Low Level Design (LLD)

Malicious URL Detection

Written By	Diana Laveena DSouza
Document Version	0.1
Last Revised Date	05/01/2023

Document Control

Change Record:

Version	Date	Author	Author
0.1	05/01/2023	Diana Laveena DSouza	Introduction & Architecture Defined

Reviews:

Version	Date	Reviewer	Comments
0.1	05/01/2023		Document Content, Version Control and Unit Test Cases to be added

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1 Introduction ……4

1.1 What is Low-Level design document? ……..4

1.2 Scope ……4

2 Architecture ……5

3 Architecture Description ……6

3.1 Data Description ……6

3.2 Data Collection and Transformation……..6

3.3 Data Insertion into Database ……6

3.4 Export Data from the Database...……6

3.5 Data Preparation... ……7

3.6 Model Building and Evaluation ……7

3.7 Data from the User ……..……..7

3.8 Data Validation ……..7

3.9 Prediction ……..……7

3.10 Deployment ……7

4 Unit Test Cases ……...8

1 Introduction

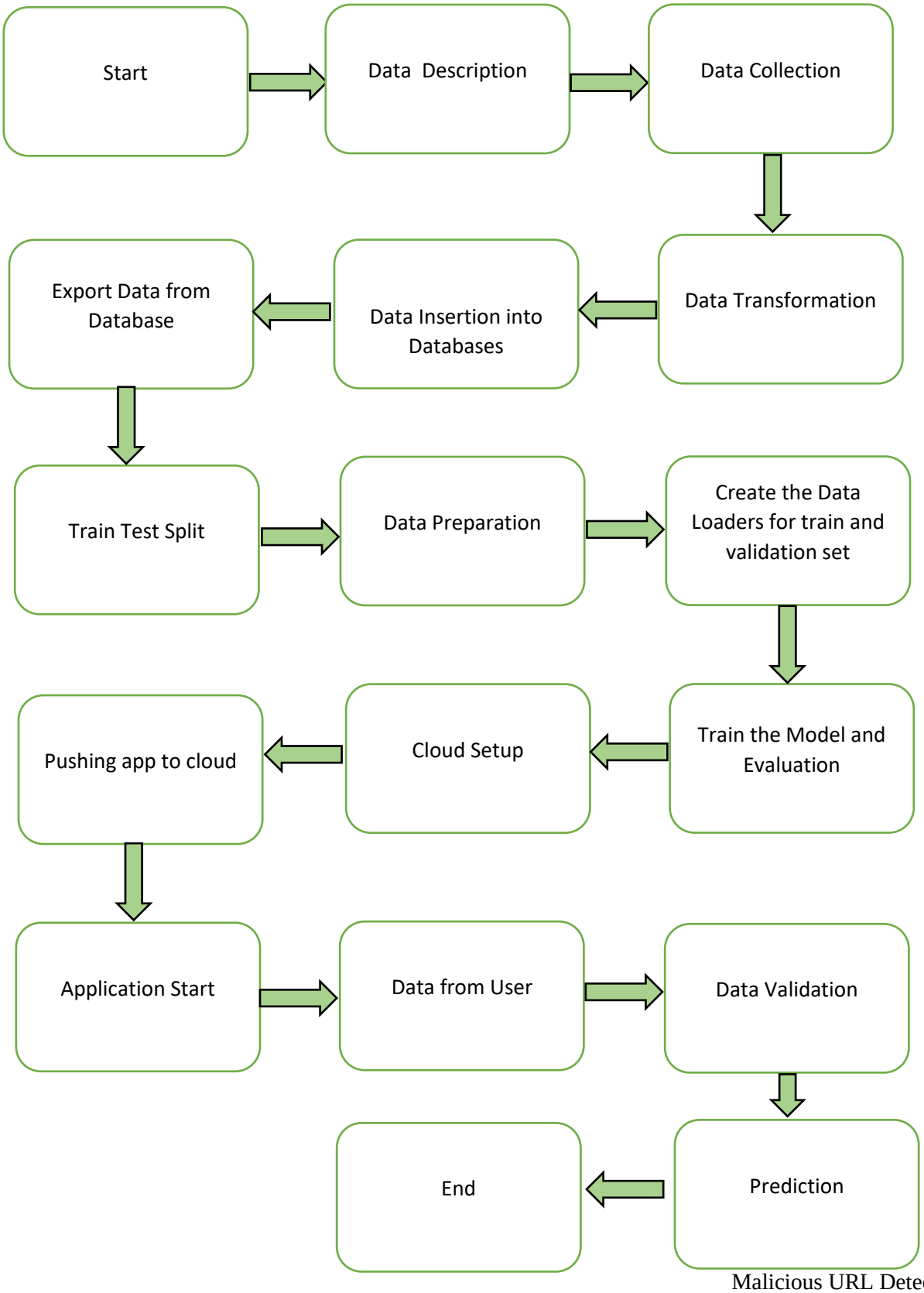
1.1 Why is Low-Level Design Document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for News Summarization. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so the programmer can directly code the program from the document.

1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2 Architecture



3 Architecture Description

3.1 Data Description

Context: Collection of URLs and their Labels helps greatly in NLP Classification tasks.

Content: List of URLs and their Labels; Dataset is split into train & validation for building the transformer model.

3.2 Data Collection and Transformation

Data was collected from the Kaggle Website. Transform the collected data into the required file format.

3.3 Data Insertion into Databases

- Database creation and connection – Create a Cassandra database with a namespace key.
- Table creation in the database.
- Insertion of files in the table.

3.4 Export Data from the Database

Data Export from Database – The data in a stored database is exported as a CSV file for Data Preparation and Model Training.

3.5 Train Test Split

We train the model with 626182 records and 6326 records for validation.

3.6 Data Preparation

Data Preparation involves extracting URL domain, splitting URL domains into lists of characters, Replacing characters with special characters which are not in the token repository. Pad the URLs whose length is smaller than 256 and truncate those whose length is larger than 256. Finally, Tokenize the URL array.

3.7 Model Building and Evaluation

Build the Transformer Block with six Encoder Layers. Each Layer consists of four-head attention and point-wise feed-forward networks of size 128. Calculate the accuracy Score of the model.

3.8 Data from the User

Here we will collect data from URLs from the user.

3.9 Data Validation

Here Data Validation will be done, given by the user.

3.10 Prediction

The model will Predict whether the URL is malicious or not.

3.11 Deployment

We will be deploying the model to GCP.

This is a workflow diagram for the Malicious URL Detection.

4 Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	The application URL should be accessible to the user.
Verify whether the Application loads entirely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load entirely for the user when the URL is accessed.
Verify whether the user can input the text in all input fields	1. Application is accessible	The user should be able to input the text in all input fields.
Verify whether the user gets Submit button to submit the inputs.		The user should get Submit button to submit the inputs.
Verify whether the user is presented with results on clicking submit.		The user should be presented with results on clicking submit