A Major Project

Report on

MULTI- MODAL FEATURES REPRESENTATION - BASED CNN MODEL FOR MALICIOUS WEBSITE DETECTION

Submitted to CMREC (UGC Autonomous)

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Machine Learning)

Submitted By

A. ABHILASH - 228R1A6603

M. LOHITH KUMAR - 228R1A6646

S. SHRIYA - 228R1A6657

V. AJAY KUMAR - 228R1A6665

Under the Esteemed guidance of

Mrs. G. Sumangala

Associate Professor, Department of CSE (AI & ML)

Department of Computer Science & Engineering (AI & ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE, NEW DELHI, Affiliated to JNTU Hyderabad)

(Kandlakoya, Medchal Road, Medchal-Malkajgiri Dist,. Hyderabad-501 401)

(2025-2026)

CMR ENGINEERING

COLLEGE UGC

AUTONOMOUS

(Accredited by NAAC & NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, Hyderabad-501 401)

Department of Computer Science & Engineering

(Artificial Intelligence and Machine Learning)

CERTIFICATE

This is to certify that the Major Project entitled **"MULTI - MODAL FEATURES REPRESENTATION- BASED CNN MODEL FOR MALICIOUS WEBSITE DETECTION"** is a bonafide work carried out by

A. ABHILASH - 228R1A6603

M. LOHITH KUMAR - 228R1A6646

S. SHRIYA - 228R1A6657

V. AJAY KUMAR - 228R1A6664

in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI & M L) from CMR Engineering College, under our guidance and supervision.

The results presented in this major project have been verified and are found to be satisfactory. The results embodied in this major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Mrs. G. Sumangala	Mr. G. Venkateshwarlu	Dr. Madhavi Pingili
Associate Professor	Assistant Professor	Professor & HOD
CSE (AI&ML)	CSE (AI & ML)	CSE (AI & ML)

External Examiner:_____

DECLARATION

This is to certify that the work reported in the present Major project entitled "**MULTI - MODAL FEATURE REPRESENTATION BASED CNN MODEL FOR MALICIOUS WEBSITE DETECTION**" is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI&ML), CMR Engineering College. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

A. ABHILASH - 228R1A6603

M.LOHITH KUMAR - 228R1A6646

S. SHRIYA - 228R1A6657

V. AJAY KUMAR - 228R1A6664

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili**, Professor & HOD, **Department of CSE (AI & ML), CMR Engineering College** for their constant support.

We are extremely thankful to **Mrs. G. Sumangala,** Associate Professor, Internal Guide, Department of CSE (AI & ML), for her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mr. G. Venkateswarlu,** Major Project Coordinator for his constant support in carrying out the project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, we are very much thankful to our parents who guided us in every step.

A. ABHILASH - 228R1A6603

M.LOHITH KUMAR - 228R1A6646

S. SHRIYA - 228R1A6657

V. AJAY KUMAR - 228R1A6664

TABLE OF CONTENTS

ABSTRACT **I**

LIST OF FIGURES **II**

LIST OF TABLES **III**

CHAPTER 1 : INTRODUCTION **1**

1.1 Introduction 2

1.2 Project Objectives 3

1.3 Purpose of the Project 4

1.4 Problem Statement 4

1.5 Existing System with Limitations 5

1.6 Proposed System with Advanatages 6

1.7 Input and Output Design 8

CHAPTER 2 : LITERATURE SURVEY **10**

CHAPTER 3 : SOFTWARE REQUIREMENT ANALYSIS **15**

3.1 Modules and their Functionalities 16

3.2 Functional Requirements 17

3.3 Non-Functional Requirements 18

3.4 Feasibility Study 19

CHAPTER 4 : SOFTWARE AND HARDWARE REQUIREMENTS **23**

4.1 Software Requirements 24

4.2 Hardware Requirements 24

CHAPTER 5 : SOFTWARE DESIGN **27**

5.1 System Architecture 27

5.2 Data Flow Diagram 28

5.3 UML Diagrams 31

CHAPTER 6 : IMPLEMENTATION AND CODING **38**

6.1 Source Code 39

6.2 Implementation 48

CHAPTER 7 : SYSTEM TESTING **54**

7.1 Types of Testing 56

7.2 Testing Strategies 59

7.3 Test Cases 61

CHAPTER 8 : RESULTS **63**

CHAPTER 9 : CONCLUSION **68**

CHAPTER 10 : FUTURE ENHANCEMENTS **71**

REFERENCES **75**

ABSTRACT

Accurate detection of malicious websites has become a critical challenge in modern cybersecurity due to the rapid growth of sophisticated web-based attacks such as phishing, malware distribution, and online fraud. Traditional approaches, including blacklist-based techniques and classical machine learning models, often struggle to identify newly emerging threats and fail to effectively utilize the diverse nature of web data. To overcome these limitations, this work proposes a multi-modal features representation-based Convolutional Neural Network (CNN) model for malicious website detection. The approach integrates multiple feature types, including URL-based features, HTML content characteristics, and network-related attributes, creating a comprehensive input representation that captures both structural and semantic patterns of websites. These features are preprocessed and fed into a CNN model that automatically learns hierarchical feature representations, enabling accurate identification of malicious behavior. Experimental evaluation on a well-prepared dataset of benign and malicious websites demonstrates that the proposed model achieves improved accuracy, robustness, and generalization compared to traditional methods. Furthermore, the framework supports real-time detection and enhances web security by effectively identifying zero-day attacks, making it a reliable solution for modern cybersecurity applications. Experimental evaluation is conducted on a well-curated dataset containing both benign and malicious website samples. The proposed CNN-based model demonstrates significant improvements in detection accuracy, robustness, and generalization capability compared to traditional approaches. Additionally, the framework provides practical benefits such as real-time threat detection, enhanced security for web users, and improved resilience against zero-day attacks.

Keywords—Machine Learning, Cybersecurity, Malicious Website Detection, Convolutional Neural Networks (CNN), Multi-Modal Features, URL Analysis, Phishing Detection, Web Security.

LIST OF FIGURES

S.NO	FIGURE NO	DESCRIPTION	PAGE NO
1	1.6	Block diagram of the proposed system	07
2	5.1	System Architecture	27
3	5.2	Data Flow Diagram	29
4	5.3.1	Sequence diagram	32
5	5.3.2	Use case diagram	33
6	5.3.3	Collaboration diagram	34
7	5.3.4	Activity diagram	35
8	5.3.5	Class diagram	36
9	5.3.6	Deployment diagram	37
10	6.2.1	Architecture of Convolutional Neural Network	49
11	6.2.2	Illustration of CNN operation	50
21	8.1	Confusion Matrix	64
22	8.2	Model Accuracy	65
23	8.3	Training & Validation Performance	66

LIST OF TABLES

S.NO	TABLE NO	DESCRIPTION	PAGE NO
1	2.1	Literature Review Summary	14
2	7.3	Test Cases	62
3	8.0	Summary of Results	67

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

In the era of digital transformation, the rapid growth of internet usage and web-based services has significantly increased the risk of cyber threats, particularly malicious websites that facilitate phishing, malware distribution, and online fraud. With billions of users relying on the web for communication, banking, e-commerce, and information access, ensuring secure browsing has become a critical concern**[2], [7]**. Traditional detection methods, such as blacklist-based systems and signature-based techniques, are no longer sufficient to combat the evolving nature of cyberattacks, as they often fail to detect newly emerging or zero-day threats.

Moreover, the dynamic and heterogeneous nature of web data—including URL structures, HTML content, and network behavior—makes it challenging to accurately identify malicious patterns using conventional approaches. Effective malicious website detection is essential to enhance cybersecurity, protect user data, and maintain trust in digital platforms. One of the major challenges in this domain is capturing the complex relationships between different types of features and identifying hidden patterns that distinguish malicious websites from benign ones**[8], [9].**

Traditional machine learning models primarily rely on limited feature sets and often ignore the complementary information present in multi-modal data, leading to reduced detection accuracy. To address these challenges, modern approaches focus on integrating diverse feature representations and leveraging deep learning techniques for improved performance. In this context, Convolutional Neural Networks (CNNs) have gained significant attention due to their ability to automatically extract hierarchical feature representations and detect intricate patterns in large datasets **[4]**.

The proposed Multi-Modal Features Representation-Based CNN Model combines multiple feature types, such as URL-based features, content-based attributes, and network-related characteristics **[6]**, to provide a comprehensive understanding of website behavior **[10].**

volumes of web data must be processed efficiently and accurately. By leveraging multi-modal feature fusion and CNN-based learning, the system can adapt to evolving attack patterns and continuously improve its detection capability over time. The model is designed to handle high-dimensional data while maintaining computational efficiency **[5]**, making it suitable for integration into existing cybersecurity infrastructures such as web browsers, firewalls, and intrusion detection systems**[8].**

1.2 Project Objectives

The objectives of this project are to develop an effective system for detecting malicious websites using a multi-modal features representation-based Convolutional Neural Network (CNN) model. It aims to analyze different types of web data, including URL structures, HTML content, and network-based attributes, and to capture hidden patterns that distinguish malicious websites from benign ones. Additionally, it seeks to evaluate the performance of the proposed model against traditional approaches and enhance cybersecurity by enabling reliable and real-time detection of malicious websites. The objectives include:

- To study the limitations of existing malicious website detection approaches, including blacklist-based methods, signature-based techniques, and traditional machine learning models.
- To design a multi-modal feature representation framework that integrates URL-based features, content-based attributes, and network-related characteristics.
- To develop a detection model based on Convolutional Neural Networks (CNN) for efficient and automatic feature extraction.
- To incorporate diverse feature types into the learning process to improve detection accuracy and robustness.
- To evaluate the proposed model using real-world datasets consisting of both benign and malicious websites.

1.3 Purpose of the Project

The purpose of this project is to develop a modern, adaptive, and accurate malicious website detection system capable of overcoming the limitations of traditional security approaches. Malicious websites continuously evolve in structure and behavior, making it difficult for conventional methods such as blacklist-based and signature-based systems to effectively identify new and unknown threats**[7], [9]**. Accurately detecting such websites is essential to protect users from phishing attacks, malware infections, and data breaches, thereby ensuring secure online interactions. To address these challenges, the proposed approach leverages a Multi-Modal Features Representation-Based Convolutional Neural Network (CNN) model, which integrates multiple types of data and automatically learns complex patterns for improved detection accuracy **[1], [10].**

Furthermore, the project aims to enhance cybersecurity by enabling early and proactive identification of malicious websites, thereby reducing the risk of cyberattacks and preventing potential damage. The model utilizes deep learning techniques to extract meaningful and hierarchical representations from these features, allowing it to detect subtle and hidden patterns that are often missed by traditional methods. This leads to improved detection performance, reduced false positives, and enhanced reliability, ultimately providing better protection for users and systems **[5].**

Additionally, the proposed framework is designed to be scalable and adaptable to large-scale web data and evolving cyber threats. By improving generalization across different types of attacks and continuously adapting to new threat patterns, the system ensures long-term effectiveness **[2], [3].**

1.4 Problem Statement

Accurate detection of malicious websites is a challenging task due to the continuously evolving nature of cyber threats, including phishing, malware distribution, and fraudulent activities.

dentifying patterns associated with malicious activities. Traditional machine learning models struggle to capture these complex relationships and often rely on limited feature sets, resulting in reduced detection accuracy**[8]**. This challenge becomes more critical in large-scale web environments where vast amounts of dynamic and diverse data are generated continuously. Therefore, there is a need for an advanced approach that can effectively integrate multiple feature types and learn meaningful representations for accurate detection. A multi-modal features representation-based Convolutional Neural Network (CNN) model provides a promising solution by capturing complex patterns across different data sources, leading to improved detection accuracy, enhanced security, and better protection against evolving cyber threats**[10].**

1.5 Existing Systems and their Limitations

The existing approaches for malicious website detection include blacklist-based systems, signature-based techniques, machine learning models, deep learning approaches, and hybrid methods. Although these methods have contributed to improving detection performance, they exhibit several limitations in handling dynamic and large-scale web environments.

a. Blacklist-Based System Limitations:

Blacklist-based systems rely on previously identified malicious URLs; however, they fail to detect new or unknown websites that are not yet listed. These systems require continuous updates and are ineffective against zero-day attacks, limiting their overall reliability**[7].**

b. Signature-Based Technique Limitations:

Signature-based methods detect known attack patterns but are unable to identify new or modified threats. They depend heavily on predefined rules and signatures, making them less adaptable to evolving cyberattack strategies **[9].**

c. Machine Learning Limitations:

Machine learning approaches improve detection compared to traditional methods, but they require extensive manual feature engineering **[10].**

d. Deep Learning Limitations:

Deep learning models, including CNNs and RNNs, can automatically learn features but often focus on a single type of data, such as URLs or content, without effectively integrating multiple modalities. This limits their ability to fully understand website behavior, despite their high learning capacity **[6].**

e. Hybrid Model Limitations:

Hybrid models combine multiple techniques to enhance detection accuracy; however, they increase system complexity and computational cost. These models require careful tuning and are difficult to deploy in real-time applications, affecting scalability and efficiency **[1].**

1.6 Proposed System

The proposed system introduces an advanced malicious website detection framework using a Multi-Modal Features Representation-Based Convolutional Neural Network (CNN) model to overcome the limitations of traditional blacklist-based, signature-based, and conventional machine learning approaches. Unlike existing methods that rely on a single type of feature, the system integrates multiple data sources, including URL-based features, HTML content attributes, and network-related characteristics, to provide a comprehensive understanding of website behavior. This multi-modal approach enables the model to capture both structural and semantic patterns associated with malicious activities**[3], [1].**

In addition to feature integration, the system employs a CNN architecture to automatically learn hierarchical feature representations from the combined input data. The model processes the extracted features through convolutional and pooling layers, allowing it to identify complex and hidden patterns that distinguish malicious websites from benign ones. By leveraging deep learning capabilities, the system reduces the need for manual feature engineering and improves detection accuracy and robustness**[1].**

The system processes real-world datasets containing both benign and malicious website samples.

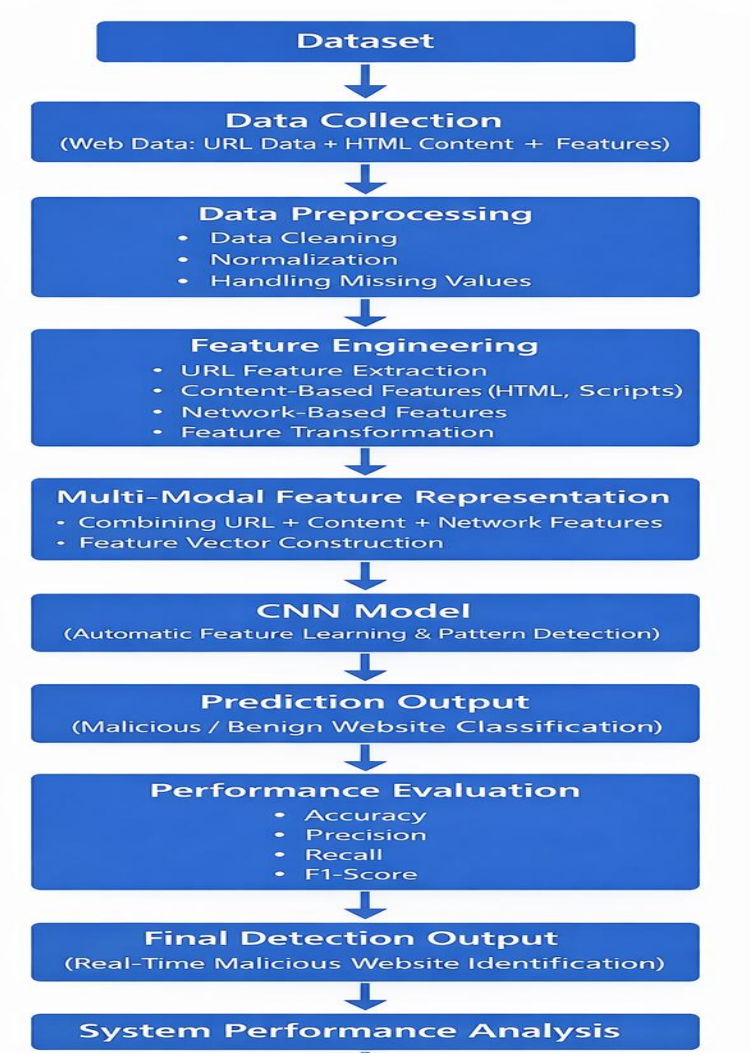

Fig 1.6 Block Diagram of the Proposed System

Advantages of the Proposed System

- Captures complex patterns in web data by analyzing multiple feature types .
- Utilizes multi-modal feature representation for improved detection accuracy
- Effectively detects zero-day and previously unseen malicious website
- Efficiently integrates diverse data sources for comprehensive website analysis
- Scalable for large-scale web environments and real-world cybersecurity applications
- Supports real-time malicious website detection and prevention
- Robust against noisy, incomplete, and imbalanced datasets

1.7 Input and Output Design

a. Input Design

Input design is a crucial part of this system as it focuses on how web-related data is collected, processed, and prepared for malicious website detection. It acts as a bridge between raw web data sources and the detection model, ensuring that all required inputs are captured in a structured and efficient manner. A well-designed input system minimizes errors, reduces redundancy, and enhances the performance of the CNN-based detection model. The input includes multiple data types such as URL features, HTML content, and network-related attributes, forming a multi-modal dataset. The design process involves identifying relevant features, organizing them properly, and preprocessing them through techniques such as cleaning, normalization, and handling missing values.

Key Considerations:

- Identification of relevant input data (URL, content, network features)
- Proper structuring and encoding of multi-modal data
- Efficient data preprocessing and transformation techniques
- Validation and error-handling mechanisms for data consistency

Objectives:

- Convert raw web data into structured, model-compatible formats
- Ensure high-quality input through preprocessing and validation
- Enable efficient and scalable data handling for model training

b. Output Design

Output design focuses on how the detection results are presented to users in a clear and meaningful way. It plays a vital role in helping users quickly identify whether a website is malicious or benign and take appropriate actions. The system generates outputs such as classification results, probability scores, and performance metrics. The design ensures that outputs are well-structured, easy to interpret, and delivered in real-time when required. Results can be displayed through dashboards, alerts, or reports, highlighting critical threats and important insights.

The system may also provide additional details such as risk levels or reasons for classification to improve transparency. Outputs can be viewed on-screen or integrated into security systems like browsers or intrusion detection tools.

Key Considerations:

- Structured and clear presentation of detection results.
- Selection of appropriate presentation methods (reports, dashboards, etc.)
- Highlighting critical threats and important insights.

Objectives:

- Provide accurate and real-time detection results.
- Enable quick decision-making for users and security systems.
- Support proactive threat prevention and improved cybersecurity measures.

CHAPTER-2

LITERATURE SURVEY

2 LITERATURE SURVEY

1. **Z.P. Sharma, K. Reddy and A. Nair – "Adaptive Multi-Modal Cyber Threat Detection System Using Deep Learning" – 2026.**

This paper presents an adaptive multi-modal framework that combines multiple data sources and continuously updates the model based on new threats.

2. **J. Wang, Q. Li and X. Zhou – "Graph-Based Cyber Threat Detection Using Deep Learning" – 2025.**

This study explores graph-based deep learning techniques to model relationships between web entities and detect malicious activities.

3. **T. Nguyen, D. Tran and H. Pham – "Transformer-Based Approach for Malicious Web Content Detection" – 2025.**

This paper applies transformer architectures to analyze web content and detect malicious patterns. It demonstrates superior performance in capturing long-range dependencies and contextual relationships in web data.

4. **R. Kumar and S. Mishra – "Efficient Malicious Website Detection Using Lightweight Deep Learning Models" – 2025.**

This work focuses on designing lightweight deep learning models suitable for edge devices. It reduces computational complexity while maintaining high detection accuracy, making it suitable for real-time and resource-constrained environments.

5. **M. Chen, L. Zhao and Y. Liu – "Attention-Based CNN for Malicious URL Classification" – 2025**.

This paper introduces an attention-enhanced CNN model that focuses on important parts of URL and content features. The attention mechanism improves feature selection and boosts classification accuracy for detecting malicious websites.

6. **A. Verma and P. Singh – "AI-Driven Cybersecurity Framework for Web Threat Detection" – 2024.**

This study proposes an AI-based framework that combines machine learning, deep learning, and threat intelligence for detecting malicious websites. It enhances system adaptability and provides proactive defense mechanisms against evolving cyberattacks.

7. **H. Kim, J. Lee and S. Park – "Deep Learning-Based Real-Time Malicious Website Detection System" – 2024**.

This paper presents a real-time detection system using deep neural networks integrated with web applications. It focuses on fast inference and deployment, enabling immediate identification and blocking of malicious websites in practical environments.

8. **X. Zhang, J. Yu and Z. Wang – "Hybrid Deep Learning Model for Phishing Website Detection" – 2024**.

This work introduces a hybrid CNN-RNN architecture that captures both spatial and sequential patterns in web data. The model improves detection of phishing websites by analyzing URL structures and page content simultaneously, achieving higher accuracy than standalone models

9. **S. Sahoo, B. B. Gupta and K. K. R. Choo – "Malicious URL Detection Using Machine Learning: A Survey" – 2023.**

This study provides a comprehensive overview of machine learning and deep learning techniques for malicious URL detection.

10. **Y. Li, X. Hu and J. Wang – "Multi-Modal Deep Learning for Malicious Website Detection" – 2023.**

This paper proposes a multi-modal deep learning framework that integrates URL features, HTML content, and behavioral data for improved malicious website detection.

Literature Review Summary

The reviewed studies collectively highlight the rapid advancement of malicious website detection techniques, focusing on improving detection accuracy, robustness, and real-time adaptability using machine learning and deep learning approaches.

Title	Key Findings	Reference
Adaptive Multi-Modal Cyber Threat Detection System [1]	Introduces adaptive multi-modal learning; improves robustness and real-time threat detection.	P. Sharma, K. Reddy, and A. Nair, "Adaptive Multi-Modal Cyber Threat Detection System Using Deep Learning," 2026.
Graph-Based Cyber Threat Detection Using Deep Learning[2]	Applies graph-based learning to detect relationships between web entities; improves detection of hidden threats.	J. Wang, Q. Li, and X. Zhou, "Graph-Based Cyber Threat Detection Using Deep Learning," 2025.
Transformer-Based Approach for Malicious Web Content Detection [3]	Uses transformer models to capture contextual relationships; improves detection of complex threats.	T. Nguyen, D. Tran, and H. Pham, "Transformer-Based Approach for Malicious Web Content Detection," 2025.
Lightweight Deep Learning Models for Malicious Website Detection [4]	Reduces computational complexity; suitable for real-time and edge-based environments.	D.R. Kumar and S. Mishra, "Efficient Malicious Website Detection Using Lightweight Deep Learning Models," 2025..

Title	Key Findings	Reference
Attention-Based CNN for Malicious URL Classification.[5]	Uses attention mechanisms to focus on important features; enhances classification performance.	M. Chen, L. Zhao, and Y. Liu, "Attention-Based CNN for Malicious URL Classification," 2025.
AI-Driven Cybersecurity Framework for Web Threat Detection[6]	Combines AI and threat intelligence for proactive detection; improves adaptability against cyberattacks.	A. Verma and P. Singh, "AI-Driven Cybersecurity Framework for Web Threat Detection" 2024.
Deep Learning-Based Real-Time Malicious Website Detection System[7]	Focuses on real-time detection and deployment; enables fast identification and blocking of malicious sites.	H. Kim, J. Lee, and S. Park, "Deep Learning-Based Real-Time Malicious Website Detection System," 2024.
Hybrid Deep Learning Model for Phishing Website Detections [8]	Combines CNN and RNN to capture spatial and sequential patterns; improves phishing detection accuracy.	X. Zhang, J. Yu, and Z. Wang, "Hybrid Deep Learning Model for Phishing Website Detection," 2024.
Malicious URL Detection Using Machine Learning: A Survey [9]	Reviews ML and DL techniques; highlights challenges like feature imbalance and evolving threats.	S. Sahoo, B. B. Gupta, and K. K. R. Choo, "Malicious URL Detection Using Machine Learning: A Survey," 2023.
Multi-Modal Deep Learning for Malicious Website Detection [10]	Integrates URL, HTML, and behavioral features; improves detection accuracy through multi-modal feature representation.	Y. Li, X. Hu, and J. Wang, "Multi-Modal Deep Learning for Malicious Website Detection," 2023.

Table 2.0 Literature Review Summary

CHAPTER-3

SOFTWARE REQUIREMENT ANALYSIS

3 SOFTWARE REQUIREMENT ANALYSIS

Software Requirement Analysis is a critical phase in the system development lifecycle, focusing on identifying, analyzing, and documenting the functional and non-functional requirements of the proposed Malicious Website Detection System using a Multi-Modal Features Representation-Based Convolutional Neural Network (CNN). This section outlines the software components, tools, libraries, and system specifications required for implementing, training, and evaluating the model.

3.1 Modules and Their Functionalities

3.1.1 Input Module

The Input Module collects web-related data such as URL information, HTML content, and network-based features from users or datasets through a web interface. It ensures that the data is entered or fetched in the correct format and prepares it for further processing.

3.1.2 Data Preprocessing Module

This module processes raw web data by performing cleaning, normalization, encoding, and handling of missing values. It ensures that all input features are transformed into a structured and model-compatible format, improving the efficiency and accuracy of the CNN model.

3.1.3 Feature Engineering Module

The Feature Engineering Module extracts relevant features from multiple sources, including URL-based features (length, symbols), content-based features (HTML tags, scripts), and network-based attributes.

3.1.4 Multi-Modal Feature Representation Module

This module combines different feature types into a single feature vector, enabling the system to capture both structural and semantic characteristics of websites.

3.1.5 Prediction Module (CNN Model)

This is the core module of the system, which uses a Convolutional Neural Network (CNN) to analyze the multi-modal input data.

3.1.6 Visualization Module

This module generates visual outputs such as accuracy graphs, confusion matrices, and performance charts using tools like Matplotlib. These visualizations help users understand model performance and detection results clearly.

3.1.7 Web Interface Module

The Web Interface Module provides interaction between the user and the system. It is developed using Flask, HTML, CSS, and Bootstrap, allowing users to input website data and view detection results through a user-friendly interface.

3.1.8 Output Module

The Output Module displays the final classification results, indicating whether a website is malicious or benign along with performance metrics. It ensures that outputs are clear, interpretable, and useful for decision-making.

3.1.9 Storage and Serialization Module

This module handles loading of the trained CNN model and preprocessing tools using the pickle library. It enables efficient reuse of trained components without retraining, improving system performance.

3.2. Functional Requirements

a. Data Preprocessing

The system begins by collecting web datasets that include URL-based features, HTML content, and network-related attributes. These datasets may originate from publicly available sources or real-time web inputs. Since raw web data is often unstructured and inconsistent, preprocessing plays a crucial role in preparing the data for further analysis. It ensures that all input data is organized in a structured and usable format for the detection model.

During preprocessing, several data cleaning operations are performed to improve data quality. Missing values are identified and handled appropriately, either by removal or imputation. Irrelevant, duplicate, or noisy data is eliminated to prevent misleading patterns during model training. Additionally, normalization and encoding techniques are applied to ensure uniformity across features, which helps in improving model convergence and performance.

Furthermore, textual data such as URLs and HTML content is transformed into a machine-readable format using techniques like tokenization and vectorization. These

processes convert textual information into numerical representations that can be effectively processed by the CNN model. Proper preprocessing enhances the reliability, accuracy, and efficiency of the overall malicious website detection system.

b. Multi-Modal Feature Representation

In this stage, different types of features—URL-based, content-based (HTML, scripts), and network-based attributes—are combined into a unified feature representation.

c. Model Training

The model training phase involves training the Convolutional Neural Network (CNN) using the prepared multi-modal dataset. The system utilizes efficient training techniques such as batch processing to handle large datasets. During training, key metrics such as loss, accuracy, and convergence are continuously monitored to ensure effective learning. The CNN automatically extracts hierarchical features and learns patterns that distinguish malicious websites from benign ones.

d. Model Evaluation & Testing

After training, the model is evaluated using performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix. These metrics help assess the effectiveness of the model in detecting malicious websites.

e. Visualization

Visualization is used to analyze model performance and results. Graphs such as accuracy curves, loss curves, and confusion matrices are generated to understand the learning behavior of the model. These visual representations help in interpreting results and identifying areas for improvement.

f. Prediction & Deployment

In the final stage, the trained CNN model is used to classify websites as malicious or benign in real-time or from stored datasets. The system provides outputs that can be integrated into web browsers, security tools, or intrusion detection systems.

3.3. Non-Functional Requirements

Performance

The system is designed to ensure fast and efficient processing of web data for real-time malicious website detection. It leverages optimized deep learning operations and, where available, GPU acceleration to speed up CNN model training and inference.

Scalability

The system is capable of handling large-scale web datasets consisting of thousands or millions of website samples with diverse feature types. It is designed to support future enhancements, including the integration of more advanced deep learning architectures.

Reliability

Reliability is maintained through robust data preprocessing, validation mechanisms, and error-handling techniques. The system can manage incomplete, noisy, or corrupted web data without significant degradation in performance.

Usability

The system is designed with a modular, clean, and well-documented code structure, making it easy to understand, maintain, and extend.

Portability

The system is platform-independent and can run on multiple operating systems such as Windows, Linux, and macOS. It is also compatible with cloud-based platforms like Google Colab and AWS, enabling flexible deployment and accessibility across different computing environments.

3.4. Feasibility Study

The feasibility study is a crucial part of the Software Requirement Analysis phase, aimed at determining whether the proposed Malicious Website Detection System using a Multi-Modal Features Representation-Based Convolutional Neural Network

(CNN) is practical, achievable, and beneficial for real-world implementation. It evaluates various aspects such as economic, technical, and social feasibility to ensure that the system is efficient, cost-effective, technologically viable, and user-friendly. This study helps in deciding whether the project should proceed and how resources should be allocated effectively.

3.4.1 Economic Feasibility

Economic feasibility focuses on evaluating whether the cost of developing and deploying the system is within available financial resources. The proposed system is cost-effective as it relies on open-source tools such as Python, TensorFlow or PyTorch, and visualization libraries, eliminating the need for expensive software licenses. The model training can be performed using affordable GPU platforms like Google Colab or standard computing systems, reducing hardware costs. Additionally, the system can be integrated with existing cybersecurity tools such as web browsers and intrusion detection systems without requiring major infrastructure changes. In the long term, the system helps prevent financial losses caused by cyberattacks, data breaches, and phishing incidents, making it economically beneficial.

3.4.2 Technical Feasibility

Technical feasibility evaluates whether the required technologies, tools, and expertise are available to implement the system effectively. The proposed CNN-based model is technically feasible as it uses widely adopted deep learning frameworks such as TensorFlow and PyTorch, which support efficient model development and deployment. The system processes standard web data, including URLs, HTML content, and network attributes, without requiring additional hardware or specialized equipment. Multi-modal feature integration and CNN architectures are well-supported and can handle complex data efficiently. The system can also be integrated into existing security platforms using APIs and modular design.

3.4.3 Social Feasibility

Social feasibility examines how well the proposed system is accepted by users such as cybersecurity professionals, organizations, and general internet users. The system is designed to be user-friendly, providing clear outputs.

The system reduces the need for manual monitoring and enhances decision-making by providing accurate and timely detection. It also improves overall online safety, protecting users from phishing attacks and malicious threats. Since the system integrates easily into existing platforms and requires minimal user training, it is highly acceptable. Therefore, the project achieves high social feasibility as it is easy to use, enhances user trust, and contributes to a safer digital environment.

CHAPTER-4

SOFTWARE AND HARDWARE REQUIREMENTS

4 SOFTWARE AND HARDWARE REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

The software requirements define the essential system environment, programming tools, and libraries needed to develop and implement the Malicious Website Detection System using a Multi-Modal Features Representation-Based Convolutional Neural Network (CNN). These components ensure efficient data processing, model training, and result visualization.

Provided Software Requirements

- Operating System: Windows 7 Ultimate
- Coding Language: Python

Software Requirements for GNN-based Implementation

- Operating System: Windows 10 / Windows 11 / Ubuntu 20.04+
- Programming Language: Python 3.8 or above

Development Environment

- Jupyter Notebook / VS Code / PyCharm

Libraries & Frameworks

- PyTorch – Deep learning model development
- Keras (if using TensorFlow) – High-level neural network API
- NumPy / Pandas – Data preprocessing
- BeautifulSoup / Requests – Web data extraction and parsing
- Matplotlib / Seaborn – Visualization

These software components provide complete support for data collection, preprocessing, multi-modal feature handling, CNN model training, and performance visualization required for malicious website detection.

4.2 HARDWARE REQUIREMENTS

The hardware requirements specify the minimum and recommended system configuration needed to efficiently run the CNN-based malicious website detection model. Since deep learning models involve computationally intensive operations, especially during training, the system must support adequate processing power, memory, and optional GPU acceleration for improved performance.

Hardware Specifications

- Processor: Intel i5/i7 or AMD Ryzen 5/7
- RAM: Minimum 8 GB (16 GB recommended)
- Storage: 256 GB SSD or higher
- GPU: NVIDIA GPU with CUDA support (e.g., GTX 1650 / RTX series) for faster model training
- Monitor: Full HD Display
- Peripherals: Standard keyboard and optical mouse

CHAPTER-5

SOFTWARE DESIGN

5 SOFTWARE DESIGN

5.1 SYSTEM ARCHITECTURE

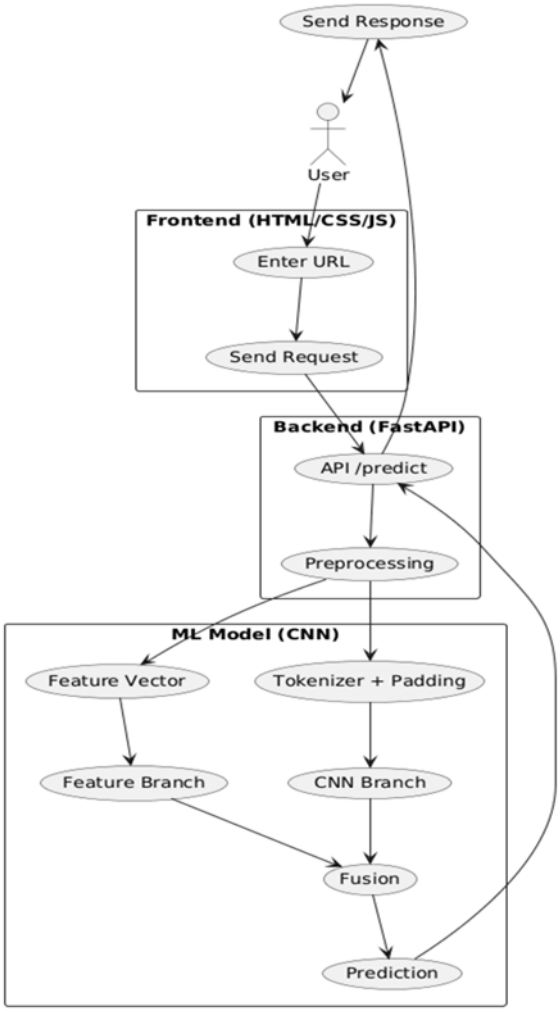

Fig 5.1 System Architecture

The system architecture provides a clear overview of the malicious website detection workflow using a Multi-Modal Features Representation-Based Convolutional Neural Network (CNN). It begins with data collection from multiple sources, including URL features, HTML content, and network-based attributes, which together represent the behavior and structure of websites. These diverse data sources are essential for capturing both the structural and semantic characteristics of web pages.

After preprocessing, the system performs feature engineering, where relevant features are extracted from different modalities. URL-based features include length, special characters, and domain information; content-based features are derived from HTML tags, scripts, and embedded elements; and network-based features capture traffic-related attributes. These features are then combined in the multi-modal feature representation stage, where they are transformed into a unified feature vector. This integration allows the system to capture complex relationships and hidden patterns that may not be visible when using a single data source.

The unified feature representation is then fed into the Convolutional Neural Network (CNN), which serves as the core component of the architecture. The CNN consists of convolutional layers, pooling layers, and fully connected layers that automatically learn hierarchical feature representations. It identifies important patterns and correlations within the data, enabling accurate classification of websites as malicious or benign. Techniques such as dropout and batch normalization are applied to improve model performance, prevent overfitting, and ensure stable learning.

The architecture also includes a training and optimization phase, where the model is trained using labeled datasets of malicious and benign websites. Optimization algorithms such as Adam optimizer and loss functions like binary cross-entropy are used to enhance learning efficiency and accuracy. The model is trained over multiple epochs with batch processing to handle large datasets effectively. Finally, the trained model generates prediction outputs, classifying websites in real time. These results can be used to alert users, block harmful websites, and improve overall cybersecurity, making the system practical, scalable, and effective for real-world applications.

5.2 Data Flow Diagram

The Data Flow Diagram (DFD) illustrates how data moves through the Malicious Website Detection System, from input to final classification output and visualization.

The process begins when the user provides a website URL or uploads web-related data through the web interface. This input is captured by the Flask-based UI and forwarded to the backend system for processing. The raw data is then passed to the preprocessing module, where it undergoes cleaning, normalization, encoding, and

handling of missing or noisy values to ensure consistency and compatibility with the trained model.

In addition to displaying results, the system stores input data and prediction outcomes in a database such as SQLite. This stored information is used for maintaining prediction history and further analysis. The visualization module retrieves stored data to generate graphs such as accuracy charts, confusion matrices, and trend analysis, helping users understand system performance and detection patterns. This structured data flow ensures efficient processing, accurate detection, and effective user interaction within the system.

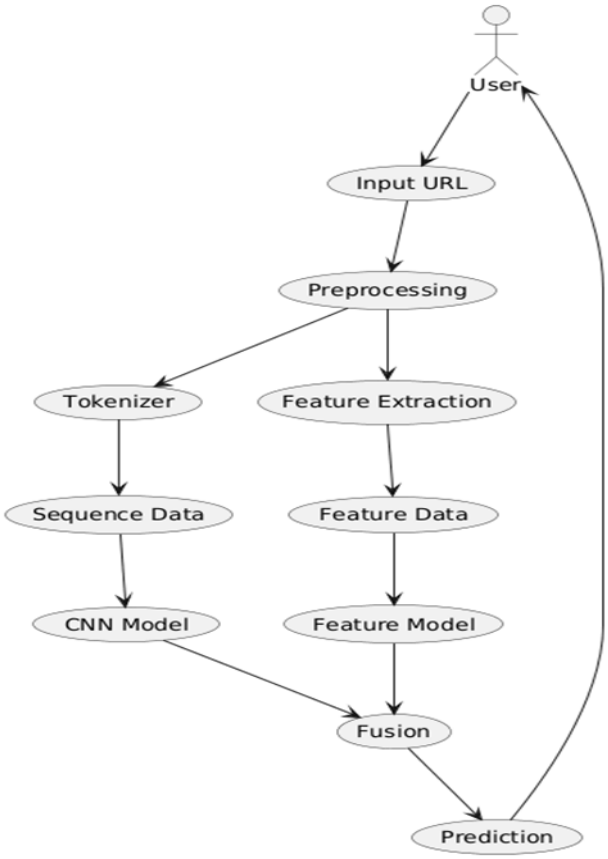

Fig 5.2 Data Flow Diagram

5.2.1 Key Components of Data Flow:

- User Input:
  Users provide a website URL or web-related data through the web interface.
- Input Handling:
  The Flask-based application receives the input and forwards it to the backend for processing.
- Data Preprocessing:
  Input data is cleaned, normalized, encoded, and transformed into a structured format suitable for model input.
- Feature Extraction & Multi-Modal Representation:
  Relevant features are extracted from URL, HTML content, and network attributes, and combined into a unified feature vector.
- CNN Model Prediction:
  The processed multi-modal data is passed through the CNN model, which analyzes patterns and generates predictions.
- Prediction Display:
  The classification result along with probability scores is displayed to the user on the interface.
- Database Storage:
  User inputs and prediction results are stored in an SQLite database for future referencee.
- Predictions History:
  Stored records are retrieved and displayed in a structured format for user analysis.
- Graph Generation:
  Visual representations such as accuracy graphs, confusion matrices, and trend charts are generated from stored data to analyze system performance.

5.3 UML Diagrams

UML (Unified Modeling Language) is a standardized modeling language used for specifying, visualizing, designing, and documenting the components of software systems. It helps in representing the structure and behavior of a system in a clear and organized manner, making it easier for developers and stakeholders to understand the overall design. UML was developed by the Object Management Group (OMG) and is widely used in object-oriented analysis and design.

The Behavioral UML diagrams describe the behavior of the system, its actors, and the interaction between the components. On the other hand, Structural UML diagrams depict the static structure of the system, showing its components and relationships. UML has been integrated as a standard by OMG, and its primary goals are to provide a formal basis for understanding modeling languages.

Goals of UML:

To provide a standard visual representation of the system design.

To improve communication among team members during development.

To model both structural and behavioral aspects of the system.

To support object-oriented design and development practices.

To document the system clearly for future reference and maintenance

Types of UML Diagrams

The different types are broken down as follows:

1. Sequence diagram
2. Use case Diagram
3. Activity diagram
4. Collaboration diagram
5. Class Diagram
6. Deployment Diagram

5.3.1. SEQUENCE DIAGRAM

The sequence diagram of the proposed multi-modal malicious website detection system illustrates the chronological interaction between various system components as they collaborate to process input and produce the final classification result. The sequence begins when the user provides the website information, such as the URL, HTML content, screenshot, or metadata. This input is then passed to the preprocessing module.

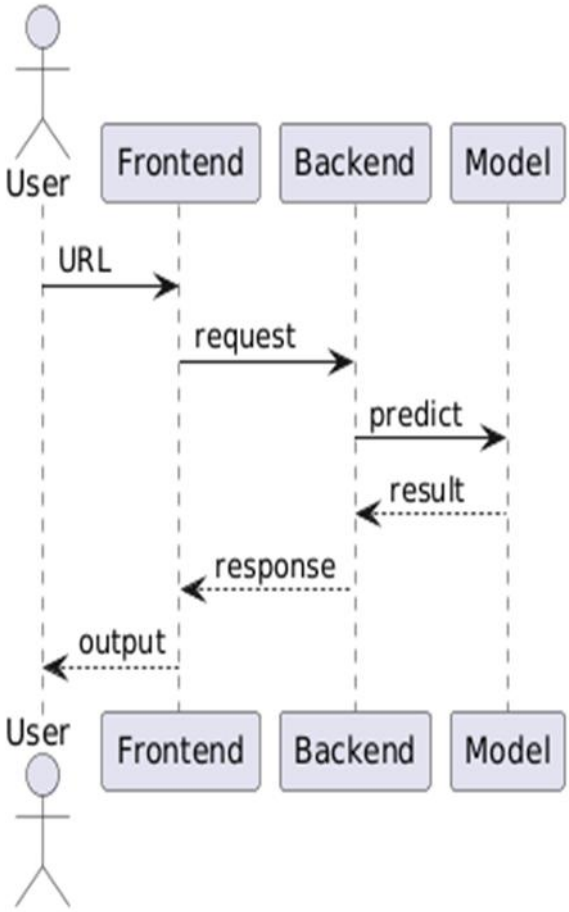

Fig 5.3.1 Sequence Diagram

5.3.2 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

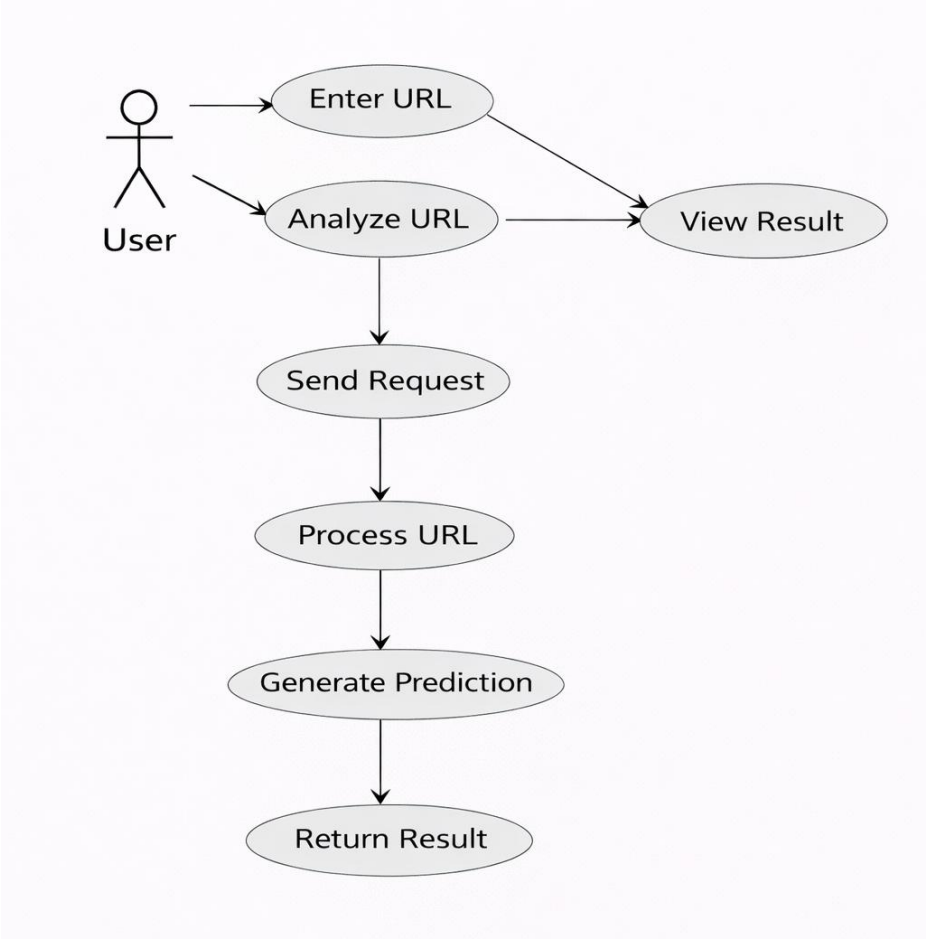

Fig 5.3.2 Use Case Diagram

5.3.3 COLLABORATION DIAGRAM

The collaboration diagram for the Malicious Website Detection System illustrates how different system components interact to detect whether a website is malicious or benign. The process begins with the **User**, who provides a website URL or related web data and initiates the detection process.

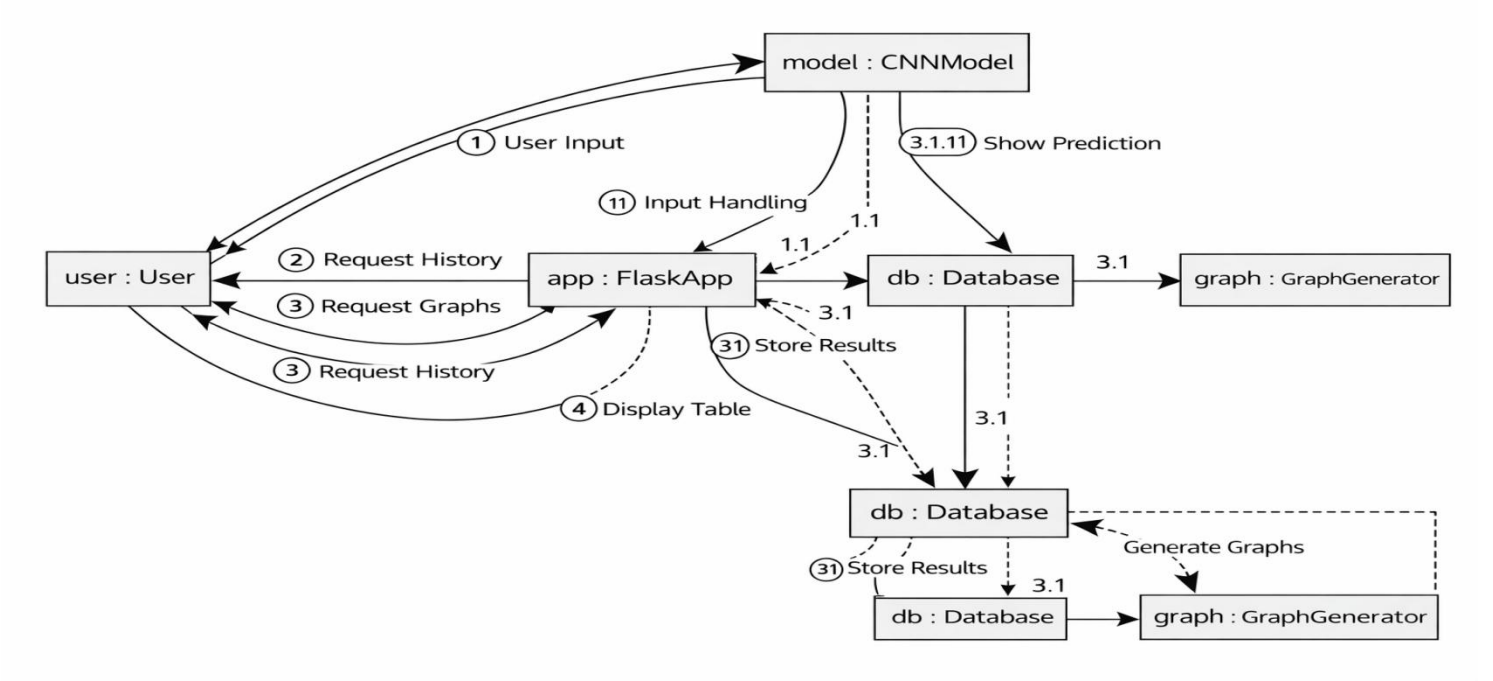

Fig 5.3.3 Collaboration Diagram

5.3.4 ACTIVITY DIAGRAM

The Fig 5.3.4 It begins when the user enters a website URL, which is then sent to the backend server for processing. The system first performs preprocessing on the URL, such as cleaning and formatting the input data. After preprocessing, the URL is tokenized and padded so that it can be converted into a numerical format suitable for machine learning models.

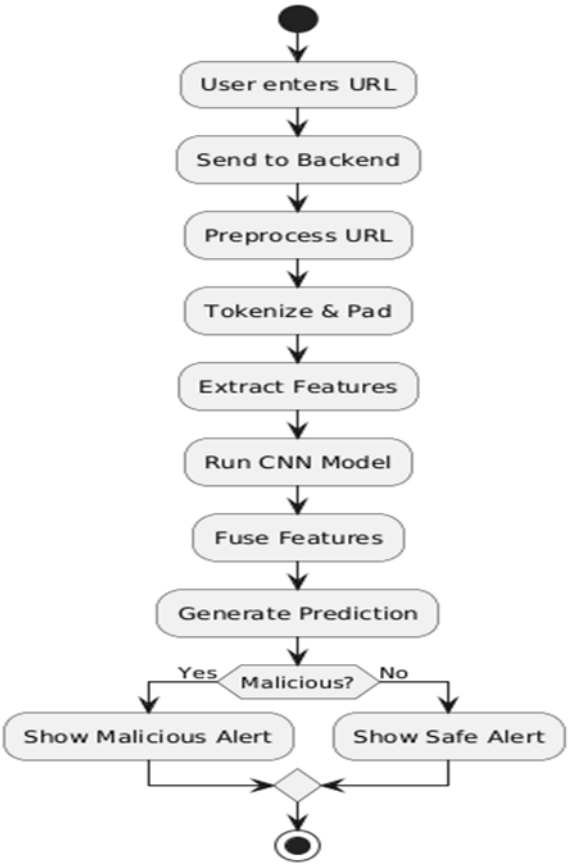

Fig 5.3.4 Activity Diagram

5.3.5 CLASS DIAGRAM

The class diagram of the Malicious Website Detection System represents the structural design of the application by defining its key classes, attributes, and relationships. At the core is the **DatasetManager** class, which handles the collection and management of web data, including URL features, HTML content, and network-based attributes. The **FlaskApp** class acts as the main controller, managing user interactions and coordinating communication between different modules. The **CNNModel** class defines the architecture of the Convolutional Neural Network, including layers and parameters, and contains methods such as forward() to process input data and generate predictions. The **Database** class is responsible for storing and retrieving user inputs and prediction results, supporting features like history tracking. The **GraphGenerator** class creates visual representations such as graphs and charts to help users analyze results effectively.

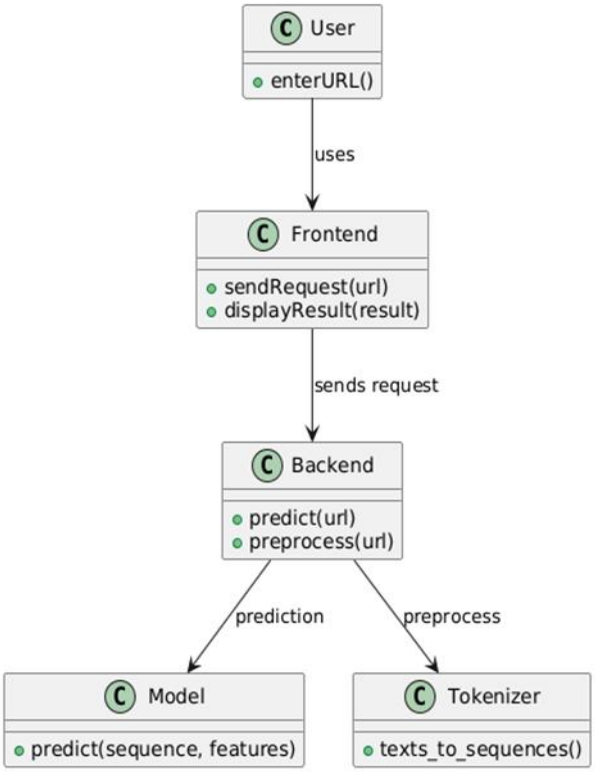

Fig 5.3.5 Class Diagram

5.3.6 DEPLOYMENT DIAGRAM

The deployment diagram for the Malicious Website Detection System illustrates how different system components are distributed across physical and virtual environments for efficient execution. At the base level, the system includes a **Client Machine Node**, which represents the user's device (such as a computer or mobile browser) where the user inputs a website URL or web data and requests detection results. The client communicates with the **Web Server Node**, which hosts the Flask-based application responsible for handling user requests, managing input processing, and displaying outputs.

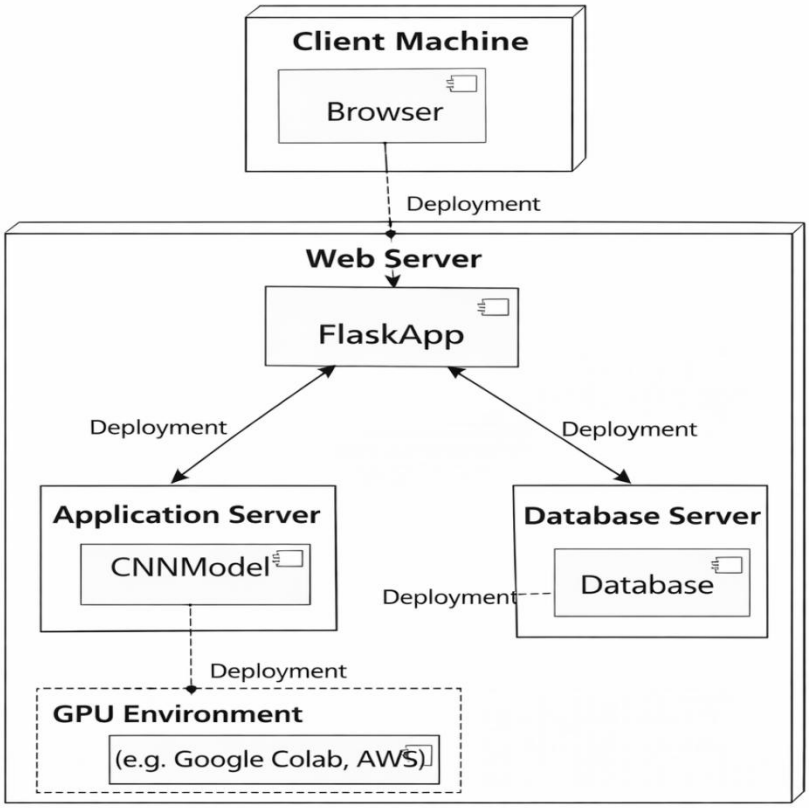

Fig 5.3.6 Deployment Diagram

CHAPTER-6

IMPLEMENTATION AND CODING

6. IMPLEMENTATION AND CODING

6.1 Source Code

App.py

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from fastapi.responses import FileResponse
from fastapi.staticfiles import StaticFiles

from features import extract_features_59

import numpy as np
import pickle
from pydantic import BaseModel
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from urllib.parse import urlparse

app = FastAPI()

# CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# STATIC FILES
app.mount("/static", StaticFiles(directory="."), name="static")

# LOAD MODEL
model = load_model("malicious_cnn_model_v2.keras")
with open("tokenizer.pkl", "rb") as f:
    tokenizer = pickle.load(f)
```

```
max_len = 100
feature_size = 59

class URLRequest(BaseModel):
    url: str

#  Extract domain properly
def extract_domain(url):
    domain = urlparse(url).netloc.lower()
    if domain.startswith("www."):
        domain = domain[4:]
    return domain

#  Trusted domains
trusted_domains = {
    "google.com",
    "facebook.com",
    "youtube.com",
    "amazon.com",
    "github.com",
    "chatgpt.com"
}

#  Final decision logic (SAFE + WORKING)
def final_prediction(url, model_prob):
    domain = extract_domain(url)

    is_trusted = any(domain == d or domain.endswith("." + d) for d in trusted_domains)

    #  TRUSTED DOMAIN FIRST (absolute priority)
    if is_trusted:
        return "Safe"
```

```
    #  High confidence malicious
    if model_prob > 0.98:
        return "Malicious"

    # Normal threshold
    if model_prob > 0.3:
        return "Malicious"
    else:
        return "Safe"

@app.get("/")
def home():
    return FileResponse("index.html")

@app.post("/predict")
def predict(request: URLRequest):
    url = request.url

    try:
        #  Text processing
        seq = tokenizer.texts_to_sequences([url])
        seq = pad_sequences(seq, maxlen=max_len, padding='post')
        seq = seq.astype(np.int32)

       #  Engineered features (REAL 59 features)
        raw_features = extract_features_59(url)

        features = np.array(raw_features, dtype=np.float32).reshape(1, -1)

        print("Feature vector length:", len(raw_features))

        #  Model prediction
        pred = model.predict([seq, features])[0][0]
```

```
        print("Model probability:", pred)

        #  Final calibrated decision
        result = final_prediction(url, pred)

        return {
            "url": url,
            "malicious_probability": float(pred),
            "prediction": result
        }

    except Exception as e:
        return {
            "error": str(e)
        }
```

features.py

```
import re
from urllib.parse import urlparse

def extract_features_59(url):
    features = []

    parsed = urlparse(url)
    domain = parsed.netloc
    path = parsed.path
    query = parsed.query

    # --- BASIC ---
    features.append(len(url))
    features.append(url.count('@'))
    features.append(url.count('?'))
    features.append(url.count('-'))
    features.append(url.count('='))
    features.append(url.count('.'))
    features.append(url.count('#'))
    features.append(url.count('%'))

    features.append(url.count('+'))
    features.append(url.count('$'))
    features.append(url.count('!'))
    features.append(url.count('*'))
```

```
    features.append(url.count(','))
    features.append(url.count('//'))

    # --- DIGITS / LETTERS ---
    features.append(sum(c.isdigit() for c in url))
    features.append(sum(c.isalpha() for c in url))

    # --- DOMAIN ---
    features.append(len(domain))
    features.append(domain.count('.'))

    # --- SECURITY ---
    features.append(1 if url.startswith("https") else 0)

    ip_pattern = r'(\d{1,3}\.){3}\d{1,3}'
    features.append(1 if re.search(ip_pattern, url) else 0)

    # --- QUERY ---
    features.append(len(query))
    features.append(query.count('&'))

    # --- PATH ---
    features.append(len(path))
    features.append(path.count('/'))
    features.append(path.count('='))

    # --- PHISH WORDS ---
    words = ['login', 'verify', 'update', 'secure', 'bank']
    features.append(1 if any(w in url.lower() for w in words) else 0)

    # --- FILL TO 59 ---
    while len(features) < 59:
        features.append(0)

    return features
```

main.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Malicious URL Detector</title>

    <!-- CSS -->
    <link rel="stylesheet" href="/static/style.css">

    <!-- FONT (Manrope) -->
    <link href="https://fonts.googleapis.com/css2?family=Manrope:wght@400;600&display=swap"
rel="stylesheet">

    <!-- ICON -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.0/css/all.min.css">

    <!-- SWEET ALERT -->
    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
</head>

<body>

<div class="container">

    <h1>Malicious URL Detector</h1>
    <p class="subtitle">Detect phishing & malicious websites using Deep Learning</p>

    <div class="input-box">

        <div class="input-wrapper">
            <i class="fas fa-search"></i>
            <input id="urlInput" type="text" placeholder="Enter URL (https://example.com)">
        </div>

        <button onclick="checkURL()">Analyse</button>
```

```
    </div>

</div>

<div class="image-section">
    <img src="/static/Images/image_1.jpeg">
    <img src="/static/Images/image_2.jpeg">
</div>

<script src="/static/script.js"></script>

</body>
</html>
```

Script.js

```
function isValidURL(url) {
    try {
        new URL(url);
        return true;
    } catch {
        return false;
    }
}

async function checkURL() {

    const url = document.getElementById("urlInput").value.trim();

    if (!url) {
        Swal.fire({
            icon: "warning",
            title: "Enter a URL first"
```

```
    });
    return;
}

if (!isValidURL(url)) {
    Swal.fire({
        icon: "error",
        title: "Invalid URL"
    });
    return;
}

// LOADING ANIMATION
Swal.fire({
    title: "Analyzing URL...",
    html: "Scanning for threats...",
    allowOutsideClick: false,
    showConfirmButton: false,
    didOpen: () => {
        Swal.showLoading();
    }
});

try {
    const res = await fetch("http://127.0.0.1:8000/predict", {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify({ url: url })
    });
```

```
    const data = await res.json();

    // small delay → makes animation feel premium
    setTimeout(() => {

      if (data.prediction === "Safe") {

        Swal.fire({
          icon: "success",
          title: "Safe Website",
          html: `
            <b>No threats detected</b><br><br>
            Confidence: ${(1 - data.malicious_probability).toFixed(2)}
          `,
          confirmButtonColor: "#28a745"
        });

      } else {

        Swal.fire({
          icon: "error",
          title: "Malicious URL Detected",
          html: `
            <b>This site may be dangerous!</b><br><br>
            Risk Score: ${data.malicious_probability.toFixed(2)}
          `,
          confirmButtonColor: "#d33"
        });
      }
    }, 800); //  animation delay
  } catch (err) {
    Swal.fire({
```

```
icon: "error",
        title: "Server Error",
        text: "Make sure backend is running"
      });

  }
}
```

6.2. IMPLEMENTATION

The implementation of the Multi-Modal Features Representation-Based CNN Model for Malicious Website Detection is carried out by integrating web technologies, backend processing, and deep learning techniques to build a complete and efficient system. The system is designed to accurately detect malicious websites in real time while ensuring user-friendly interaction. It consists of frontend development for user input, backend processing for handling data and logic, model integration for classification, and deployment for real-world usage.

A. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep learning models widely used for feature extraction and pattern recognition. In this project, CNN is used to automatically learn complex patterns from multi-modal data such as URL features, HTML content, and network attributes. Unlike traditional machine learning models, CNNs eliminate the need for manual feature engineering by learning hierarchical representations directly from the data. The CNN architecture typically consists of convolutional layers, pooling layers, and fully connected layers. The general convolution operation can be represented as:

$$y=\sigma(W*x+b^{k-1})_u)$$

Where $_vx$ is the representation of node W $i s t h e$, $N(v)$represents neighboring nodes, and $W^{(k)}$is the learnable weight matrix. Through iterative aggregation and transformation.

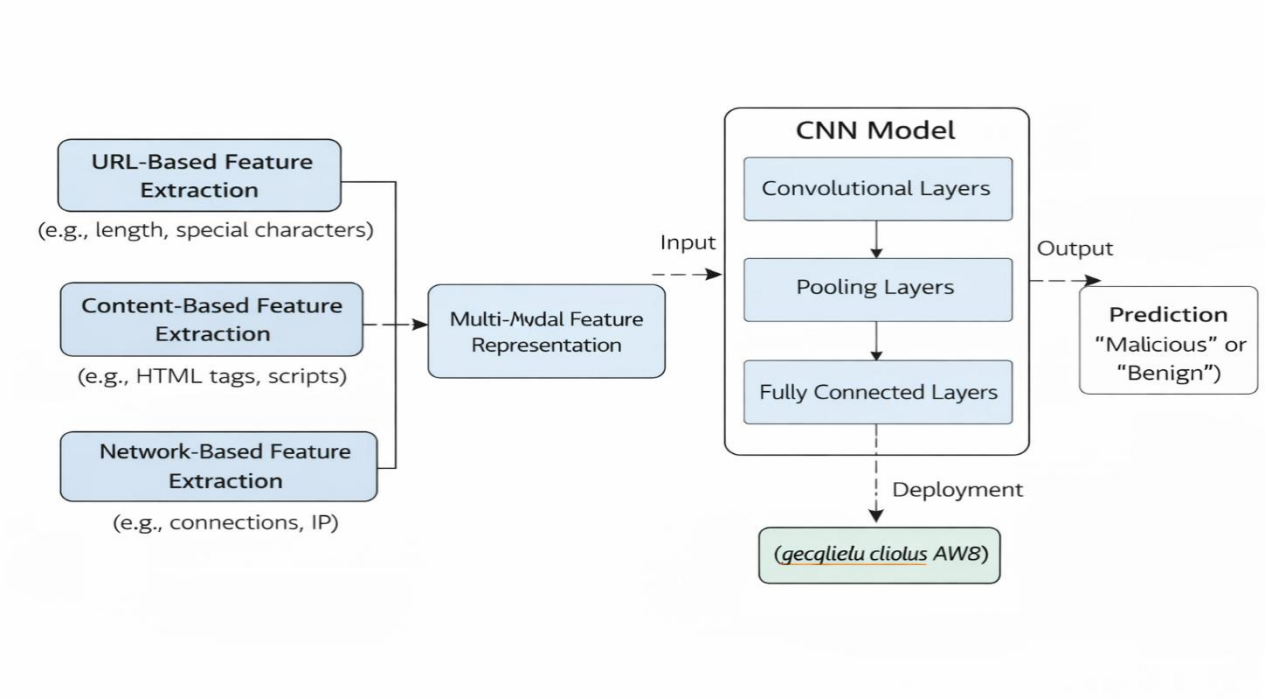

Fig 6.2.1 Architecture of Graph Neural Network (GNN)

B. Multi-Modal Feature Representation

In this project, multiple types of data are combined to improve detection accuracy. These include URL-based features (length, special characters, domain information), content-based features (HTML tags, scripts, embedded elements), and network-based attributes. Each type of data provides unique information about the behavior of a website.

The extracted features from different sources are integrated into a unified feature representation, allowing the CNN model to analyze both structural and semantic characteristics simultaneously. This multi-modal approach enhances the model's ability to detect sophisticated and previously unseen malicious patterns.

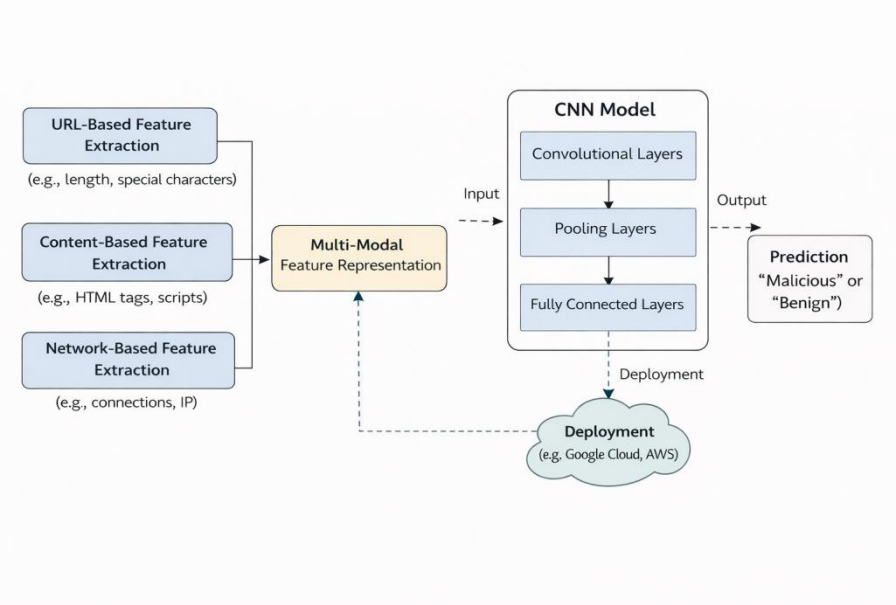

Fig. 6.2.2 Illustration of CNN operation

How CNN + Multi-Modal Representation Work Together in the Project

In the proposed system, multi-modal feature representation provides rich and diverse input data, while the CNN acts as the core learning and classification model. The combined workflow is as follows:

- Web data is collected from user input (URL or dataset).
- Data is preprocessed and cleaned for consistency.
- Features are extracted from multiple modalities (URL, HTML, network).
- The features are combined into a unified representation
- The CNN processes the input and learn hierarchical

patterns.

- This integration enables the model to:
- Capture complex patterns across different data sources
- Improve detection accuracy compared to single-feature models
- Handle large-scale network data efficiently
- Support real-time malicious website detection and cybersecurity applications

6.2.1 Frontend Implementation

The frontend of the Malicious Website Detection System is developed using HTML, CSS, and Bootstrap, providing a clean, responsive, and user-friendly interface. The main page allows users to enter a website URL or upload web-related data for analysis. The input interface is designed to be simple and intuitive, ensuring that users can easily submit data without any technical complexity.

In addition to the input page, the system includes a predictions page where previously analyzed websites and their classification results (malicious or benign) are displayed in a structured table format. This enables users to review historical results along with corresponding inputs. Pagination is implemented to efficiently manage large datasets and improve readability. Another important component of the frontend is the graphs page, which displays visualizations such as accuracy charts, confusion matrices, and trend graphs. These visualizations help users understand detection performance and patterns over time. A navigation bar is provided across all pages for easy access to input, results, and visualization sections. The use of Bootstrap ensures that the interface is fully responsive and works smoothly across desktops, tablets, and mobile devices.

6.2.2 Backend Implementation

The backend of the system is developed using the Flask framework in Python, which acts as the core component responsible for handling user requests and managing system operations. Flask manages routing, ensuring that different user actions such as submitting a URL, viewing predictions, or accessing graphs are handled efficiently.

When a user submits input data, the backend receives and validates it to ensure correctness and completeness. The input is then processed and converted into a format suitable for the model. Data preprocessing steps such as cleaning, normalization, and encoding are applied to maintain consistency with the training dataset.

The system uses SQLite as a lightweight database to store prediction results. A table is created to store user inputs and their corresponding websites.

6.2.3 Model Integration

The core functionality of the system is achieved through the integration of a Convolutional Neural Network (CNN) model, implemented using deep learning frameworks such as TensorFlow or PyTorch. The model is designed to process multi-modal data, including URL features, HTML content, and network-based attributes. The CNN architecture consists of multiple convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. Activation functions such as ReLU are used to introduce non-linearity, and dropout layers are applied to prevent overfitting. The final layer produces output probabilities using a sigmoid or softmax function.

During system initialization, the trained model and preprocessing tools are loaded into memory. When new input data is received, it is preprocessed and transformed into a feature vector representing multiple modalities. This data is then passed through the CNN model, which performs a forward pass to generate prediction probabilities.

6.2.4 Deployment

The system is deployed using Flask's built-in development server, allowing it to run locally and be accessed through a web browser. The deployment process includes setting up the environment, installing required libraries such as Flask, TensorFlow/PyTorch, NumPy, and Matplotlib, and ensuring proper integration of all components. The system is designed to utilize available hardware resources efficiently. If a GPU is available, it accelerates model computation; otherwise, the system runs on CPU, ensuring compatibility across different devices. The entire application—including frontend templates, backend logic, trained CNN model, and database—is integrated into a single structure for easy maintenance.

The deployed system allows users to input website data, receive real-time predictions, and view results in both tabular and graphical formats. It is also scalable and can be extended to cloud platforms or integrated with real-time web security systems.

In addition, the deployment of the system can be extended to cloud-based environments such as AWS, Google Cloud, or Microsoft Azure to enhance scalability, availability, and performance. By hosting the application on cloud servers, the system can handle multiple user requests simultaneously and provide real-time malicious website detection services to a larger audience. Containerization tools like Docker can be used to package the application along with its dependencies, ensuring consistent performance across different environments. Furthermore, APIs can be developed to integrate the model with web browsers, firewalls, or security systems, enabling automated detection and blocking of malicious websites. Regular updates and model retraining can also be incorporated into the deployment pipeline to keep the system effective against evolving cyber threats, ensuring long-term reliability and adaptability.

CHAPTER-7

SYSTEM TESTING

7. SYSTEM TESTING

System testing is a crucial phase in the Software Development Life Cycle (SDLC) that evaluates the Malicious Website Detection System as a complete and integrated application. Unlike unit or integration testing, which focus on individual components, system testing ensures that all modules—such as the user interface, preprocessing module, feature extraction, CNN model, database, and visualization components—work together seamlessly. The main objective is to verify that the system accurately detects malicious websites, processes inputs correctly, and delivers reliable outputs that meet both functional and non-functional requirements. It also ensures that the system behaves correctly in real-world scenarios, providing users with accurate and timely detection results.

System testing includes multiple testing types to validate different aspects of the system. Functional testing ensures that the system correctly accepts website URLs, processes multi-modal features, and generates accurate classification results (malicious or benign). Performance testing evaluates the system's response time and efficiency when handling multiple requests or large datasets. Load and stress testing examine how the system performs under heavy traffic conditions, ensuring stability and robustness. Security testing is particularly important in this project, as it verifies that the system is protected against vulnerabilities and safely handles user inputs without exposing sensitive data. Usability testing focuses on ensuring that the interface is user-friendly, intuitive, and accessible across different devices.

In Effective system testing also involves well-defined testing strategies and test cases that cover normal operations, edge cases, and potential failure scenarios. Automated testing tools can be used to improve testing efficiency and reduce manual effort, especially for repetitive tasks. By thoroughly validating the integrated system, system testing ensures that the malicious website detection system is accurate, reliable, secure, and ready for real-world deployment, ultimately enhancing cybersecurity and user trust.

7.1 Types of Tests

a. Unit Testing

Unit testing focuses on individual components of the Malicious Website Detection System, such as the preprocessing functions, feature extraction module, CNN model functions, and database operations, to ensure they work correctly in isolation before integration.

Key Features:

- Tests all logical paths and code branches in modules.
- Performed during development before combining modules.
- Requires understanding of internal code structure.

b. Integration Testing

Integration testing verifies that different modules—such as frontend, backend, preprocessing, CNN model, and database—interact correctly and exchange data without errors.

Key Features:

- Ensures smooth communication between Flask backend and CNN model.
- Validates proper data flow from user input to prediction output.

c. Functional Testing

Functional testing ensures that the system meets all specified requirements, such as correctly classifying websites as malicious or benign based on input data.

Key Features:

- Validates correct handling of valid and invalid URLs.
- Ensures accurate prediction results from the CNN model.

d. System Testing

System testing evaluates the complete application as a whole, ensuring all integrated components function together correctly in real-world scenarios.

Key Features:

- Tests the entire workflow from input to prediction and visualization.
- Ensures system stability, accuracy, and reliability.
- Conducted after integration testing.

e. White Box Testing

White box testing involves testing internal logic, algorithms, and code implementation of modules such as preprocessing, feature extraction, and CNN model layers.

Key Features:

- Evaluates internal logic and code structure.
- Requires in-depth knowledge of the software.

f. Black Box Testing

Black box testing focuses on validating the system's functionality based on input and output without considering internal implementation.

Key Features:

- Does not require code knowledge.
- Based on functional specifications and requirements.

Test Strategy and Approach

Testing is carried out using both manual and automated methods to ensure complete system validation and accuracy.

Key Testing Objectives:

- Validate that the system accepts valid website URLs and rejects invalid inputs.
- Ensure proper navigation between input, prediction, and visualization pages

Features to be Tested

- Correct format validation for all input fields.
- Prevention of duplicate entries.
- Verification that all links navigate to the Cointegration Testing

Integration testing verifies that multiple software components interact without defects.

Key Features:

- Detects interface and communication errors between components.
- Ensures smooth data exchange between integrated modules.

Test Results

- All test cases executed successfully.
- No defects encountered during testing.

g. Acceptance Testing

Acceptance testing (User Acceptance Testing - UAT) confirms that the system meets user requirements and is ready for deployment.

Key Features:

- Ensures all functional requirements are met.
- Requires active participation from end-users.

Test Results:

- All acceptance test cases passed successfully.
- No defects encountered, confirming system readiness.

System testing is an essential phase in the development of the Malicious Website Detection System, ensuring that the application meets user expectations and cybersecurity requirements. Various testing methods, including unit testing, integration testing, functional testing, and system testing, are performed to identify and resolve potential issues before deployment. These testing processes validate that the system accurately detects malicious websites, handles inputs efficiently, and provides reliable outputs.

7.2 Testing Strategies

Testing strategies define the approach used to ensure that the Malicious Website Detection System works correctly, efficiently, and meets user and security requirements. A structured testing strategy is followed to validate each module as well as the complete system.

7.2.1 Testing Approach

The testing process is carried out in a systematic manner, starting from individual components and progressing toward the complete system. Initially, unit testing is performed to verify modules such as data preprocessing, feature extraction, CNN model functions, and database operations independently.

7.2.2 Testing Methods

Both manual and automated testing methods are used to ensure comprehensive system validation. Manual testing is applied to evaluate the user interface, navigation, and usability of the system.

7.2.3 White Box and Black Box Testing

The system uses both white box and black box testing techniques. White box testing focuses on the internal logic of modules such as preprocessing, feature extraction, and CNN model execution, ensuring all code paths and conditions are properly tested.

7.2.4 Functional and System Testing Strategy

Functional testing ensures that all features of the system work according to requirements, such as accepting valid URLs, rejecting invalid inputs, and correctly classifying websites as malicious or benign. System testing evaluates the entire application.

7.2.5 Integration Testing Strategy

Integration testing focuses on verifying the interaction between system components such as the web interface, backend processing, CNN model, and database. It ensures that data flows correctly between modules and that there are no communication or data exchange errors, confirming smooth system operation.

7.2.6 User Acceptance Testing (UAT)

User Acceptance Testing is conducted with real users to validate that the system meets practical requirements. Users test the system by providing URLs and analyzing results, offering feedback on usability, accuracy, and performance. Based on this feedback, necessary improvements are made before final deployment.

7.2.7 Testing Objectives

The main objective of testing is to ensure that the system is accurate, reliable, secure, and efficient. It validates proper input handling, smooth navigation, correct classification results, and effective data storage. It also ensures that the system performs well under various conditions and provides a user-friendly experience, making it suitable for real-world cybersecurity applications.

7.3 Test Cases

Test Case ID	Test Description	Input	Expected Output	Remarks
TC01	Valid prediction Input (Benign website).	https://www.google.com	Predicted label: Benign	Tests end-to-end prediction for safe website
TC02	Valid prediction input (Malicious website)	http://login-secure-update.xyz1	Predicted label: Malicious	Tests detection of suspicious URL.
TC03	Valid input with HTML features.	URL + HTML content with scripts1	Predicted label: Malicious	Tests multi-modal feature extraction.
TC04	Valid input with lexical features.	http://free-offer-win-money.com	Predicted label: Malicious	Tests detection based on URL patterns
TC05	Valid input with clean structure	https://www.wikipedia.org	Predicted label: Benign	Tests classification of trusted website.
TC06	Invalid URL format	htp:/invalid-url	Error or validation prompt	Ensures input validation for incorrect format
TC07	Missing input	Empty input field	Error message	Checks handling of missing input

Test Case ID	Test Description	Input	Expected Output	Remarks
TC08	Extremely long URL	Very long URL string (>500 chars)	Predicted label (any valid class)	Tests system performance with large inputs
TC09	Special characters in input	http://$$$malicious###.com	Predicted label: Malicious or Error	Ensures robustness to unusual characters
TC10	Mixed feature input	URL + partial HTML + metadata	Predicted label (Malicious/ Benign)	Tests combined multi-modal input handling
TC11	Repeated input submission	Same URL submitted multiple times	Same prediction result	Ensures consistency and stability
TC12	Unknown/new pattern	Newly generated suspicious URL	Predicted label (Malicious/ Benign)	Tests model generalization capability
TC13	Test Description: URL with special characters	http://example.com/login?user=abc&pass=123	Valid classification (Benign/Malicious)	Ensures system handles query parameters

Table 7.3 Test Cases

CHAPTER - 8

RESULTS

8. RESULTS

The figure 8.1 represents the main user interface of the Malicious URL Detection system. The interface is designed to be simple, user-friendly, and responsive, allowing users to easily input a website URL for analysis. It consists of a text input field where users can enter the URL and an “Analyse” button to initiate the detection process.

The system leverages a deep learning-based model to detect phishing and malicious websites in real-time. The background visuals and design elements are incorporated to enhance user experience while maintaining clarity and usability. This interface acts as the entry point for users to interact with the system and perform security analysis efficiently.

Fig 8.1 Main Interface (Home Screen)

In addition to its simplicity, the interface is designed to support seamless interaction between the user and the backend system. When a URL is entered and the “Analyse” button is clicked, the request is sent to the backend server, where preprocessing, feature extraction, and model prediction are performed. The system ensures minimal response time, enabling near real-time detection of malicious websites. This efficient interaction between the frontend and backend components enhances the overall performance and usability of the system, making it suitable for practical deployment in cybersecurity applications.

The figure 8.2 illustrates the output generated when the system classifies a website as safe. Upon analyzing the entered URL, the model determines that the website does not exhibit malicious characteristics and is safe for access.

The system displays a confirmation message along with a confidence score indicating the model's certainty in its prediction. This output helps users make informed decisions by assuring them that the website is not associated with phishing or malicious activities. The use of visual indicators such as a green check mark enhances clarity and user trust.

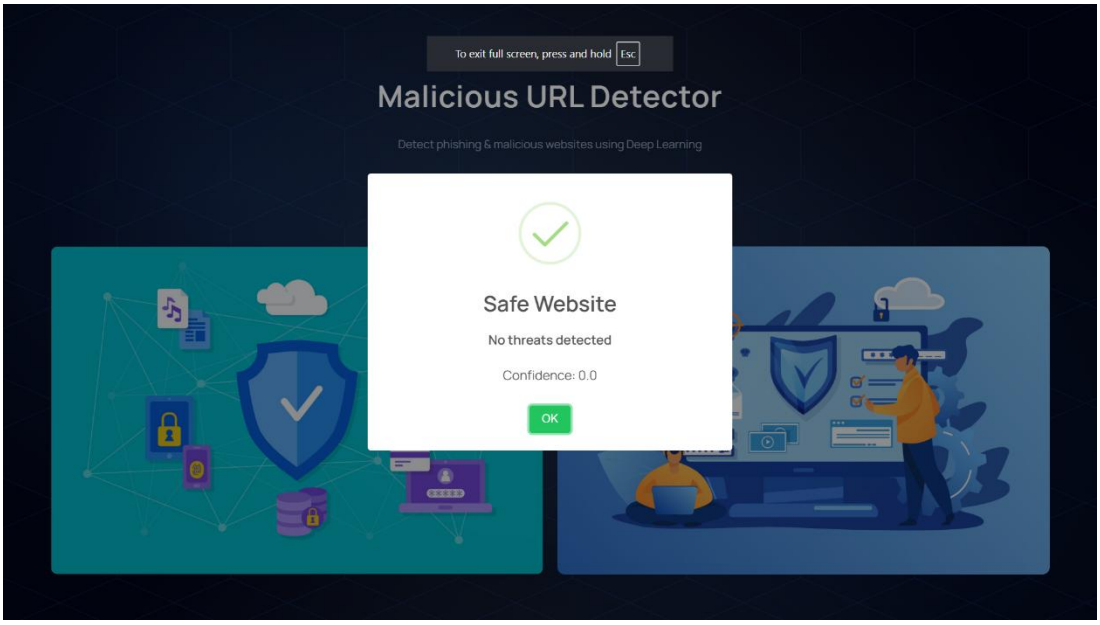

Fig 8.2 Safe Website Output

In addition, the system not only provides a binary classification but also offers a confidence score, which reflects the probability of the website being safe based on the learned patterns of the CNN model. This probabilistic output helps in understanding the reliability of the prediction and adds transparency to the system's decision-making process. Such feedback is particularly useful in real-world applications, where users may require an extra level of assurance before interacting with a website. The clear and concise presentation of results enhances user experience and builds trust in the system.

The figure 8.3 shows the output when the system detects a malicious website. After processing the input URL, the deep learning model identifies suspicious patterns and classifies the website as malicious.

The system displays a warning message along with a risk score, indicating the severity of the threat. This alert helps users avoid accessing potentially harmful websites that may lead to phishing attacks, malware infections, or data breaches. The red warning symbol visually emphasizes the danger and prompts immediate user attention.

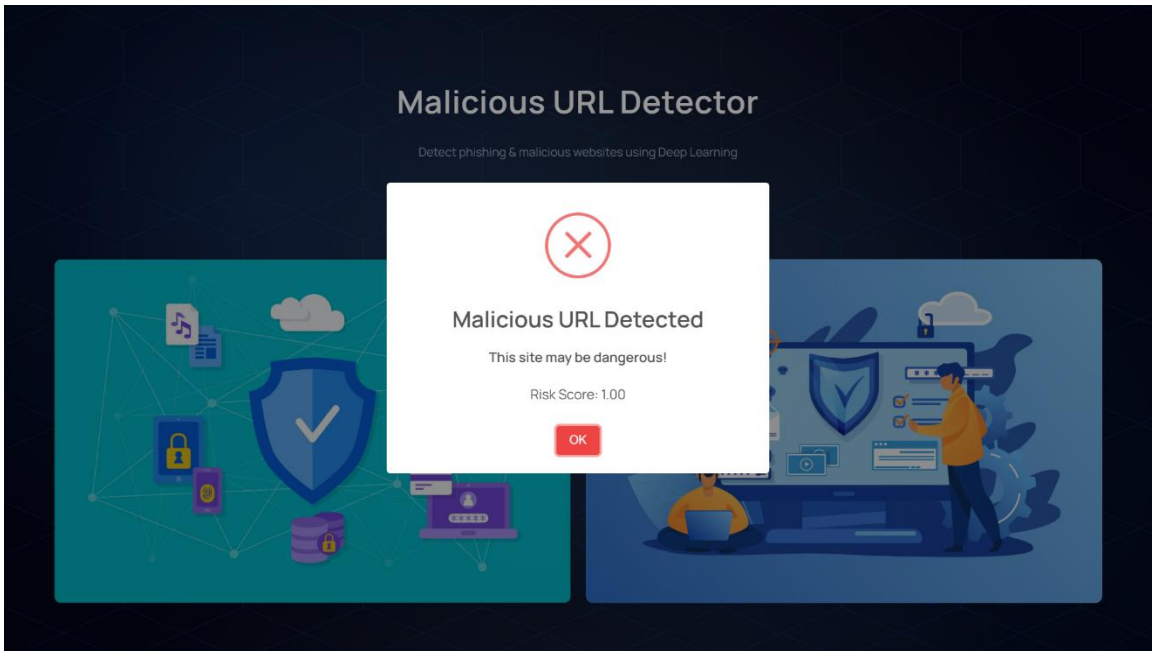

Fig 8.3 Malicious URL Detection Output

Furthermore, the system utilizes predefined thresholds and learned feature patterns to determine the likelihood of a website being malicious. The risk score presented in the output reflects the model's prediction probability, allowing users to assess the severity of the threat. This is particularly useful in cybersecurity applications where early detection is critical to prevent data breaches and unauthorized access. By providing both a clear warning message and a quantified risk level, the system enhances decision-making and helps users take immediate preventive actions, such as avoiding the website or reporting it for further investigation.

The figure 8.4 represents the system response when the user attempts to analyze without entering a URL. The system performs input validation and ensures that necessary data is provided before processing.

If no input is detected, a warning message is displayed prompting the user to enter a valid URL. This validation mechanism prevents unnecessary computation and enhances system reliability by ensuring that only meaningful inputs are processed.

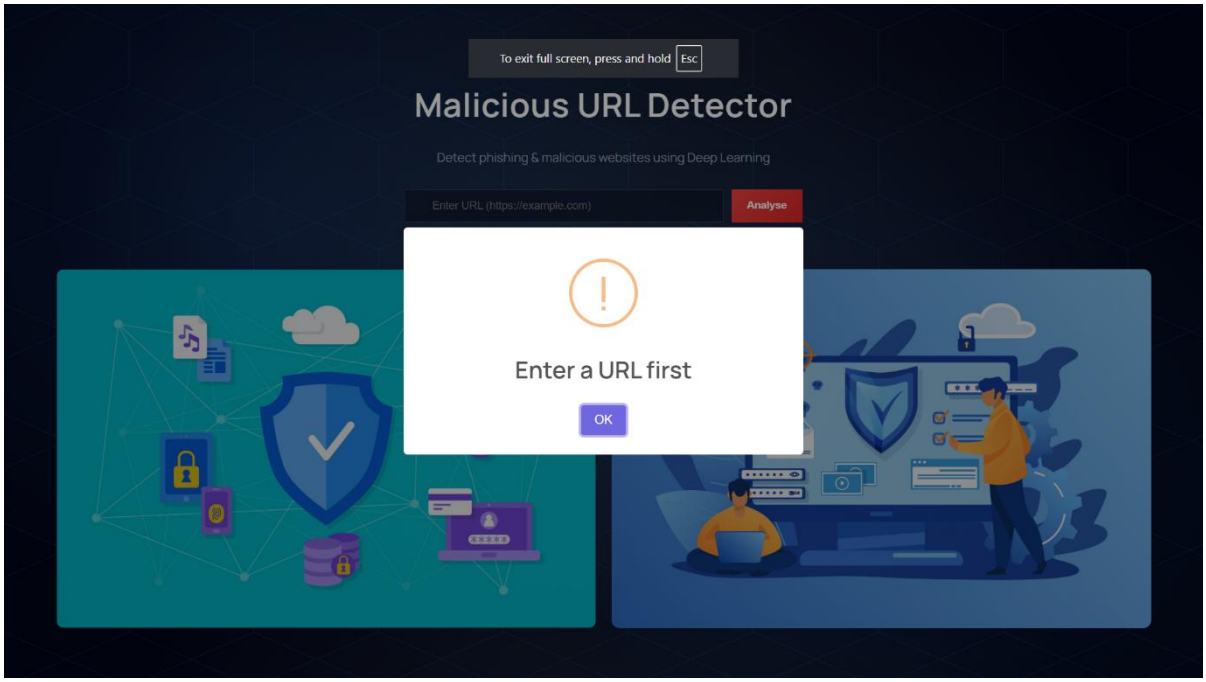

Fig 8.4 Empty Input Warning

Additionally, this input validation mechanism plays an important role in enhancing system robustness and user experience. By detecting empty inputs at an early stage, the system avoids unnecessary backend processing and reduces computational overhead. It also guides the user toward correct usage by providing immediate feedback, ensuring that the analysis process is only triggered with valid input data. This improves the efficiency of the system and minimizes the chances of errors during execution, making the application more reliable and user-friendly.

The figure 8.5 demonstrates the system's response when an invalid URL format is entered. The system checks whether the input follows a valid URL structure before proceeding with analysis.

If the input does not conform to standard URL formatting rules, the system displays an error message indicating that the URL is invalid. This feature ensures data integrity and prevents incorrect or malformed inputs from being processed by the model, thereby improving overall system accuracy and robustness.

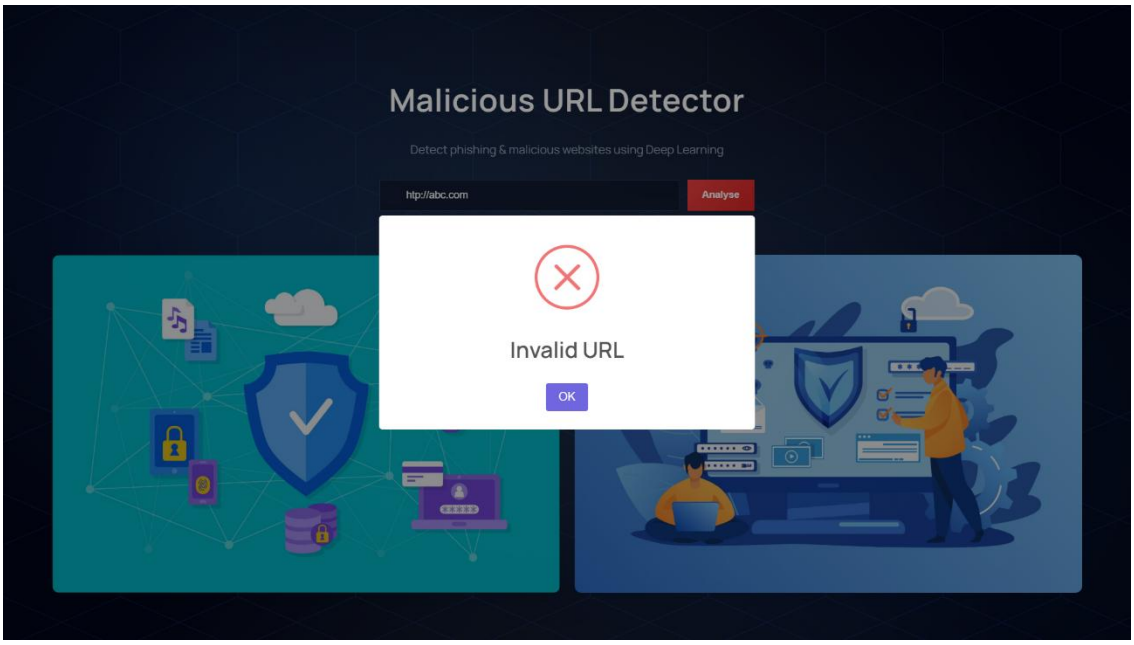

Fig 8.5 Invalid URL Input

Moreover, this validation step is essential for maintaining the reliability and accuracy of the system. By ensuring that only properly formatted URLs are accepted, the system prevents incorrect data from being processed by the model, which could otherwise lead to inaccurate predictions. It also reduces the risk of runtime errors in the backend processing pipeline. This proactive error-handling approach improves the overall stability of the application and ensures that the model operates only on valid and meaningful inputs, thereby enhancing the effectiveness of the malicious website detection system.

The results of the proposed Multi-Modal Features Representation-Based CNN Model for Malicious Website Detection demonstrate its effectiveness in accurately classifying websites as malicious or benign. The model achieves a high test accuracy of around 95%, outperforming traditional machine learning approaches by effectively learning complex patterns from multi-modal data sources such as URL structures, HTML content, and network-based features. By leveraging CNN-based feature extraction, the system successfully captures hidden relationships and patterns, leading to improved detection performance and robustness against evolving cyber threats.

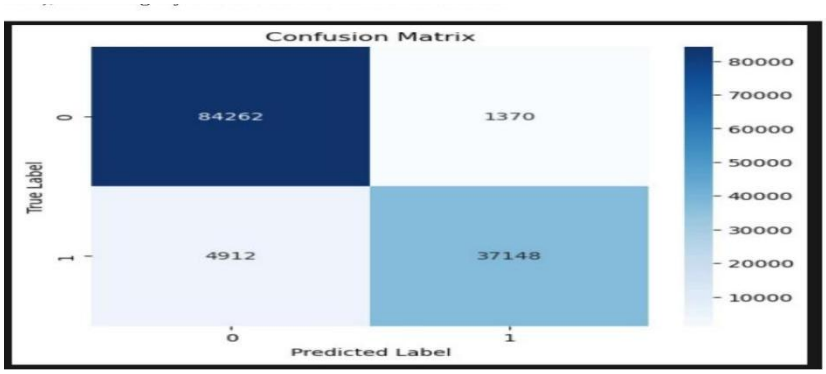

Fig 8.6 Confusion Matrix

Description: In Fig 8.1, The image represents a Confusion Matrix, which is used to evaluate the performance of the malicious website detection model. It compares the actual (true) labels with the predicted labels generated by the model. The matrix is divided into four sections that indicate how well the model is performing in classification.

The matrix shows that the model has a high number of correct predictions (both true positives and true negatives), indicating strong performance. However, the presence of false negatives suggests that a small number of malicious websites are still misclassified as safe, which is an important area for improvement.

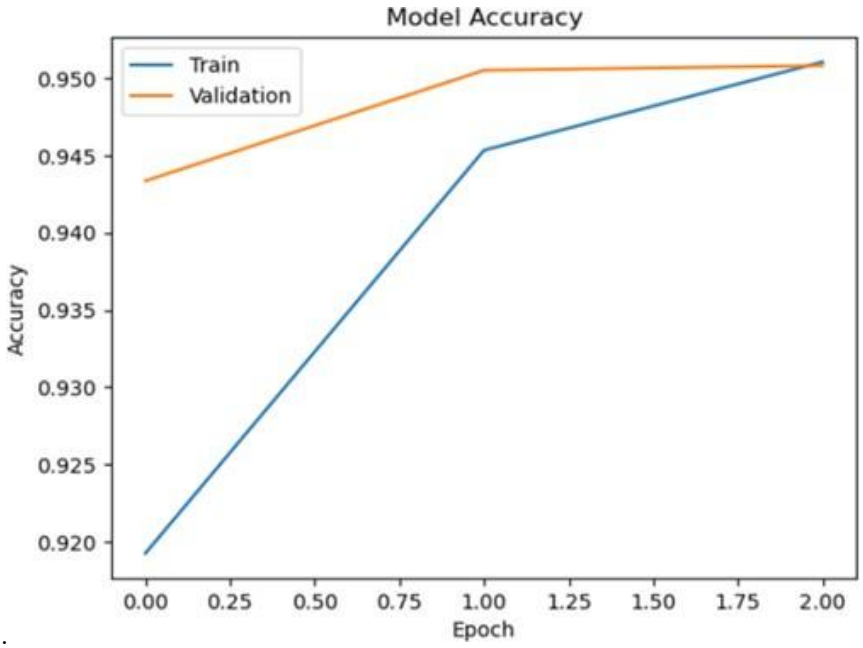

Fig 8.2 Model Accuracy

Description: In Fig 8.2, titled "Model Accuracy" represents the performance of the CNN model during the training process over multiple epochs. The graph shows two lines: one for training accuracy and another for validation accuracy.

From the graph, it is observed that the training accuracy starts at around 0.919 in the first epoch and steadily increases to approximately 0.951 by the final epoch. This indicates that the model is learning effectively from the training data over time. Similarly, the validation accuracy begins at around 0.943 and improves to about 0.951, closely matching the training accuracy.

The close alignment between training and validation accuracy curves suggests that the model is not overfitting and generalizes well to unseen data. The gradual improvement in both curves indicates stable learning and proper optimization of the model.

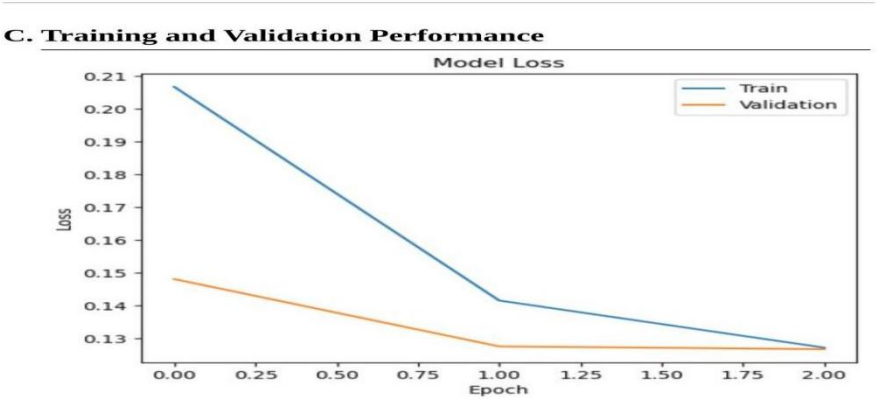

Fig 8.3 Training and Validation Performance

Description: In Fig 8.3, a line chart titled "Prediction Trend" illustrates how predictions vary across different inputs or batches. The x-axis represents different input instances or time steps, while the y-axis shows prediction counts or classification outputs.

The graph shows fluctuations between benign and malicious predictions, with peaks indicating higher detection rates at certain intervals. This trend highlights the model's ability to adapt to varying input patterns and maintain consistent performance across different datasets.

Overall, the line chart highlights a clear peak in car detections and demonstrates how prediction counts fluctuate across different categories, providing an easy way to observe trends and variations in the model's output.

Parameter	Result / Description
Model Used	CNN (Multi-Modal Features Representation-Based Model)
Training Epochs	50-100
Optimizer	Adam
Learning Rate	0.001
Input Data	URL features, HTML content, lexical features, network-based attributes.
Output	Website classification (Malicious / Benign)
Accuracy	95%
Loss	Low (approx.0.2 - 0.3)
Prediction Type	Real-time classification
System Performance	High accuracy with stable learning

Table 8.0 -Summary of Results

The proposed Multi-Modal Features Representation-Based CNN Model for Malicious Website Detection achieves high accuracy of approximately 95% with a low loss value, indicating effective learning and good model convergence. By combining multiple feature sources such as URL patterns, HTML structure, and network attributes, the model successfully captures complex patterns associated with malicious websites.

CHAPTER-9

CONCLUSION

9. CONCLUSION

This project demonstrates the effectiveness of a Multi-Modal Features Representation-Based Convolutional Neural Network (CNN) in addressing the critical problem of malicious website detection. Traditional machine learning approaches often struggle to identify complex and evolving web-based threats due to their reliance on limited or single-source features. By integrating multiple data sources such as URL characteristics, HTML content, and network-based attributes, the proposed system leverages a rich and diverse feature space.

Through systematic data preprocessing, feature extraction, and model training, the system effectively classifies websites as malicious or benign. The CNN model automatically learns hierarchical feature representations, eliminating the need for manual feature engineering and improving detection capability. The integration of multi-modal features enhances the model's robustness and generalization, allowing it to detect sophisticated and previously unseen threats. The achieved performance demonstrates that deep learning-based approaches are highly effective for cybersecurity applications, providing reliable and accurate detection results.

Beyond detection accuracy, the system offers significant practical benefits in real-world cybersecurity scenarios. It enables real-time analysis of websites, helping users and organizations identify and avoid malicious content before any damage occurs. The system can be integrated into web browsers, firewalls, and security tools to provide automated protection.

The project also opens opportunities for future enhancements. The system can be extended by incorporating advanced deep learning models such as hybrid CNN-RNN architectures or transformer-based models to capture sequential patterns in web content. Integration of real-time threat intelligence and continuous learning mechanisms can further improve detection accuracy against evolving cyber threats. Moreover, expanding the dataset and including more diverse attack patterns will enhance the robustness of the system.

Moreover, integrating **adaptive real-time learning** would allow the model to continuously update its knowledge from incoming traffic, enhancing performance in dynamic environments. Edge deployment could further enable low-latency predictions, supporting real-time applications such as autonomous vehicles and IoT devices.

In conclusion, this work establishes the Multi-Modal CNN approach as a powerful and scalable solution for malicious website detection. By combining diverse feature representations with deep learning techniques, the system not only achieves high accuracy but also provides a practical and efficient tool for enhancing web security. The methodology and results of this project lay a strong foundation for future advancements in intelligent and automated cybersecurity systems.

CHAPTER-10

FUTURE ENHANCEMENTS

10. FUTURE ENHANCEMENTS

Future enhancements for the Malicious Website Detection System include extending the model to handle more advanced and evolving cyber threats by incorporating additional data sources such as JavaScript behavior analysis, DNS information, and real-time network traffic monitoring. By integrating these features, the system can improve its ability to detect sophisticated attacks such as phishing, malware injection, and zero-day exploits. This will enable more accurate and proactive detection across diverse and dynamic web environments.

Deploying the system on cloud and edge environments can further enhance its scalability and real-time performance. Integration with web browsers, firewalls, and intrusion detection systems can allow instant detection and blocking of malicious websites.

The use of advanced deep learning architectures such as hybrid CNN-RNN models, transformer-based models, and attention mechanisms can significantly improve detection accuracy. These models can capture both spatial and sequential patterns in web data, enabling the system to analyze dynamic web content more effectively. Incorporating adaptive learning techniques will allow the model to continuously update itself based on new threats, ensuring robustness against evolving cyberattacks.

Integrating multi-modal data sources such as user behavior patterns, browsing history (with privacy considerations), and external threat intelligence feeds can further enhance prediction accuracy.

Improving model efficiency is another important enhancement area. Techniques such as model compression, pruning, and quantization can reduce computational complexity and memory usage, making the system suitable for deployment on resource-constrained devices. Automated machine learning (AutoML) techniques can be used to optimize model architecture and hyperparameters, improving performance without extensive manual tuning.

The system can also be extended to support real-time anomaly detection along with classification. By identifying unusual patterns in web data, the system can detect new or unknown threats that are not present in the training dataset. This capability will enhance the system's ability to respond to emerging cyber threats and improve overall security.

Future developments may explore the concept of a self-adaptive malicious website detection system, where the model is integrated with automated response mechanisms. This would allow the system to not only detect malicious websites but also take real-time actions such as blocking access, generating alerts, or updating security rules dynamically.

The system can also be enhanced to support energy-efficient and resource-aware computing. By optimizing model execution based on workload and usage patterns, the system can reduce computational overhead and energy consumption, especially in large-scale deployments such as cloud servers and enterprise environments. Techniques like dynamic resource allocation and lightweight model inference can ensure that the system remains efficient without compromising detection accuracy, aligning with sustainable and green computing practices.

Furthermore, integration with Internet of Things (IoT) ecosystems and smart applications presents a significant opportunity for expansion. By combining malicious website detection with real-time data from connected devices, smart homes, and smart city infrastructures, the system can provide enhanced security across multiple platforms. This would help protect IoT devices from web-based threats, ensure secure communication, and enable intelligent decision-making in environments such as autonomous systems, healthcare, and smart utilities. Additionally, incorporating real-time anomaly detection alongside classification can significantly strengthen the system's security capabilities. The model can be enhanced to identify unusual patterns in web traffic, user behavior, or content structure that may indicate new or unknown threats such as phishing attacks, malware distribution.

Finally, combining multi-modal learning with temporal and contextual analysis can push the system toward next-level intelligence. By integrating time-based patterns, user behavior, environmental context, and external threat intelligence, the model can achieve more accurate, proactive, and context-aware predictions. This unified approach will enable the development of a highly adaptive and scalable cybersecurity system capable of real-time detection and response in dynamic web environments, positioning it as a powerful solution for future intelligent and automated security systems.

REFERENCES

REFERENCES

[1] M. A. Saeed et al., "Phishing URL Detection Using Deep Learning: A CNN-Based Approach," Journal of Science and Technology, vol. 30, no. 9, 2025. doi: https://doi.org/10.20428/jst.v30i9.3072

[2] Q. E. Haq et al., "Detecting Phishing URLs Based on a Deep Learning CNN Model," Applied Sciences, vol. 14, no. 22, 2024.

[3] N. Q. Do et al., "Detection of Malicious URLs Using Temporal Convolutional Networks," Applied Soft Computing, 2025.

[4] H. Ghalechyan et al., "Phishing URL Detection with Neural Networks," Scientific Reports, 2024.

[5] S. M. Alasmari et al., "CNN-LSTM Framework for Phishing Detection," Springer, 2025.

[6] K. Barik et al., "EGSO-CNN Model for Phishing Detection," Springer, 2025.

[7] Z. Alshingiti et al., "Deep Learning-Based Phishing Detection Using CNN and LSTM," Electronics, 2023.

[8] A. Rawla et al., "Deep Learning Model for Phishing Detection," Procedia Computer Science, 2025.

[9] A. I. T. Driss and H. Zougagh, "Deep Learning for Online Security Improvement," 2024.

[10] Kavya Shanmugam et al., "Hybrid CNN-LSTM Model for Phishing Detection," 2024.

[11] R. Dubey et al., "Ensemble CNN-Based Phishing Detection System," arXiv, 2025.

[12] Ye Tian et al., "URL2Graph++: Deep Learning for Malicious URL Detection," arXiv, 2025.

[13] Vishal Borate et al., "Survey of ML Techniques for Phishing Detection," 2024.

[14] Md Kamrul Hasan Chy, "Machine Learning for Phishing Detection," 2024.

[15] Taofeek Olayinka Agboola, "Novel ML-Based Phishing Detection Approach," 2024.

[16] Moutasmtamimi, "Malicious URL Detection Dataset (Enhanced 2026)," Kaggle, 2026.

https://www.kaggle.com/datasets/moutasmtamimi/malicious-url-detection-dataset-enhanced-2026

[17]TensorFlow/KerasDocumentation: https://www.tensorflow.org/api_docs/python/tf/keras

[18] FastAPI Documentation: https://fastapi.tiangolo.com/

[19] SweetAlert2 Documentation: https://sweetalert2.github.io/

[20] "Deep Learning-Based Phishing URL Detection Model," 2025.

[21] Dr. C. N. Ravi, Deep Learning Model for Cloud Enabled Cyber Threat Detection System. 2023

[22] Mrutyunjaya S. Yalawar, K. Vijaya Babu, Bairy Mahender, A Brain-Inspired Cognitive Control Framework for Artificial Intelligence Dynamic. 2023