PHISHING DETECTION APPLICATION

A Major Project Report Submitted in Partial Fulfillment for the Award of the Degree of Bachelor of Technology in Computer Science and Engineering

Submitted to

Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

Submitted by:

Uday Raj Singh (2100100100182)

Zoya Ashiyam (2100100100198)

Md Shabi Ahmed (2100100100090)

Swati Kesarwani (2200100109023)

UNDER THE SUPERVISION OF

Mr. Amit Roy

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNITED COLLEGE OF ENGINEERING AND RESEARCH, PRAYAGRAJ

MAY 2025

DECLARATION

We, hereby certify that the project entitled "Phishing Detection Application" submitted by us in partial fulfillment of the requirement for the award of degree of the B. Tech. (Computer Science & Engineering) submitted to Dr. A.P.J. Abdul Kalam Technical University, Lucknow at United College of Engineering and Research, Prayagraj is an authentic record of our own work carried out during a period from June, 2024 to May, 2025 under the guidance of Mr. Amit Roy Assistant Professor of Computer Science and Engineering. The matter presented in this project has not formed the basis for the award of anyother degree, diploma, fellowship or any other similar titles.

Signature of the Student

Uday Raj Singh

Signature of the Student

Zoya Ashiyam

Signature of the Student

Md Shabi Ahmed

Signature of the Student

Swati Kesarwani

Place:

Date:

CERTIFICATE

This is to certify that the project titled “Phishing Detection Application” is the bonafide work carried out by (Uday Raj Singh, Zoya Ashiyam, Md Shabi Ahmed, Swati Kesarwani) & (2100100100182, 2100100100198, 2100100100090, 2200100109023) in partial fulfillment of the requirement for the award of degree of the B. Tech. (Computer Science & Engineering) submitted to Dr. A.P.J Abdul Kalam Technical University, Lucknow at United College of Engineering and Research, Prayagraj is an authentic record of their own work carried out duringa period from June, 2024 to May, 2025 under the guidance of Mr. Amit Roy Assistant Professor of Computer Science and Engineering. The Major Project Viva-Voce Examination has been held on __________________.

Signature of the Guide ________________________________

[Mr. Amit Roy]

Signature of Project Coordinator _______________________

[Mr. Shyam Bahadur Verma]

Signature of the Head of Department ____________________

[Dr. Vijay Kumar Dwivedi]

Place:

Date:

ABSTRACT

Phishing attacks remain one of the most widespread and damaging forms of cybercrime, threatening the security and privacy of individuals and organizations alike. This project aims to develop an advanced **Phishing Detection Application** designed to identify and mitigate phishing attempts in real-time, leveraging innovative technologies such as machine learning and natural language processing. By integrating a robust threat intelligence framework and employing behavioural analysis, the system enhances detection accuracy while adapting to the dynamic nature of phishing schemes. Through detailed risk assessments and proactive alerts, it empowers users with actionable insights to navigate digital communication securely. The proposed solution promises to make a significant contribution to the cybersecurity domain by fostering resilience against evolving cyber threats.

Phishing, a prevalent form of cyber-attack, poses severe risks to individuals and organizations by exploiting trust to steal sensitive information and cause financial and reputational harm. As phishing techniques grow increasingly sophisticated, there is an urgent need for advanced detection mechanisms that can adapt to evolving threats. This project presents a **Phishing Detection Application** that leverages cutting-edge machine learning algorithms and behavioural analysis to identify phishing attempts in real-time. By integrating threat intelligence databases and providing actionable insights, the system ensures robust protection against fraudulent activities while remaining user-friendly and scalable for diverse applications. This innovative solution represents a vital step toward safeguarding digital environments from cyber threats.

ACKNOWLEDGEMENT

We express our sincere gratitude to the Dr. A.P.J Abdul Kalam Technical University, Lucknow for giving us the opportunity to work on the Major Project during our final year of B.Tech. (CSE) is an important aspect in the field of engineering.

We would like to thank Dr. Swapnil Srivastava, Principal and Dr. Vijay Kumar Dwivedi, Head of Department, CSE at United College of Engineering and Research, Prayagraj for their kind support.

We also owe our sincerest gratitude towards Mr. Amit Roy for his valuable advice and healthy criticism throughout our project which helped us immensely to complete our work successfully.

We would also like to thank everyone who has knowingly and unknowingly helped us throughout our work. Last but not the least, a word of thanks for the authors of all those books and papers which we have consulted during our project work as well as for preparing the report.

Table of Contents

Introduction

In today's digital age, the threat of phishing has become a significant challenge for individuals and organizations alike. Phishing scams exploit human vulnerabilities and technical loopholes, causing substantial financial losses and compromising sensitive data. Our **Phishing Detection Application** aims to address this pressing issue by providing a robust, user-friendly solution designed to safeguard users against fraudulent attempts.
Utilizing advanced machine learning algorithms and real-time analysis, this application identifies and flags suspicious emails, links, and messages with remarkable accuracy. By equipping users with actionable insights and proactive warnings, it not only mitigates risks but also fosters a safer and more secure online environment. Whether you're a business looking to protect employees or an individual striving for digital safety, our application empowers users to stay ahead of cyber threats.

Definition

Phishing attacks have emerged as one of the most pervasive threats in the cybersecurity landscape. These malicious attempts exploit the trust of individuals and organizations, leading to the theft of sensitive information, financial losses, and reputational damage. Despite existing measures, many users and systems remain vulnerable due to the increasingly sophisticated tactics employed by cybercriminals.
The primary problem lies in the detection and prevention of phishing attempts in real-time. Conventional solutions often fail to accurately identify phishing content, especially when attackers leverage subtle deception techniques, such as well-disguised email addresses, fraudulent links, or social engineering methods. This necessitates the development of a robust and intelligent solution capable of proactively identifying and mitigating phishing threats with high accuracy while remaining user-friendly and adaptable to evolving threats.

PROBLEM DEFINITION

1.Nature of the Problem

Phishing is a malicious activity in which attackers deceive individuals or organizations to gain sensitive information such as passwords, financial details, or personal data. This problem has escalated due to the sophistication of phishing techniques, which can bypass traditional security measures like spam filters and firewalls.

2. Causes

- Increasing use of online platforms for communication and transactions.
- Lack of awareness among users about phishing tactics.
- Advanced techniques such as URL spoofing, email impersonation, and social engineering.
- Limited capability of existing security tools to detect dynamic phishing patterns.

3. Consequences

- Compromise of sensitive data leading to privacy breaches.
- Financial losses for individuals and businesses.
- Reputational damage to organizations targeted by phishing.
- Increased vulnerability to further cyber-attacks once a phishing attack succeeds.

4. Need for a Solution

Current anti-phishing solutions often fail to provide real-time detection or adapt to evolving threats. Therefore, an intelligent, adaptable system is needed to:

- Detect and block phishing attempts promptly.
- Educate users about recognizing phishing attempts.
- Protect sensitive information and mitigate risks of cyber fraud.

5. Goals of the Project

- Develop an innovative phishing detection system using machine learning and natural language processing techniques.
- Provide real-time alerts and actionable insights to users.
- Ensure user-friendly operation and scalability for individual and organizational use.

In this section we shall also discuss the limitation and drawback of the existing system that forced us to take up this project. Really that work was very typical to manage the daily errors free records and adding or removing any node from server. This problem produces a need to change the existing system. Some of these shortcomings are being discussed below: -

- Low Functionality

With the existing system, the biggest problem was the low functionality. The problem faced hampered the work. For small task like adding any new node to server or deleting a node or keeping daily record, we have to appoint minimum two or three employee.

- Erroneous Input and Output

In the existing system, humans performed all the tasks. As in the human tendency, error is also a possibility. Therefore, the inputs entered by the person who is working in the Company, in the registers may not be absolutely foolproof and may be erroneous. As a result of wrong input, the output reports etc. will also be wrong which would in turn affect the performance.

- Portability Problem

System that existed previously was manual. As a result, the system was less portable. One has to carry the loads of many registers to take the data from one place to another. A big problem was that the system was less flexible and if we wanted to calculate yearly or monthly maintenance report or efficiency report, then it was a big headache.

- Security

Security concerns were also one of the motives of the Company for the need of software. In the registers, the data is not secure as anybody can tamper with the data written in the registers. While in this software, just a password makes it absolutely secure from the reach of unauthorized persons.

- Data Redundancy

In the case of manual system, the registers are maintained in which, a lot of data is written.

- Processing Speed

In manual system maintaining a register and performing the necessary calculation has proved to be a troublesome job, which takes a lot of time and may affect the performance of the Company. But with this software we can have all the tasks performed in a fraction of second by a single click thus making the troublesome job much easier.

- Manual Errors

When a number of tough tasks are prepared by the humans like preparation of reports, performing long calculation then some human error are obvious due to a number of factors like mental strain, tiredness etc. But as we all know that computer never get tired irrespective of the amount of work it has to do. So this software can nullify the probability of manual error that improve the performance.

- Complexity in Work

In manual system whenever a record is to be updated or to be deleted a lot of cutting and overwriting needs to be done on the registers that are concerned that are deleted or updated record, which makes the work very complex.

Overview

System Overview:

- **Purpose**:
  To identify and prevent phishing attacks by analysing emails, URLs, and other digital communication channels.
- **Target Users**:
  Individuals, businesses, and organizations seeking protection from phishing threats.

Key Features:

1. **Real-Time Detection**:
   - Employs machine learning and natural language processing to analyse content instantly.
   - Flags suspicious links, email addresses, and attachments.
2. **Threat Database Integration**:
   - Maintains an up-to-date database of known phishing sites and patterns.
   - Automatically checks against the database during analysis.
3. **Behavioural Analysis**:
   - Monitors user interactions with links and prompts warnings for unusual or risky activities.
4. **User Education**:
   - Provides detailed alerts explaining why content was flagged as phishing.
   - Includes tutorials and tips for users to recognize phishing attempts.
5. **Customizability**:
   - Enables users to set personal sensitivity levels for detection.
   - Offers configurations for business-specific needs.

Technical Specifications:

- **Input**: Emails, URLs, and file attachments.
- **Output**: Phishing risk score, threat category, and warnings.
- **Machine Learning Models**: Pre-trained models fine-tuned for phishing detection.
- **Data Sources**:Integration with threat intelligence feeds, public cybersecurity databases, and proprietary data.
- **Platform Compatibility**: Cross-platform support (Windows, macOS, Linux, and mobile OS).

Security:

- **Encryption**: End-to-end encryption for analysed data to ensure privacy.
- **Authentication**: Multi-factor authentication for application access.
- **Data Retention**: Minimal and compliant with data protection regulations.

Performance:

- **Accuracy**: High precision in detecting phishing attempts.
- **Scalability**: Supports large-scale operations for enterprise use.

Hardware Specification

Hardware Requirements

For Client Side –

- **Processor**: Dual-core processor (e.g., Intel Core i3 or equivalent) for running lightweight detection algorithms.
- **Memory (RAM)**: 4GB or more for smooth operation of browser extensions or small applications.
- **Storage**: Minimal storage (e.g., 128GB SSD) for storing temporary data or logs.
- **Network Interface**: Standard Wi-Fi or Ethernet adapter for internet connectivity.
- **Operating System**: Windows, macOS, or Linux, depending on the user's device.

For Server Side –

- **Processor**: Multi-core server-grade processor (e.g., Intel Xeon or AMD EPYC) for handling high workloads.
- **Memory (RAM)**: At least 32GB for processing large datasets and running complex algorithms.
- **Storage**: High-capacity SSDs (e.g., 1TB or more) for storing phishing databases and logs.
- **Graphics Processing Unit (GPU)**: Dedicated GPUs (e.g., NVIDIA A100) for training deep learning models.
- **Network Interface**: High-speed network adapters for real-time data exchange.
- **Operating System**: Linux-based systems (e.g., Ubuntu Server) for stability and security.

Software Specification

Software Requirements

• Client on Internet: Web Browser, Operating System (any).

• Client on Intranet: Client Software, Web Browser, Operating System (any). • Web Server: IIS 7 or higher, Operating System (windows server).

• Application framework: Microsoft .NET Framework 4.0.

• Data Base Server: MS SQL SERVER 2008 R2, Operating System (windows Server).

• Development End: Visual Studio 2012 (ASP.NET, HTML, JavaScript), MS SQL SERVER 2008, OS (Windows Server), Web Server (IIS 6 or higher).

• Management studio 2012

LITERATURE SURVEY

Existing System

literature survey focusing on existing systems for phishing detection: --

Introduction

With the rise of phishing threats, various systems have been developed to detect and mitigate these cyberattacks. This survey explores the methodologies, strengths, and limitations of existing phishing detection systems to identify opportunities for improvement and innovation.

1. Existing Approaches

Rule-Based Systems:

- These systems rely on static rules, such as detecting suspicious keywords, mismatched URLs, or unusual email formatting.
- **Strengths**: Simple and quick to implement.
- **Limitations**: Ineffective against sophisticated or evolving phishing tactics.

Machine Learning-Based Systems:

- Models like Random Forest, Decision Trees, Support Vector Machines, and Naive Bayes are trained on datasets to classify emails or websites as phishing or legitimate.
- **Strengths**: Greater accuracy compared to static rules.
- **Limitations**: Dependent on the quality and size of training datasets, and prone to false positives**.**

Natural Language Processing (NLP)-Enhanced Systems:

- Analyse email content and website text for linguistic markers, sentiment, or deceptive patterns.
- **Strengths**: Ability to detect phishing attempts based on the tone and structure of language.
- **Limitations**: Struggles with language nuances and domain-specific terminologies.

Threat Intelligence Integration:

- Systems integrate with databases of known phishing URLs or blacklisted domains to enhance detection.
- **Strengths**: Effective against previously identified threats.
- **Limitations**: Less effective against new or zero-day phishing attacks.

Browser-Based Extensions:

- Browser add-ons flag suspicious links and warn users before they interact with potential phishing sites.
- **Strengths**: User-friendly and easily accessible.
- **Limitations**: Limited scope, as they only protect within the browser environment.

2. Strengths of Current Systems

- Real-time detection is implemented in many modern solutions.
- Advanced algorithms provide higher detection rates compared to older rule-based systems.
- User education modules in some systems help raise awareness.

3. Limitations and Challenges

- **Evasion Tactics**: Phishers employ obfuscation techniques, such as domain squatting or embedding malicious code in images.
- **False Positives**: Some systems erroneously flag legitimate emails or websites, which could disrupt workflows.
- **Adaptability**: Existing systems struggle to adapt to the constantly evolving phishing techniques.
- **Scalability**: Many solutions face challenges when deployed at enterprise scale, dealing with massive volumes of data.

4. Conclusion

- While significant progress has been made in phishing detection systems, there is still room for improvement, particularly in enhancing real-time detection accuracy, reducing false positives, and improving adaptability to sophisticated attacks. Hybrid approaches combining machine learning, NLP, and threat intelligence integration hold great promise for addressing the challenges of current systems.

Proposed System

Literature survey focusing on the **proposed system** for phishing detection:

Introduction

Phishing attacks continue to be a critical cybersecurity challenge, with increasingly sophisticated techniques used by attackers to deceive users. The proposed system aims to address the limitations of existing solutions by leveraging advanced technologies such as machine learning, natural language processing (NLP), and real-time threat intelligence. This survey reviews relevant research that informs the development and design of the proposed system.

1. Research on Machine Learning Models

- **Supervised Learning**: Studies demonstrate the effectiveness of algorithms like Random Forest, Support Vector Machines, and Logistic Regression for phishing detection. These models excel in classifying data when labeled datasets are available.
- **Deep Learning**: Recent research highlights the use of neural networks, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to identify complex phishing patterns. These models offer better adaptability to evolving threats.
- **Proposed Integration**: The system plans to incorporate deep learning for enhanced pattern recognition and anomaly detection.

2. Research on Natural Language Processing

- **Text Analysis**: NLP techniques such as sentiment analysis, keyword extraction, and syntactic parsing have proven effective in identifying deceptive language used in phishing emails and websites.
- **Proposed Integration**: The proposed system will employ advanced NLP methods to analyze email content, URLs, and messages in real-time, flagging suspicious elements with high accuracy.

3. Research on Behavioural Analysis

- **User Interaction Tracking**: Behavioural analysis studies focus on user interactions with links and attachments, identifying abnormal or risky patterns indicative of phishing.
- **Proposed Integration**: The system will monitor user behaviour and provide contextual warnings based on anomalies, enhancing protection against sophisticated phishing techniques.

4. Research on Threat Intelligence

- **Real-Time Threat Feeds**: Research has emphasized integrating dynamic threat databases for identifying known phishing URLs and domains.
- **Proposed Integration**: The system aims to incorporate a constantly updated threat intelligence framework, improving detection accuracy and response speed.

5. Innovation in Detection Accuracy

- **Hybrid Models**: Studies advocate combining rule-based approaches, machine learning, and NLP for robust detection. Hybrid models have shown significant improvement in reducing false positives while ensuring scalability.
- **Proposed Integration**: The proposed system will utilize a hybrid approach to maximize accuracy and adaptability across varied phishing scenarios.

Conclusion

- The proposed phishing detection system builds upon existing research to address key challenges such as evasion tactics, false positives, and adaptability. By integrating machine learning, NLP, behavioral analysis, and real-time threat intelligence, the system aims to provide a robust and scalable solution for phishing prevention.

Feasibility Study: -

Feasibility study is these conducted steps of the system development life cycle. Things are always easy at the beginning in any software process. In fact, nothing is infeasible with unlimited time and resources. But it is not the fact. So, practically we have to do in limited resources in a restricted time margin. So, for the system to be feasible, following points we have to consider.

The feasibility study is conducted to check whether the candidate system is feasible. The system which is selected to be the best against the criteria is thereafter designed and developed. The feasibility study takes in to consideration, the risks involved in the project development beforehand. Therefore, in this phase we have to do feasibility study which is the test of the website according to its workability, impact on the organization, ability to meet user need and effective use of resources. We do the feasibility study for website to analyze the risks, costs and benefits relating to economics, technology and user organization. There are several types of feasibility depending on the aspect they cover. Import of these includes:

Technical Feasibility:

This is an important outcome of preliminary investigation. It comprises of following questions: -

- Can the work of project be done with the current equipment, existing software and available man power resource?
- If Technology is required what are the possibilities that it can be developed?

Economic Feasibility:

It deals with question related to the economy. It comprises of the following questions: -

- Are there sufficient benefits in creating the system to make the cost acceptable?
- Are the costs of not creating the system so great that the project must be undertaken?

Legal Feasibility:

It deals with the question related to the legal issues. It comprises of the following questions: -

- Contract Signing
- Software License agreement
- Issues related to cyber laws.
- Legal issues relating to the man power contract.

Operational Feasibility:

The operational feasibility consists of the following activity: -

- Will the system be useful if it is developed &implemented?
- Will there be resistance from employee?

Social & Behavioral Feasibility:

It deals with the various issues related to the human behavior like: -
• Whether the user be able to adapt a new change or not?
• Whether the ambiance we are providing suits the user or not?

Request Approval: -

Request approval is the third phase of system development lifecycle. Request approval is the phase in which all the requirements which would be provide in the system are stated. The request approval is a sort of agreement between the client and the company which is building this software. Both the parties should be mutually agreed on the stated requirements.

System Analysis: -

System analysis is the phase following the phase of the request approval. In this phase wet end to analyze the overall system which we have to build. System analysis is the crucial part in SDLC.

System Design: -

System design means the designing of the system. The System can be done in either of the following two ways: -

• Logical System Design
• Physical System Design

Coding: -

Coding is the phase in which a developer codes using any programming languages. Coding constitutes only20 %of the whole project and which is easier to write. The coding work is also done in the teams; development of the system is usually done under the modular programming style, which can be either top-down approach or bottom-up approach.

Testing: -

Testing is the phase in which the system that has been developed is tested. Testing comprises of the 60%ofthe overall development of the system. Testing of the system is important because testing aims to uncover the different errors in the system. There are various different testing techniques that can be used for the testing of the system.

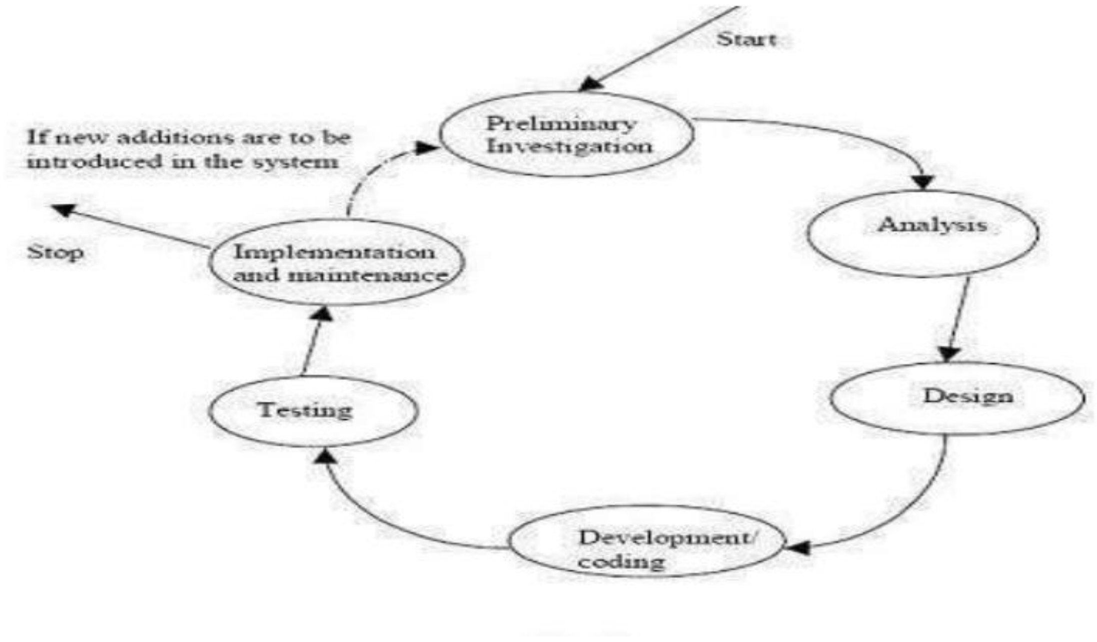
Implementation: -

Implementation process involved the installation of software on user’s side. Implementation process actually depends on type of a system &various. Opting for suitable conversion approach is as implementation. The conversion processes are as follows:-

- Parallel Conversion
- Direct Conversion Approach
- Pilot Conversion Approach
- Phase In Conversion Approach

Maintenance: -

Merely developing the system is not important but also maintenance is important. The company that has built the system provides for some time free of cost maintenance to the client and after that period it is usually a paid service.


Start
Preliminary Investigation
Analysis
Design
Development/ coding
Testing
Implementation and maintenance
If new additions are to be introduced in the system
Stop

Various Stages in Software Development

Process Description

Gantt charts mainly used to allocate resources to activities. The resources allocated to activities include staff, hardware, and software. Gantt charts (named after its developer Henry Gantt) are useful for resource planning. A Gantt chart is special type of bar chart where each bar represents an activity. The bars are drawn along a timeline. The length of each bar is proportional to the duration of the time planned for the corresponding activity. Gantt chart is a project scheduling technique. Progress can be represented easily in a Gantt chart, by coloring each milestone when completed. The project will start in the month of January and end after 4 months at the beginning of April.

SYSTEM ANALYSIS & DESIGN

OBJECTIVE:

The objective of the application is to develop a system using which job applicants and recruiters can communicate with each other. The purpose is to enable applicants to search for jobs in a convenient manner and to enable employers to find suitable candidates.

1). To provide online discussion board, between all registered user and admin.

2). Easily accessible from any corner of the world if you have internet connection.

3). If user complaint is reasonable, we will attempt to secure a satisfactory resolution for user.

Phases:

System Development Life Cycle (SDLC) mainly consists of the following7 phases which can be detailed: -

Preliminary Investigation: -

This is the first phase of the system development life cycle. In this phase we tend to find out the needs of the client ―what exactly does the client want? Before the development of any system the important point is to know the needs, objectives and scope of the system.

Requirement Specification

1. Functional Requirements

1. **Phishing Email Detection**:
    - Identify and flag emails with suspicious content, links, or attachments.
    - Provide risk scores and reasons for detection.
2. **Real-Time URL Analysis**:
    - Analyze URLs in real-time to detect phishing attempts before users access them.
    - Use domain reputation and content similarity checks.
3. **Threat Intelligence Integration**:
    - Integrate with global threat databases to identify known phishing URLs and domains.
    - Update threat intelligence in real-time.
4. **Machine Learning Model**:
    - Train and deploy machine learning models to detect patterns associated with phishing.
    - Continuously improve accuracy through feedback and retraining.
5. **User Notifications**:
    - Provide real-time alerts and warnings for suspicious content.
    - Include detailed explanations to help users understand and act appropriately.
6. **Behavioural Analysis**:
    - Monitor user behaviour for interactions with risky content.
    - Flag unusual activity patterns indicative of phishing.
7. **Customizability**:
    - Allow users or organizations to set sensitivity levels for phishing detection.
    - Enable rules to adapt to specific use cases (e.g., enterprise policies).

2. Non-Functional Requirements

1. **Performance**:
    - High-speed processing to ensure real-time phishing detection.
    - Support for high volumes of data in enterprise environments.

2. **Scalability**:
    - Ability to scale from individual use to large organizations.
    - Handle increasing numbers of phishing attempts and users without degradation in performance.
3. **Usability**:
    - Intuitive and user-friendly interface for non-technical users.
    - Provide actionable insights with minimal technical jargon.
4. **Security**:
    - Ensure encrypted communication during data analysis.
    - Comply with data protection regulations (e.g., GDPR, CCPA).
5. **Reliability**:
    - Maintain a high detection rate while minimizing false positives and negatives.
    - Operate consistently under different network conditions.
6. **Compatibility**:
    - Support cross-platform environments (Windows, macOS, Linux, mobile OS).
    - Ensure integration with popular email clients and browsers.

3. System Requirements

1. **Hardware Requirements**:
    - Server with high processing power for machine learning models.
    - Sufficient storage for maintaining threat intelligence databases.
2. **Software Requirements**:
    - Programming languages (e.g., Python, Java) for system development.
    - Machine learning frameworks (e.g., TensorFlow, PyTorch).
    - Integration APIs for threat intelligence services.
3. **Network Requirements**:
    - Stable internet connection for real-time updates and analysis.
    - Secure communication channels (e.g., HTTPS, VPN).

ER-Diagram

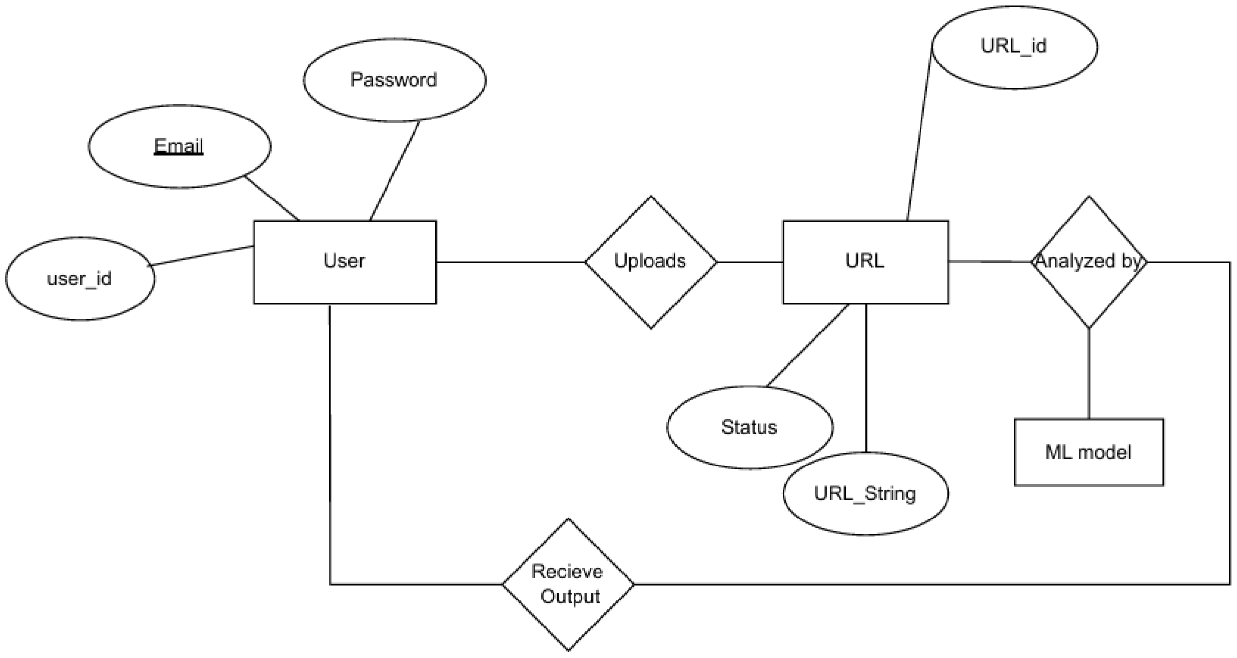
Introduction: -

In software engineering, an entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs. ER Diagrams depicts relationship between data objects. The attribute of each data objects noted in the entity-relationship diagram can be described using a data object description. Entity relationship diagram is very basic, conceptual model of data and it is fundamental to the physical database design. This analysis is then used to organize data as relations, normalizing relations, and obtaining a Relational database. The entity-relationship model for data uses three features to describe data. These are:

1.Entities which specify distinct real-world items in an application.

2. Relationship, which connect entities and represent meaningful dependencies between them.

3.Attributes which specify properties of entities & relationships.



ER-diagram

Data Flow Diagram

Introduction: -

Data Flow Diagram DFD is an acronym for the word Data Flow Diagram. DFD is pictorial representation of the system. DFD is a graphical representation of the —flow of data through the information system. DFD are also used forth visualization of data processing (structured design). ADFD provides no information about the timings of the process, or about whether process will operate in parallel or sequence. DFD is an important technique for modeling a system's high-level detail by showing how input data is transformed too input results through as sequence of functional transformations. DFD reveal relationships among between the various components in a program or system. Of DFD lies in the fact that using few symbols we are able to express program design in an easier manner. ADFD can be used to represent the following: -

- External Entity sending and receiving data.
- Process that change the data.
- Flow of data within the system.
- Data Storage locations.

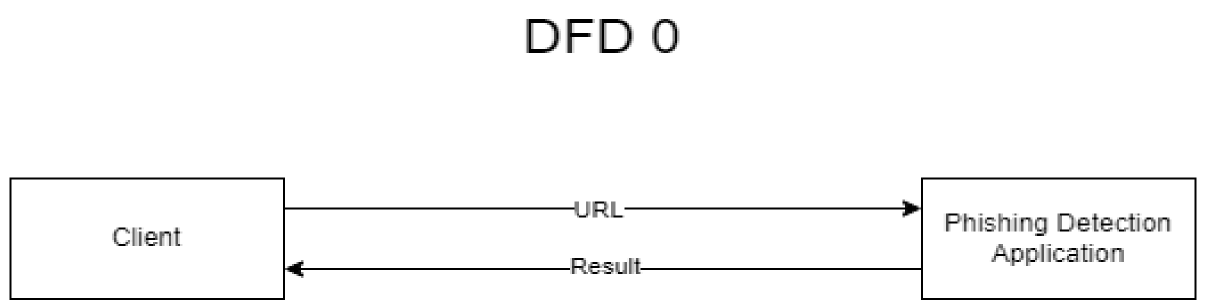
Uses of DFD: -

The main uses of data flow diagrams are as follows: -

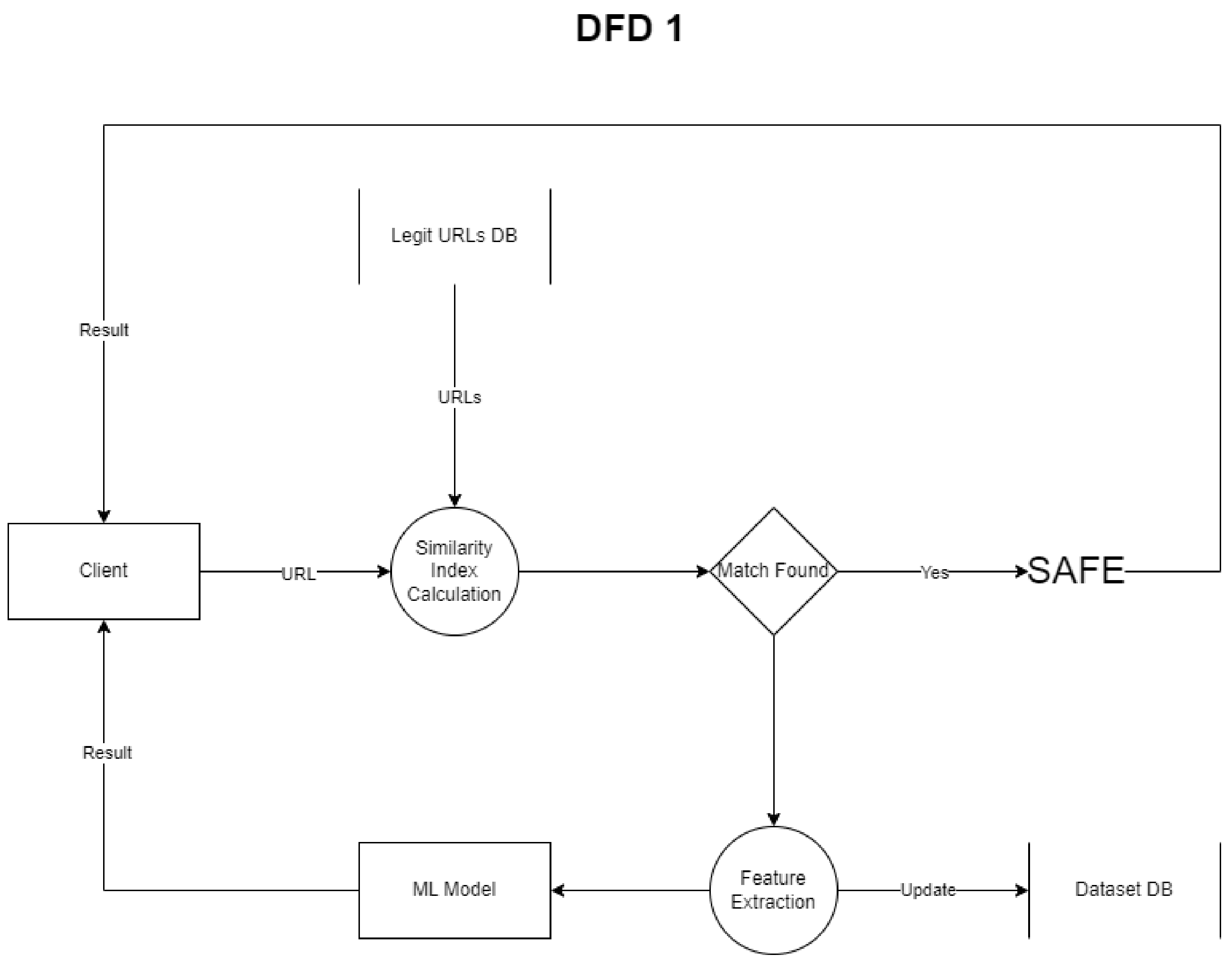
DFD is a method of choice for representation of showing of information through a system because of the following reasons: -

•DFDs are easier to understand by technical and non-technical audiences.
• DFDs can provide a high-level system overview, complete with boundaries and connections to other system.
•DFDs can provided tailed representation of system components

0 Level DFD



1 Level DFD



Design and Test Steps

Design Steps

1. **Requirement Analysis:**
    - Gather and document functional and non-functional requirements.
    - Define the scope and purpose of the phishing detection system.
2. **System Architecture:**
    - Design the overall architecture, including data flow, modules (e.g., input validation, analysis, and output generation), and integration points with threat databases.
3. **Database Design:**
    - Create the schema for data storage, including user information, analysis results, and threat intelligence records.
4. **Algorithm Selection:**
    - Choose appropriate machine learning models and techniques (e.g., Random Forest, NLP models).
    - Define preprocessing steps for data (e.g., tokenization, vectorization).
5. **Interface Design:**
    - Develop mockups or wireframes for the user interface (UI) to ensure it is intuitive and user-friendly.
    - Decide on cross-platform compatibility (web, desktop, or mobile).
6. **Integration Plan:**
    - Identify APIs for threat intelligence and any external systems the application needs to connect to.
    - Plan for incorporating security features (e.g., encryption, authentication).

Test Steps

1. **Unit Testing:**
    - Test individual components or modules (e.g., URL validation, ML model inference) to ensure they work as expected.
    - Use mock data to verify each unit's functionality.
2. **Integration Testing:**
    - Verify interactions between modules (e.g., analysis engine and threat intelligence database).

- Test end-to-end workflows, such as data input → analysis → results generation.

3. **Performance Testing:**
   - Evaluate the system’s speed and scalability under different loads.
   - Simulate multiple users submitting emails or URLs simultaneously.

4. **Accuracy Testing:**
   - Assess the system's accuracy in detecting phishing attempts.
   - Use labelled datasets to calculate precision, recall, and F1 score.

5. **User Acceptance Testing (UAT):**
   - Test with actual users to ensure the system meets their expectations.
   - Gather feedback on the UI, detection results, and overall usability.

6. **Security Testing:**
   - Validate that sensitive information is encrypted during transmission and storage.
   - Test for vulnerabilities such as injection attacks, data leaks, or unauthorized access.

7. **Regression Testing:**
   - After updates or fixes, ensure previous functionalities are not broken.
   - Use automated testing tools to streamline this process**.**

8. **Deployment Testing:**
   - Test the system in its live environment to verify its performance and reliability.
   - Check that real-time updates from the threat database work seamlessly.

Software Requirements Specification

A requirements specification for a software system is a complete description of the behavior of a system to be developed and it includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints). Requirements are a sub-field of software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software. The software requirement specification document enlists all necessary requirements for project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with project team and the customer

PROJECT MODEL USED

Iterative Enhancement Model

- ➢ This model has the same phases as the waterfall model, but with fewer restrictions. Generally, the phases occur in the same order as in the waterfall model, but they may be conducted in several cycles.
- ➢ Useable product is released at the end of each cycle, with each release providing additional functionality. Customers and developers specify as many requirements as possible and prepare a SRS document. Developers and customers then prioritize these requirements. Developers implement the specified requirements in one or more cycles of design, implementation and test based on the defined priorities.

The procedure itself consists of the initialization step, the iteration step, and the Project Control List. The initialization step creates a base version of the system. The goal for this initial implementation is to create a product to which the user can react. It should offer a sampling of the key aspects of the problem and provide a solution that is simple enough to understand and implement easily. To guide the iteration process, a project control list is created that contains a record of all tasks that need to be performed. It includes such items as new features to be implemented and areas of redesign of the existing solution. The control list is constantly being revised as a result of the analysis phase.

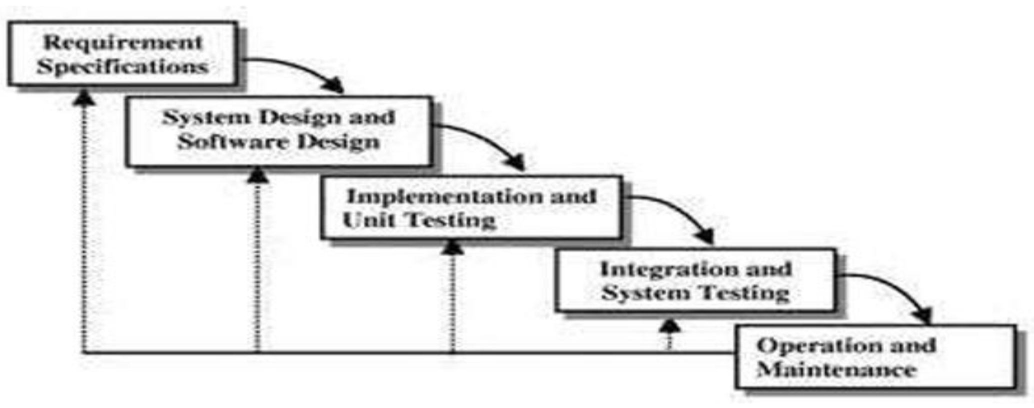
The iteration involves the redesign and implementation of iteration is to be simple, straightforward, and modular, supporting redesign at that stage or as a task added to the project control list. The level of design detail is not dictated by the iterative approach. In a light-weight iterative project the code may represent the major source of documentation of the system; however, in a critical iterative project a formal Software Design Document may be used. The analysis of an iteration is based upon user feedback, and the program analysis facilities available. It involves analysis of the structure, modularity, usability, reliability, efficiency, & achievement of goals. The project control list is modified in light of the analysis results.

Phases:

Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment. The unified process groups increments/iterations into phases: inception, elaboration, construction, and transition. • Inception identifies project scope, requirements (functional and non-functional) and risks at a high level but in enough detail that work can be estimated. • Elaboration delivers a working architecture that mitigates the top risks and fulfills the non- functional requirements.

•Construction incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.

Transition delivers the system into the production operating environment.


Requirement Specifications
System Design and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Phases

Algorithm

The project uses Random Forest algorithm to train the model. Random Forest is a versatile and powerful machine learning algorithm often used in projects requiring classification or regression tasks. If your project employs the Random Forest algorithm, its functionality can be described as follows:

Overview of Random Forest:

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. Each tree in the forest is built on a subset of the dataset, and the final output is derived through majority voting (for classification) or averaging (for regression).

Why Random Forest in the Project:

1. **Accuracy and Robustness**: By aggregating the outputs of many decision trees, Random Forest reduces the risk of overfitting and produces more reliable predictions.
2. **Feature Importance**: The algorithm provides insights into which features are most influential, helping in data analysis and model refinement.
3. **Handling Missing Data**: Random Forest can work well even if some data points are missing, making it suitable for real-world datasets.
4. **Versatility**: It performs effectively on both structured and unstructured data, making it ideal for a range of applications, including phishing detection.

Application in the Project:

In a phishing detection project, Random Forest can analyse features such as URL structure, email metadata, or website content to identify patterns associated with phishing attempts. By training on labelled examples of phishing and legitimate data, the algorithm learns to classify new instances with high precision.

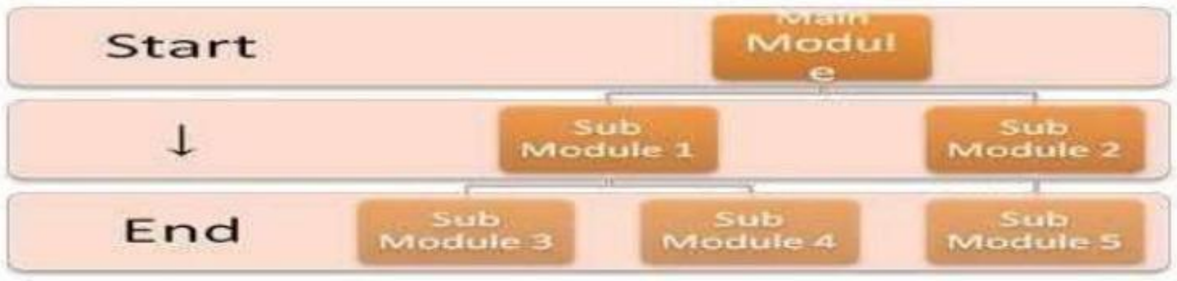
SUPPORT AND MAINTENANCE

One year free support for rectifying system bugs including front end and beck end will be provided. During warranty period Software Engineers will be responsible for removing bugs and improving it. After one year support can be extended @ 20% of the total product deployment cost

SYSTEM DESIGN APPROACH

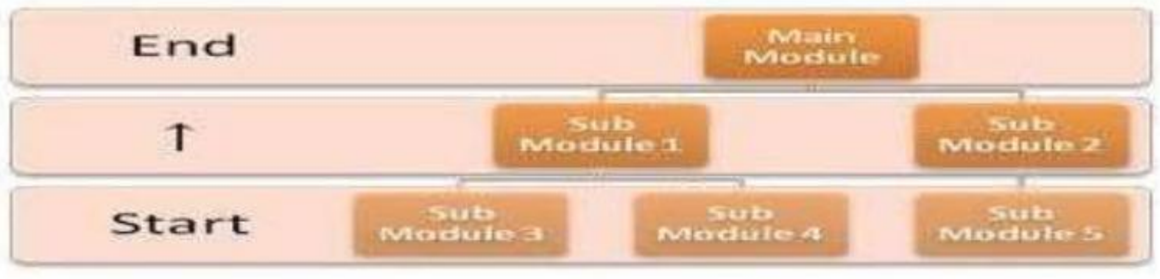
Top – Down designing:

The top - down designing approach started with major components of the system. It is a stepwise refinement which starts from an abstract design, in each steps the design is refined two or more concrete levels until we reach a level where no – more refinement is possible or not needed.



Bottom – Up designing:

In bottom – up designing the most basic and primitive components are designed first, and we proceed to higher level components. We work with layers of abstractions and abstraction are implemented until the stage is reached where the operations supported by the layer is complete.



Approach we are following:

In this project we are following Mixed Approach i.e. A combination of top – down and bottom – up. We are developing some of the components using top – down designing approach (e.g. the Webpages) and some components in bottom – up designing approach (e.g. the middle tier classes).

Low Level Design

Description: Low Level Design creation is one of the most important activities in the development of any software product. The low-level design document gives the design of the actual software application. Low level design document is based on High Level Design document. It defines internal logic of every sub module. A good low level design document will make the application very easy to develop by the developer. An effective design document results in very low efforts in developing a Software product.

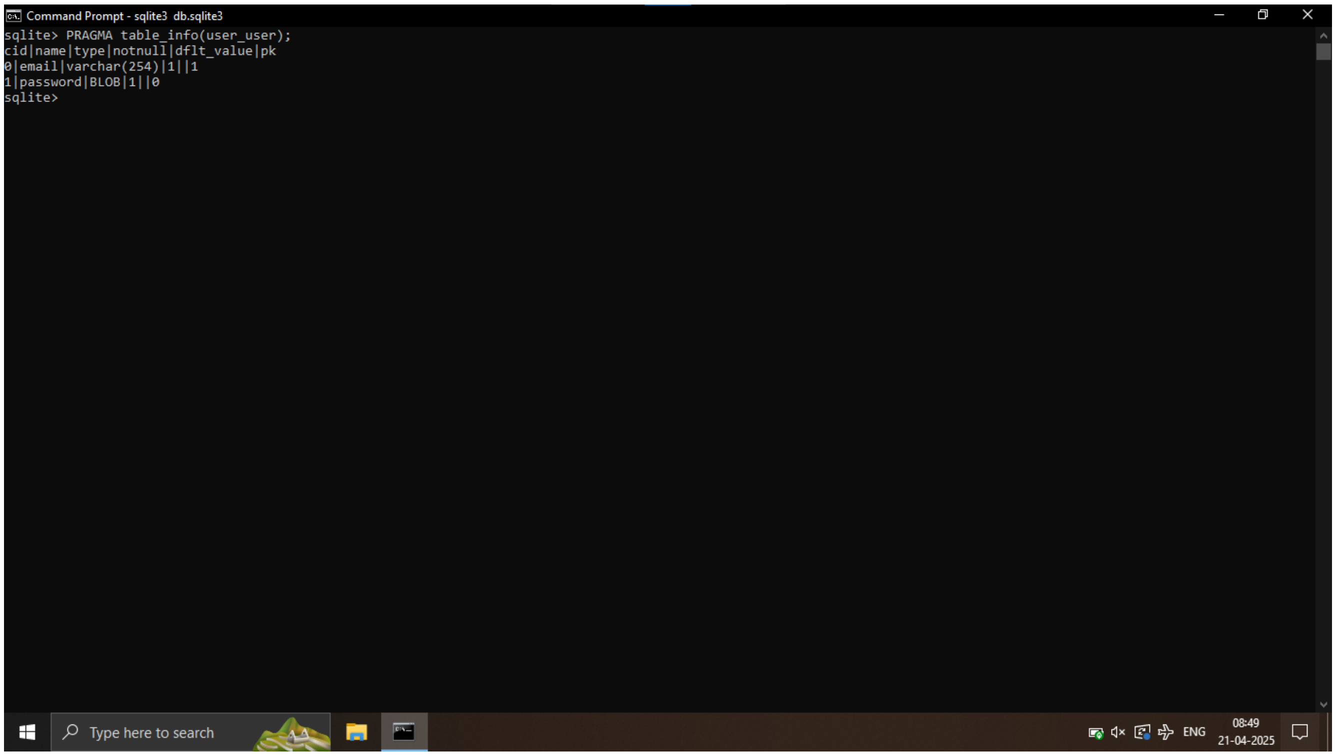
Each project's low level design document should provide a complete and detailed specification of the design for the software that will be developed in the project, including the classes, member and non-member functions, and associations between classes that are involved.

The low-level design document should contain a listing of the declarations of all the classes, non-member-functions, and class member functions that will be defined during the subsequent implementation stage, along with the associations between those classes and any other details of those classes (such as member variables) that are firmly determined by the low-level design stage. The low-level design document should also describe the classes, function signatures, associations, and any other appropriate details, which will be involved in testing and evaluating the project according to the evaluation plan defined in the project's requirements document.

DATA MODELING

It contains only 1 table that is registration table.

Schema



```
sqlite> PRAGMA table_info(user_user);
cid|name|type|notnull|dflt_value|pk
0|email|varchar(254)|1||1
1|password|BLOB|1||0
sqlite>
```

TESTING

Testing is the integral part of any System Development Life Cycle insufficient and interested application tends to crash and result in loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation.

"Software Testing can be looked upon as one among much process, an organization performs, and that provides the last opportunity to correct any flaws in the developed system. Software Testing includes selecting test data that have more probability of giving errors." The first step in System testing is to develop the plan that all aspect of system. Complements, Correctness, Reliability and Maintainability.

Software is to be tested for the best quality assurance, an assurance that system meets the specification and requirement for its intended use and performance. System Testing is the most useful practical process of executing the program with the implicit intention of finding errors that makes the program fail.

Types of Testing

Black Box (Functional) Testing:

Testing against specification of system or component. Study it by examining its inputs and related outputs. Key is to devise inputs that have a higher likelihood of causing outputs that reveal the presence of defects. Use experience and knowledge of domain to identify such test cases. Failing this a systematic approach may be necessary. Equivalence partitioning is where the input to a program fall into a number of classes, e.g. positive numbers vs. negative numbers. Programs normally behave the same way for each member of a class. Partitions exist for both input and output. Partitions may be discrete or overlap. Invalid data (i.e. outside the normal partitions) is one or more partitions that should be tested.
Internal System design is not considered in this type of testing. Tests are based on requirements and functionality.
This type of test case design method focuses on the functional requirements of the software, ignoring the control structure of the program. Black box testing attempts to find errors in the following categories:
- Incorrect or missing functions.
- Interface errors.
- Errors in data structures or external database access.
- Performance errors.
- Initialization and termination errors.

White Box (Structural) Testing

Testing based on knowledge of structure of component (e.g. by looking at source code). Advantage is that structure of code can be used to find out how many test case need to be performed. Knowledge of the algorithm (examination of the code) can be used to identify the equivalence partitions. Path testing is where the tester aims to exercise every independent execution path through the component. All conditional statements tested for both true and false cases. If a unit has n control statements, there will be up to 2n possible paths through it. This demonstrates that it is much easier to test small program units than large ones. Flow graphs are a pictorial representation of the paths of control through a program (ignoring assignments, procedure calls and I/O statements). Use flow graph to design test cases that execute each path. Static tools may be used to make this easier in programs that have a complex branching structure. Tools support. Dynamic program analyzers instrument a program with additional code. Typically, this will count how many times each statement is executed. At end print out report showing which statements have and have not been executed. Problems with flow graph derived testing:

- Data complexity could not take into account.
- We cannot test all paths in combination.
- In really only possible at unit and module testing stages because beyond that complexity is too high.

This testing is based on knowledge of the internal logic of an application's code. Also known as a Glass Box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

Unit Testing:

Unit testing concentrates on each unit of the software as implemented in the code. This is done to check syntax and logical errors in programs. At this stage, the test focuses on each module individually, assuring that it functions properly as a unit. In our case, we used extensive white- box testing at the unit testing stage. A developer and his team typically do the unit testing do the unit testing is done in parallel with coding; it includes testing each function and procedure.

Incremental Integration Testing:

Bottom-up approach for testing i.e. continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately done by programmers or by testers.

Integration Testing:

Testing of integration modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server and distributed systems.

Functional Testing:

This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black box type testing geared to functionality requirements of an application.

System Testing:

Entire system is tested as per the requirements. Black box type test that is based on overall requirement specifications covers all combined parts of a system.

End-to-End Testing:

Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with hardware, applications, or system if appropriate.

Regression Testing:

Testing the application as a whole for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.

Acceptance Testing:

Normally this type of testing is done to verify if system meets the customer specified requirements. User or customers do this testing to determine whether to accept application.

Performance Testing:

Term often used interchangeably with “stress” and “load” testing, to check whether system meets performance requirements, used different performance and load tools to do this.

Alpha Testing:

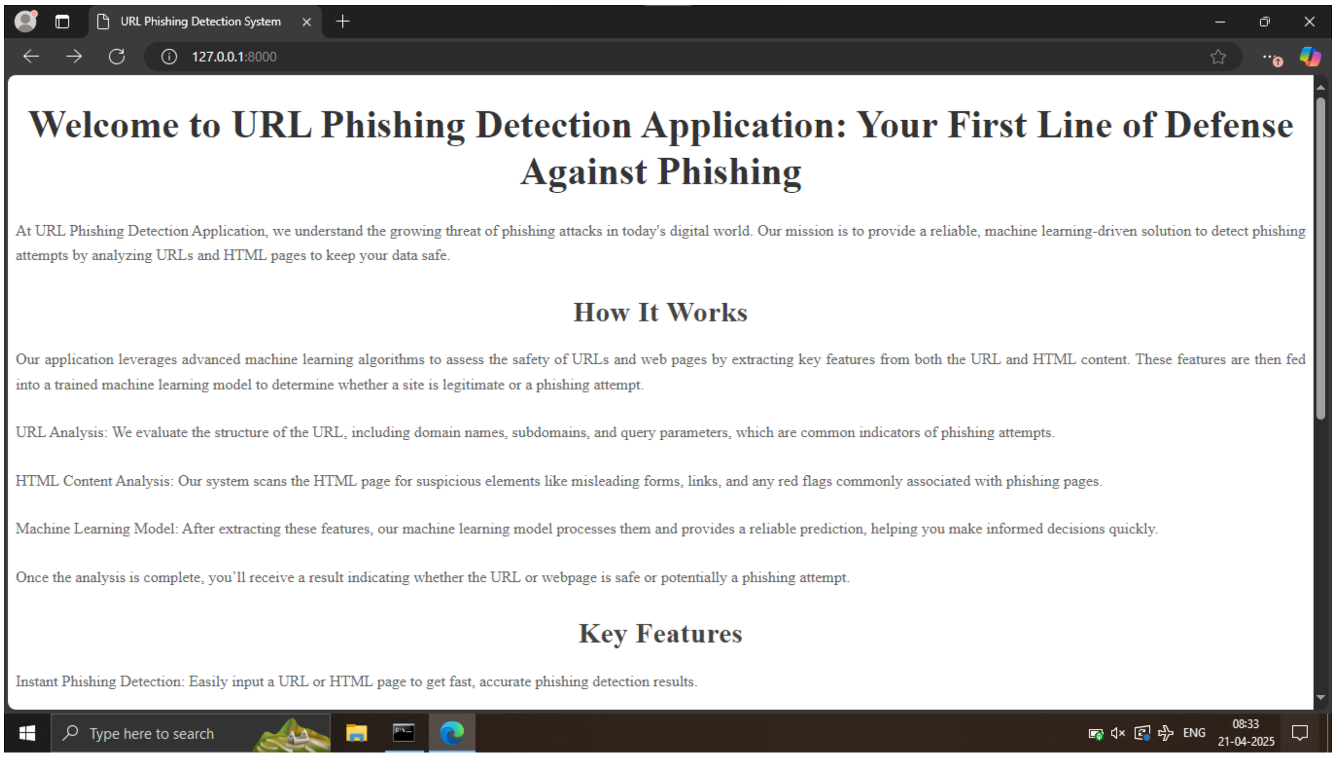
In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

Beta Testing:

Testing typically done by end-users or others. This is final testing before releasing application for commercial purpose.

Result/Output: --

Home_page


URL Phishing Detection System

127.0.0.1:8000

Welcome to URL Phishing Detection Application: Your First Line of Defense Against Phishing

At URL Phishing Detection Application, we understand the growing threat of phishing attacks in today's digital world. Our mission is to provide a reliable, machine learning-driven solution to detect phishing attempts by analyzing URLs and HTML pages to keep your data safe.

How It Works

Our application leverages advanced machine learning algorithms to assess the safety of URLs and web pages by extracting key features from both the URL and HTML content. These features are then fed into a trained machine learning model to determine whether a site is legitimate or a phishing attempt.

URL Analysis: We evaluate the structure of the URL, including domain names, subdomains, and query parameters, which are common indicators of phishing attempts.

HTML Content Analysis: Our system scans the HTML page for suspicious elements like misleading forms, links, and any red flags commonly associated with phishing pages.

Machine Learning Model: After extracting these features, our machine learning model processes them and provides a reliable prediction, helping you make informed decisions quickly.

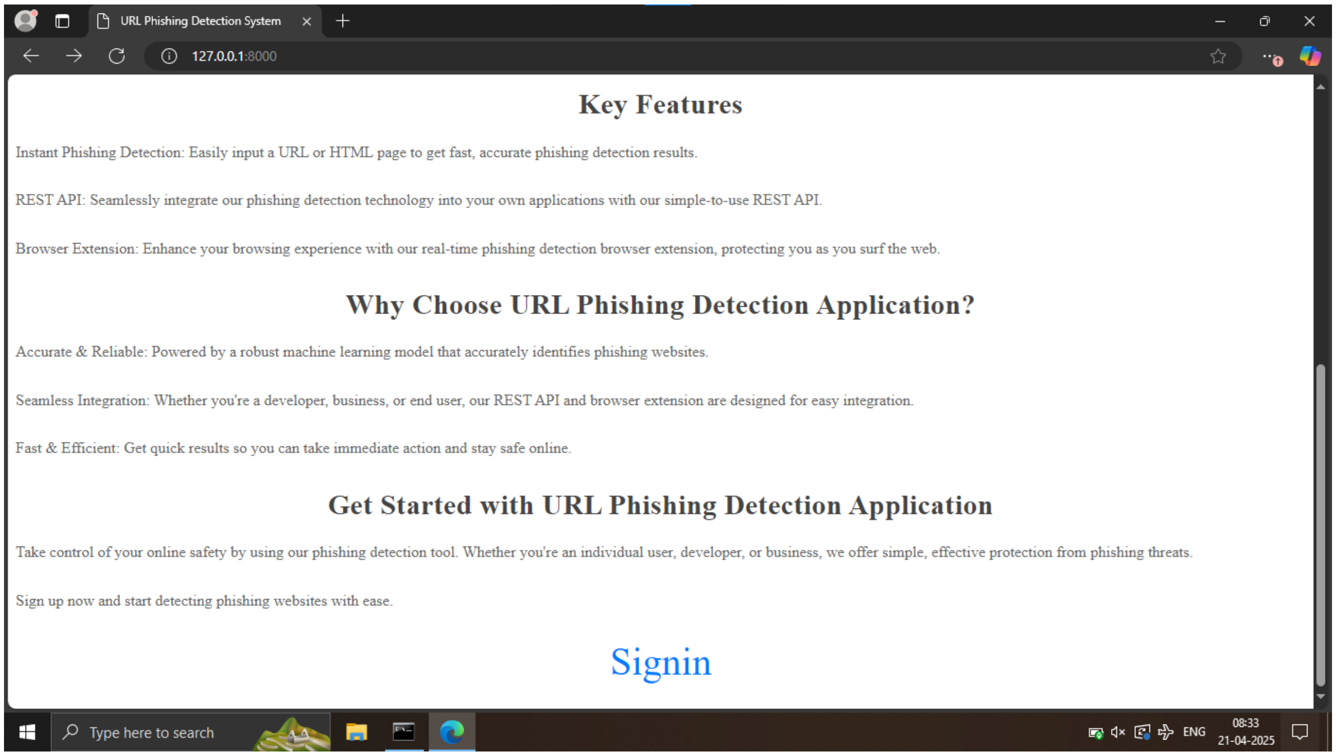
Once the analysis is complete, you'll receive a result indicating whether the URL or webpage is safe or potentially a phishing attempt.

Key Features

Instant Phishing Detection: Easily input a URL or HTML page to get fast, accurate phishing detection results.

Type here to search

ENG 08:33 21-04-2025


URL Phishing Detection System

127.0.0.1:8000

Key Features

Instant Phishing Detection: Easily input a URL or HTML page to get fast, accurate phishing detection results.

REST API: Seamlessly integrate our phishing detection technology into your own applications with our simple-to-use REST API.

Browser Extension: Enhance your browsing experience with our real-time phishing detection browser extension, protecting you as you surf the web.

Why Choose URL Phishing Detection Application?

Accurate & Reliable: Powered by a robust machine learning model that accurately identifies phishing websites.

Seamless Integration: Whether you're a developer, business, or end user, our REST API and browser extension are designed for easy integration.

Fast & Efficient: Get quick results so you can take immediate action and stay safe online.

Get Started with URL Phishing Detection Application

Take control of your online safety by using our phishing detection tool. Whether you're an individual user, developer, or business, we offer simple, effective protection from phishing threats.

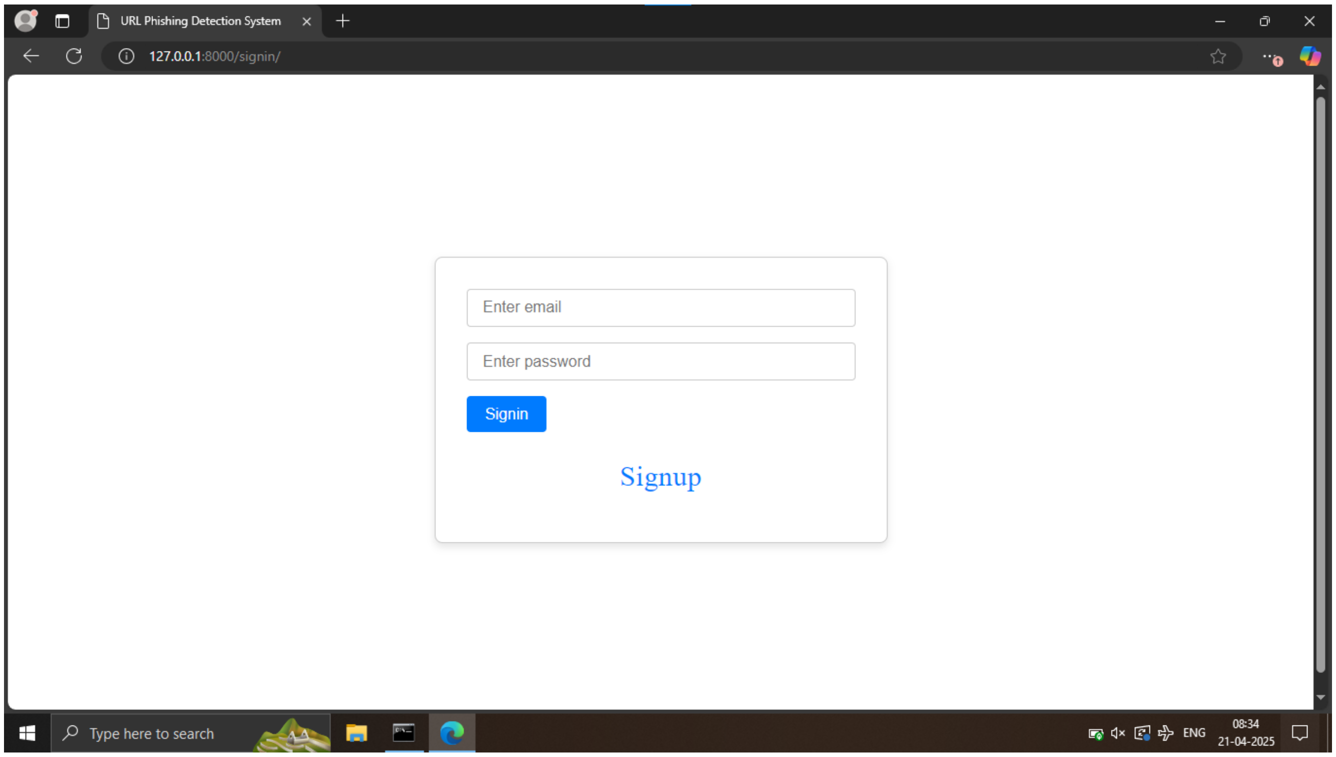
Sign up now and start detecting phishing websites with ease.

Signin

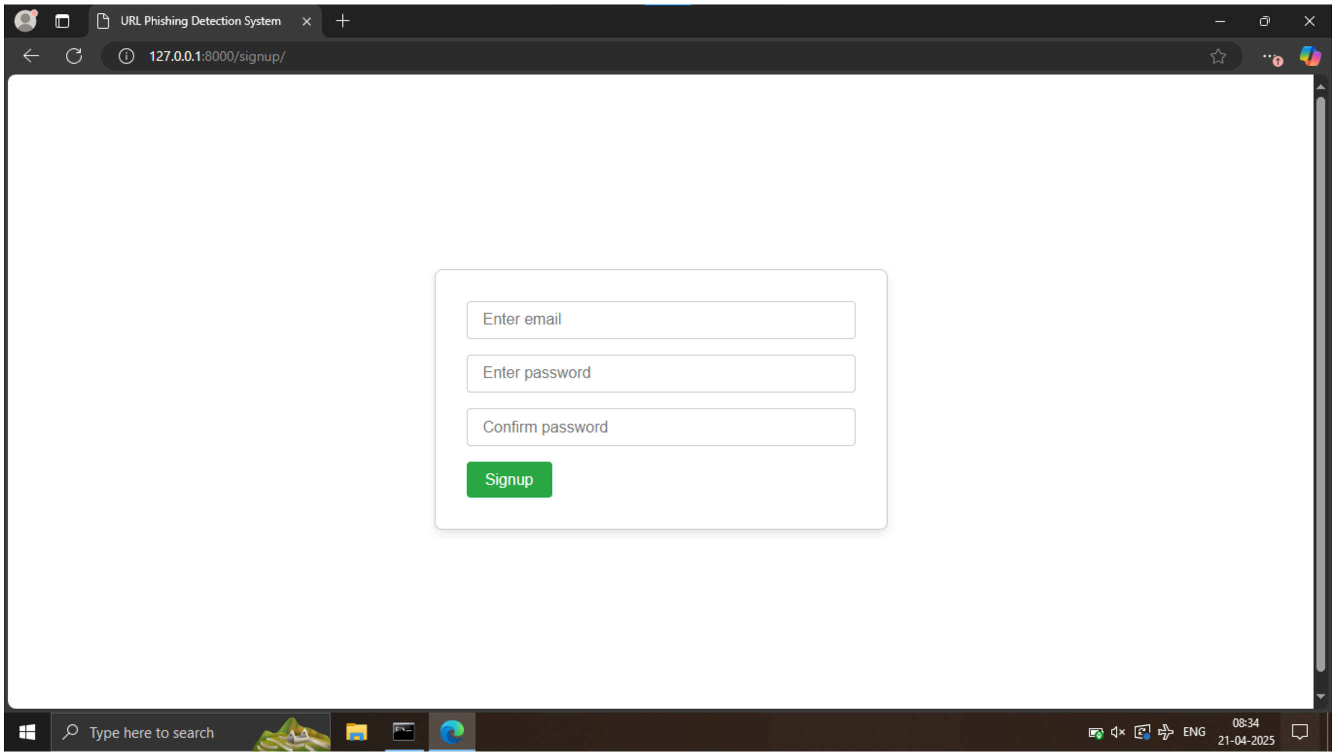
Type here to search

ENG 08:33 21-04-2025

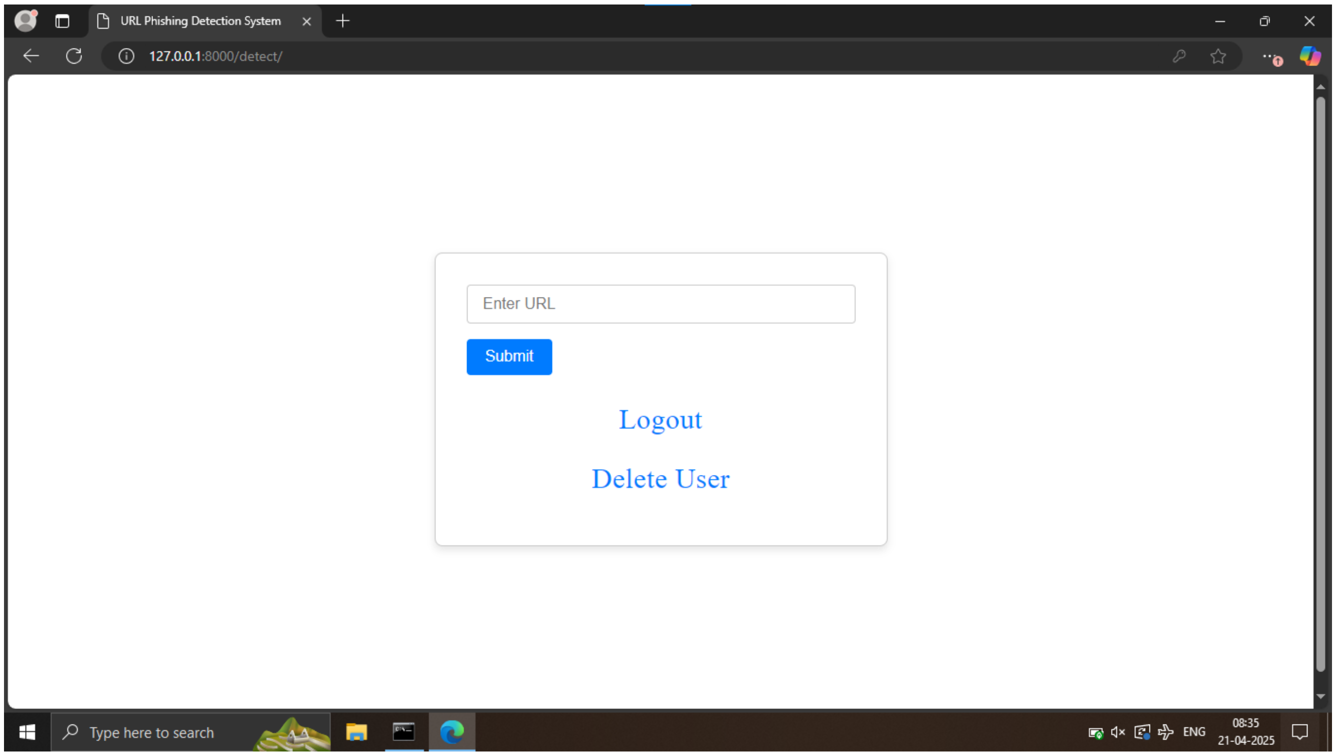
Signin_page



Signup_page



URL detection page



Conclusion

In the face of rising cyber threats, phishing remains one of the most critical challenges for individuals and organizations worldwide. The development of an advanced **Phishing Detection Application** addresses this pressing issue by leveraging cutting-edge technologies, such as machine learning and natural language processing, to identify and prevent phishing attempts effectively.
The application not only enhances detection accuracy through real-time analysis and behavioural monitoring but also integrates threat intelligence for proactive protection. Its user-friendly interface and customizable features empower users to safeguard their digital environments effortlessly. By bridging the gap between awareness and action, this solution provides a robust defence against phishing attacks, contributing significantly to a safer and more secure online experience.

Future Scope

1. Advanced Threat Detection

- Incorporation of deep learning techniques to improve detection accuracy for sophisticated phishing attacks, such as zero-day threats and spear-phishing.
- Development of multi-modal detection methods, analysing not just text but also images, videos, and other embedded media in emails and websites.

2. Real-Time Adaptability

- Enhanced integration with global threat intelligence platforms for real-time updates on phishing trends and patterns.
- Autonomous learning systems that adapt to evolving phishing tactics without manual intervention.

3. Cross-Platform Integration

- Expansion to protect users across multiple platforms, such as messaging apps, social media platforms, and enterprise collaboration tools (e.g., Slack, Teams).
- Browser extensions and mobile apps for seamless user experiences and protection on-the-go.

4. User Awareness and Training

- Integration of interactive modules for user education, including phishing awareness training and simulated phishing tests.
- Gamification techniques to make cybersecurity learning more engaging and effective.

5. Enterprise-Level Features

- Support for large-scale deployment in enterprises with centralized threat monitoring dashboards.
- Customizable policies and configurations tailored to specific organizational needs.

6. Enhanced Data Privacy and Security

- Adoption of advanced encryption protocols and privacy-preserving technologies for user data.

- Compliance with evolving data protection regulations, such as GDPR and CCPA, ensuring trust among users.

7. Collaboration with AI

- Collaborating with other AI systems to create a comprehensive cybersecurity framework addressing a broader range of cyber threats.
- Developing APIs for seamless integration with other cybersecurity tools, such as firewalls and antivirus software.

8. Global Reach

- Supporting multiple languages to cater to a global audience, ensuring effectiveness across diverse regions and cultures.
- Customizing phishing detection to consider cultural and regional nuances in phishing techniques.

References

Here are some potential references you can use for the phishing detection application project:

Research Papers and Articles

- PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index
- Ahammad, S.H., Kale, S.D., Upadhye, G.D., Pande, S.D., Babu, E.V., Dhumane, A.V., Bahadur, M.D.K.J., 2022. Phishing URL detection using machine learning methods. Adv. Eng. Softw. 173, 103288. https://doi .org /10 .1016 /j .advengsoft .2022 .103288.
- Alani, M.M., Tawfik, H., 2022. PhishNot: a cloud-based machine-learning approach to phishing URL detection. Comput. Netw. 218, 109407. https://doi .org /10 .1016 /j .comnet .2022 .109407.
- Anon, 2023a. Retrieved May, 12, 2023. https://isc .sans .edu /diary /The +zip +gTLD + Risks +and +Opportunities /29838.

Books

- **"Cybersecurity and Machine Learning"** by Shai Shalev-Shwartz and Shai Ben-David.
- **"Introduction to Natural Language Processing"** by Jacob Eisenstein.

Web Resources

1. **Microsoft Security Blog**:
   - Regular updates on phishing trends and detection techniques.
   - Microsoft Security Blog
2. **Open Threat Intelligence Platforms**:
   - Tools such as VirusTotal for real-time threat database lookups.
   - VirusTotal
3. **GitHub Repositories**:
   - Open-source projects related to phishing detection and machine learning.
   - Example: GitHub

Datasets

- **Enron Email Dataset**: Often used for training phishing detection systems.
- **Phishing Websites Dataset**: Available on platforms like Kaggle.