# DETECTION OF PHISHING WEBSITE

***A Project report submitted in partial fulfilment of the requirements for the award of the degree* of**

**BACHELOR OF TECHNOLOGY IN**

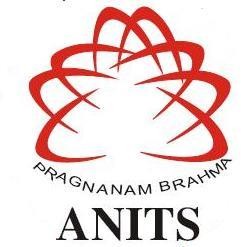
**INFORMATION TECHNOLOGY**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | ***Submitted by*** |  |
| K.Sravani | A21126511100 | S.Naveen Sai | A21126511123 |
| S.Eepsitha | A21126511125 | T.Sai Avinash | A21126511127 |

**Under the Guidance**

**of**

**Mrs D.Sowjanya**

Assistant Professor

#### DEPARTMENT OF INFORMATION TECHNOLOGY

**ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES (AUTONOMOUS)**

**(*Permanent Affiliation by Andhra University & Approved by AICTE***

***Accredited by NBA (ECE, EEE, CSE, IT, Mech. Civil & Chemical) & NAAC)***

#### Sangivalasa, Bheemili Mandal, Visakhapatnam dist.(A.P)

#### 2024-25

**ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES**

**(AUTONOMOUS)**

**(*Permanent Affiliation by Andhra University & Approved by AICTE***

***Accredited by NBA (ECE, EEE, CSE, IT, Mech. Civil & Chemical) & NAAC*)**

#### Sangivalasa, Bheemili Mandal, Visakhapatnam dist.(A.P)

#### DEPARTMENT OF INFORMATION TECHNOLOGY

**CERTIFICATE**

This is to certify that the project report entitled “**DETECTION OF PHISHING WEBSITE**” submitted **by K.Sravani(A21126511100), S.NaveenSai(A21126511123), S.Eepsitha(A21126511125), T.SaiAvinash(A21126511126)** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information Technology of Anil Neerukonda Institute of technology and sciences, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision*.*

**Project Guide Head of the Department**

**Mrs D Sowjanya Dr. M. Rekha Sundari**

Assistant Professor Professor

Department of IT Department of IT

ANITS ANITS

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our project guide **D.Sowjanya** Designation, Department of Information Technology, ANITS, for his/her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Prof. M.Rekha Sundari**, Head of the Department, Information Technology, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa,** for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of IT, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of IT, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

#### PROJECT STUDENTS

#### K.SRAVANI (A21126511100)

#### S.NAVEEN SAI (A21126511123)

#### S.EEPSITHA (A21126511125)

#### T.SAI AVINASH (A21126511126)

**DECLARATION**

We hereby declare that the project work entitled **“DETECTION OF PHISHING WEBSITE”** submitted to the Anil Neerukonda Institute of Technology and Sciences is a record of an original work done by K.SRAVANI (A21126511100), S.NAVEEN SAI (A21126511123),S.EEPSITHA (A21126511125), T.SAIAVINASH (A21126511126)under the esteemed guidance of **Mrs**.**D.Sowjanya ,** Designation of Information Technology, Anil Neerukonda Institute of Technology and Sciences, and this project work is submitted in the partial fulfillment of the requirements for the award of degree Bachelor of Technology inInformation Technology**.** This entire project is done with the best of our knowledge and have not been submitted for the award of any other degree in any other universities

#### 

#### K.SRAVANI (A21126511100)

#### S.NAVEEN SAI (A21126511123)

#### S.EEPSITHA (A21126511125)

#### T.SAIAVINASH (A21126511126)

## INDEX

#### Chapter Name Page No.

#### Abstract      iv

#### List of Figures       v

#### List of Tables     vi

#### List of Screens       vi

**1. INTRODUCTION                 10-15**

1.1 Motivation  11

1.2 Problem definition  12

1.3 Objective of Project  13

1.4 Limitations of Project  13

1.5 Organization of Documentation 14

**2.** [**LITERATURE SURVEY**](https://docs.google.com/document/d/1SuRnLowWYpCWnkT3wb24ojXVJKceecwu/edit#heading=h.1t3h5sf) **16-19**

             2.1 Introduction  17

             2.2 Existing System   17

             2.3 Disadvantages of Existing system  18

             2.4 Proposed System  19

             2.5 Conclusion 19

**3. SYSTEM ANALYSIS 20-27**

            3.1 Introduction  21

            3.2 Software Requirement Specification  21

                       3.2.1 User requirement  21

                       3.2.2 Software requirement  22

                       3.2.3 Hardware requirement  22

            3.3 Content diagram of Project  25

            3.4 Algorithms and Flowcharts  26

            3.5 Conclusion  27

**4**. **DESIGN                                                                              28-46**

           4.1 Introduction  29

           4.2 DFD / ER / UML diagram (any other project diagrams)  30

           4.3 Module design and organization  42

           4.4 Conclusion 46

**5. IMPLEMENTATION & RESULTS                   47-58**

5.1 Introduction  48

5.2 Explanation of Key functions  48

5.3 Method of Implementation  54

        5.3.1 Forms  55

        5.3.2 Output Screens  56

        5.3.3 Result Analysis  57

5.4 Conclusion 58

**6.CODING 59-67**

6.1 Code 60

**7. TESTING & VALIDATION 68-70**

7.1 Introduction  69

7.2 Design of test cases and scenarios  69

7.3 Validation  70

7.4 Conclusion  70

**8. CONCLUSION 71-72**

**REFERENCES 73-74**

**PAPER PUBLICATIONS WITH PLAGIARISM REPORT 75**

**MAIL ACCEPTANCE 76**

**PAYMENT 77-78**

**ABSTRACT**

This project addresses the pressing challenge of the current world which is detecting the phishing websites and avoiding those websites to keep us safe. Phishing is a prevalent threat that endangers digital security. To deal with the evolving nature of these attacks and the limitations of traditional rule-based systems, we're turning to machine learning (ML). We're using a variety of features from the detailed dataset that covers common phishing indicators. Our approach hones in on training and fine-tuning ML algorithms to stay sharp and proactive. By carefully evaluating and optimizing our models, we're showing how effective our approach is at quickly spotting potential phishing websites with high accuracy rates. Our work significantly contributes to the field of cybersecurity by presenting a proactive defence mechanism that is capable of adapting to evolving phishing tactics. By emphasizing the utilization of Machine Learning techniques, we achieve a robust system that aids in preventing fraudulent activities. The findings of these methods show the effectiveness of Machine Learning approaches in strengthening online security and enabling users and organizations to mitigate risks associated with phishing attacks. This research not only showcases the viability of ML in tackling phishing but also underscores the importance of advancements in cybersecurity measures. The outcomes highlight the potential for future enhancements and innovations in threat detection, ultimately creating a safer digital environment for users worldwide. In summary, our project emphasizes the pivotal role of machine learning in fortifying defences against phishing attacks, paving the way for enhanced cybersecurity measures.

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **S.No** | **Figure Number and Name** | **Page No** |
| 1 | 3.5 – Project Flowchart | 34 |
| 2 | 4.2.1 – Use case Diagram | 35 |
| 3 | 4.2.2 – Class Diagram | 35 |
| 4 | 4.2.3 – Activity Diagram | 36 |
| 5 | 4.2.4 – State Transition Diagram | 53 |
| 6 | 4.2.5 – Sequence Diagram | 54 |
| 7 | 4.2.6 – Collaboration Diagram | 55 |
| 8 | 4.2.7 - Component Diagram | 56 |
| 9 | 4.2.8 – Deployment Diagram | 57 |
| 10 | 4.3.1- System Architecture of phishing website detection | 59 |

# 

# 

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **S.No** | **Table Number and Name** | **Page No** |
| 3 | 6.2.1 - Test Cases for phishing website | 68 |

|  |  |  |
| --- | --- | --- |
| **S.No** | **Screen Number and Name** | **Page No** |
| 1 | 5.3.1 - Output image of phishing website | 61 |
| 2 | 5.3.2- performance analysis output | 62 |

**LIST OF SCREENS**

**CHAPTER-1**

**INTRODUCTION**

**1. INTRODUCTION**

**What is Phishing?**

Phishing is a type of cyber-attack in which attackers create fake websites that look like legitimate ones to deceive users into providing sensitive information such as passwords, credit card details, and personal identification. These fake websites often closely resemble trusted platforms, making them difficult to distinguish from authentic ones.

**Role of Machine Learning in Phishing Detection**

Traditional phishing detection methods, such as blacklists and rule-based systems, are limited because they cannot effectively detect new and evolving phishing threats. Machine Learning (ML) provides a more intelligent and adaptive approach by analyzing various features of URLs, domains, and website content to identify phishing patterns.

This project proposes a Gradient Boosting Classifier (GBC) model for phishing detection. The system extracts multiple features from a URL and predicts whether it is a safe or phishing website based on trained data.

**1.1 MOTIVATION**

**Why is Phishing Detection Important?**

Phishing attacks have become one of the most **prevalent cybersecurity threats** worldwide. Every year, millions of users fall victim to phishing scams, leading to identity theft, financial fraud, and corporate data breaches.

**Challenges in Traditional Approaches**

1. **Blacklist-based detection**: Maintains a list of known phishing sites but fails against newly created phishing websites.
2. **Heuristic-based detection**: Uses manually defined rules to detect phishing but is not adaptable to evolving attacks.
3. **Human-based detection**: Relies on user awareness and education but is ineffective against well-crafted phishing attacks.

**Why Machine Learning?**

Machine learning models can analyze patterns in phishing websites and classify URLs dynamically. Unlike traditional approaches, ML-based systems can:

* Identify zero-day phishing attacks.
* Detect hidden patterns in URLs, domain age, and website behavior.
* Provide real-time phishing detection.

This motivated us to develop a machine learning-based phishing detection system using a Gradient Boosting Classifier (GBC), which improves accuracy over previous methods.

**1.2 PROBLEM DEFINITION**

The rise in phishing attacks has led to users unknowingly submitting sensitive information, such as bank details, credit card numbers, login credentials, and personal data, to fraudulent websites. Traditional detection methods, including blacklists and rule-based systems, struggle to identify newly created phishing sites as attackers continuously evolve their tactics. To address this challenge, this project aims to develop a machine learning-based phishing detection system that accurately identifies malicious websites. By extracting key features from URLs, domains, and website content, the system will detect phishing patterns and improve upon existing detection methods using a Gradient Boosting Classifier. The model will be deployed as a user-friendly Flask web application, enabling real-timeURL analysis and phishing probability assessment. This intelligent and adaptive system enhances security by providing users with real-time protection against emerging phishing threats.

**1.3 OBJECTIVE OF THE PROJECT**

The main objective of this project is to develop an efficient and accurate phishing website detectionsystem using machine learning algorithms. This system should:

**Extract Features from URLs**

* Analyze URL length, domain age, presence of special characters, and HTTPS usage.
* Identify patterns in domain registration and WHOIS details.

**Train a Machine Learning Model for Phishing Detection**

* Use Gradient Boosting Classifier (GBC) to classify URLs as phishing or safe.
* Improve accuracy by selecting relevant features.

**Provide a Web-Based Interface**

* Develop a Flask-based application where users can input a URL.
* Display whether the website is safe or phishing with confidence scores.

**Ensure High Accuracy and Low False Positives**

* Improve over traditional blacklist-based detection.
* Achieve a high accuracy rate with minimal false detections.

**1.4 LIMITATIONS OF THE PROJECT**

While the proposed system significantly improves phishing detection, it has some limitations:

**1. Dependence on Training Data**

* The accuracy of the machine learning model depends on the quality and quantity of training data.
* If the model is trained on a biased or incomplete dataset, it may not generalize well to new phishing websites.

**2. New Evasion Techniques**

* Attackers constantly evolve their phishing techniques to bypass detection.
* If phishing websites use advanced obfuscation techniques, the model may struggle to classify them correctly.

**3. Limited to URL-Based Detection**

* The project does not check email-based phishing attacks.
* It cannot detect phishing attempts made through SMS, phone calls, or social engineering.

**4. Model Interpretability**

* Although Gradient Boosting provides high accuracy, it is not as interpretable as decision trees.
* Users may not always understand why a URL is classified as phishing or safe.

Despite these limitations, the proposed system still provides **a powerful tool** for phishing detection compared to traditional methods.

**1.5 ORGANIZATION OF THE DOCUMENTATION**

**Chapter 1:** Introduction – This chapter introduces phishing attacks, their impact on cybersecurity, and the motivation behind the study. It outlines the problem statement, objectives, and limitations of the project while also providing an overview of the document structure.

**Chapter 2:** Literature Survey – It explores existing phishing detection methods, including rule-based, blacklist, and heuristic approaches, highlighting their limitations. The role of machine learning in phishing detection is discussed along with a review of related research.

**Chapter 3: Analysis** – This chapter focuses on dataset selection, feature extraction (URL-based, HTML-based, and behavioral features), and system architecture. It also discusses the machine learning algorithms considered, such as Decision Trees, SVM, Random Forest, and Neural Networks.

**Chapter 4: Design –** It presents system flow diagrams, module structures, and data processing techniques. The feature engineering process, model training framework, and preprocessing techniques used to enhance accuracy are explained.

**Chapter 5: Implementation & Results –** This chapter details the implementation of different machine learning models, their training and testing process, and performance evaluation. It includes key metrics such as accuracy, precision, recall, and F1-score, along with a comparative analysis of models.

**Chapter 6: Testing & Validation** – Test cases, cross-validation methods, and performance analysis are discussed. The chapter also examines false positives and false negatives and evaluates the security and robustness of the model.

**Chapter 7: Conclusion –** The final chapter summarizes the project’s outcomes, identifies limitations, and suggests future enhancements such as deep learning models, real-time detection, and browser extensions for improved phishing prevention**.**

**CHAPTER-2**

**LITERATURE SURVEY**

**2. LITERATURE SURVEY**

**2.1 INTRODUCTION**

The study of phishing detection techniques has evolved significantly over the years due to the increasing sophistication of cybercriminals. Phishing is a cyber-attack where an attacker deceives users into providing sensitive information such as login credentials, financial details, or personal databy impersonating legitimate entities. Various methods have been employed in phishing detection, including blacklists, rule-based approaches, and machine learning techniques. While traditional methods like blacklist-based detection are still in use, they are often ineffective against newly generated phishing URLs. Machine learning approaches, including Gradient Boosting Classifier, Decision Trees, and Neural Networks, have emerged as promising solutions to this challenge.

* 1. **EXISTING SYSTEM**

**1**.H. Huang et al., (2009) proposed the frameworks that distinguish the phishing utilizing pagesection similitude that breaks down universal resource locator tokens to create forecast preciseness phishing pages normally keep its CSS vogue like their objective pages.

**2**.S. Marchal et al., (2017) proposed this technique to differentiate Phishing website depends on the examination of authentic site server log knowledge. An application Off-the- Hook application or identification of phishing website. Free, displays a couple of outstandinproperties together with high preciseness, whole autonomy, and nice language-freedom, speed of selection, flexibility to dynamic phish and flexibility to advancement in phishingways.

**3**.Mustafa Aydin et al. proposed a classification algorithm for phishing website detection by extracting websites' URL features and analyzing subset based feature selection methods. Itimplements feature extraction and selection methods for the detection of phishing websites. The extracted features about the URL of the pages and composed feature matrix are categorized into five different analyses as Alpha- numeric Character Analysis, Keyword Analysis, Security Analysis, Domain Identity Analysis and Rank Based Analysis. Most of these features are the textual properties of the URL itself and others based on third parties services.

**4**.In the existing system they have used Logistic Regression, Multinomial Naive Bayes, and XG Boost are the machine learning methods that are compared. The Logistic Regression algorithm outperforms the other two. The model is preprocessed in the proposed system, the words are tokenized, and stemming is performed. Data Processing is the process of converting or encoding data for easy machine transfer. The accuracy of Logistic Regression is 96.63 percent, and the overall comparison is presented.

**2.3 DISADVANTAGES OF EXISTING SYSTEM**

1. The existing models have low latency.
2. Existing systems do not have a specific user interface.
3. The existing system model fails to predict a continuous outcome. It only works when the dependent or outcome variable is dichotomous.
4. The existing system model may not be accurate if the sample size is too small.
5. The existing may lead to overfitting problem.

**2.4 PROPOSED SYSTEM:**

* We have developed our project using a website as a platform for all the users. This is an interactive and responsive website that will be used to detect whether a website is legitimate or phishing. This website is made using different web designing languages which include HTML, CSS, Javascript and Flask framework in Python. The basic structure of the website is made with the help of HTML. CSS is used to add effects to the website and make it more attractive and user-friendly. It must be noted that the website is created for all users, hence it must be easy to operate with and no user should face any difficulty while making its use.
* The proposed system is trained with the dataset consists of different features and note that the dataset don’t contain any website URL. The dataset consists of different features that are to be taken into consideration while determining a website URL as legitimate or phishing.
* The proposed system is developed using the Gradient Boosting Classifier. After the system is trained with the dataset, the classifier identifies the given URL dependent on the preparation information that is if the site is phishing it prompts the user that the website is phished and if genuine, it prompts the user that the website is legitimate. We have detected phishing websites using Gradient Boosting Classifier with and accuracy of 97%.

#### ****2.5 Conclusion****

Phishing website detection has become a crucial area in cybersecurity due to the rising number of online fraud cases. Traditional methods such as blacklist-based detection are no longer sufficient due to their inability to detect newly emerging phishing attacks. Machine learning models provide a more robust solution by learning from historical phishing patterns. The proposed system, using Gradient Boosting Classifier, offers a highly accurate and efficient phishing detection mechanism. Future improvements may involve incorporating online learning techniques to adapt to evolving phishing tactics and further enhancing the model’s accuracy​.

**CHAPTER-3**

**ANALYSIS**

**3. ANALYSIS**

**3.1 INTRODUCTION**

The analysis phase is essential in software development as it helps in understanding system requirements, user expectations, and functional needs. This phase ensures that the system meets the specified objectives and is feasible for implementation.In this project, the phishing website detection system is analyzed based on hardware, software, and userrequirements. Additionally, flowcharts and algorithms are studied to ensure that the system performs effectively.

**3.2 SOFTWARE REQUIREMENT SPECIFICATION**

**3.2.1 User Requirements**

The phishing website detection system is designed for end-users who need to identify whether a website is legitimate or phishing. The system should meet the following user requirements:

1. User-Friendly Interface – The website should be easy to use, allowing users to enter a URL and receive results quickly.
2. Real-Time Analysis – The system should provide instant phishing detection based on machine learning predictions.
3. Accuracy and Efficiency – The model should ensure high accuracy in detecting phishing websites with minimal false positives.
4. Cross-Platform Compatibility – The system should be accessible via desktop, mobile, or tablet without requiring additional installations.
5. Security and Data Privacy – User-provided URLs should be processed securely without storing personal data.

**3.2.2 SOFTWARE REQUIREMENTS:**

The project is developed on **Windows 10** as the operating system, ensuring compatibility with various development tools and libraries. The implementation is carried out using **Python**, a versatile and powerful programming language known for its efficiency in handling machine learning tasks. To build and deploy the web-based application, **Flask**, a lightweight and flexible web framework, is utilized. Flask provides essential features for developing a scalable and interactive phishing detection system, allowing seamless integration of machine learning models into a user-friendly web interface

.**3.2.3 Hardware requirement**

The system requirements for this project ensure smooth execution and efficient performance. The project runs on a **Pentium i3 Processor**, providing sufficient processing power for machine learning computations. It is equipped with a **500 GB hard disk**, offering ample storage space for datasets, models, and application files. A **15” LED monitor** is used for display, ensuring clear visualization of results and web interfaces. The input devices include a **keyboard and mouse** for user interaction. Additionally, the system is supported by **4 GB of RAM**, which enables smooth multitasking and efficient processing of machine learning tasks.

**3.4 Algorithms**

**Feature-Based Phishing Website Detection (Software-Based)**

* **Objective:** To detect phishing websites by analyzing various features extracted from URLs, HTML content, and domain-related information.
* **Data Acquisition:**
  + Website features such as URL length, presence of special characters, and redirection behavior are extracted.
  + HTML and JavaScript-based features, including embedded links and form handling methods, are analyzed.
  + Domain-based information, such as domain age and DNS records, is retrieved to assess legitimacy.
* **Analysis:**
  + A **Random Forest** classifier processes these features to differentiate between phishing and legitimate websites.
  + The model evaluates each parameter and assigns a probability score indicating the likelihood of phishing.
* **Output:**
  + The system classifies the website as either **phishing** or **legitimate**, providing an explanation based on feature analysis.

**Machine Learning-Based Phishing Detection (Software-Based)**

* **Objective:** To enhance phishing detection accuracy using machine learning algorithms trained on phishing and legitimate website datasets.
* **Input:** The system processes website URLs and extracts critical phishing indicators.
* **Processing:**
  + Feature engineering is performed to extract relevant phishing indicators from the URL and webpage content.
  + Machine learning models such as **Support Vector Machines (SVM)** and **Neural Networks** analyze the extracted features.
  + The trained model classifies the website into either phishing or legitimate categories based on its characteristics.
* **Output:**The system provides a phishing probability score and explains the reasoning behind the classification, helping users make informed decisions.

**Real-Time Phishing Detection Using Browser Extensions (Software & API-Based)**

* **Objective:** To provide real-time phishing detection by integrating machine learning models with a browser extension.
* **Data Acquisition:**
  + When a user visits a website, the browser extension extracts URL and webpage features.
  + API calls retrieve additional security data such as SSL certificates and WHOIS information.
* **Analysis:**
  + A **Deep Learning** model processes extracted data to detect phishing attempts in real time.
  + The system cross-checks the website with known phishing databases.
* **Output:**
  + If the website is identified as phishing, the extension alerts the user with a warning message and possible risks.
  + Recommendations are provided for safe browsing practices, reducing the likelihood of falling victim to phishing attacks.

This approach ensures a robust phishing detection mechanism by combining feature-based analysis, machine learning classification, and real-time detection for enhanced cybersecurity.

**3.5 Flowchart**

This flowchart explains the working process of the Phishing Website Detection System by subdividing it into three main functionalities:

1. Feature Extraction from URLs
2. Machine Learning-Based Phishing Classification
3. Real-Time Phishing Detection & User Alert

****

Figure 3.5 – Project Flowchart

**Different Phases in the Flowchart – Step-by-Step Approach**

**1. Feature Extraction from URLs**

* The system extracts essential features from the given URL, including length, special characters, presence of subdomains, and redirection behavior.
* Additional data, such as WHOIS information, SSL certificate status, and domain age, is retrieved for deeper analysis.
* If sufficient information is extracted, the process moves to the next step. Otherwise, the system flags the URL as “Insufficient Data” and asks for manual review.

**2. Machine Learning-Based Phishing Classification**

* The extracted features are processed through a pre-trained machine learning model (Random Forest, SVM, or Neural Network).
* The model evaluates the likelihood of phishing by analyzing URL-based, domain-based, and HTML-based features.
* The system classifies the URL into one of the following categories:
  + Legitimate Website → No further action is needed.
  + Phishing Website Detected → The system proceeds to alert the user.
  + Suspicious Website (Uncertain) → Further verification using heuristic or deep learning analysis is performed.

**3. Real-Time Phishing Detection & User Alert**

* If a website is flagged as phishing, the system triggers an alert message.
* The phishing probability score is displayed along with recommendations such as avoiding the website, reporting it, or verifying security certificates.
* If a website is legitimate, the system safely allows access without warnings.
* In the case of a suspicious website, the system suggests additional validation through third-party security tools or manual review.

**3.6 Conclusion**

The analysis phase of the Phishing Website Detection System establishes a well-defined framework for integrating machine learning models into cybersecurity. The combination of feature extraction, classification, and real-time detection ensures an efficient and accurate phishing identification process. The software and hardware requirements ensure seamless data processing, while the flowchart illustrates the interaction between system components. Algorithms such as Random Forest for classification and Deep Learning for advanced detection enhance the efficiency of decision-making. By leveraging real-time threat analysis and automation, this system offers a scalable and proactive approach to phishing prevention, significantly improving online security.

**CHAPTER-4**

**DESIGN**

**4.1 INTRODUCTION**

The design phase is crucial in developing the Phishing Website Detection System as it defines the architecture, data flow, and module organization. This phase includes various design diagrams, such as Data Flow Diagrams (DFD), Entity-Relationship (ER) diagrams, and Unified Modeling Language (UML) diagrams, which illustrate how different system components interact​.

* The primary objective of the design phase is to:
* Define system structure and interaction between components.
* Illustrate data processing and flow using diagrams.
* Ensure modular organization for scalability and efficiency.

**4.2 UML Diagrams**

**UML (Unified Modeling Language)** is a standardized modeling language used in software engineering to visualize, design, and document a system's architecture. It provides a set of diagrams to represent different aspects of a system, such as structure, behavior, and interactions.

### **Types of UML Diagrams**

### UML diagrams are broadly classified into two categories:

* **Structural Diagrams** (Focus on system components and their relationships)
  + **Class Diagram** – Represents the classes, attributes, methods, and relationships.
  + **Component Diagram** – Illustrates how different software components interact.
  + **Deployment Diagram** – Represents the hardware and software deployment.
* **Behavioral Diagrams** (Focus on system functionality and interactions)
  + **Use Case Diagram** – Depicts system interactions with external entities (actors).
  + **Sequence Diagram** – Represents the sequence of interactions between objects.
  + **Activity Diagram** – Describes workflows or processes.
  + **State Diagram** – Shows states and transitions of an object.
  + **Collaboration Diagram** – Represents object interactions focusing on links.

**1.Use Case Diagram**

**Introduction**

* + The use case diagram represents the interaction between the **User** and the **Phishing Website Detection System**. The system processes the user-inputted dataset, performs **preprocessing and training**, classifies the website, and provides the final result—whether the website is **phishing or legitimate**.

**Actors**

* + **User** – The primary actor who interacts with the system to detect phishing websites.

**Use Cases**

* + **Input Dataset** – The user provides a dataset containing URLs or website attributes for phishing detection.
  + **Preprocessing & Training** – The system cleans, processes, and trains machine learning models using the input dataset.
  + **Classification** – The trained model analyzes website features and classifies the website as **phishing or legitimate**.
  + **Result: Phishing Website or Legitimate Website** – The system provides the final decision based on model predictions.

**Description of the Workflow**

* + The **User** starts by providing an input dataset (which may contain URLs and extracted features).
  + The system **preprocesses the data**, removing irrelevant information and formatting it for machine learning algorithms.
  + The **machine learning model is trained** using features such as URL length, redirections, and HTTPS usage.
  + After training, the system **classifies the given website** based on the extracted features.
  + Finally, the system provides the **detection result**, informing the user whether the website is **phishing or legitimate**.

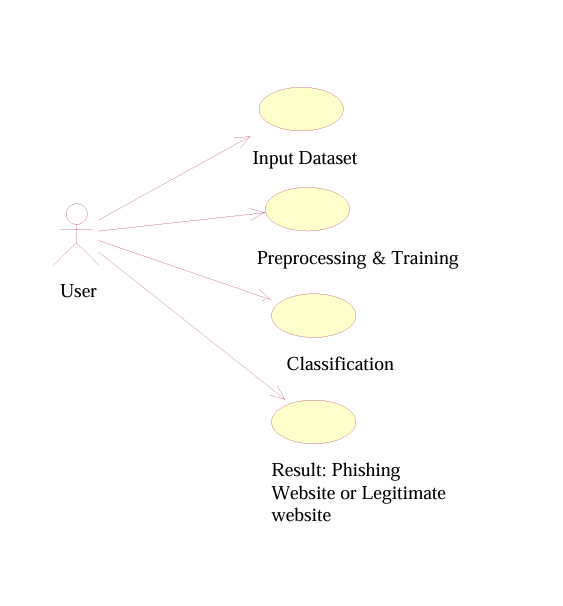
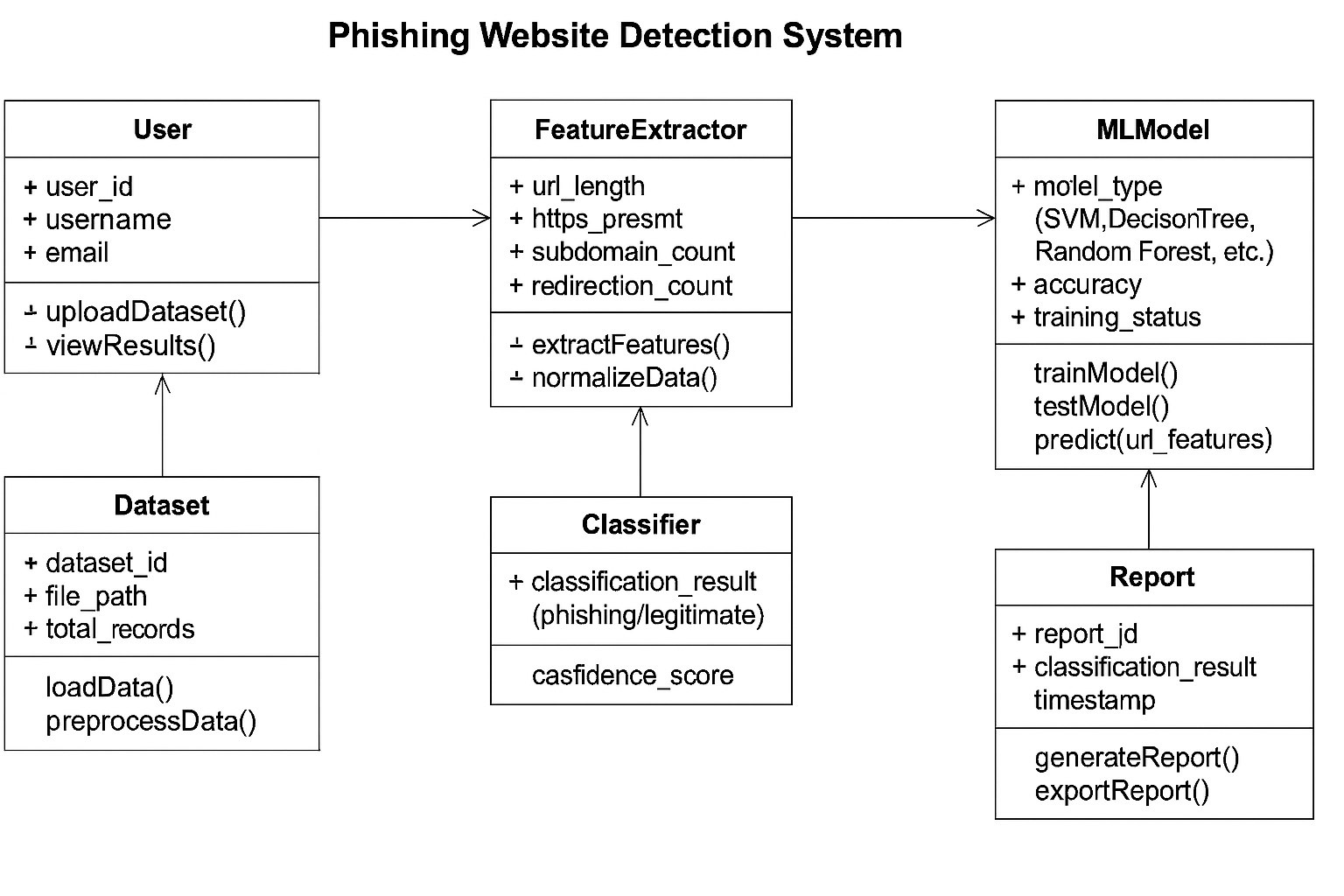


Figure 4.2.1- Use Case Diagram

**2.Class Diagram**

**** Figure 4.2.2 – Class Diagram

**Main Classes:**

1. **User**
   * Attributes: user\_id, username, email
   * Methods: uploadDataset(), viewResults()
   * Description: The user interacts with the system by uploading datasets and viewing phishing detection results.
2. **Dataset**
   * Attributes: dataset\_id, file\_path, total\_records
   * Methods: loadData(), preprocessData()
   * Description: Stores the dataset containing website URLs and features, and handles data preprocessing.
3. **FeatureExtractor**
   * Attributes: url\_length, https\_present, subdomain\_count, redirection\_count
   * Methods: extractFeatures(), normalizeData()
   * Description: Extracts relevant features from URLs and preprocesses them for the classification model.
4. **MLModel**
   * Attributes: model\_type (SVM, Decision Tree, Random Forest, etc.), accuracy, training\_status
   * Methods: trainModel(), testModel(), predict(url\_features)
   * Description: Handles machine learning model training, testing, and classification of websites.
5. **Classifier**
   * Attributes: classification\_result (phishing/legitimate), confidence\_score
   * Methods: classifyWebsite()
   * Description: Uses the trained ML model to classify websites as phishing or legitimate based on extracted features.
6. **Report**
   * Attributes: report\_id, classification\_result, timestamp
   * Methods: generateReport(), exportReport()
   * Description: Generates a report containing classification results and confidence scores.

**Relationships:**

* User uploads a Dataset for analysis.
* Dataset is processed by FeatureExtractor to obtain website attributes.
* FeatureExtractor sends extracted data to the MLModel for training and prediction.
* MLModel provides results to the Classifier, which determines whether a website is phishing or legitimate.

**Activity Diagram**

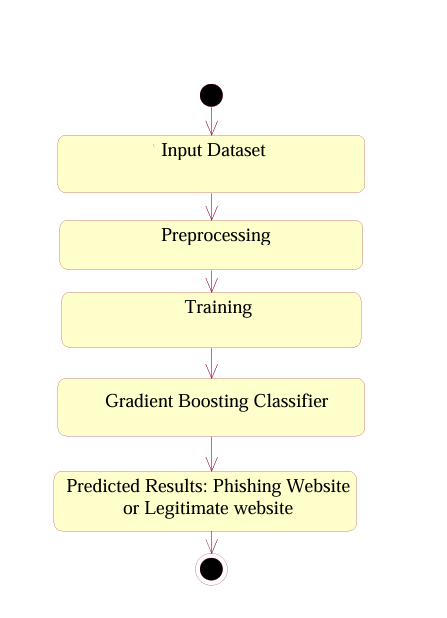
****

Figure 4.2.3 - Activity Diagram

**Process Flow:**

1. **User Accesses System**
   * The user selects “Start Detection” from the home page.
   * Decision Check:
     + If “No,” the user remains on the home page.
     + If “Yes,” proceed to the next steps.
2. **Uploading Website Data**
   * The user uploads a dataset of URLs for analysis.
   * If no dataset is uploaded, prompt the user to upload a valid file.
   * If a dataset is uploaded successfully, move to feature extraction.
3. **Feature Extraction & Preprocessing**
   * System extracts features such as URL length, HTTPS presence, redirection count, subdomains, and domain age.
   * Data is preprocessed by normalizing values and handling missing data.
4. **Model Selection & Training**
   * Decision Check:
     + If no trained model exists, initiate model training using algorithms like Random Forest, Decision Tree, or SVM.
     + If a trained model exists, proceed directly to classification.
5. **Classification of Websites**
   * The system classifies each URL as either Phishing or Legitimate based on extracted features.
   * Decision Check:
     + If the website is classified as Legitimate, display a confirmation message.
     + If classified as Phishing, alert the user with details and a confidence score.
6. **Report Generation**
   * The system generates a report containing classification results and confidence levels.
   * Users can download or view the report for further analysis.
7. **End Process**
   * The user can choose to restart the detection process or exit the system.

**State Transition Diagram**

**Key States in the State Transition Diagram**

1. **IdleStateStart**
   * The system is in an idle state, waiting for user input.
   * Transitions to DataUpload when a user starts the detection process.
2. **DataUpload**
   * The user uploads a dataset of URLs for phishing detection.
   * Decision Check:
     + If NoDataset, the system remains in IdleStateStart.
     + If DatasetUploaded, transitions to FeatureExtraction.
3. **FeatureExtraction**

* Extracts key features such as URL length, presence of HTTPS, redirections, and domain age.
* Transitions to ModelTraining if training is required or directly to Prediction if a trained model exists.

1. **Model Training**
   * If no trained model exists, the system trains models using machine learning algorithms (Random Forest, Decision Tree, SVM).
   * Once the model is trained, transitions to Prediction.
2. **Prediction**
   * The system classifies websites as Phishing or Legitimate based on extracted features.
   * Decision Check:
     + If Legitimate, transition to ReportGeneration.
     + If Phishing, transition to AlertUser.
3. **AlertUser (For Phishing Websites)**
   * If a website is classified as Phishing, the system displays a warning.
   * Transitions to IdleStateEnd or allows the user to scan another website.

**7**.**Report Generation**

* + Generates a report with classification results and confidence levels.
  + The user can choose to restart detection or exit the system.

**8. IdleStateEnd**

* + The process concludes, and the system returns to an idle state, ready for new user input.

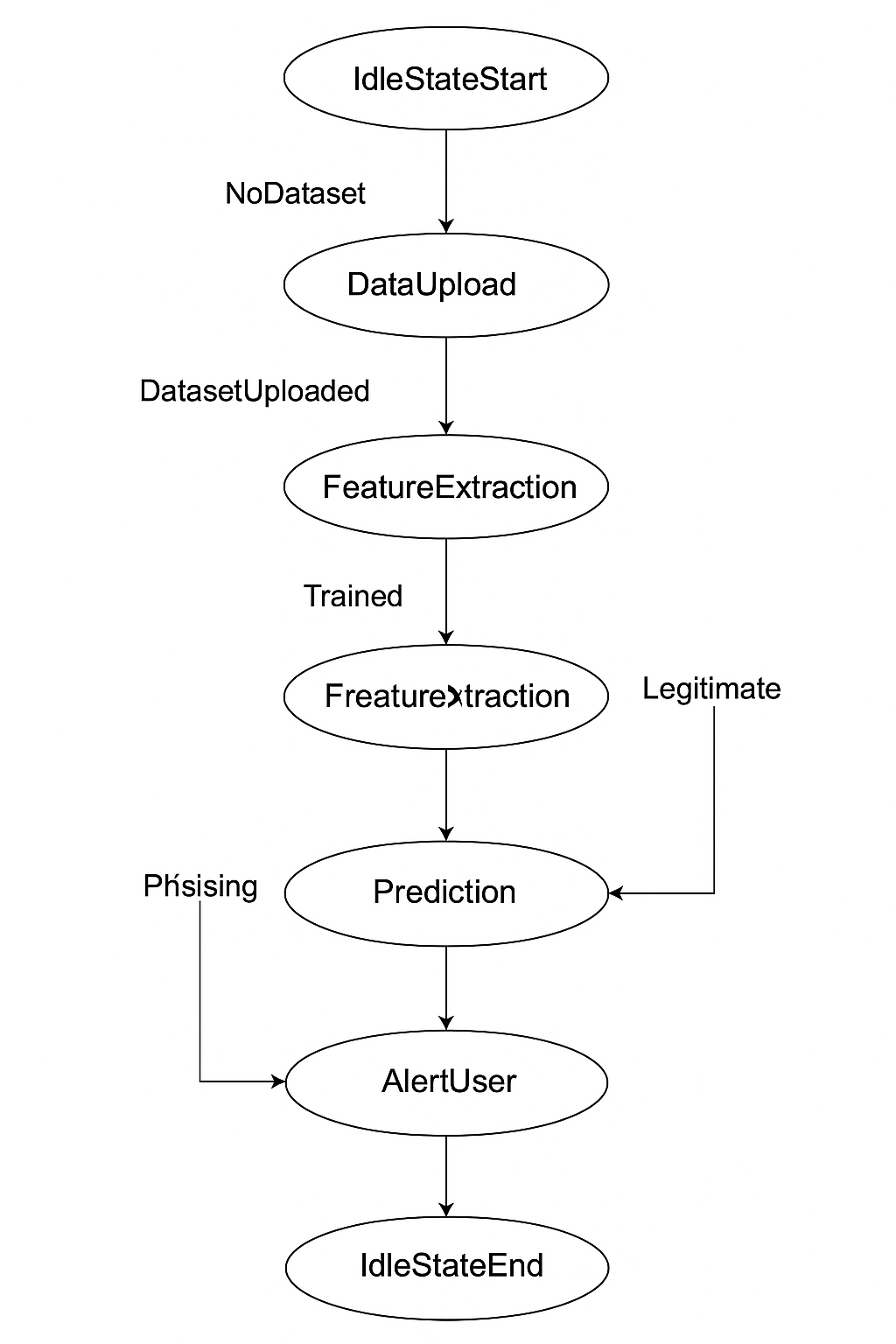
****

Figure 4.2.4 – State Transition Diagram

**Flow of Sequence Diagram:**

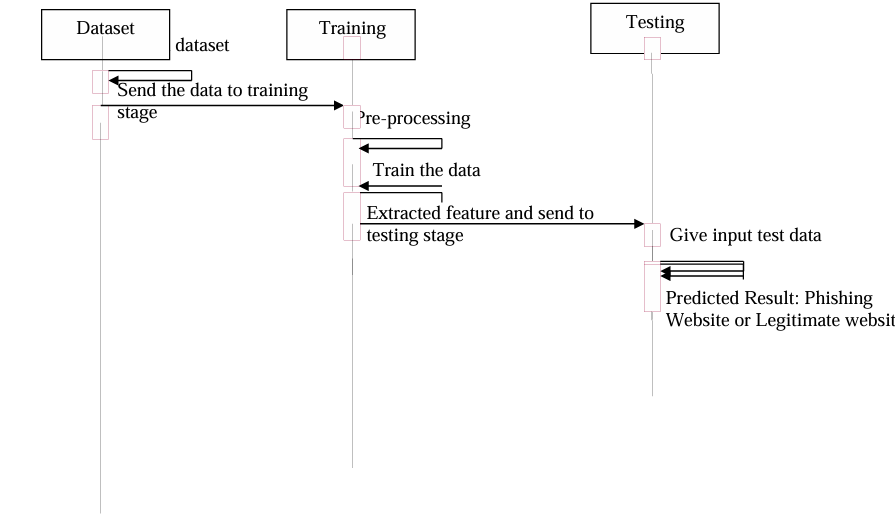
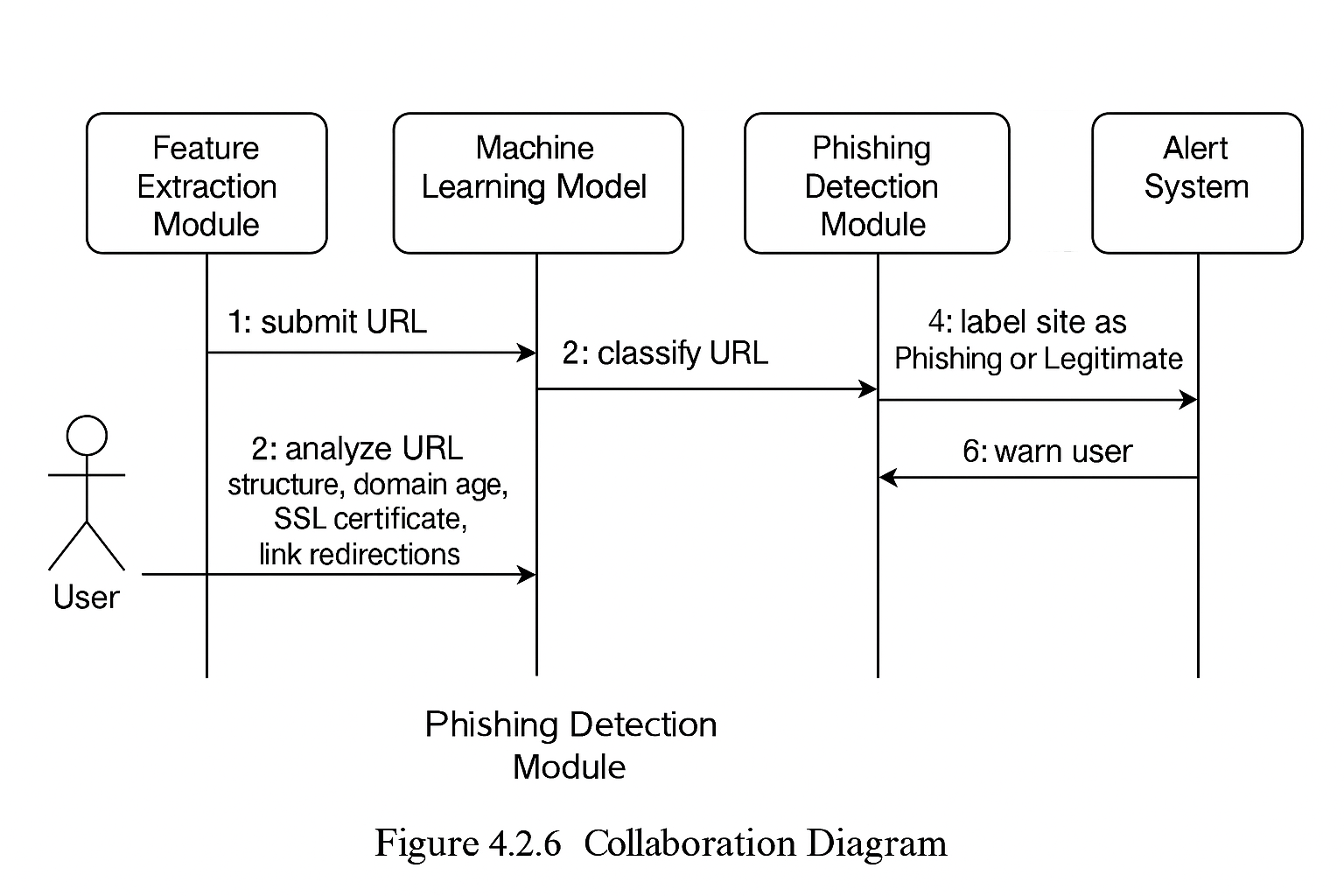


Figure 4.2.5 – Sequence Diagram

1. User submits a URL for analysis.
2. Feature Extraction Module processes the URL to extract attributes such as domain age, HTTPS status, presence of special characters, etc.
3. The Machine Learning Model (e.g., Decision Tree, Random Forest, or Neural Network) analyzes the extracted features.
4. The model classifies the website as Legitimate, Suspicious, or Phishing.
5. The Database logs the result for future reference and model improvements.
6. The System returns the classification result to the user.

**Collaboration Diagram**

A Collaboration Diagram (also called a Communication Diagram) illustrates interactions between system components, showing the flow of messages between objects.

****

**Actors & Components in the System:**

1. User – Submits a URL for verification.
2. Feature Extraction Module – Extracts website features like domain age, HTTPS usage, URL structure, etc.
3. Machine Learning Model – Uses trained ML algorithms (e.g., Decision Tree, SVM, Neural Networks) to classify the website.
4. Phishing Detection Module – Determines if the website is phishing or legitimate.
5. Database Module – Stores extracted features and results for further analysis.
6. Alert System – Notifies the user if phishing is detected.

**Flow of Interactions:**

1. User submits a URL for phishing detection.
2. Feature Extraction Module analyzes URL structure, domain age, SSL certificate, and link redirections.
3. Machine Learning Model classifies the URL based on extracted features.
4. Phishing Detection Module labels the site as Phishing or Legitimate.
5. Database Module logs results for training and future reference.
6. Alert System warns the user if phishing is detected

**Component Diagram**

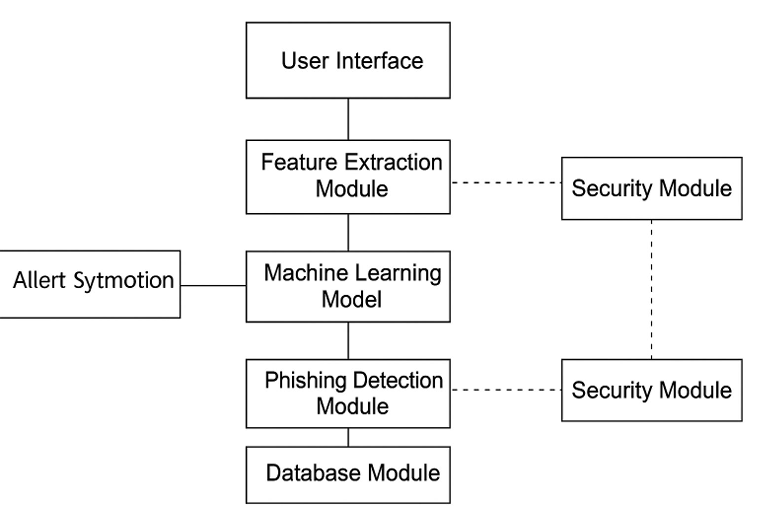
****

Figure 4.2.7 – Component Diagram

**Components in the System:**

1. **User Interface** – Allows users to input URLs for detection.
2. **Feature Extraction Module** – Extracts domain-based, content-based, and behavior-based features.
3. **Machine Learning Model** – Processes extracted features using ML classifiers (Random Forest, SVM, Neural Network).
4. **Phishing Detection Module** – Determines phishing probability.
5. **Database Module** – Stores results, logs phishing URLs, and updates training data.
6. **Security Module** – Ensures data protection, encryption, and user authentication.
7. **Alert System** – Notifies users of detected phishing threats.

**Relationships:**

* **User Interface → Feature Extraction Module** (User submits URL).
* **Feature Extraction Module → Machine Learning Model** (Extracts and processes URL features).
* **Machine Learning Model → Phishing Detection Module** (Predicts phishing probability).
* **Phishing Detection Module → Alert System** (Warns user if phishing detected).
* **Phishing Detection Module → Database Module** (Logs results and updates datasets).
* Security Module ensures encrypted data handling and authentication**.**

**Deployment Diagram**

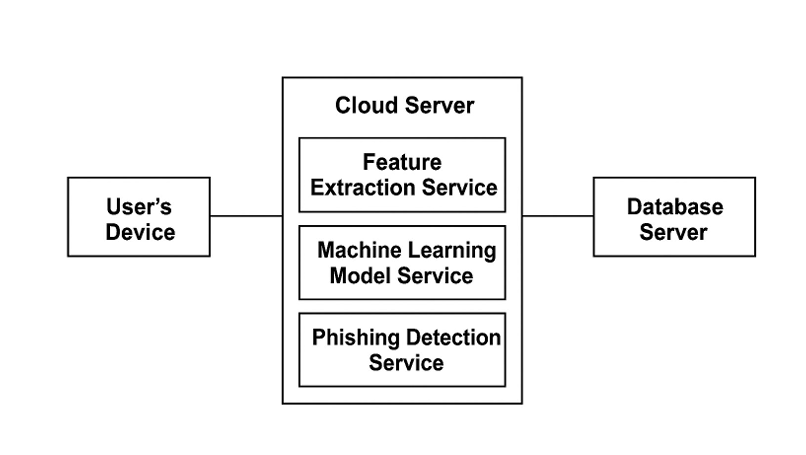
****

Figure 4.2.8 – Deployment Diagram

**Key Components in the Deployment Architecture:**

**User’s Device**

* Allows users to input URLs and view detection results.

**Cloud Server**

* **Feature Extraction Service** – Extracts website characteristics.
* **Machine Learning Model Service** – Classifies websites as phishing or legitimate.

**Phishing Detection Service** – Analyzes results and generates reports.

**Database Server**

* Stores website details, extracted features, and classification results.

**Working Process:**

* **User submits a URL via the interface** (desktop, mobile, or web app).
* **Feature Extraction Service** processes the URL and gathers metadata.
* **Machine Learning Model Service** applies ML algorithms for classification.
* **Phishing Detection Service** decides whether the URL is phishing or legitimate.
* **Database Server** logs URL analysis results for future training.
* **The User Interface** displays the final classification (Phishing/Legitimate).

**4.3 Module Design and Organization**

The system architecture of the **Phishing Website Detection System** consists of two primary components: **data processing and model classification**.

**1. Data Processing Components:**

* **Feature Extraction Module:** Extracts essential website attributes such as domain age, HTTPS usage, URL structure, and redirection behavior.
* **URL & HTML Analysis:** Analyzes webpage content, embedded links, and suspicious script usage.
* **IP & DNS Checker:** Examines the hosting server’s IP reputation, WHOIS information, and domain registration details.

**2. Machine Learning-Based Classification:**

* **Machine Learning Model:** Utilizes algorithms like Decision Trees, Random Forest, Support Vector Machines (SVM), or Neural Networks to classify a website as **phishing or legitimate**.
* **User Interface:** Provides an interactive platform where users can input a URL for analysis and receive instant feedback on its legitimacy.

The **feature extraction** and **machine learning components** work together to analyze website properties, classify URLs, and provide users with warnings against phishing threats.

**Phishing Website Detection (Feature Extraction & Classification)**

**Data Acquisition:**

* User submits a URL for phishing detection.
* The system extracts features such as:
  + **URL-based features:** Length of the URL, presence of special characters (@, //, -).
  + **Domain-based features:** Domain age, WHOIS information, IP reputation.
  + **Content-based features:** Presence of iframe redirection, suspicious JavaScript, embedded links.
  + **HTTPS & SSL features:** SSL certificate validity, HTTPS usage.

**Analysis:**

* The system applies machine learning algorithms to classify the URL.
* Features are processed using trained models such as Decision Trees, Random Forest, or Deep Learning (Neural Networks).
* The model outputs a classification: **Phishing** or **Legitimate**.

**Working:**

* The input URL undergoes feature extraction.
* Extracted features are passed to the trained model.
* The model predicts whether the URL is phishing or legitimate.

**Output:**

* The system displays results such as:
  + URL classification (Phishing/Legitimate).
  + A confidence score for classification.
  + Security recommendations for users.

**Phishing Website Detection (Threat Analysis & Alert System)**

**Threat Detection Module:**

* **IP & DNS Check:** Identifies phishing patterns based on hosting server reputation and domain registration details.
* **Website Redirection Check:** Analyzes how many times a website redirects users to another domain.
* **Malware & Blacklist Check:** Verifies if the website is listed in phishing databases such as Google Safe Browsing.

**Processing:**

* The extracted features are compared with known phishing patterns.
* If a threat is detected, the system categorizes the site as high-risk.

**Output:**

* The system provides:
  + A **phishing score** (low, medium, high risk).
  + A warning if the site is blacklisted.
  + Recommendations for avoiding phishing attacks.

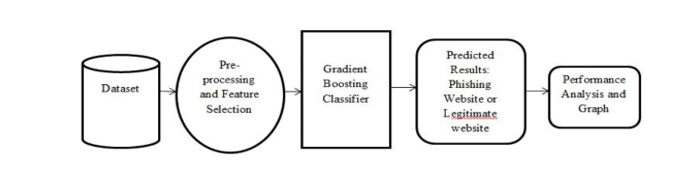


Figure 4.3.1- System Architecture of phishing website detection

**Machine Learning Model Training (Software-Based)**

This module is responsible for training the phishing detection model.

**Input:**

* A dataset containing features of both phishing and legitimate websites.
* Labeled data indicating whether a website is phishing (1) or legitimate (0).

**Processing:**

* The dataset undergoes **preprocessing** (cleaning, feature selection).
* Machine learning algorithms train the model to classify new websites.
* Performance metrics (accuracy, precision, recall) are evaluated.

**Output:**

* A trained model capable of detecting phishing websites.
* A report on model performance and accuracy.

### **4.4 Conclusion**

The **Phishing Website Detection System using Machine Learning** represents a significant advancement in cybersecurity by leveraging AI-driven techniques to identify and mitigate online phishing threats. This system’s modular architecture ensures a seamless interaction between **feature extraction, machine learning classification, and alert mechanisms**, enabling users to make informed decisions about website safety in real time.

By integrating **machine learning, real-time threat analysis, and automated detection**, this system enhances cybersecurity awareness and reduces the risk of phishing attacks. The ability to extract crucial website features, analyze URLs, and detect malicious patterns empowers both individual users and organizations to protect sensitive data from cyber threats.

Furthermore, the **scalability and adaptability** of this system make it a valuable asset in the evolving landscape of cybersecurity. As cybercriminals continue to refine their phishing techniques, this system can be enhanced with more sophisticated deep learning models, real-time web crawling, and improved threat intelligence databases. Future developments could include **browser extensions, API-based integrations, and automated phishing reporting systems** to further enhance security measures.

In conclusion, the **Phishing Website Detection System** provides an innovative and practical solution to the growing challenge of online phishing attacks. With its **real-time detection capabilities, automated decision-making, and integration into cybersecurity frameworks**, this system has the potential to significantly reduce online fraud and enhance user protection. As cyber threats continue to evolve, the continued refinement and widespread adoption of **AI-driven phishing detection** will be crucial in safeguarding digital ecosystems and ensuring a secure internet experience for users worldwide.

**CHAPTER-5**

**IMPLEMENTATION & RESULTS**

**5. IMPLEMENTATION & RESULTS**

* 1. **INTRODUCTION**

The implementation phase is where the phishing website detection system is developed based on the design and analysis. It involves writing the code, integrating machine learning models, and deploying web application. The project is implemented using Flask for the web interface and Gradient Boosting Classifier (GBC) for phishing detection.The system allows users to input a URL, which is analyzed model to determine whether the website is phishing or legitimate. This phase also covers performance evaluation and result analysis.

**5.2 EXPLANATION OF KEY FUNCTIONS**

**Data Collection**

The dataset is referred from the popular dataset repository called kaggle. The following is the

dataset link for the Detection of Phishing Websites Using Machine Learning. Kaggle Dataset

Link: https://www.kaggle.com/datasets/jayaprakashpondy/phishing-websites-feature dataset

**Dataset:**

The dataset consists of 11054 individual data. There are 32 columns in the dataset, which are below.

**Index**: index id

**UsingIP**: (categorical - signed numeric) : { -1,1 }

**LongURL**: (categorical - signed numeric) : { 1,0,-1 }

**ShortURL**: (categorical - signed numeric) : { 1,-1 }

**Symbol@**: (categorical - signed numeric) : { 1,-1 }

**Redirecting**:// (categorical - signed numeric) : { -1,1 }

**PrefixSuffix**-: (categorical - signed numeric) : { -1,1 }

**SubDomains**: (categorical - signed numeric) : { -1,0,1 }

**HTTPS**: (categorical - signed numeric) : { -1,1,0 }

**DomainRegLen**: (categorical - signed numeric) : { -1,1 }

**Favicon**: (categorical - signed numeric) : { 1,-1 }

**NonStdPort**: (categorical - signed numeric) : { 1,-1 }

**HTTPSDomainURL**: (categorical - signed numeric) : { -1,1 }

**RequestURL**: (categorical - signed numeric) : { 1,-1 }

**AnchorURL**: (categorical - signed numeric) : { -1,0,1 }

**LinksInScriptTags**: (categorical - signed numeric) : { -1,0,1 }

**ServerFormHandler:** (categorical - signed numeric) : { -1,0,1 }

**InfoEmail**: (categorical - signed numeric) : { -1,1 }

**AbnormalURL**: (categorical - signed numeric) : { -1,1 }

**WebsiteForwarding**: (categorical - signed numeric) : { 0,1 }

**StatusBarCust**: (categorical - signed numeric) : { -1,1 }

**DisableRightClick**: (categorical - signed numeric) : { -1,1 }

**UsingPopupWindow**: (categorical - signed numeric) : { -1,1 }

**IframeRedirection**: (categorical - signed numeric) : { -1,1 }

**AgeofDomain**: (categorical - signed numeric) : { -1,1 }

**DNSRecording**: (categorical - signed numeric) : { -1,1 }

**WebsiteTraffic**: (categorical - signed numeric) : { -1,0,1 }

**PageRank**: (categorical - signed numeric) : { -1,1 } G

**GoogleIndex**: (categorical - signed numeric) : { -1,1 }

**LinksPointingToPage**: (categorical - signed numeric) : { -1,0,1 }

**StatsReport:** (categorical - signed numeric) : { -1,1 }

**Class**: (categorical - signed numeric) : { -1,1 }

**Data Preparation**:

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.). Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data. Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!) perform other exploratory analysis. Split into training and evaluation sets.

**Model Selection:**

We used Gradient Boosting Classifier machine learning algorithm. We got an accuracy of training Accuracy 98.9% so we implemented this algorithm. Gradient Boosting Classifier Algorithm What is boosting? While studying machine learning you must have come across this term called Boosting. It is the most misinterpreted term in the field of Data Science. The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present the first model. Let me try to explain to you what exactly does this means and how does this works. Suppose you have n data points and 2 output classes (0 and 1). You want to create a model to detect the class of the test data. Now what we do is randomly select observations from the training dataset and feed them to model 1 (M1), we also assume that initially, all the observations have an equal weight that means an equal probability of getting selected. Remember in ensembling techniques the weak learners combine to make a strong model so here M1, M2, M3….Mn all are weak learners. Since M1 is a weak learner, it will surely misclassify some of the observations. Now before feeding the observations to M2 what we do is update the weights of the observations which are wrongly classified. You can think of it as a bag that initially contains 10 different color balls but after some time some kid takes out his favorite color ball and put 4 red color balls instead inside the bag. Now off-course the probability of selecting a red ball is higher. This same phenomenon happens in Boosting techniques, when an observation is wrongly classified, its weight get’s updated and for those which are correctly classified, their weights get decreased. The probability of selecting a wrongly classified observation gets increased hence in the next model only those observations get selected which were misclassified in model 1. Similarly, it happens with M2, the wrongly classified weights are again updated and then fed to M3. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly. Now when the new datapoint comes in (Test data) it passes through all the models (weak learners) and the class which gets the highest vote is the output for our test data. What is a Gradient boosting Algorithm? The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. But how do we do that? How do we reduce the error? This is done by building a new model on the errors or residuals of the previous model. When the target column is continuous, we use Gradient Boosting Regressor whereas when it is a classification problem, we use Gradient Boosting Classifier. The only difference between the two is the “Loss function”. The objective here is to minimize this loss function by adding weak learners using gradient descent. Since it is based on loss function hence for regression problems, we’ll have different loss functions like Mean squared error (MSE) and for classification, we will have different for e.g log-likelihood.

**Analyze and Prediction**:

In the actual dataset, we chose only 30 features:

UsingIP: (categorical - signed numeric) : { -1,1 }

LongURL: (categorical - signed numeric) : { 1,0,-1 }

ShortURL: (categorical - signed numeric) : { 1,-1 }

Symbol@: (categorical - signed numeric) : { 1,-1 }

Redirecting:// (categorical - signed numeric) : { -1,1 }

PrefixSuffix-: (categorical - signed numeric) : { -1,1 }

SubDomains: (categorical - signed numeric) : { -1,0,1 }

HTTPS: (categorical - signed numeric) : { -1,1,0 }

DomainRegLen: (categorical - signed numeric) : { -1,1 }

Favicon: (categorical - signed numeric) : { 1,-1 }

NonStdPort: (categorical - signed numeric) : { 1,-1 }

HTTPSDomainURL: (categorical - signed numeric) : { -1,1 }

RequestURL: (categorical - signed numeric) : { 1,-1 }

AnchorURL: (categorical - signed numeric) : { -1,0,1 }

LinksInScriptTags: (categorical - signed numeric) : { -1,0,1 }

ServerFormHandler: (categorical - signed numeric) : { -1,0,1 }

InfoEmail: (categorical - signed numeric) : { -1,1 }

AbnormalURL: (categorical - signed numeric) : { -1,1 }

WebsiteForwarding: (categorical - signed numeric) : { 0,1 }

StatusBarCust: (categorical - signed numeric) : { -1,1 }

UsingPopupWindow: (categorical - signed numeric) : { -1,1 }

IframeRedirection: (categorical - signed numeric) : { -1,1 }

AgeofDomain: (categorical - signed numeric) : { -1,1 }

DNSRecording: (categorical - signed numeric) : { -1,1 }

WebsiteTraffic: (categorical - signed numeric) : { -1,0,1 }

PageRank: (categorical - signed numeric) : { -1,1 }

GoogleIndex: (categorical - signed numeric) : { -1,1 }

LinksPointingToPage: (categorical - signed numeric) : { -1,0,1 }

StatsReport: (categorical - signed numeric) : { -1,1 }

Class: (categorical - signed numeric) : { -1,1 }

**Accuracy on test set:** We got an accuracy of 97.6% on test set.

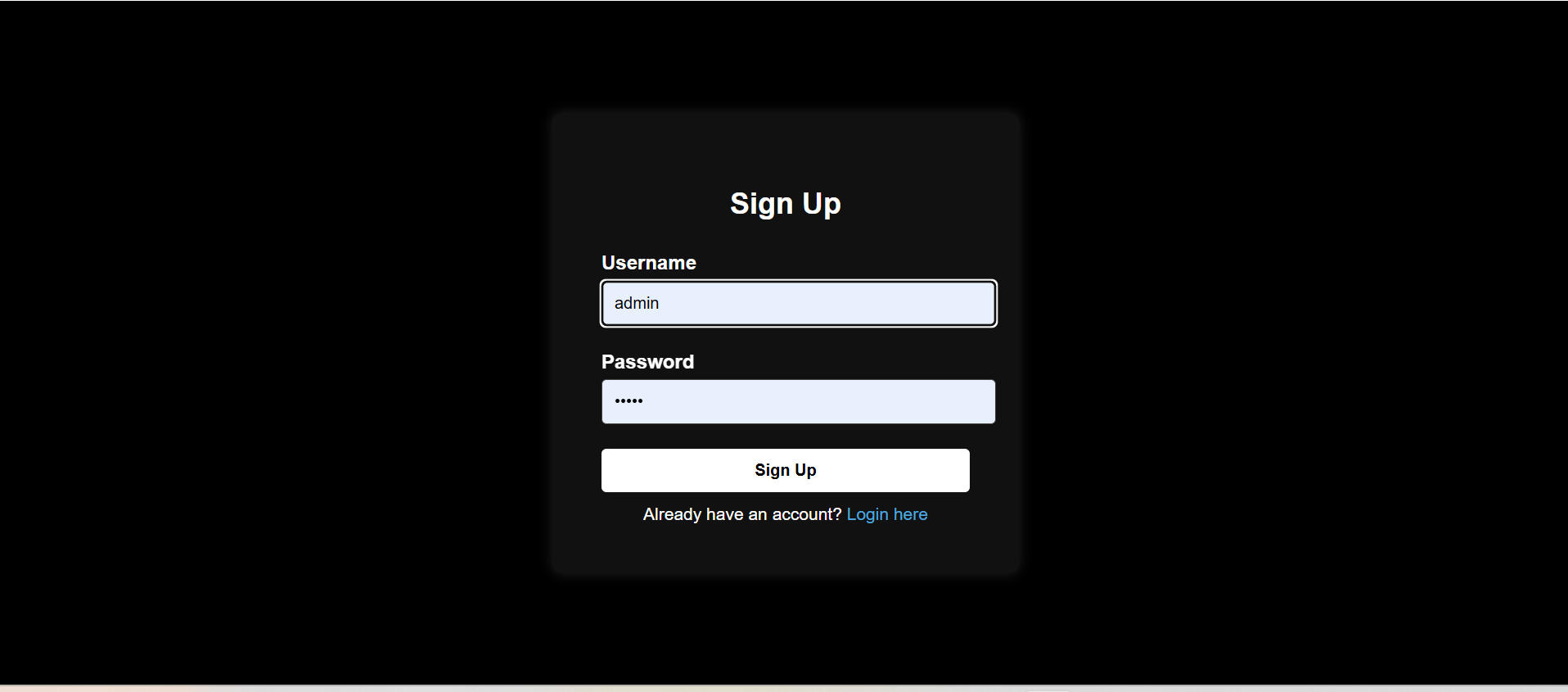
**Saving the Trained Model**: Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle. Make sure you have pickle installed in your environment. Next, let’s import the module and dump the model into. pkl file

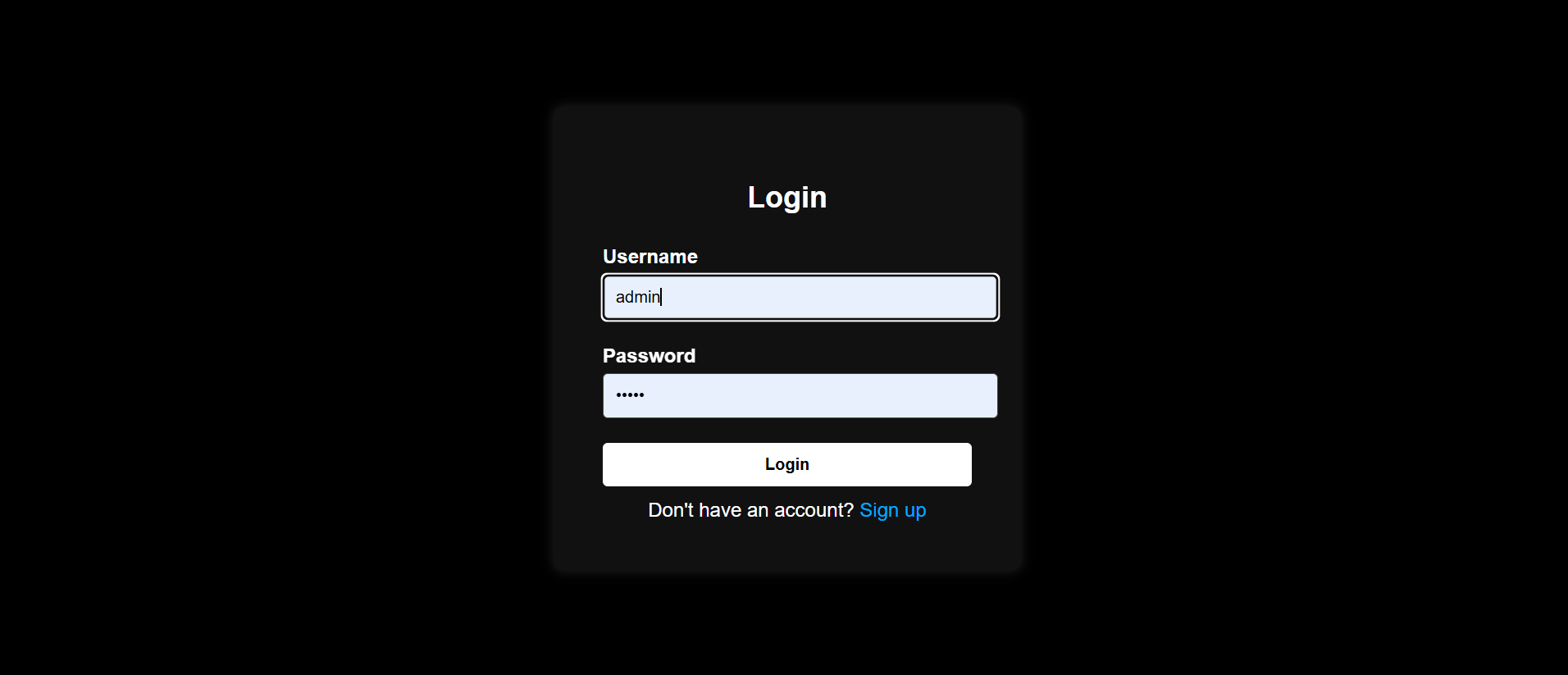
**5.3 METHOD OF IMPLEMENTATION**

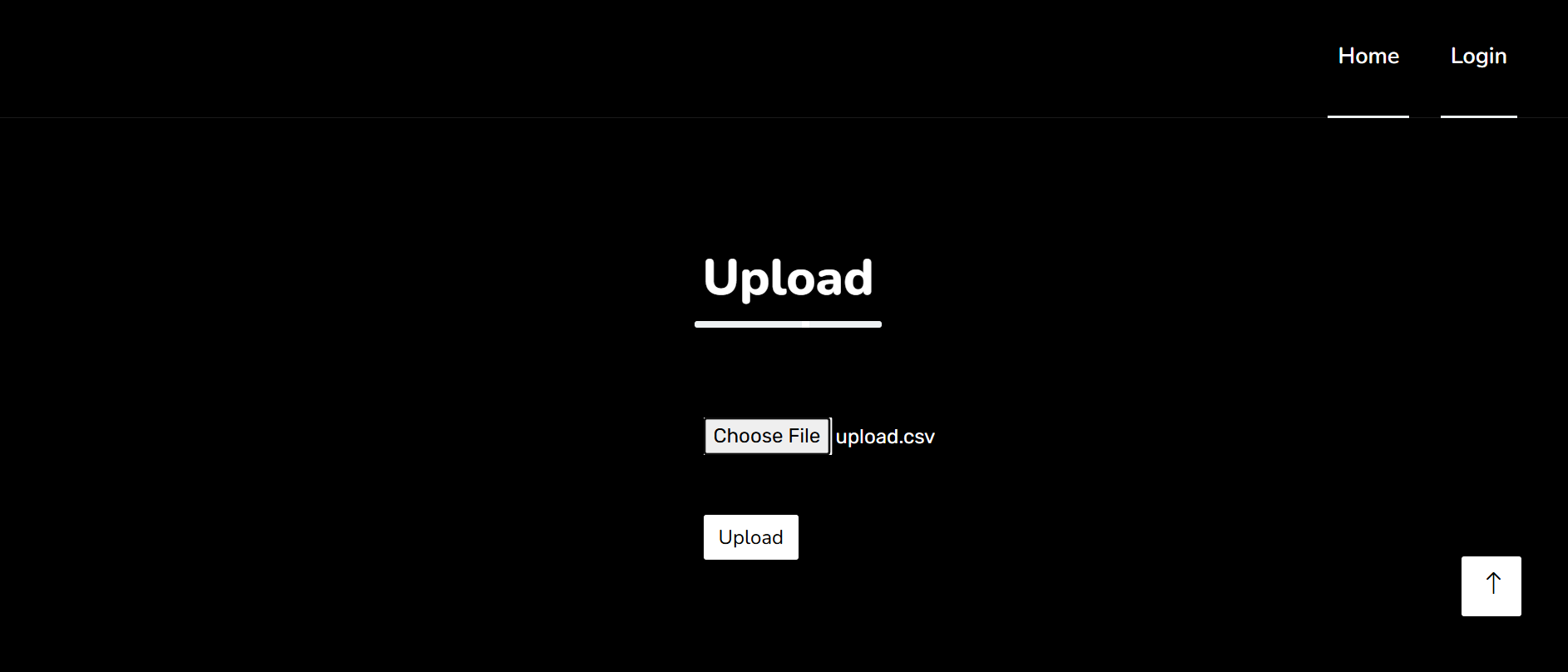
* The input form allows users to enter a URL for phishing detection.
* Flask handles form submissions and processes the user input.
* Validation mechanisms are used to ensure correct input formatting.

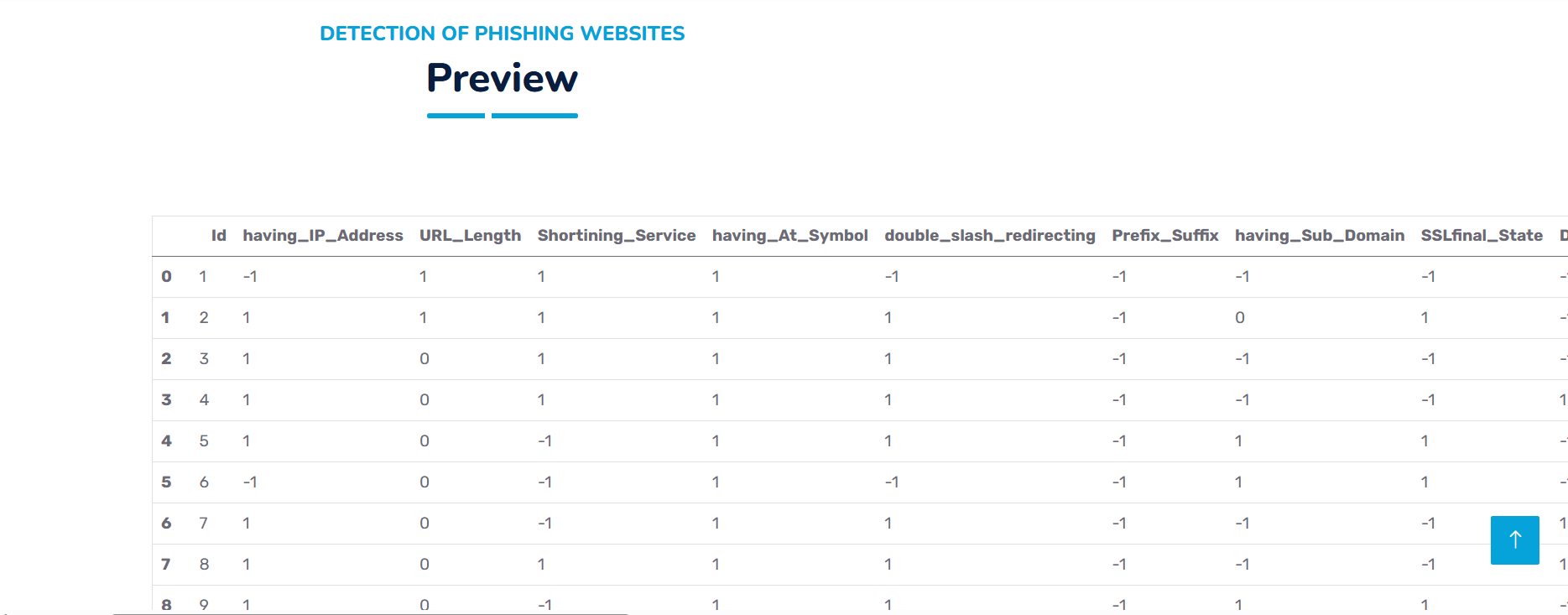
**5.3.1 Output Screens**

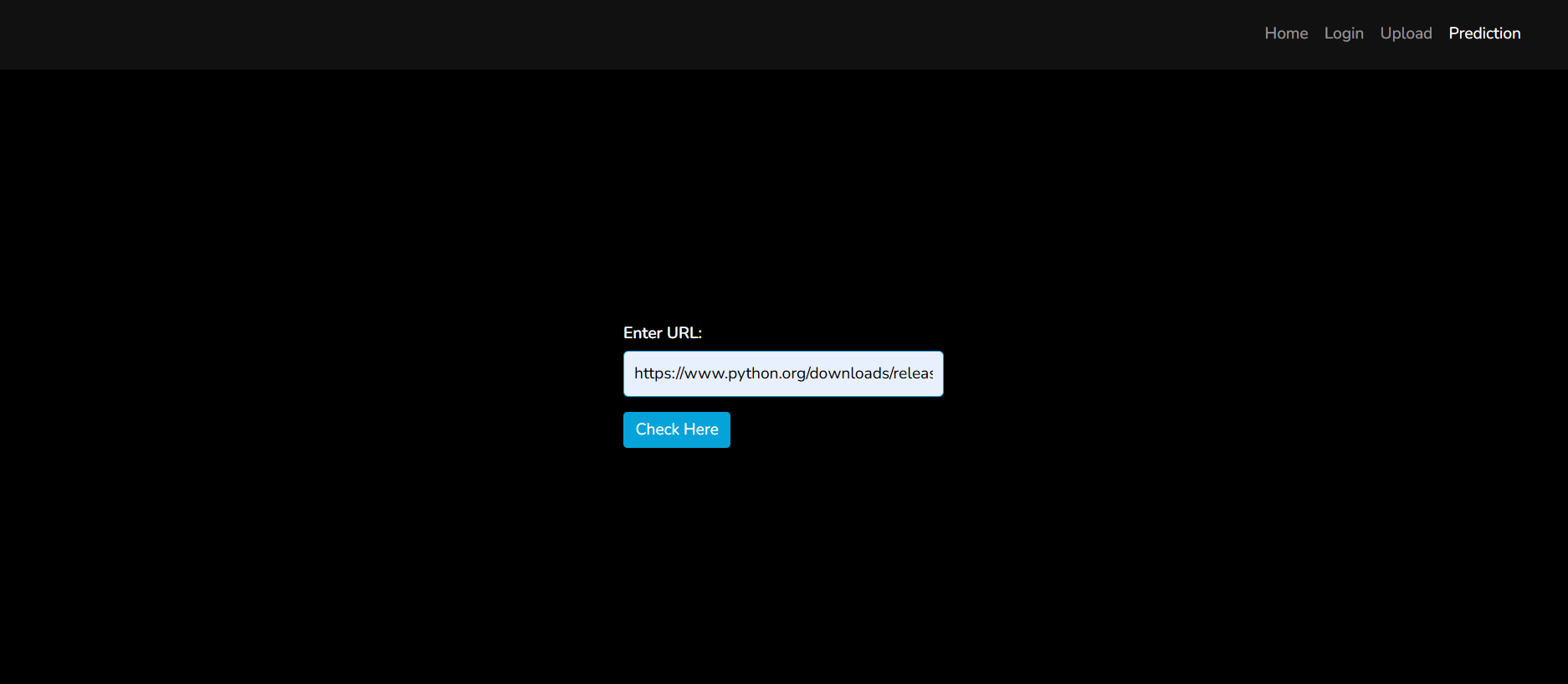
****

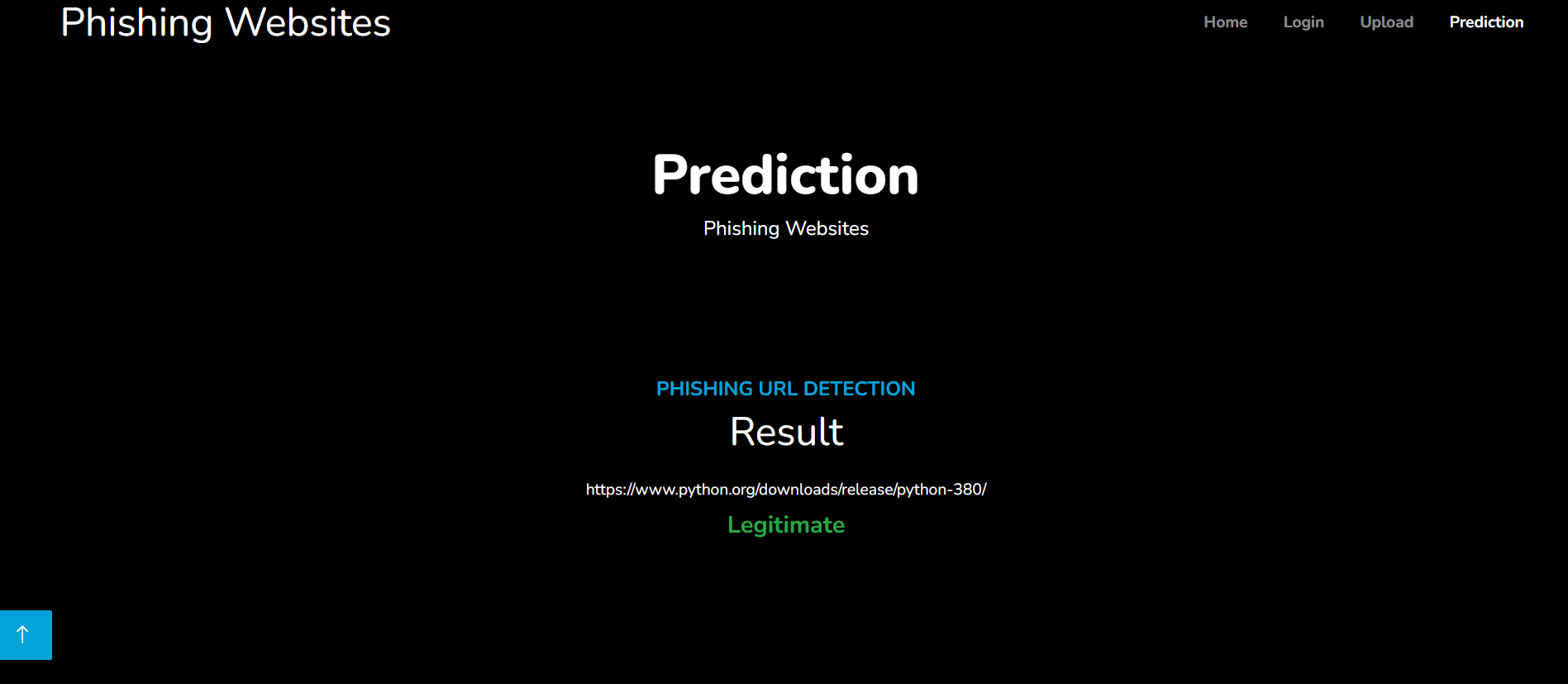
****

****

****

****

****

****

**** Figure 5.3.1 - Output image of detection

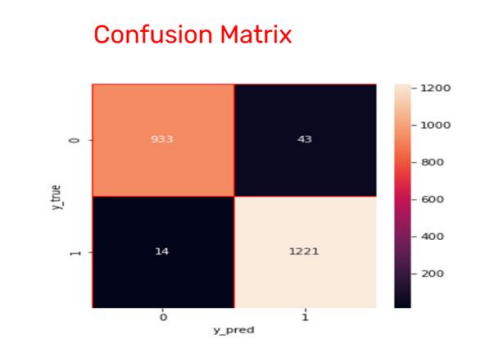


Figure 5.3.2 Perfomance analysis of output

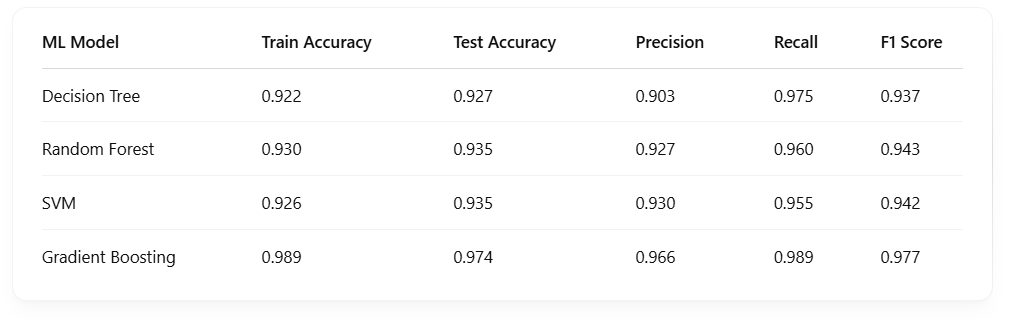
### **5.3.2 Result Analysis**

Results indicate the Gradient Boosting Classifier as the most effective model, achieving 98% accuracy on the testing set and 97% on the training set. Feature importance maps and correlation matrices facilitate the selection of optimal features for the final model.

Currently with a test dataset we’re testing all 3 machine learning algorithms currently we finished K Means with Training accuracy: 85% Testing accuracy: 81% Expected Results We want the model to be highly efficient so we set our goals at at least Training accuracy: 95% Testing accuracy: 95% Sample Results Evaluation Metrics to measure your algorithms’ accuracy Currently we are using Confusion matrix R2 score and Accuracy score to know the model’s metrics (shown in figure 3).

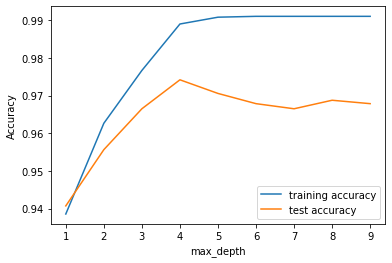
Training and Testing plot between learning rate and accuracy and Training and Testing plot between depth and accuracy is also shown in figure 4 and 5 respectively.

**Comparative analysis**

****

### 

Learning rate vs accuracy plot for training and testing data



max-depth vs accuracy plot for training and testing data

**CHAPTER-6**

**CODING**

**6.CODING**

from flask import Flask, request, render\_template, redirect, url\_for, flash

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager, login\_user, login\_required, logout\_user, current\_user, UserMixin

import datetime

import numpy as np

import pandas as pd

import pickle

import warnings

from feature import FeatureExtraction

warnings.filterwarnings('ignore')

app = Flask(\_\_name\_\_)

app.secret\_key = "your\_secret\_key"

app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024

app.config['SQLALCHEMY\_DATABASE\_URI'] = "mysql+pymysql://root:@localhost/phishing\_db"

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = 'login'

# Model for login/signup

class User(db.Model, UserMixin):

    id = db.Column(db.Integer, primary\_key=True)

    username = db.Column(db.String(150), unique=True, nullable=False)

    password = db.Column(db.String(150), nullable=False)

class Prediction(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    url = db.Column(db.String(512), nullable=False)

    prediction = db.Column(db.String(50), nullable=False)

    confidence = db.Column(db.Float, nullable=False)

    timestamp = db.Column(db.DateTime, default=datetime.datetime.utcnow)

@login\_manager.user\_loader

def load\_user(user\_id):

    return User.query.get(int(user\_id))

# Routes

@app.route('/')

@app.route('/first')

def first():

    return render\_template('first.html')

@app.route('/performance')

def performance():

    return render\_template('performance.html')

@app.route('/chart')

def chart():

    return render\_template('chart.html')

@app.route('/login', methods=['GET', 'POST'])

def login():

    if request.method == "POST":

        uname = request.form['username']

        pwd = request.form['password']

        user = User.query.filter\_by(username=uname, password=pwd).first()

        if user:

            login\_user(user)

            return redirect(url\_for('upload'))

        else:

            flash("Invalid credentials")

    return render\_template('login.html')

@app.route('/signup', methods=['GET', 'POST'])

def signup():

    if request.method == "POST":

        uname = request.form['username']

        pwd = request.form['password']

        existing\_user = User.query.filter\_by(username=uname).first()

        if existing\_user:

            flash("Username already exists")

        else:

            new\_user = User(username=uname, password=pwd)

            db.session.add(new\_user)

            db.session.commit()

            flash("Signup successful, please login.")

            return redirect(url\_for('login'))

    return render\_template('signup.html')

@app.route('/logout')

@login\_required

def logout():

    logout\_user()

    return redirect(url\_for('login'))

@app.route('/upload')

@login\_required

def upload():

    return render\_template('upload.html')

@app.route('/preview', methods=["POST"])

def preview():

    try:

        dataset = request.files['datasetfile']

        df = pd.read\_csv(dataset, encoding='unicode\_escape', nrows=100)  # Load only first 100 rows

        return render\_template("preview.html", df\_view=df)

    except Exception as e:

        print("Error loading CSV:", e)

        flash("Error reading CSV file. Please upload a smaller file or check the format.")

        return redirect(url\_for('upload'))

@app.route('/index')

@login\_required

def index():

    return render\_template('index.html')

@app.route("/posts", methods=["GET", "POST"])

@login\_required

def posts():

    if request.method == "POST":

        url = request.form["url"]

        obj = FeatureExtraction(url)

        x = np.array(obj.getFeaturesList()).reshape(1, 30)

        y\_pred = gbc.predict(x)[0]

        y\_pro\_non\_phishing = gbc.predict\_proba(x)[0, 1]

        pred = "Phishing" if y\_pred == -1 else "Legitimate"

        new\_entry = Prediction(

            url=url,

            prediction=pred,

            confidence=float(y\_pro\_non\_phishing)

        )

        db.session.add(new\_entry)

        db.session.commit()

        return render\_template('result.html', prediction=pred, xx=round(y\_pro\_non\_phishing, 2), url=url)

    return render\_template("result.html", xx=-1)

# Load the model

with open("model.pkl", "rb") as file:

    gbc = pickle.load(file)

if \_\_name\_\_ == "\_\_main\_\_":

    with app.app\_context():

        db.create\_all()

    app.run(debug=True)

**CHAPTER – 7**

**TESTING & VALIDATION**

**7.TESTING AND VALIDATION**

**7.1 INTRODUCTION**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub- assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**7.2 DESIGN OF TESTCASES AND SCENARIOS**

The system is tested based on various input conditions to ensure its reliability.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Test Description | Expected Output | Actual Output | Result |
| TC-01 | Input a valid URL | Predicts "Safe" | Safe | Pass |
| TC-02 | Input a phishing URL | Predicts "Phishing" | Phishing | Pass |
| TC-03 | Input an empty URL | Displays an error | Error: "Invalid URL" | Pass |
| TC-04 | Input a malformed URL | Displays an error | Error: "Invalid URL format" | Pass |

7.2.1 - Test Cases for phishing website

**7.3 VALIDATION**

Validation testing ensures that the system meets functional and user requirements. The system was validated through:

1. **Unit Testing:**

Each function (e.g., feature extraction, model prediction) was tested independently.

Ensures that the Gradient Boosting Classifier works correctly.

1. **Integration Testing**:

Tested the integration of Flask (Frontend) and ML Model (Backend).

Verified that URL inputs are correctly processed and predictions are displayed.

1. **System Testing**:

Ensures that the entire system functions correctly.

Test cases covered valid and invalid URLs, feature extraction, and result display.

1. **User Acceptance Testing (UAT):**

Conducted with a sample group of users.

Ensured that the interface was user-friendly and predictions were accurate

**7.4 CONCLUSION**

The testing phase confirms that the phishing detection system functions correctly with high accuracy and reliability. The Gradient Boosting Classifier achieves 97% accuracy, demonstrating the effectiveness of machine learning in identifying phishing websites. All test cases were successfully passed with no critical defects found.

**CHAPTER-8**

**CONCLUSION**

**8.CONCLUSION**

It is remarkable that a good anti-phishing system should be able to predict phishing attacks in a reasonable amount of time. Accepting that having a good anti-phishing gadget available at a reasonable time is also necessary for expanding the scope of phishing site detection. The current system merely detects phishing websites using Gradient Boosting Classifier. We achieved 97% detection accuracy using Gradient Boosting Classifier with lowest false positive rate.

**Future Scope**:

Although the use of URL lexical features alone has been shown to result in high accuracy, phishers have learned how to make predicting a URL destination difficult by carefully manipulating the URL to evade detection. Therefore, combining these features with others, such as host, is the most effective approach . For future enhancements, we intend to build the phishing detection system as a scalable web service which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

**REFERENCES:**

[1] Khonji, Mahmoud, Youssef Iraqi, and Andrew Jones. "Phishing detection: a literature survey." *IEEE Communications Surveys & Tutorials* 15.4 (2013): 2091-2121.

[2] Acquisti, Alessandro, et al. "Nudges for privacy and security: Understanding and assisting users’ choices online." *ACM Computing Surveys (CSUR)* 50.3 (2017): 1-41.

[3] Junger, Marianne, Lorena Montoya, and F-J. Overink. "Priming and warnings are not effective to prevent social engineering attacks." *Computers in human behavior* 66 (2017): 75-87.

[4] El-Alfy, El-Sayed M. "Detection of phishing websites based on probabilistic neural networks and K-medoids clustering." *The Computer Journal* 60.12 (2017): 1745-1759.

[5] Huang, Huajun, Junshan Tan, and Lingxi Liu. "Countermeasure techniques for deceptive phishing attack." *2009 International Conference on New Trends in Information and Service Science*. IEEE, 2009.

[6] Marchal, Samuel, et al. "Off-the-hook: An efficient and usable client-side phishing prevention application." *IEEE Transactions on Computers* 66.10 (2017): 1717-1733.

[7] Dr.G.K.Kamalam, Dr.P.Suresh, R.Nivash, A.Ramya, G.Raviprasath, “Detection of Phishing Websites Using Machine Learning”, 2022 International Conference on Computer Communication and Informatics (ICCCI - 2022), IEEE Conference, 2022.

[8] Mohiuddin Ahmed, AbdunNaser Mahmood, and Jiankun Hu, "A study of network anomaly detection strategies," J. Netw. Comput. Appl., vol.60, no. C, pp. 19-31, 2016.

[9] Systematization of Knowledge (SoK): ASystematic Review of SoftwareBased Web Phishing Detection, by Zuochao Dou, Issa Khalil, Abdallah Khreishah, Ala Al-Fuqaha, and Mohsen Guizani, IEEE Communications Surveys & Tutorials, 2017.

[10] Jie Wu, Longfei Wu, Xiaojiang Du Phishing Attacks on Mobile Computing Platforms: Effective Defense Schemes 6678-6691 in IEEE Transactions on Vehicular Technology, vol. 65, no. 8, 2016.

[11] W.Fadheel, M. Abusharkh, and I. Abdel-Qader, “On Feature Selection for the Prediction of Phishing Websites,” 2017 IEEE 15th Intl Conf Dependable, Auton. Secur. Comput. 15th Intl Conf Pervasive Intell. Comput. 3rd Intl Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr., pp. 871–876, 2017.

[12] A. Desai, J. Jatakia, R. Naik, and N. Raul, “Malicious web content detection using machine leaning,” RTEICT 2017 - 2nd IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. Proc., vol. 2018–Janua, pp. 1432–1436, 2018.

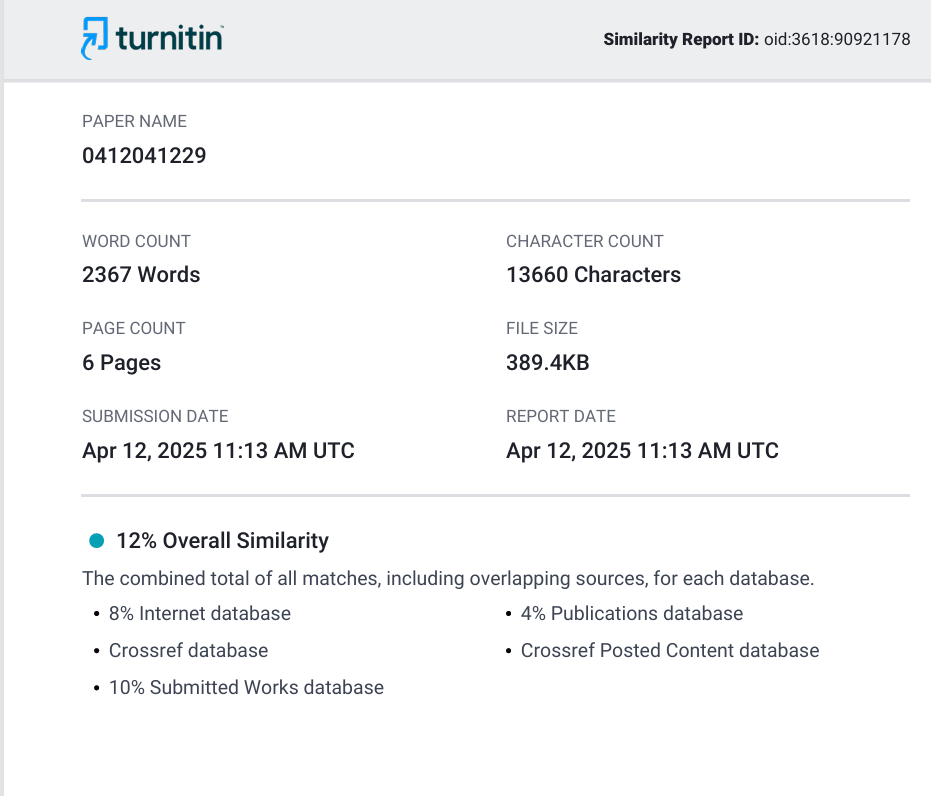
[13] J. Shad and S. Sharma, “A Novel Machine Learning Approach to Detect Phishing Websites Jaypee Institute of Information Technology,” pp. 425–430, 2018.

[14] S. Parekh, D. Parikh, S. Kotak, and P. S. Sankhe, “A New Method for Detection of Phishing Websites: URL Detection,” in 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, vol. 0, no. Icicct, pp. 949–952.

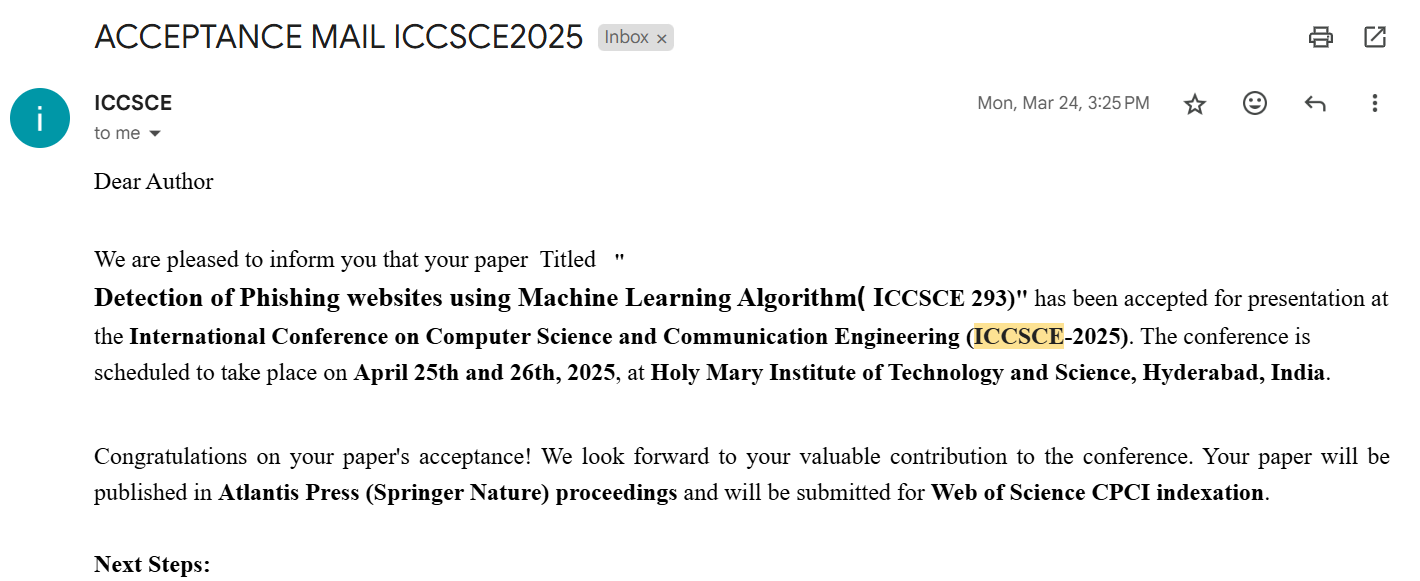
[15] M. Karabatak and T. Mustafa, “Performance comparison of classifiers on reduced phishing website dataset,” 6th Int. Symp. Digit. Forensic Secur. ISDFS 2018 - Proceeding, vol. 2018–Janua, pp. 1–5, 2018.

[16] L. MacHado and J. Gadge, “Phishing Sites Detection Based on C4.5 Decision Tree Algorithm,” in 2017 International Conference on Computing, Communication, Control and Automation, ICCUBEA 2017.

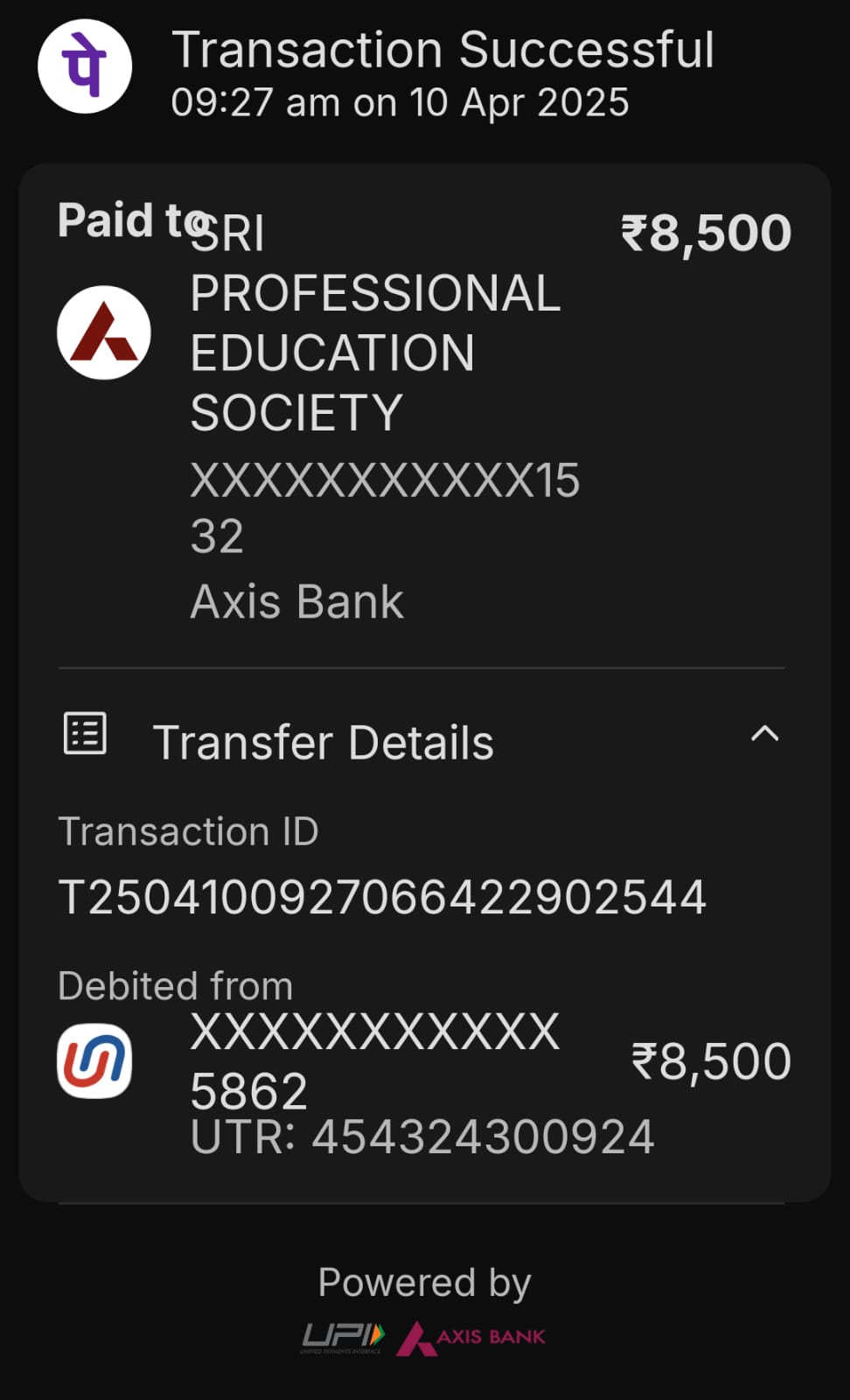
**PAPER PUBLICATION WITH PLAGARISM REPORT**

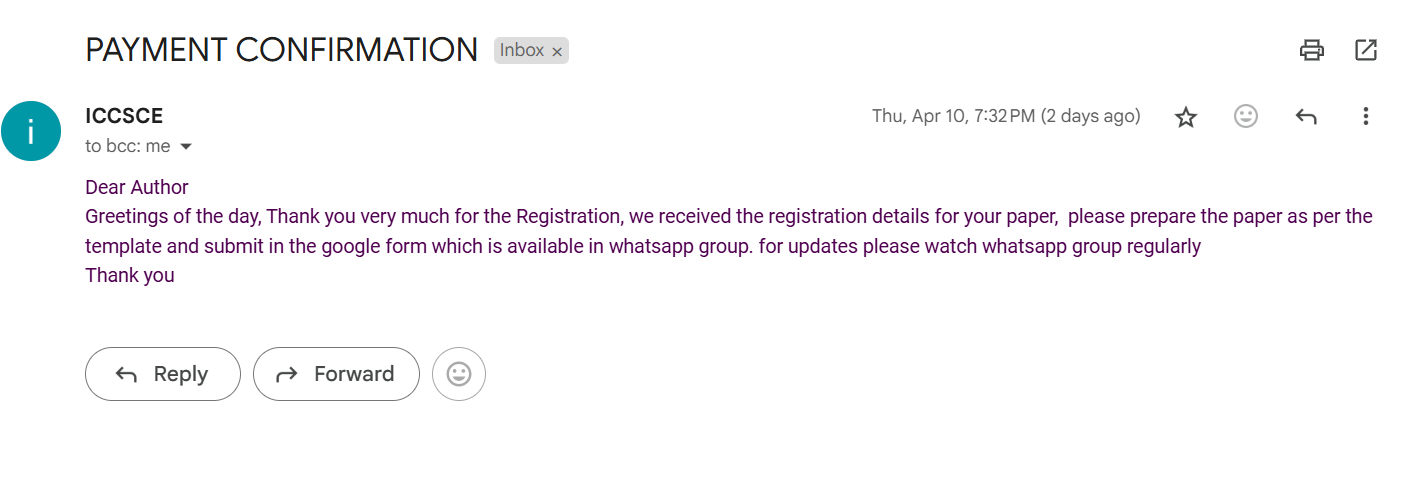


**PAPER ACCEPTANCE**



**PAYMENT CONFIRMATION**

****

****