# High Level Design (HLD)

# Malicious URL Detection

# Document Version Control

| Date Issued | Version | Description | Author |
|---|---|---|---|
| 05/01/2023 | 1 | HLD – V1.0 | Diana Laveena DSouza |
| | | | |
| | | | |
| | | | |
| | | | |

# Contents

# Abstract

With the development of Internet technology, network security is under diverse threats. In particular, attackers can spread malicious uniform resource locators (URLs) to carry out attacks such as phishing and spam. The research on malicious URL detection is significant for defending against these attacks. However, there are still some problems. For instance, malicious features cannot be extracted efficiently. Some existing detection methods are easy to evade attackers. We design malicious URL detection using Transformers models to solve these problems.

# 1 Introduction

## 1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions before coding, and can be used as a reference manual for how the modules interact at a high level.

**The HLD will:**

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

## 1.2 Scope

The HLD documentation presents the system's structure, including the database architecture, application architecture, application flow and technology architecture. The HLD uses non-technical to mildly-technical terms, which should be understandable to the system's administrators.

## 1.3 Definitions

| ***Term*** | ***Description*** |
|---|---|
| *Database* | Collection of all the information monitored by this system |
| *IDE* | Integrated Development Environment |

| | |
|---|---|
| *Local System* | Local System |

# 2 General Description

## 2.1 Product Perspective

Transformer models are used to solve cyber security problems. They are used to detect Malicious domain names. As soon as it detects a Malicious URL, it sends information to the cyber security team. Maintain a database to store every Malicious URL.

## 2.2 Problem Statement

To create a Transformer model that should detect Malicious domain names.

## 2.3 Proposed Solution

The solution proposed here is to create a TransformerEncoder with MultiheadAttention model that detects Malicious domain names.

## 2.4 Data Requirements

Data requirements depend entirely on our problem statement.

- We need a large amount of data, nearly 6,26,182 records.

We use the Cassandra database by inserting records into it and then exporting them to a CSV file for further use.

## 2.5 Tools used

Python programming language and frameworks such as NumPy, Pandas, and transformers are used to build the whole model.

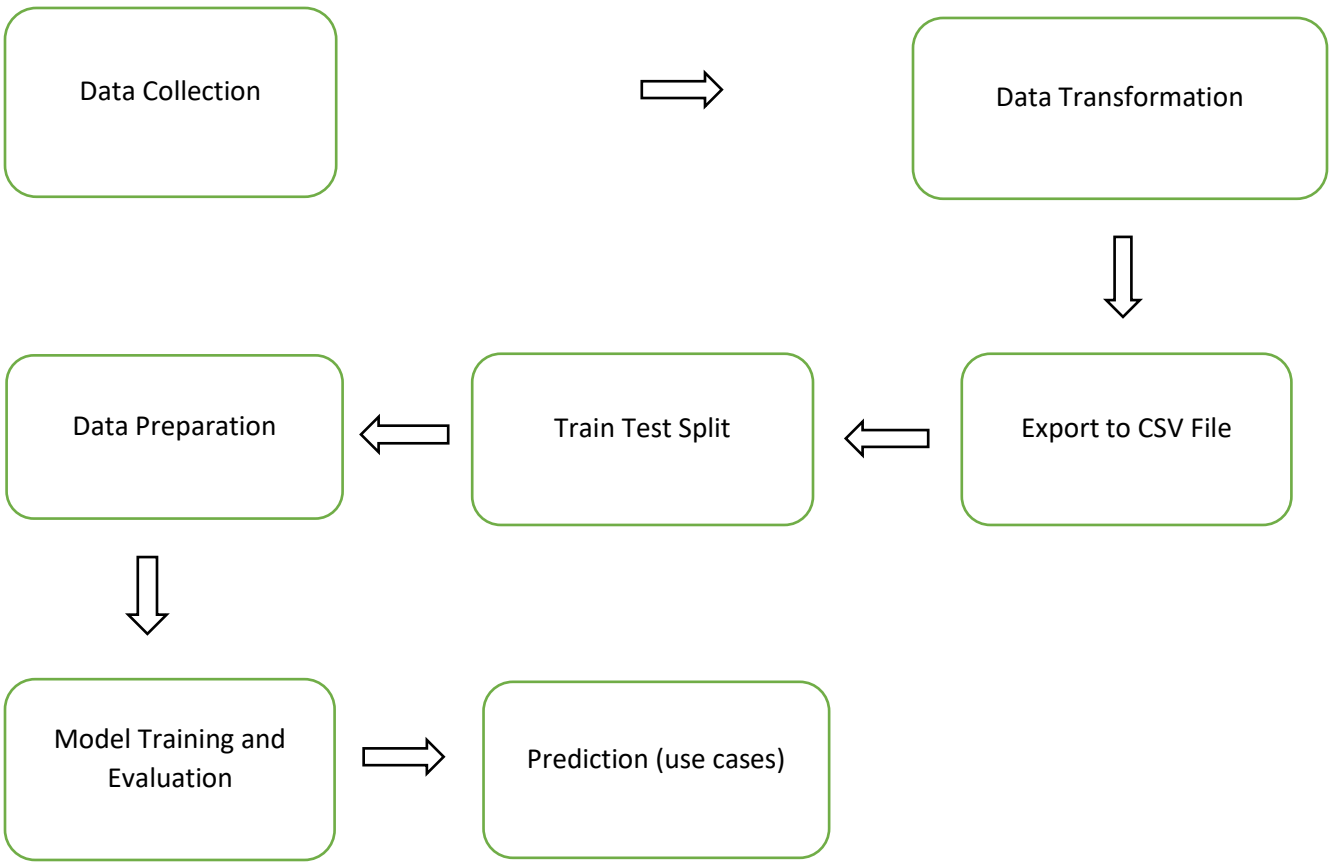

- PyCharm is used as IDE.
- Local System is used for the deployment of the model.
- Front-end development is done using HTML.
- Python Flask is used for backend development.
- GitHub is used as a version control system.
- Torch is used to create Data Loaders and build Transformer models.
- DVC is used to create the AIops pipeline.

# 3 Design Details

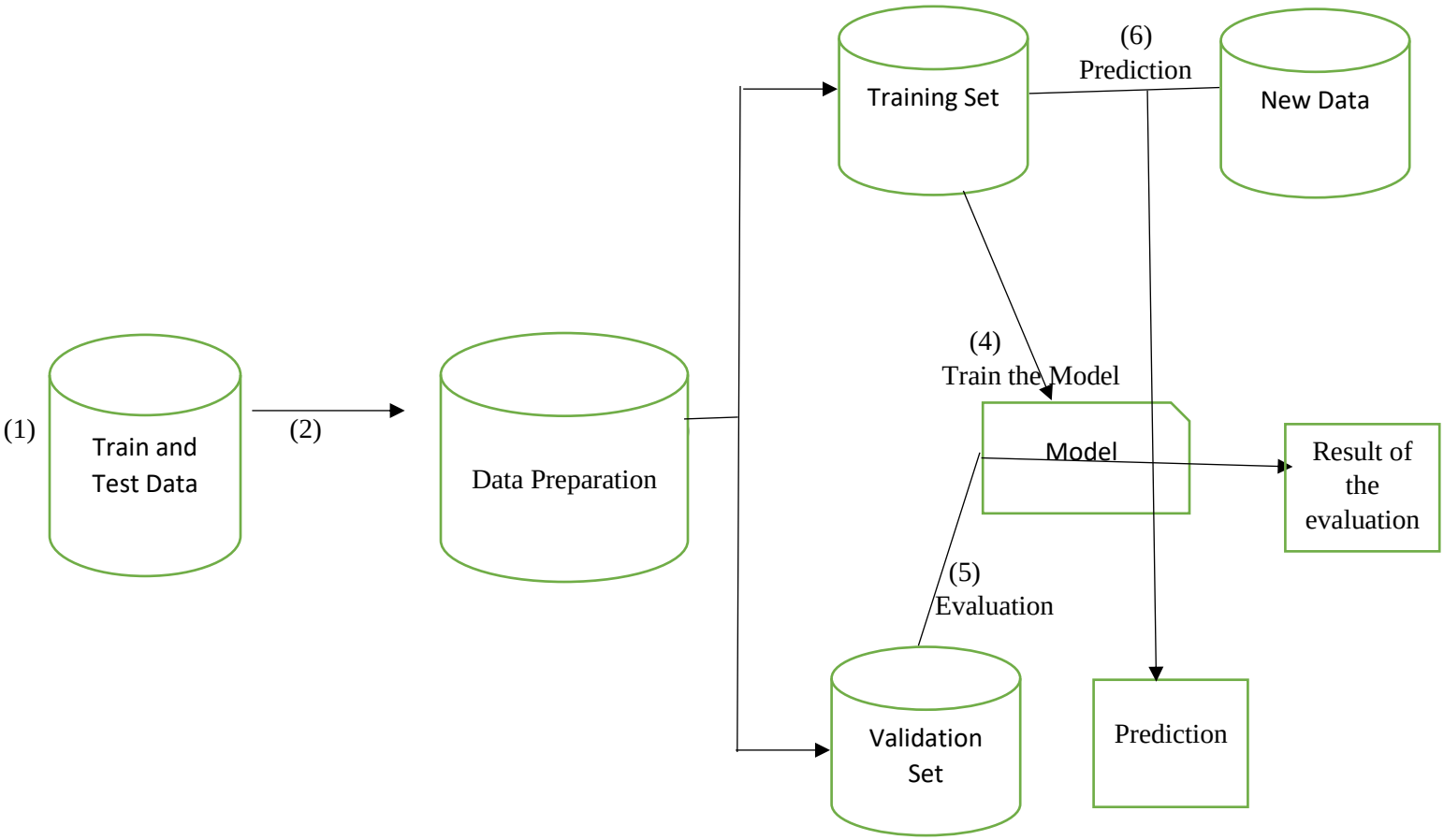
## 3.1 Process Flow

To detect Malicious URLs, we will use the transformer model. Below is the process flow diagram is as shown below.

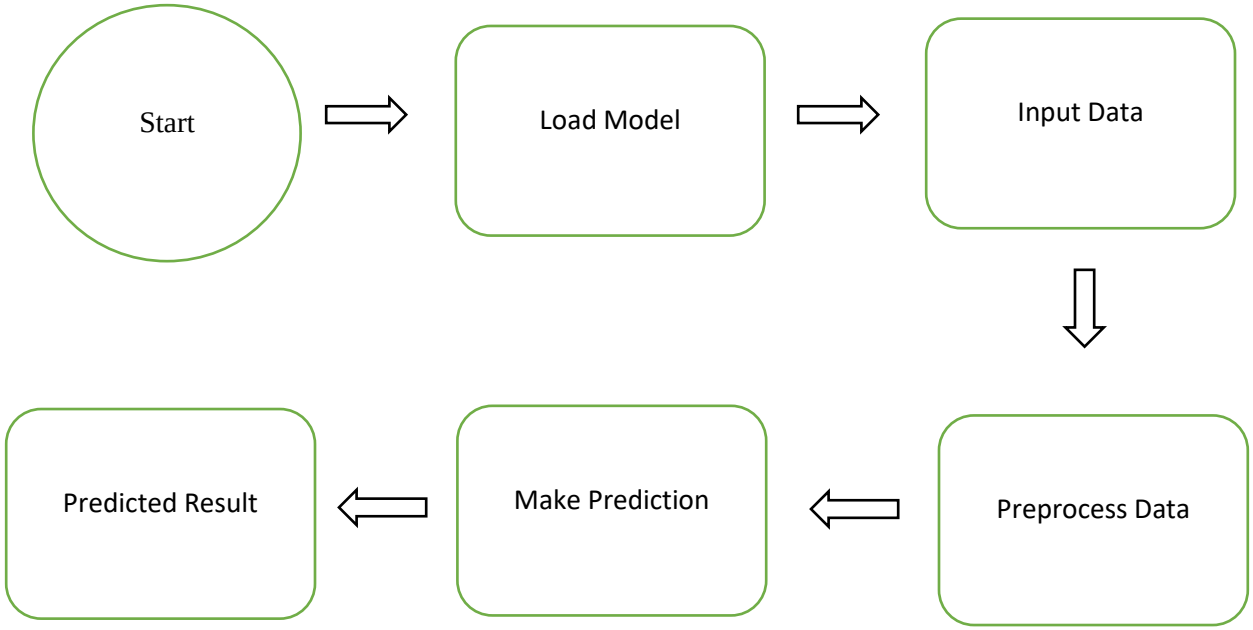
# Proposed Methodology



## 3.1.1 Model Training and Evaluation



### 3.1.2 Deployment Process




## 3.2 Event Log

The system should log every event, so the user will know what process is running internally.

**Initial Step-By-Step Description:**

1. The System identifies at what step logging is required.
2. The System should be able to log each system flow.
3. Developer can choose the logging method. You can choose database logging/File logging as well.
4. System should not hang even after using so many loggings. Logging is just because we can easily debug issues, so logging is mandatory too.

## 3.3 Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything outside the normal and intended usage.

# 4 Performance

The Transformer model is used to detect Malicious domain names to help avoid cybercrime.

## 4.1 Reusability

The code written and the components used should have the ability to be reused with no problems.

## 4.2 Application Compatibility

The different components for this project will use Python as an interface between them. Each component will have its task to perform, and it is the job of Python to ensure the proper transfer of information.

## 4.3 Resource Utilization

When any task is performed, it will likely use all the preprocessing power available until that function is finished.

## 4.4 Deployment




# 5 Conclusion

The dynamic convolutional neural network is used to detect Malicious domain names to help avoid cybercrime.

# 6 References

1. https://www.kaggle.com/code/kawiswara/malicious-web-detection-with-1d-cnn
2. https://www.hindawi.com/journals/scn/2021/5518528/
3. https://www.mandiant.com/resources/blog/training-transformers-for-cyber-security-tasks-malicious-url-prediction
4. https://pdf.sciencedirectassets.com
5. https://phishtank.com/
6. https://www.researchgate.net/publication/359921090_A_HYBRID_PHISHING_DETECTION_MODEL_BASED_ON_TRANSFORMER_CHARACTERBERT_FROM_URLS
7. https://deepai.org/publication/a-transformer-based-model-to-detect-phishing-urls