websites

[1] https://www.w3schools.com/python/python_conditions.asp

[2] https://docs.djangoproject.com/en/5.0/intro/tutorial01/#creating-a-project

[3] https://www.codecademy.com/catalog/language/html-css

[4] https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-sql-command.html

[5] https://www.w3schools.com/html/default.asp

Books

[1] Think Python: An Introduction to Software Design, Allen B. Downey, CreateSpace Independent Pub, 2011

[2] MySQL in a Nutshell, Russell J. T. Dyer, O'Reilly Media, 2008

[3] Mastering Django, Gnw Independent Publishing, Nigel George, 2020

[4] Code with Python, S. Chand's, S Chand and Company Ltd, 2023

[5] Mastering PyCharm, Quazi Nafiul Islam, Packt Publishing Limited, 2015