A

Industry Oriented Mini Project Report

On

PHISHING EMAIL DETECTION USING IMPROVED RCNN

(Submitted in partial fulfilment of the requirements for the award of Degree)

Bachelor of Technology

in

COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

By

BALASANI SIDHARTHA (217R1A6713)

Under the guidance of

P.N. SANTHOSH KUMAR

Associate Professor

Department of Computer Science & Engineering (Data Science)

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

Accredited by NBA & NAAC with 'A' Grade

Approved by AICTE, New Delhi and JNTU, Hyderabad.

2024-2025

Department of Computer Science & Engineering (Data Science)

CERTIFICATE

This is to certify that the project entitled "**Phishing Email Detection Using Improved RCNN"** being submitted by **BALASANI SIDHARTHA (217R1A6713)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (Data Science) to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2024-25.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

P.N.Santhosh Kumar
Associate Professor
Internal Guide

Dr. K.Murali
Associate Professor & HoD

External Examiner

Submitted for viva-voce Examination held on: _____

ACKNOWLEDGMENT

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I take this opportunity to express my profound gratitude and deep regard to my guide **"P.N. Santhosh Kumar"**, Designation for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry me a long way in the journey of life on which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Mr. Sanjib Kumar Nayak, Mrs. V. Prema Tulasi, Mr. B. Ramesh, Mr. A. Veerender, Mrs. V. Sandya, Ms. Sandhyarani** for their cordial support, valuable information and guidance, which helped me in completing this task through various stages.

I am also thankful to **Dr. K. Murali**, Head, Department of Computer Science and Engineering (Data Science) for providing encouragement and support for completing this project successfully.

I am obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. I also express our sincere gratitude to **Sri. Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. I am grateful for their constant support and help.

Finally, I would like to take this opportunity to thank my family for their constant encouragement, without which this assignment would not be completed. I sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

BALASANI SIDHARTHA (217R1A6713)

ABSTRACT

Phishing emails remain a major cybersecurity threat, deceiving individuals and organizations by pretending to be legitimate sources to steal information. Traditional detection methods, like rule-based filters and blacklists, are limited in keeping up with evolving phishing tactics. These systems often miss advanced phishing attempts and produce many false positives, which disrupt email communications and lower user confidence. To overcome these challenges, this project proposes a phishing detection system based on an enhanced Recurrent Convolutional Neural Network (RCNN) model. The RCNN model combines the capabilities of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to recognize spatial and sequential patterns in phishing emails. CNNs excel at spotting structural features, such as links and attachments, while RNNs analyse the sequence of text to detect deceptive language patterns. Merging these techniques allows the RCNN model to catch sophisticated phishing attempts that other methods might miss. This project trained the RCNN model on a dataset consisting of phishing and legitimate emails. The results of the model reduces false positives significantly and improves detection accuracy. The hybrid system is especially effective at identifying complex phishing attacks that use advanced social engineering and evade standard detection methods. Additionally, the system's real-time detection capability makes it well-suited for high-traffic email environments where quick response to phishing threats is essential. The project also examines the limitations of current phishing detection systems, which often struggle with adaptability and scalability. The RCNN model addresses these gaps, offering a more reliable approach to phishing detection. In conclusion, the RCNN model provides a robust and promising solution to modern phishing threats by enhancing detection accuracy and protecting users from increasingly sophisticated cyberattacks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SCREENSHOTS

LIST OF ABBREVIATIONS

RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
RCNN	Recurrent Convolutional Neural Networks
GNN	Graph Neural Networks
LSTM	Long Short-Term Memory
URL	Uniform Resource Locator
ORM	Object-Relational Mapping

Chapter 1
INTRODUCTION

1.1 Project Scope

The project aims to develop an advanced phishing email detection system utilizing a sophisticated approach combining deep learning techniques, specifically an improved Recurrent Convolutional Neural Network (RCNN) model. The system will analyse multiple components of an email, including the header, body, character level, and word level, to identify phishing threats with high accuracy. Given the growing threat of phishing emails and their increasing sophistication, this project seeks to provide a solution that goes beyond traditional methods like blacklists or basic machine learning models. By incorporating an attention mechanism into the email analysis, the model will be able to prioritize the most relevant parts of an email that indicate phishing activity. The system will also be tested on an unbalanced dataset that mirrors real-world conditions, containing both legitimate and phishing emails in realistic proportions. The final solution is expected to significantly outperform existing models in identifying phishing emails while minimizing false positives, ensuring the safety of users from phishing attacks, data breaches, and financial fraud. The integration of machine learning, data mining, and NLP (Natural Language Processing) techniques in this project will contribute to the cybersecurity domain by providing a robust, effective, and highly efficient tool to combat phishing threats. This system is aimed at improving the accuracy of phishing detection by capturing both the structural elements, such as links and email headers, and the sequential nature of the email content.

1.2 Project Purpose

The primary purpose of this project is to develop a robust and efficient phishing email detection system that addresses the escalating threat of phishing attacks in the digital age. Phishing, which involves the fraudulent practice of sending deceptive emails to trick users into divulging sensitive information such as passwords, credit card details, and personal identification, has caused immense financial and data losses worldwide. Despite the introduction of various anti-phishing measures, existing methods often fall short due to their reliance on static approaches like blacklists or their inability to adapt to the evolving tactics of attackers. This project seeks to bridge that gap by utilizing an advanced deep learning model that leverages improved Convolutional Neural Networks (CNN) and multilevel attention mechanisms to analyse the intricate components of an

email--such as the header, body, word level, and character level—in a more detailed and dynamic manner. By doing so, the system aims to identify phishing attempts with greater precision, even in scenarios where phishing emails closely mimic legitimate communications. Furthermore, the project intends to minimize false positives, ensuring that genuine emails are not erroneously classified as phishing attempts, which can enhance user trust and overall email system performance.

1.3 Project Features

The central feature of the project is the implementation of an Improved Recurrent Convolutional Neural Network (RCNN). This model integrates the strengths of the Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs are used to detect structural elements such as suspicious links, attachments, or email headers, while RNNs analyse the sequential flow of text, identifying deceptive language or patterns common in phishing emails. The system aims to provide superior detection accuracy by capturing both spatial and temporal patterns in emails. Another feature is its ability to detect phishing emails in real time. As emails are received, the system immediately analyses and flags potentially malicious emails, providing instant protection for users and preventing phishing attacks before any damage can be done. This feature is critical for environments with high email traffic where timely detection is essential. The model is designed to continuously learn from new phishing examples, making it adaptable to evolving phishing tactics. This approach improves user trust in the system while ensuring a high level of security without interrupting normal email communication. Phishing emails often use obfuscation techniques like misleading URLs, deceptive links, or well-crafted language to trick users. The RCNN's ability to analyse both the structure and sequential flow of emails makes it resilient to these tactics, enabling it to detect phishing attempts that might bypass simpler detection systems.

Chapter 2

SYSTEM ANALYSIS

2.1 Problem Definition

The project enhances phishing email detection using an Improved Recurrent Convolutional Neural Network (RCNN) model. Phishing emails trick users into revealing sensitive data, and traditional systems, like rule-based filters and blacklists, often fail to detect evolving phishing techniques. The proposed RCNN model combines Convolutional Neural Networks (CNNs), which detect structural elements such as links and suspicious headers, with Recurrent Neural Networks (RNNs) that analyse the sequential patterns in email content. This hybrid approach captures both spatial and temporal features, enabling more accurate identification of phishing emails. The model is trained on large datasets, making it adaptable to new phishing tactics while significantly reducing false positives. The system also supports real-time detection, offering a scalable and robust solution for modern email security, which is critical in today's cybersecurity landscape.

2.2 Existing System

The existing systems for phishing email detection primarily rely on traditional techniques such as rule-based filters, blacklists, and signature-based approaches. Rule-based systems detect phishing emails by identifying suspicious keywords, malformed URLs, or unusual email structures, while blacklist-based methods compare incoming emails to a known database of phishing sources Rule-based systems detect phishing emails by identifying suspicious keywords, malformed URLs, or unusual email structures, while blacklist-based methods compare incoming emails to a known database of phishing sources. While these techniques offer some protection, they struggle with evolving phishing tactics that bypass static rules and blacklists. Additionally, signature-based systems, which rely on recognizing previously identified phishing patterns, are unable to detect new or sophisticated phishing attempts. More recent advancements have incorporated machine learning models, such as Support Vector Machines (SVM), Decision Trees, and Random Forests, which help improve detection accuracy by learning from past data. However, these models are often limited in handling complex, real-time phishing attacks and ca result in high false-positive rates. Overall, the existing systems face challenges in adaptability, scalability, and real-time detection, leaving room for further improvement in addressing phishing threats.

2.2.1 Limitations of the Existing System

The existing phishing email detection systems face several disadvantages:

1. **Inability to Detect Evolving Phishing Techniques**: Rule-based and blacklist systems are static and rely on predefined patterns or known phishing sources. As phishing tactics constantly evolve, these systems fail to detect new or sophisticated phishing attempts that deviate from known patterns.
2. **High False-Positive Rates**: Many Machine Learning-based systems, especially those using simpler models like SVM or Decision Trees, often flag legitimate emails as phishing (false positives). This reduces user trust in the detection system and disrupts normal email communication.
3. **Limited Scalability**: Traditional detection systems struggle to handle the massive volume of emails in real-time, especially in large organizations. The need for constant updates to blacklists or rule sets further strains their scalability and responsiveness.
4. **Inadequate Handling of Complex Data**: Many existing models are not capable of understanding complex structures, links, or contextual relationships within an email. Phishing emails often use advanced obfuscation techniques, such as misleading URLs or visually deceptive content, which simple models fail to interpret effectively.

2.3 Proposed System

The proposed system introduces an enhanced phishing email detection approach using an **Improved Recurrent Convolutional Neural Network (RCNN)** model, designed to overcome the limitations of existing systems. By combining the strengths of both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), the RCNN captures both the structural and sequential features of phishing emails. CNN layers analyse spatial patterns, such as embedded links and suspicious email headers, while RNN layers capture the sequential flow of text, identifying deceptive language patterns commonly used in phishing attempts. This hybrid architecture allows the model to detect more sophisticated and evolving phishing attacks that bypass traditional rule-based or blacklist methods. Furthermore, the proposed system is capable of learning from large datasets, adapting to new phishing techniques, and improving

detection accuracy over time. It also significantly reduces false positives and negatives, ensuring that legitimate emails are not incorrectly flagged. By incorporating real-time detection and scalability, the proposed system offers a robust solution for modern email security, effectively addressing the shortcomings of current phishing detection approaches.

2.3.1 Advantages of Proposed System

1. **Enhanced Detection Accuracy**: By combining the strengths of CNN and RNN, the proposed system captures both structural (e.g., links, images) and sequential (e.g., email content flow) features, leading to a more comprehensive analysis of phishing emails and improved detection accuracy.
2. **Adaptability to Evolving Threats**: Unlike traditional systems that rely on static rules or blacklists, the RCNN model learns from large datasets and continually adapts to new phishing tactics, making it more effective at detecting emerging and sophisticated phishing attacks.
3. **Reduced False Positives and Negatives**: The hybrid RCNN architecture minimizes false positives (flagging legitimate emails as phishing) and false negatives (missing phishing emails), resulting in more reliable and accurate detection compared to existing models.
4. **Real-Time Detection**: The proposed system is designed for real-time phishing detection, allowing it to identify and flag malicious emails as they arrive, ensuring that potential threats are addressed immediately.
5. **Contextual Understanding**: By analysing both the structure and content of emails, the RCNN is capable of understanding deceptive language patterns and social engineering tactics used in phishing emails, improving its ability to detect attacks based on email semantics.

2.4 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Feasibility studies employ various tools and techniques, such as cost-benefit analysis, break-even analysis, return on investment analysis, decision trees and sensitivity analysis. Common mistakes to avoid include insufficient data collection, poor risk assessment, inadequate stakeholder engagement, lack of contingency planning, and unrealistic assumptions.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.4.1 Economical Feasibility

Economic feasibility involves assessing whether the financial benefits of a project surpass its costs, ensuring that the investment is worthwhile. In this project, it involves evaluating the necessary resources such as software, hardware, and human effort to determine if the expenses are justified by the anticipated advantages. The project makes use of cost-efficient technologies like open-source platforms, particularly Python and Django, which significantly lower software expenses. These free resources enable the development of a sophisticated phishing detection system without the need for costly proprietary tools. Additionally, the project can be executed using existing hardware, such as standard desktop computers with moderate processing capabilities, ensuring minimal additional hardware investments

2.4.2 Technical Feasibility

This study manages to see the technical possibilities, i.e. the technical requirements of the system. There should not be a huge need for the technical resources available for any system created. This could lead to greater demands on available technology resources. This may put more demands on the client. There should be a normal need

for the developed system as it requires little or no modification to implement this technique. This is done to identify the technical feasibility of a given product, not the high proportion of the technical requirements of the system. The equipment used has the technical ability to hold data and can be used by the new system. Many users can use it for just about any particular purpose, and data protection is available because it relates to every email user.

2.4.3 Social Feasibility

The point of the study is to measure the acceptability of the system by the user. This includes teaching the user to use the system efficiently. The user should not feel vulnerable to the system and instead accept it as a requirement. This review is done based on the customer's acceptance of the system, which will provide an idea of the system and its use. The customer should not be harmed in the use of this program. The client should be able to design with the techniques used to create the project and be able to articulate his ideas to broaden the need, not to oppose it. The use of the program should feel acceptable without any confusion.

2.5 Hardware & Software Requirements

2.5.1 Hardware Requirements

Processor : Pentium IV or higher

RAM : 4 GB

Hard Disk : 20 GB

2.5.2 Software Requirements

Operating System : Windows 7 Ultimate

Coding Language : Python

Framework : Django

Database : MySQL

Application : WampServer

Debugger : Any Browser

2.6 Python

Python is currently the most widely used multi-purpose, high-level programming language. It allows programming in Object-Oriented and Procedural paradigms. Also Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. We don't need to use data types to declare variables because they are dynamically typed, so we can write a = 10 to assign an integer value to an integer. Debugging is faster in Python development because no package steps are included in Python development, and the revision-test-debug cycle is much faster.

2.6.1 Python Features

1. Easy to Learn and Use
2. Expressive Language
3. Cross-Platform Language
4. Free and Open Source
5. Object-Oriented Language
6. Extensible
7. GUI Programming support
8. Integrated

2.6.2 Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development. Here, we are specifying applications areas where python can be applied.

1. Web Applications
2. Desktop GUI Applications
3. Software Development
4. Applications for Images

2.7 DJANGO

Django is a free, open-source, high-level Python web framework that enables rapid of secure, maintainable websites. Designed for flexibility and scalability, Django streamlines the development process with its modular design, Object-Relational Mapping (ORM) system, and extensive libraries. Ideal for complex data-driven sites, e-commerce platforms, and social media applications, Django powers notable projects like Instagram, Pinterest, and Dropbox. With a large community and extensive documentation, Django offers fast development, scalability, and security. Its modular architecture allows developers to build reusable components, while its authentication and authorization system ensures robust user management. Whether building a simple web application or a complex enterprise solution, Django provides a robust foundation for efficient and effective web development.

2.7.1 Features of DJANGO

1. Rapid Development
2. Secure
3. Scalable
4. Fully loaded
5. Versatile
6. Open Source
7. Vast and Supported Community

Chapter 3

ARCHITECTURE

3.1 Project Architecture

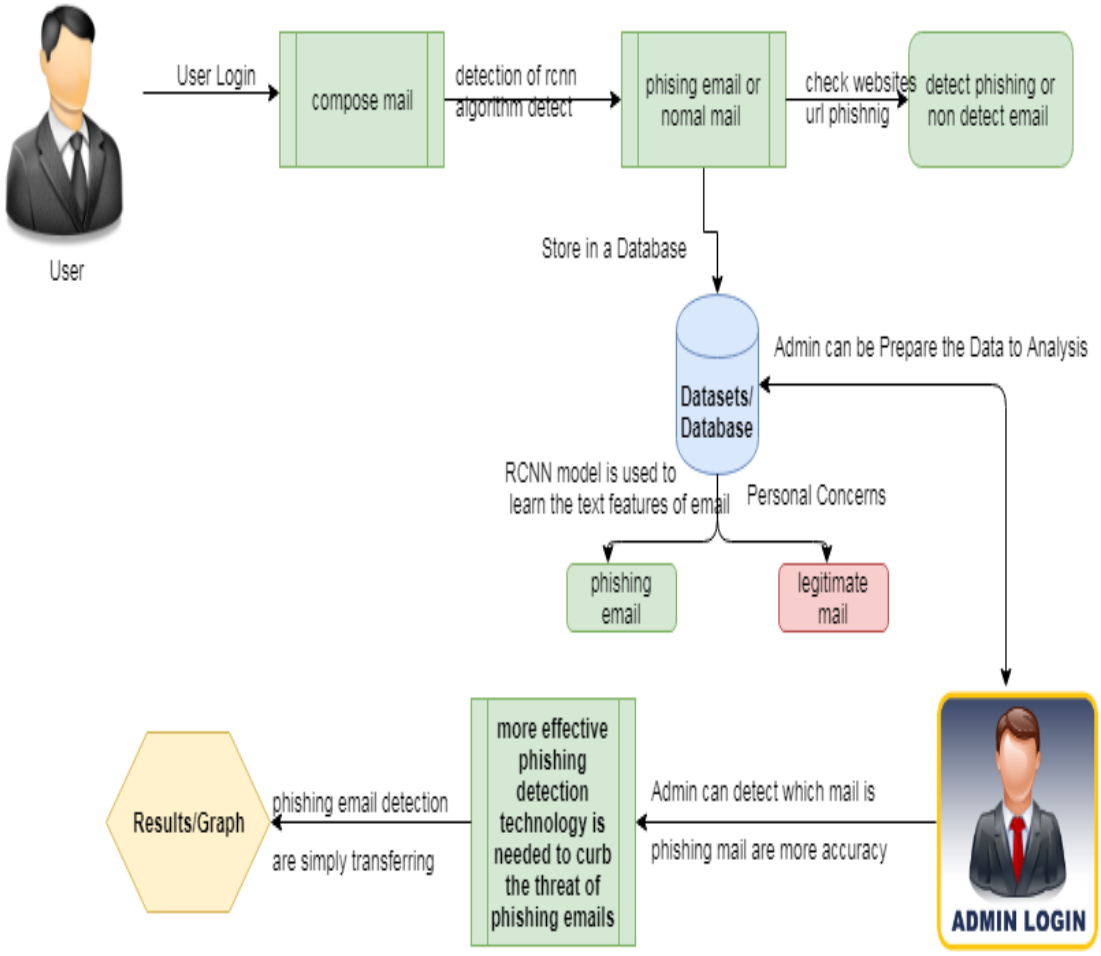


Fig 3.1 Project Architecture

The architecture of this phishing email detection system is structured to efficiently analyse and classify emails by integrating multiple components. It uses an advanced deep learning model, specifically an improved Recurrent Convolutional Neural Network (RCNN), to process emails at various levels. An attention mechanism is included in the model, enabling it to focus on the most relevant parts of the email content for accurately detecting phishing threats. The architecture involves several stages of data processing: initially, the system extracts important features from the email data, and then the RCNN model is applied for classification. The backend, built with Python and Django, handles communication between the user interface and the admin interface.

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. Use Case Diagrams help identify key system functionalities, prioritize requirements, and clarify system boundaries. They are essential tools in software development, business analysis, and system design, facilitating communication among stakeholders and ensuring the system meets user needs.

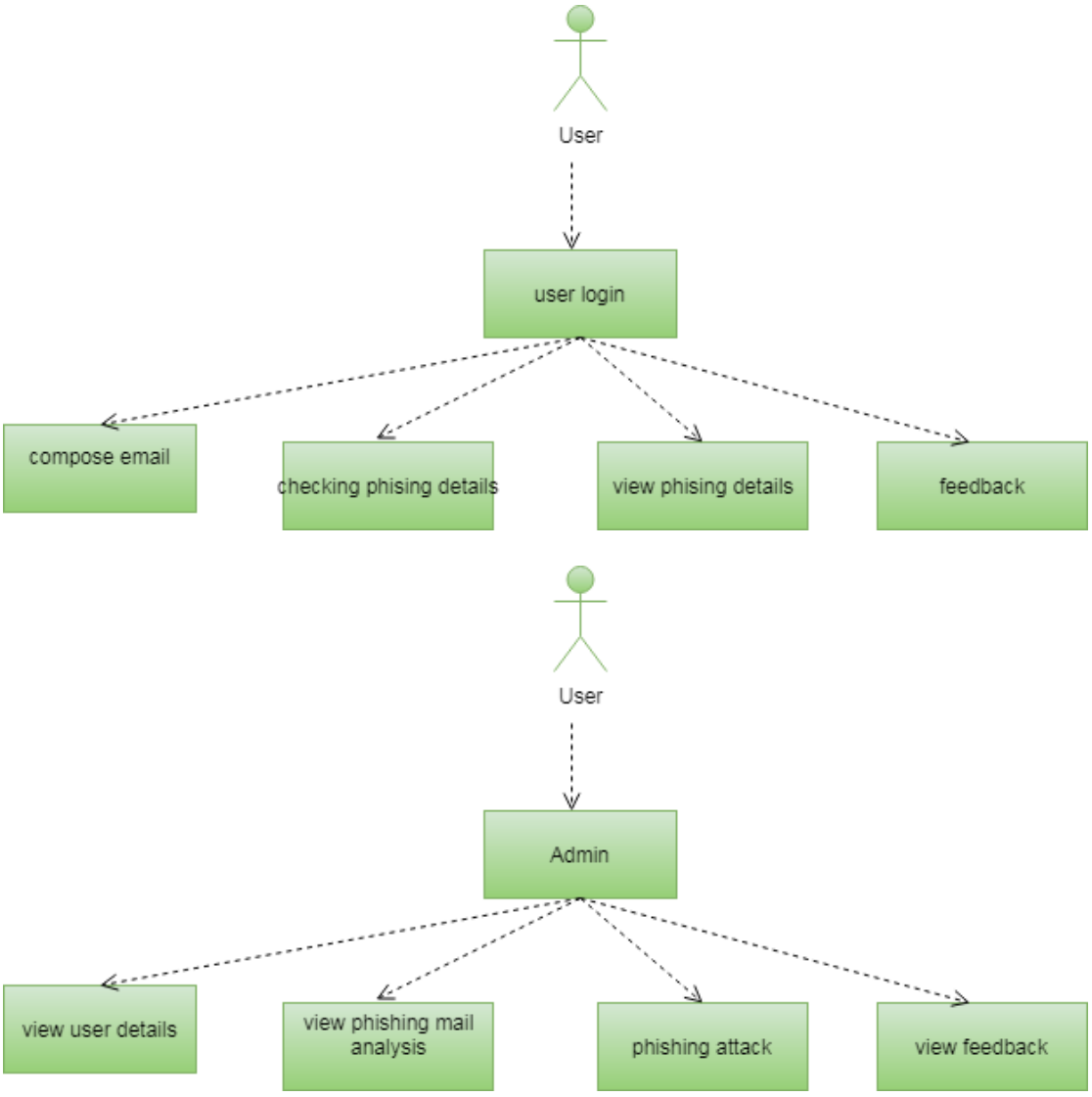


Fig 3.2 Use Case Diagram

3.3 Class Diagram

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. Active class is used in a class diagram to represent the concurrency of the system. Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose.

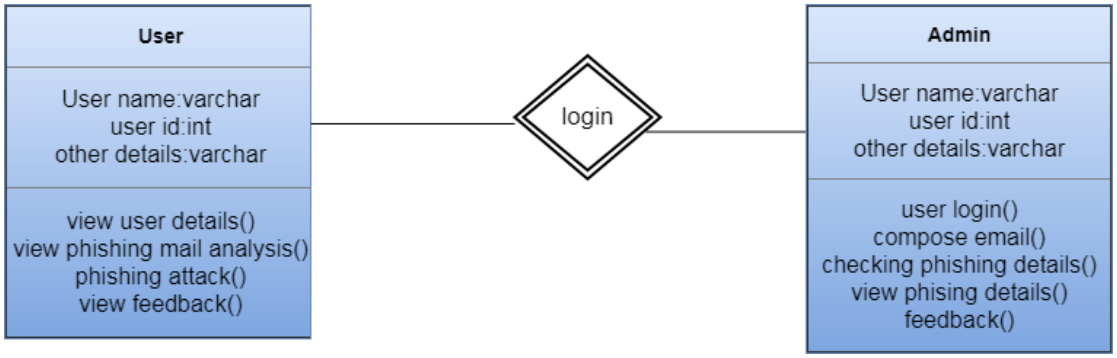


Fig 3.3 Class Diagram

3.4 Sequence Diagram

A Sequence Diagram is a type of interaction diagram in UML (Unified Modelling Language) that shows how objects interact with each other through a time sequence. It details the flow of messages or events between different objects, illustrating how operations are carried out step by step. Sequence diagrams emphasize the order of messages exchanged between objects, typically aligning vertically to show the progression of time. Each object has a lifeline represented by a dashed line, and interactions between objects are depicted as arrows. It shows the sequence of messages, requests, and responses exchanged between actors, classes, or objects to achieve a specific goal.

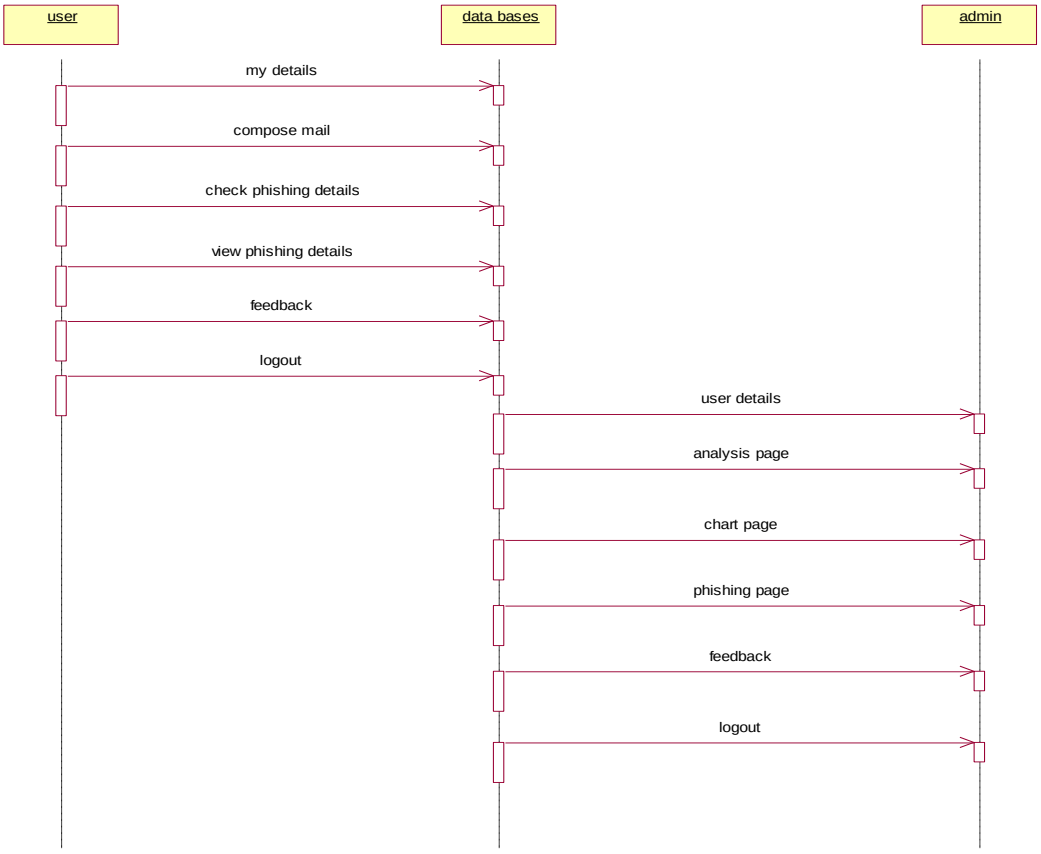


Fig 3.4 Sequence Diagram

3.5 Activity Diagram

An Activity Diagram is a UML (Unified Modeling Language) diagram that represents the flow of control or activities in a system. It is used to model the dynamic behaviour of a system by depicting the flow of activities sequentially or concurrently. Activity diagrams focus on the workflow and describe how tasks or operations progress from one to the next. The diagram includes various elements such as activities (represented by rectangles with rounded corners), decisions (diamond shapes), and transitions (arrows) that show the flow between activities. Activity diagrams can also represent parallel processes using synchronization bars, making them useful for visualizing

complex workflows, including both sequential and concurrent actions.The activity diagram is as follows:

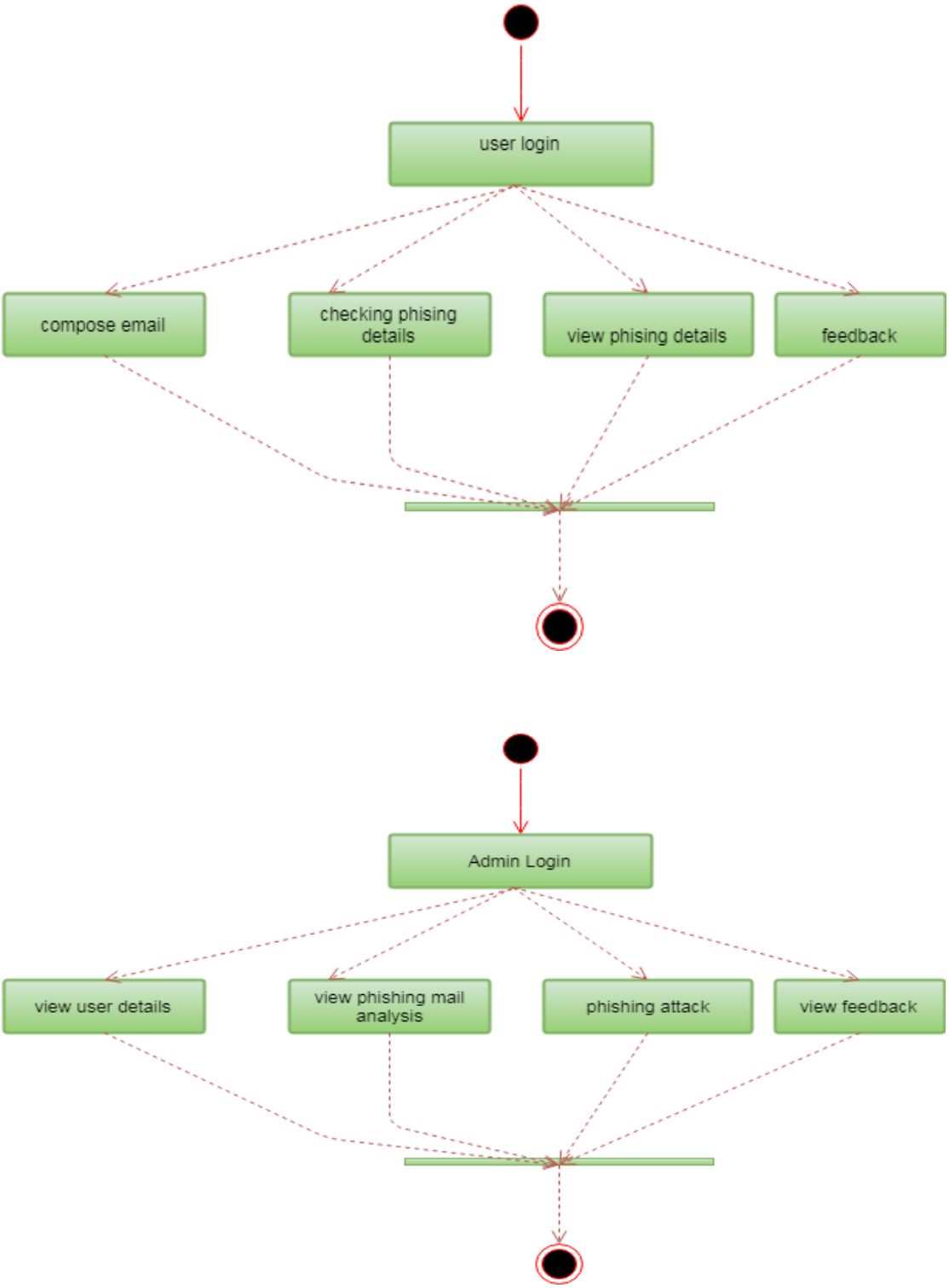


Fig 3.5 Activity Diagram

Chapter 4

IMPLEMENTATION

4.1 Machine Learning Algorithms

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it. The different kinds of Machine Learning are:

1. **Supervised learning** involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into tasks and classification regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities
2. **Unsupervised learning** involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.

4.1.2 Challenges and Applications of Machine Learning

Challenges in Machine Learning

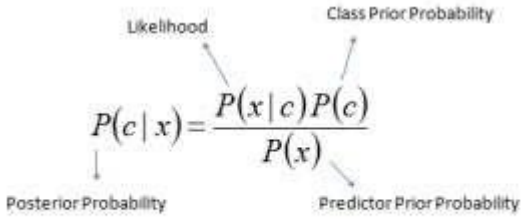
- Quality of Data
- Time Consuming Task
- Issue of overfitting & underfitting
- Difficulty in Deployment

Applications of Machine Learning

- Emotion analysis

- Sentiment analysis
- Stock market and forecast analysis
- Fraud detection

4.1.3 Naive Baye -Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. Naive Bayes is a simple but surprisingly powerful algorithm for predictive modelling. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

4.1.4 Decision Trees- The Decision Tree algorithm in machine learning is a simple yet powerful method used for classification and regression tasks. It works by recursively splitting the dataset into smaller subsets based on the most important features, using metrics like Gini impurity or information gain for classification, and mean squared error for regression. The tree consists of nodes representing features, branches indicating decisions, and leaves showing final predictions.

4.1.5 Logistic Regression- Logistic regression is a statistical method used for binary classification in machine learning, predicting the probability that an input belongs to a specific category (e.g., yes/no) based on one or more predictor variables. It employs the logistic function (sigmoid) to transform linear combinations of inputs into probabilities between 0 and 1. The model's decision boundary is defined by a linear equation, and it minimizes the log loss (binary cross-entropy) during training

4.1.6 Random Forests- Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of their predictions for

classification or the mean for regression tasks. Each tree is built using a random subset of the training data and a random subset of features. This approach improves accuracy and robustness against noise. Random Forest is particularly effective for handling large datasets with high dimensionality and can also provide insights into feature importance, making it useful for both classification and regression problems.

4.1.7 Support Vector Machines (SVM)- Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that separates data points of different classes with the maximum margin. SVM can efficiently perform non-linear classification using kernel functions, which transform the input space into higher dimensions. This flexibility allows SVM to handle complex datasets with intricate class boundaries.

4.1.8 K-Nearest Neighbors- K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm used for classification and regression tasks. It classifies a data point based on the majority class among its K nearest neighbors in the feature space, typically using Euclidean distance to measure proximity. KNN does not involve any explicit training phase, making it easy to implement. However, it can be computationally intensive during prediction, especially with large datasets.

4.2 DataSet

The dataset used in this phishing email detection system is designed to reflect real-world conditions by containing a mix of phishing and legitimate emails. It is unbalanced, meaning that there are more legitimate emails than phishing ones, which is typical in actual email traffic. This design ensures that the model learns to detect phishing attempts without being skewed by an unrealistic dataset composition.

Before the data is fed into the model, it undergoes preprocessing, which involves cleaning the emails and extracting relevant features. This step is crucial for ensuring that the data is formatted correctly and irrelevant information is removed. Stratified random sampling is used to ensure that the dataset maintains a proper balance during training and testing phases, with phishing and legitimate emails being proportionally represented.

The dataset is divided into training and testing sets, with the larger portion used to train the model. The testing set is used to validate the model's ability to correctly classify emails it has not seen before. By using a realistic ratio of phishing and legitimate emails,

the dataset allows the model to adapt to actual email environments and ensures that it performs well in detecting phishing attempts without generating a large number of false positives.

4.3 Deep Learning

Deep learning is a subset of machine learning that employs neural networks with multiple layers to analyse and learn from vast amounts of data. Inspired by the human brain's architecture, deep learning models automatically extract features from raw data, making them particularly effective for complex tasks such as image and speech recognition, natural language processing, and autonomous systems. These models consist of interconnected neurons organized into input, hidden, and output layers, where each neuron processes input data and passes the result through nonlinear activation functions. While deep learning eliminates the need for extensive manual feature engineering and has achieved remarkable advancements in various fields, it requires large labeled datasets and significant computational resources, presenting challenges such as overfitting. Despite these hurdles, deep learning remains a powerful tool driving innovations across diverse industries.

4.3.1 Types of Deep Learning

1. **Convolutional Neural Networks (CNNs) -**It a specialized type of deep learning architecture primarily designed for processing grid-like data, particularly images. They utilize convolutional layers to automatically learn spatial hierarchies of features through local receptive fields, enabling the detection of patterns such as edges, textures, and shapes. A typical CNN consists of several layers, including convolutional layers, pooling layers, and fully connected layers, which work together to progressively extract high-level features from the input data. CNNs are particularly effective in tasks like image classification, object detection, and segmentation, as they reduce the need for manual feature extraction and are robust to variations in the input.
2. **Recurrent Neural Networks (RNNs)-** Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to recognize patterns in sequences of data, such as time series or natural language. Unlike traditional neural networks, RNNs have loops that allow information to persist, making them well-suited for tasks where context and sequence are important. This architecture enables RNNs to take

into account both current inputs and past information, which makes them effective for tasks like language modeling, speech recognition, and machine translation. However, standard RNNs struggle with long-term dependencies due to issues like vanishing gradients, where gradients become too small during training, making it difficult for the network to learn from earlier time steps.

3. **Transformers-** Transformers represent a significant advancement in the field of deep learning, particularly in handling sequential data such as text. Unlike previous architectures like recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), transformers utilize a mechanism called self-attention, which enables them to consider the entire context of a sequence simultaneously rather than processing it step by step. Since their introduction, transformers have revolutionized natural language processing and have been adapted for various applications, leading to the development of powerful models that have set new benchmarks across numerous tasks.
4. **Graph Neural Networks-** These are a specialized class of neural networks designed to work with graph-structured data, where relationships between entities are represented as nodes and edges. Unlike traditional neural networks that operate on fixed-size, grid-like data, GNNs can naturally handle variable-sized graphs, making them ideal for applications in social network analysis, recommendation systems, biological networks, and more. GNNs operate by iteratively aggregating information from a node's neighbors, allowing each node to learn representations that capture both its features and its relationships within the graph.

5. **RCNN-**Recurrent Convolutional Neural Networks (RCNNs) are an advanced neural network architecture that integrates the strengths of both convolutional neural networks (CNNs) and recurrent neural networks (RNNs), making them particularly effective for processing spatial and temporal data. In RCNNs, convolutional layers extract spatial features from input data, such as images or video frames, while recurrent layers capture temporal dependencies by processing sequences over time.

4.4 Sample Code

<?xml version="1.0" encoding="UTF-8"?>

<project version="4">

<component name="ChangeListManager">

<list default="true" id="c7cd87dd-2bf9-455d-a572-1f0e82358d78" name="Default Changelist" comment="" />

<option name="EXCLUDED_CONVERTED_TO_IGNORED" value="true" />

</component>

<component name="FileEditorManager">

<leaf SIDE_TABS_SIZE_LIMIT_KEY="300">

<state relative-caret-position="1258">

<caret line="104" column="61" selection-start-line="104" selection-start-column="61" selection-end-line="104" selection-end-column="61" />

</state>

</provider>

</entry>

</file>

<file pinned="false" current-in-tab="false">

<entry

</folding>

</state>

</provider>

</entry>

</file>

<file pinned="false" current-in-tab="true">

<entry file="file://$PROJECT_DIR$/Users/views.py">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="-1411">

<caret line="163" column="59" selection-start-line="163" selection-start-column="59" selection-end-line="163" selection-end-column="59" />

<folding>

<elementsign="e#0#9#0" expanded="true" />

</folding>

</state>

</provider>

</entry>

</file>

<file pinned="false" current-in-tab="false">

<entry file="file://$PROJECT_DIR$/design/htmlfiles/users/userlogin.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="203">

<caret line="103" column="119" selection-start-line="103" selection-start-column="119" selection-end-line="103" selection-end-column="119" />

<folding>
<elementsign="n#style#0;n#h2#0;n#body#0;n#html#0;n#!!top"expanded="true" />

</folding>

</state>

</provider>

</entry>

</file>

<file pinned="false" current-in-tab="false">

<entryfile="file://$PROJECT_DIR$/design/htmlfiles/users/viewmailpage.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="299">

<caret line="111" column="6" lean-forward="true" selection-start-line="111" selection-start-column="6" selection-end-line="111" selection-end-column="6" />

<folding>
<elementsign="n#style#0;n#a#0;n#div#3;n#body#0;n#!!top"expanded="true" />

<elementsign="n#style#0;n#a#1;n#div#3;n#body#0;n#!!top"expanded="true" />

<elementsign="n#style#0;n#0;n#a#1;n#div#3;n#body#0;n#!!top"expanded="true" />

<elementsign="n#style#0;n#a#2;n#div#3;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#2;n#div#3;n#body#0;n#!!top" expanded="true" />

</folding>

</state>

</provider>

</entry>

</file>

<file pinned="false" current-in-tab="false">

<entry file="file://$PROJECT_DIR$/design/htmlfiles/users/design.html">ne="99" selection-end-column="76" />

<folding>

<elementsign="n#style#0;n#a#0;n#div#3;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#0;n#div#3;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#a#1;n#div#3;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#1;n#div#3;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#a#2;n#div#3;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#2;n#div#3;n#body#0;n#!!top" expanded="true" />

</folding>

</state>

</provider>

</entry>

</file>

</leaf>

</component>

<component name="FileTemplateManagerImpl">

<option name="RECENT_TEMPLATES">

<list>

<option value="HTML File" />

</list>

</option>
</component>
<component name="IdeDocumentHistory">
<option name="CHANGED_PATHS">
<list>
<option value="$PROJECT_DIR$/Phishing_Email_Detection/settings.py" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/userpage1.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/checking_attack.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/viewmailcopy.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/spamcopy.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/userpagecopy.html" />
<option value="$PROJECT_DIR$/Admins/models.py" />
<option value="$PROJECT_DIR$/Users/logics.py" />
<option value="$PROJECT_DIR$/Users/models.py" />
<option value="$PROJECT_DIR$/design/htmlfiles/admins/analysis_page.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/checking.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/feedback.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/mydetails.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/admins/login_page.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/admins/admin_page.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/admins/viewfeedback.html" />
<option value="$PROJECT_DIR$/Phishing_Email_Detection/urls.py" />
<option value="$PROJECT_DIR$/Admins/views.py" />
<option value="$PROJECT_DIR$/design/htmlfiles/admins/chart.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/admins/attactdetails.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/admins/admin_design.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/design.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/userlogin.html" />
<option value="$PROJECT_DIR$/design/htmlfiles/users/userregister.html" />
<option value="$PROJECT_DIR$/Users/views.py" />

<option value="$PROJECT_DIR$/design/htmlfiles/users/userpage.html" />

<option value="$PROJECT_DIR$/design/htmlfiles/users/viewmailpage.html" />

<option value="$PROJECT_DIR$/design/htmlfiles/users/spampage.html" />

<item name="Phishing_Email_Detection" type="462c0819:PsiDirectoryNode" />

<item name="design" type="462c0819:PsiDirectoryNode" />

</path>

<path>

<item name="Phishing_Email_Detection" type="b2602c69:ProjectViewProjectNode" />

<item name="Phishing_Email_Detection" type="462c0819:PsiDirectoryNode" />

<item name="design" type="462c0819:PsiDirectoryNode" />

<item name="htmlfiles" type="462c0819:PsiDirectoryNode" />

</path>

<path>

<item name="Phishing_Email_Detection" type="b2602c69:ProjectViewProjectNode" />

<item name="Phishing_Email_Detection" type="462c0819:PsiDirectoryNode" />

<item name="design" type="462c0819:PsiDirectoryNode" />

<item name="htmlfiles" type="462c0819:PsiDirectoryNode" />

<item name="admins" type="462c0819:PsiDirectoryNode" />

</path>

<path>

<item name="Phishing_Email_Detection" type="b2602c69:ProjectViewProjectNode" />

<item name="Phishing_Email_Detection" type="462c0819:PsiDirectoryNode" />

<item name="design" type="462c0819:PsiDirectoryNode" />

<item name="htmlfiles" type="462c0819:PsiDirectoryNode" />

<item name="users" type="462c0819:PsiDirectoryNode" />

</path>

<path>

<item name="Phishing_Email_Detection" type="b2602c69:ProjectViewProjectNode" />

</task>

<servers />

</component>

<component name="ToolWindowManager">

<frame x="-8" y="-8" width="1382" height="744" extended-state="7" />

<editor active="true" />

<layout>

<window_info active="true" content_ui="combo" id="Project" order="0" visible="true" weight="0.18973215" />

<window_info id="Structure" order="1" side_tool="true" weight="0.25" />

<window_info id="Favorites" order="2" side_tool="true" />

<window_info anchor="bottom" id="Message" order="0" />

<window_info anchor="bottom" id="Find" order="1" />

<window_info anchor="bottom" id="Run" order="2" />

<window_info anchor="bottom" id="Debug" order="3" weight="0.4" />

<window_info anchor="bottom" id="Cvs" order="4" weight="0.25" />

<window_info anchor="bottom" id="Inspection" order="5" weight="0.4" />

<window_info anchor="bottom" id="TODO" order="6" />

<window_info anchor="bottom" id="Terminal" order="8" visible="true" weight="0.29411766" />

<window_info anchor="bottom" id="Event Log" order="9" side_tool="true" />

<window_info anchor="bottom" id="Python Console" order="10" />

<window_info anchor="right" id="Commander" internal_type="SLIDING" order="0" type="SLIDING" weight="0.4" />

<window_info anchor="right" id="Ant Build" order="1" weight="0.25" />

<window_info anchor="right" content_ui="combo" id="Hierarchy" order="2" weight="0.25" />

</layout>

</component>

<component name="VcsContentAnnotationSettings">

<option name="myLimit" value="2678400000" />

</component>

<component name="editorHistoryManager">

<entry file="file://$PROJECT_DIR$/Phishing_Email_Detection/wsgi.py">

<provider selected="true" editor-type-id="text-editor" />

</entry>

<entry file="file://$PROJECT_DIR$/manage.py">

<provider selected="true" editor-type-id="text-editor" />

</entry>

<entry file="file://$PROJECT_DIR$/Users/tests.py">

<provider selected="true" editor-type-id="text-editor" />

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/users/checking_attack.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="969">

<caret line="57" column="17" selection-start-line="57" selection-start-column="7" selection-end-line="57" selection-end-column="17" />

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/Phishing_Email_Detection/settings.py">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="961">

<caret line="80" column="28" selection-start-line="80" selection-start-column="28" selection-end-line="80" selection-end-column="28" />

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/users/viewmailcopy.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="251">

<caret line="72" column="21" selection-start-line="72" selection-start-column="20"

selection-end-line="72" selection-end-column="21" />

<folding>

<elementsign="n#style#0;n#a#0;n#td#2;n#tr#1;n#table#0;n#div#0;n#body#0;n#!!top"

expanded="true" />

</folding>

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/users/spamcopy.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="177">

<caret line="101" column="19" selection-start-line="101" selection-start-column="19" selection-end-line="101" selection-end-column="19" />

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/users/userpagecopy.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="285">

<caret line="95" lean-forward="true" selection-start-line="95" selection-end-line="95" />

<folding>

<elementsign="n#style#0;n#input#0;n#td#0;n#tr#0;n#table#0;n#form#0;n#div#0;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#a#0;n#div#1;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#0;n#div#1;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#a#1;n#div#1;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#1;n#div#1;n#body#0;n#!!top" expanded="true"/>

<elementsign="n#style#0;n#a#2;n#div#1;n#body#0;n#!!top”expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#2;n#div#1;n#body#0;n#!!top" expanded="true" />

</folding>

</state>

</provider>

entry file="file://$PROJECT_DIR$/design/htmlfiles/users/feedback.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="95">

<caret line="60" column="30" selection-start-line="60" selection-start-column="30" selection-end-line="60" selection-end-column="30" />

<folding>

<elementsign="n#style#0;n#td#0;n#tr#0;n#form#0;n#table#0;n#div#0;n#!!top" expanded="true"/>

<elementsign="n#style#0;n#input#0;n#td#0;n#tr#1;n#form#0;n#table#0;n#div#0;n#!! top" expanded="true" />

</folding>

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/admins/admin_page.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="313">

<caret line="47" column="35" selection-start-line="47" selection-start-column="35" selection-end-line="47" selection-end-column="35" />

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/admins/viewfeedback.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="228">

<caret line="47" column="41" selection-start-line="47" selection-start-column="41" selection-end-line="47" selection-end-column="41" />

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/admins/analysis_page.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="34">

<caret line="2" column="22" lean-forward="true" selection-end-line="2" selection-end-column="22" />

<caret line="42" lean-forward="true" selection-start-line="42" selection-end-line="42"/>

<folding>

<elementsign="e#0#34#0" expanded="true" />

</folding>

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/design/htmlfiles/admins/chart.html">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="306">

<caret line="54" column="57" selection-start-line="54" selection-start-column="57" selection-end-line="54" selection-end-column="57" />

<folding>

<elementsign="n#style#0;n#a#0;n#h2#2;n#div#0;n#!!top" expanded="true" />

</folding>

<elementsign="n#style#0;n#input#0;n#td#0;n#tr#3;n#table#0;n#form#0;n#div#0;n#body#0;n#!!top" expanded="true" />

</folding>

</state>

<elementsign="n#style#0;n#p#0;n#a#1;n#div#3;n#body#0;n#!!top"expanded="true" />

<elementsign="n#style#0;n#a#2;n#div#3;n#body#0;n#!!top" expanded="true" />

<elementsign="n#style#0;n#p#0;n#a#2;n#div#3;n#body#0;n#!!top" expanded="true" />

</folding>

</state>

</provider>

</entry>

<entry file="file://$PROJECT_DIR$/Users/views.py">

<provider selected="true" editor-type-id="text-editor">

<state relative-caret-position="-1411">

<caretline="163" column="59" selection-start-line="163" selection-start-column="59" selection-end-line="163" selection-end-column="59" />

<folding>

<elementsign="e#0#9#0" expanded="true" />

</folding>

</state>

</provider>

</entry>

</component>

</project>

Python Code:

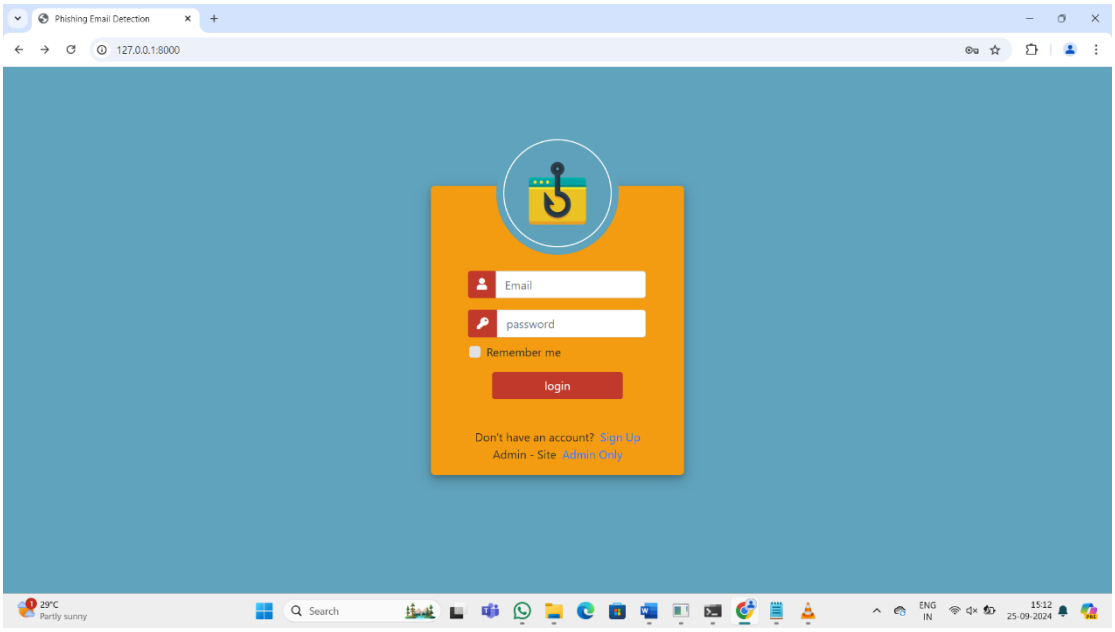
```
from django.db.models import Count

from django.shortcuts import render, redirect, get_object_or_404

from Users.models import SendMailModel, UserRegister_Model, FeedbackModel, Phishing_Model

def login_page(request):

    if request.method == "POST":

        if request.method == "POST":

            usid = request.POST.get('username')
```

```
pswd = request.POST.get('password')
            if usid == 'admin' and pswd == 'admin':
                return redirect('admin_page')
    return render(request,'admins/login_page.html')
def admin_page(request):
    obj=UserRegister_Model.objects.all()
    return render(request,'admins/admin_page.html',{'objects':obj})
def analysis_page(request):
    obj=SendMailModel.objects.all()
    return render(request,'admins/analysis_page.html',{'obj':obj})
def analysisdelete(request,pk):
    obj = get_object_or_404(SendMailModel, pk=pk)
    obj.delete()
    return redirect('analysis_page')
def categoryanalysis_chart(request,chart_type):
    chart =
SendMailModel.objects.values('category').annotate(dcount=Count('category'))
    return render(request,'admins/chart.html',{'objects':chart,'chart_type':chart_type})
def viewfeedback(request):
    obj = FeedbackModel.objects.all()
    return render(request,'admins/viewfeedback.html',{'object':obj})
def attactdetails(request):
    obj=Phishing_Model.objects.all()
    return render(request,'admins/attactdetails.html',{'obj':obj})
```

Chapter 5

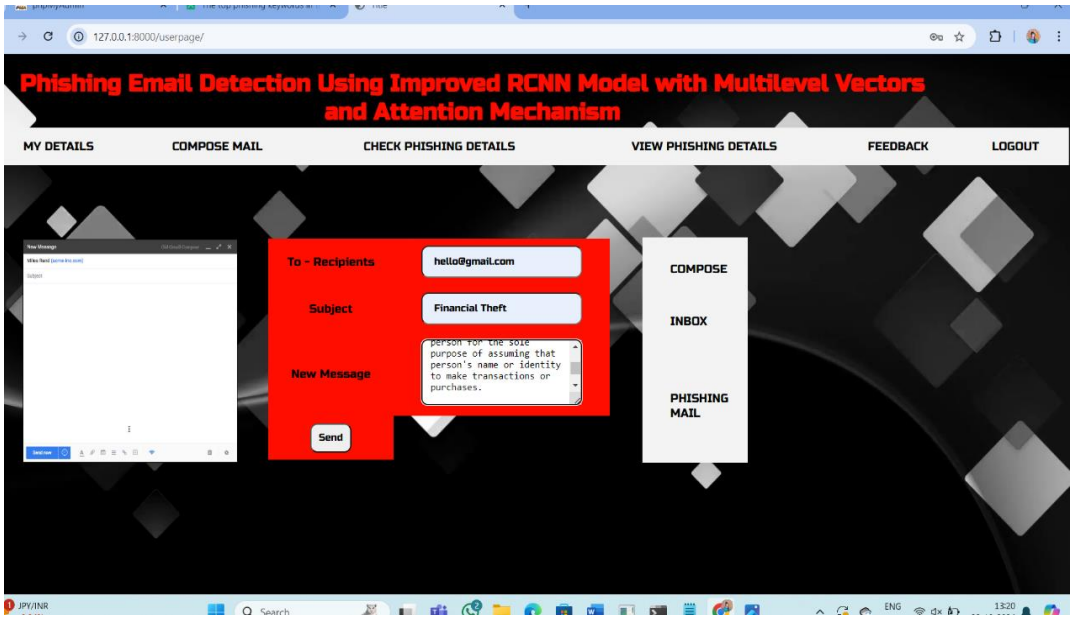
SCREENSHOTS

5.1 SCREENSHOTS



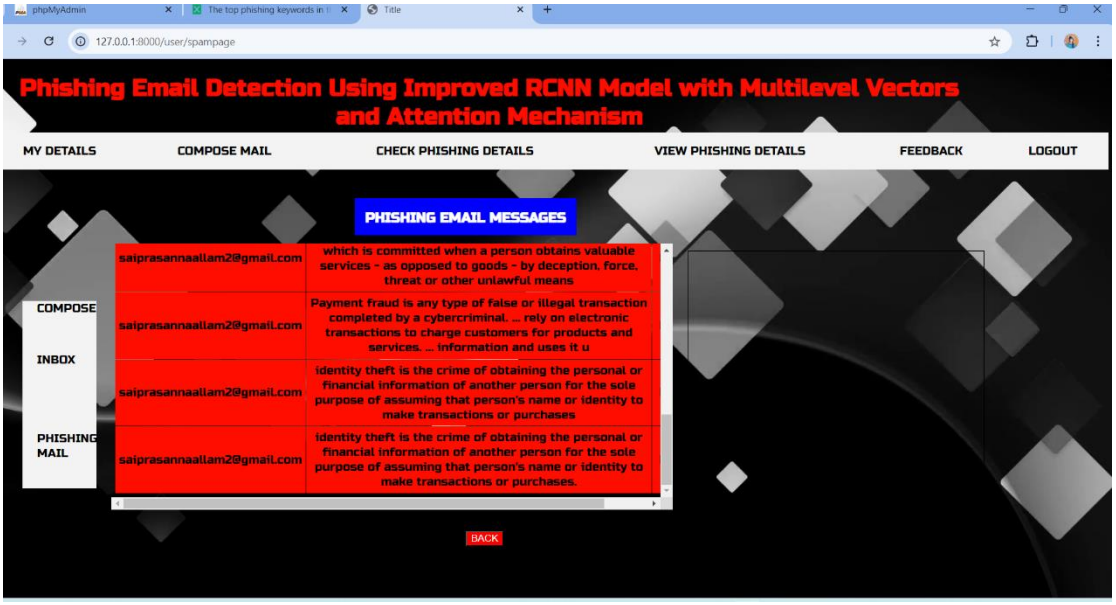
Screen Shot 5.1 User Login Page

In this page every user should login with their personal credentials. In this user name is the email id which was given in the registration page.



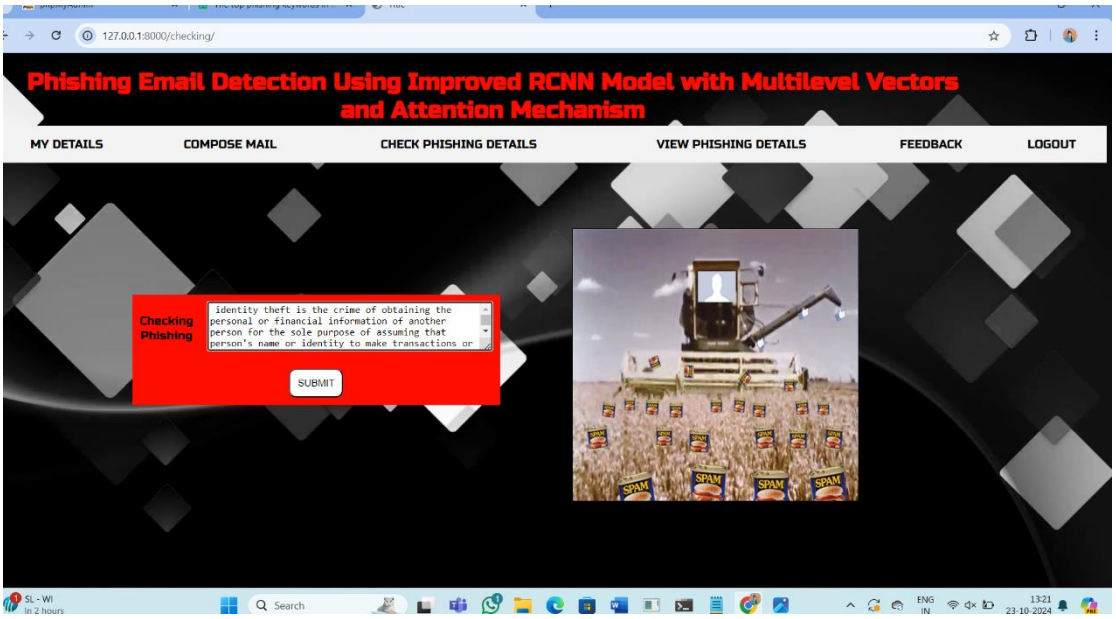
Screen Shot 5.2 Compose a mail

A mail is composed and sent to a recipient containing a subject and a message



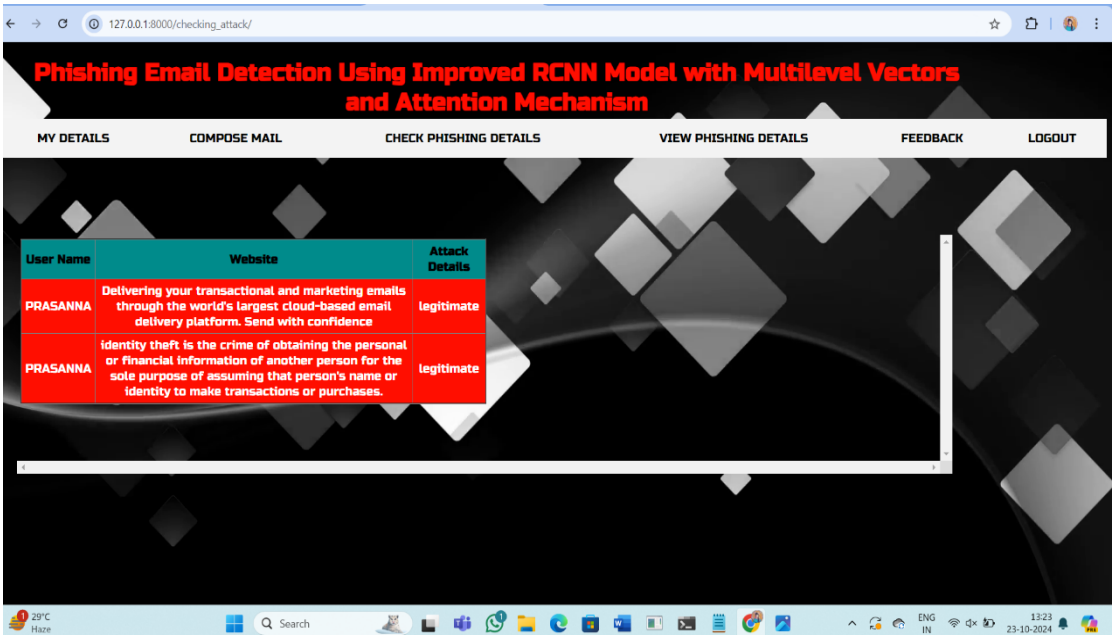
Screen Shot 5.3 Phishing details are detected

In this page the phishing mails are detected based on phishing keywords



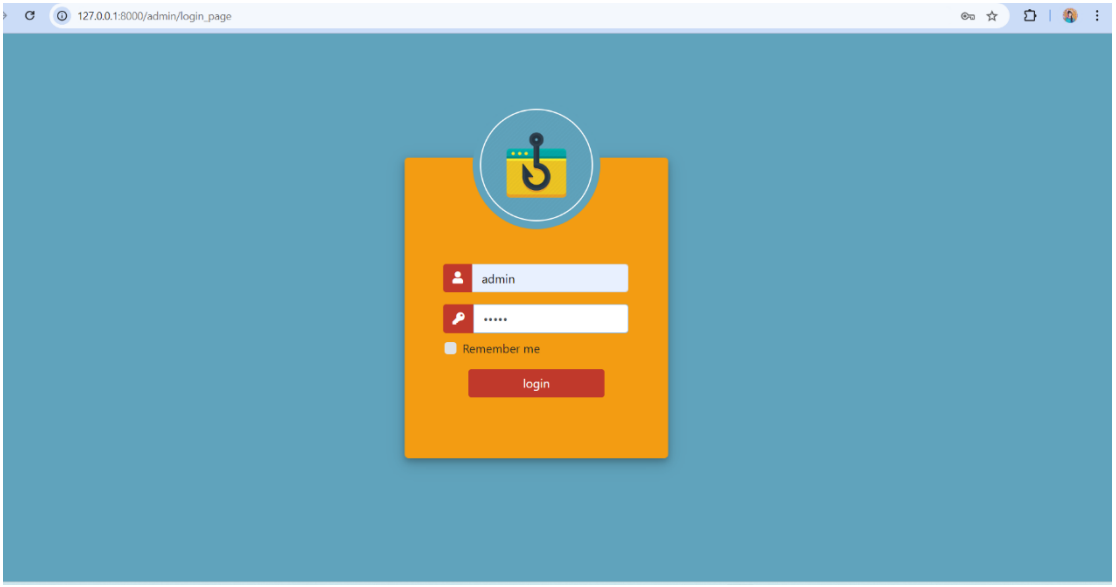
Screen Shot 5.4 Checking Phishing

The received mail is copied here and submitted.The model shows whether mail is legitimate or phished



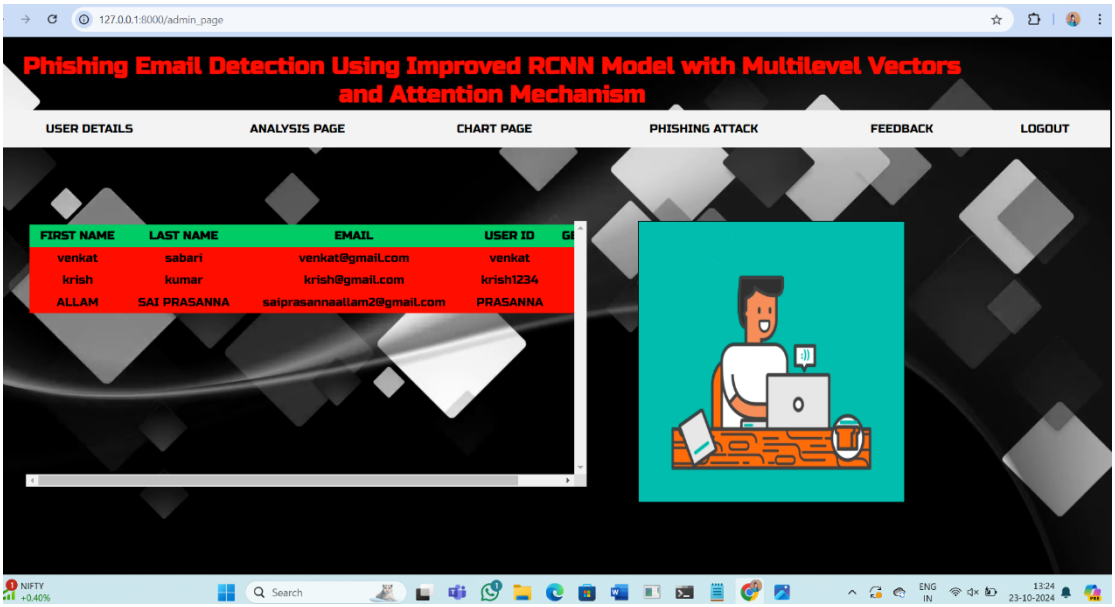
Screen Shot 5.5 View Phishing Details

All the received phishing mails are viewed here.



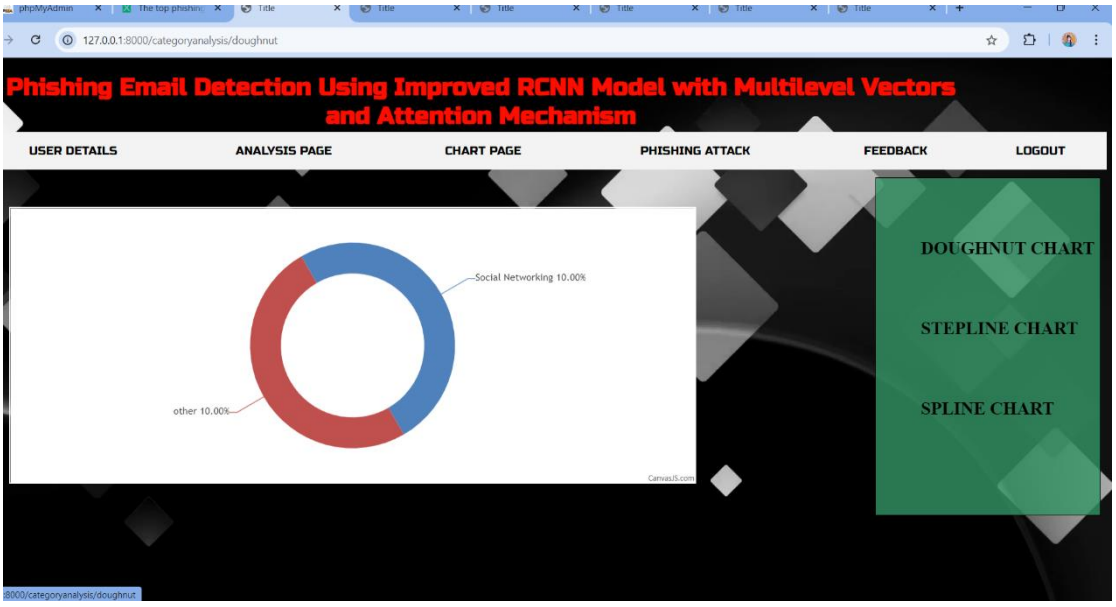
Screen Shot 5.6 Admin Login Page

This page is for admin login usage



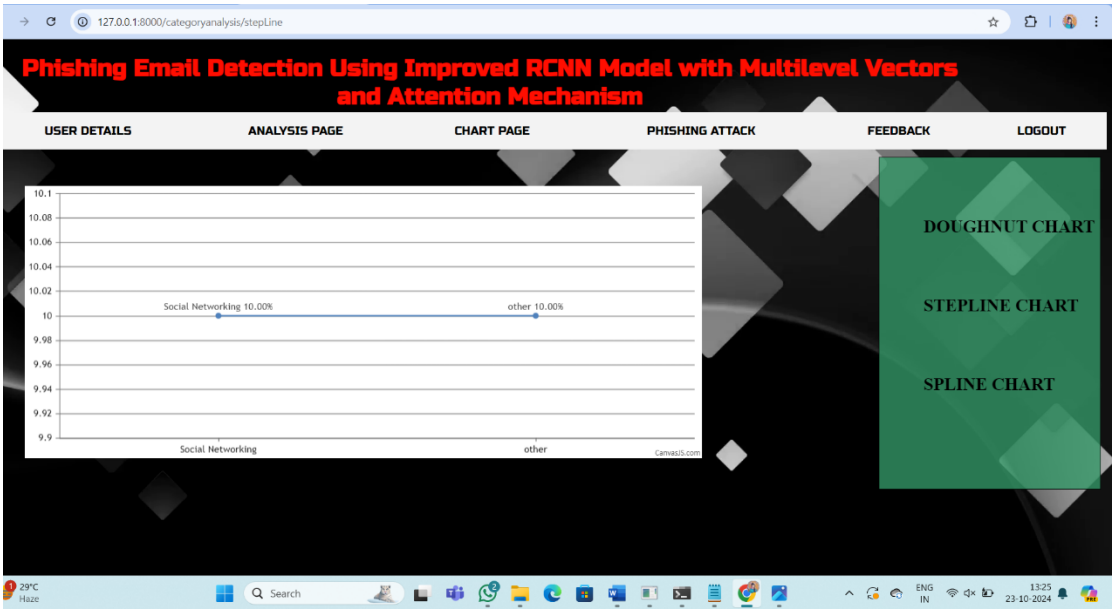
Screen Shot 5.7 Admin Analysis Page

Admin has access to all the mails received by the user.

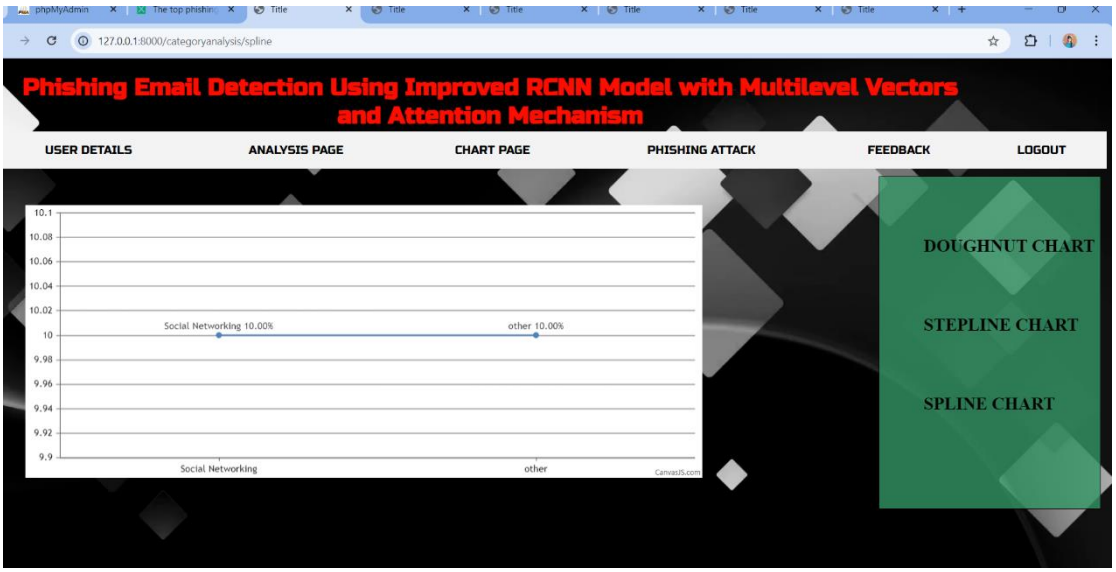


Screen Shot 5.8 DoughNut Chart

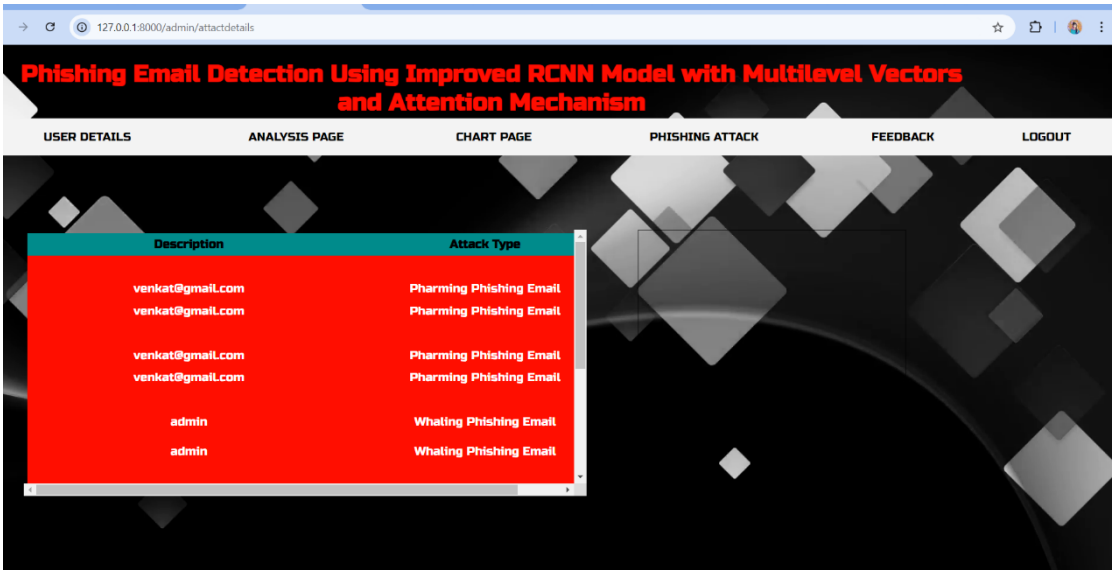
Analysed results shown in graphical representation



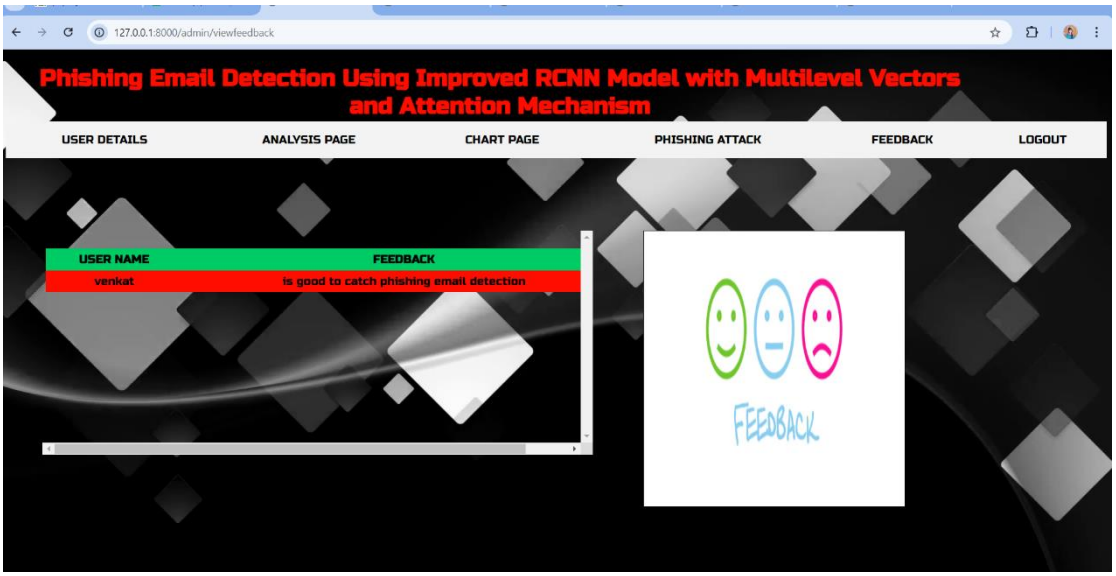
Screen Shot 5.9 Stepline Chart



Screen Shot 5.10 SPLine Chart



Screen Shot 5.11 List of mails &types



Screen Shot 5.12 Feedback of Users

Chapter 6

TESTING

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

6.1 Introduction to Testing

Testing is a crucial phase in the development and deployment of any machine learning system, including phishing email detection models, as it ensures the system operates as expected under different conditions. In phishing detection, the testing process evaluates the model's accuracy, robustness, and overall performance in identifying phishing emails versus legitimate ones. It begins with the preparation of datasets, typically split into training and testing sets, where the model is trained on one portion and evaluated on the other. Testing not only measures the model's accuracy but also helps detect any overfitting—where the model performs well on the training data but poorly on unseen data. It helps ensure that the classifier is generalizing effectively to new and unseen emails, rather than memorizing the training set. The performance of the system is often measured through metrics like accuracy, precision, recall, and the F1 score, which provide insights into how well the system can identify phishing emails and avoid false positives. Furthermore, confusion matrices are often employed to visualize the system's classification errors, helping to fine-tune the model to reduce false positives (legitimate emails classified as phishing) and false negatives (phishing emails classified as legitimate).

6.2 Types of Testing

6.2.1 Unit Testing

Unit testing involves the design of test cases that recognize that the internal program logic works correctly, and that the program inputs provide the correct outputs. All decision sections and internal code flow must be recognized. This is a test of each software unit of the application. This is done after an individual unit is finalized before consolidation. This is a basic test that relies on knowledge of its development and is aggressive. Unit testing Performs basic testing at the unit level and tests a specific business function, application and / or system configuration. Unit testing ensures that each unique path of the business process is accurate to documented features and contains clearly defined inputs and expected outputs.

6.2.2 Integration Testing

Integrated testing is designed to determine whether the integrated software units run as a program. Testing is a phenomenon driven and most prone to the underlying effects of screens or fields. The integration test determines that, although the components are particularly satisfied, as has been shown by the successful unit test, the fusion of the components is correct and stable. Integration testing aims to show problems arising from the combination of components. Software integration testing is an additional integration test of two or more integrated software items on the same platform to create failures caused by interface failures.

6.2.3 Functional Testing

Functional testing is a type of software testing that verifies the system against the functional requirements or specifications. It focuses on ensuring that each function of the software application operates in conformance with the requirement documentation. Testers input data into the system and check the output against expected results, covering areas like user commands, data manipulation, and integrations. Functional testing is essential for validating that the system performs the intended tasks correctly, without considering the internal workings of the application. It helps identify issues related to missing or incorrect features, user interface behavior, and overall system usability.

6.2.4 White Box Testing

White Box Testing involves testing a system's internal structure, focusing on code, algorithms, and data flows. Testers design tests based on the knowledge of the system's inner workings to ensure all code paths function as expected. It is useful for detecting logic errors, vulnerabilities, and optimizing performance.

6.2.5 Black Box Testing

Black Box Testing examines the system's external behaviour without any knowledge of its internal structure. Testers focus on providing inputs and verifying outputs against expected results, ensuring the system meets user expectations and functions correctly. This method is ideal for validating functionality, performance, and usability.

6.3 Test Cases

Table 6.1 Testcases

S. NO	TEST NAME	INPUT	EXPECTED OUTPUT	OUTPUT	STATUS
1	Registration test	All the registration details	Need to validate from database	Successfully validate on database	Success
2	User-Login page	User-id and password	Need to validate from database	Successfully validate on database	Success
3	Admin-login page	User-id and password	Need to validate from database	Successfully validate on database	Success
4	Sending mail	Subject and mail id	Need to validate from database	Successfully validate on database	Success
5	Phishing email detection	Subject	Need to validate from dataset	Successfully validate on dataset	Success
6	Received mail	Subject and mail address	Need to validate from Database	Successfully validate on database	Success

Chapter 7

CONCLUSION & FUTURE SCOPE

7.1 Conclusion

We use a new deep learning model to detect phishing emails. The model employs an improved RCNN to model the email header and the email body at both the character level and the word level. Therefore, the noise is introduced into the model minimally. In the model, we use the attention mechanism in the header and the body, making the model pay more attention to the more valuable information between them. We use the unbalanced dataset closer to the real-world situation to conduct experiments and evaluate the model. The model obtains a promising result. Several experiments are performed to demonstrate the benefits of the proposed model. Since the model employs an advanced Recurrent Convolutional Neural Network (RCNN) architecture that analyses both the email header and body at the character and word levels, effectively minimizing noise during feature extraction. A noteworthy feature of our model is the incorporation of an attention mechanism, allowing it to focus on critical information within the email, thereby improving its ability to identify potential phishing attempts. The results demonstrate promising detection rates, highlighting the model's potential for practical application. Our experiments showcase the advantages of the proposed approach, including its superior performance compared to existing detection methods.

7.2 Future Scope

For future work, we aim to enhance our model’s capabilities in detecting phishing emails that lack an email header and consist solely of the body content. This scenario poses a unique challenge, as the absence of header information removes critical contextual cues that can help identify phishing attempts. To address this, we will explore advanced techniques to analyse the content of the email body more effectively, focusing on extracting meaningful features that can provide insight into the intent and nature of the message. It will contribute to a deeper understanding of phishing tactics, providing valuable information for developing better security measures and educational programs aimed at raising awareness about phishing threats. By focusing on these areas, we hope to significantly advance the field of phishing detection and contribute to more secure communication practices in digital environments.

References

1. Anti-Phishing Working Group. (2018). Phishing Activity Trends Report 1st Quarter .Available: Preports/apwg_trends_report_q1_2018.pdf
2. PhishLabs. (2018). *2018* Phish Trends & Intelligence Report. [Online]. Available:https://info.phishlabs.com/hubfs/2018%20PTI%20Report/PhishLabs%20Trend %20Report_2018-digital.pdf
3. M. Nguyen, T. Nguyen, and T. H. Nguyen. (2018). ‘‘A deep learning model with hierarchical LSTMs and supervised attention for anti-phishing.’’ [Online]. Available: https://arxiv.org/abs/1805.01554
4. Anti-Phishing Working Group. (2016). Phishing Activity Trends Report 4th Quarter2016.[Online].Available:http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf
5. L. M. Form, K. L. Chiew, S. N. Sze, and W. K. Tiong, ‘‘Phishing email detection technique by using hybrid features,’’ in Proc. 9th Int. Conf. IT Asia (CITA*)*, Aug. 2015, pp.
6. R. Verma and N. Hossain, ‘‘Semantic feature selection for text with application to phishing email detection,’’ in Proc. Int. Conf. Inf. Secur. Cryptol. Cham, Switzerland: Springer, 2013.

Github Link

https://github.com/Sai2002/Phishing-Email-Detection-Using-Improved-RCNN