**Architecture**

**Phasing Domain Detection**

Document Version:1.0

Control Version Date: 22-Feb-24

Author : Akshay Kumar BM

Comments: Introduction & Architecture Defined

**Context**

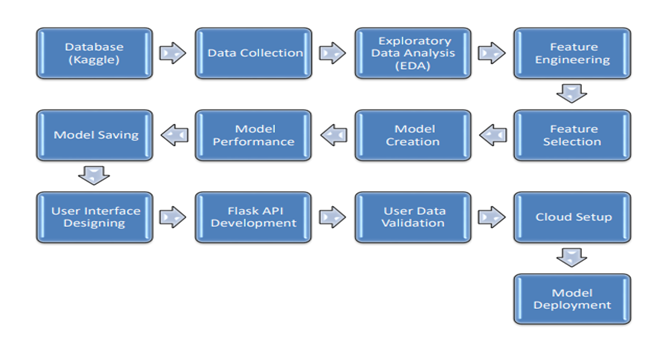
1. Introduction
   1. What is Low-Level design document?
   2. Scope
2. Architecture
3. Architecture Description
   1. Data Ingestion
   2. Data Transformation
   3. Model Training
   4. Prediction
   5. Deployment

# Low-Level Documentation

* 1. Model Training Component
  2. Training Pipeline
  3. Prediction Pipeline
  4. Web Application
  5. Logging
  6. Exception

**1. Introduction**

The phishing domain detection system is designed to predict whether a given URL is a phishing website or not. The system is built using Python and involves several components including data ingestion, data transformation, model training, and prediction.

**2. Architecture**

The system is divided into several modules, each responsible for a specific task in the pipeline. These modules include:

* Data Ingestion
* Data Transformation
* Model Training
* Prediction

**3. Architecture Description**

**3.1 Data Ingestion**

The DataIngestion class is responsible for reading the dataset from a CSV file and splitting it into training and testing datasets. The paths for these datasets are stored in the DataIngestionconfig dataclass.

**3.2 Data Transformation**

The DataTransformation class handles preprocessing tasks such as handling missing values and feature scaling. The preprocessing object is saved as a pickle file for later use in the prediction pipeline.

**3.3 Model Training**

The ModelTrainer class trains several models on the preprocessed training data and selects the best performing model based on accuracy. The trained model is saved as a pickle file.

**3.4 Prediction**

The PredictPipeline class loads the preprocessor and trained model to make predictions on new data. It includes a method extract\_features to extract relevant features from a given URL.

**3.5 Deployment**

The Flask application can be run locally for development and testing. For production, it can be deployed on a cloud platform that supports Python and Flask.

This document provides a high-level overview of the system’s design. Each module can be further broken down into smaller components for a more detailed low-level design. The actual implementation may vary based on specific requirements and constraints.

**4. Low-Level Design Document for Phishing Domain Detection Project**

**1. Data Ingestion Component**

This component is responsible for reading the dataset and splitting it into training and testing sets. The DataIngestion class has a method initiate\_data\_ingestion which reads the dataset, splits it into training and testing sets, and saves them in the specified paths.

**Python**

class DataIngestion:

def \_\_init\_\_(self):

self.ingestion\_config=DataIngestionconfig()

def initiate\_data\_ingestion(self):

...

**2. Data Transformation Component**

This component preprocesses the data by handling missing values and scaling the features. The DataTransformation class has a method initiate\_data\_transformation which applies the preprocessing steps on the training and testing datasets.

**Python**

class DataTransformation:

def \_\_init\_\_(self):

self.data\_transformation\_config=DataTransformationconfig()

def initiate\_data\_transformation(self,train\_path,test\_path):

...

**3. Model Training Component**

This component is responsible for training the model. The ModelTrainer class has a method initiate\_model\_training which trains the model on the preprocessed training dataset.

**Python**

class ModelTrainer:

def \_\_init\_\_(self) -> None:

self.model\_train\_config=ModelTrainerConfig()

def initate\_model\_training(self,train\_array,test\_array):

...

**4. Training Pipeline**

This component orchestrates the training process by calling the methods from the Data Ingestion, Data Transformation, and Model Training components sequentially.

**Python**

class Training\_pipeline:

def \_\_init\_\_(self) -> None:

self.DataIngestion\_obj=DataIngestion()

self.data\_transformation\_obj=DataTransformation()

self.model\_trainer\_obj=ModelTrainer()

def initiate\_training\_pipeline(self):

...

**5. Prediction Pipeline**

This component is responsible for making predictions on new data. The PredictPipeline class has a method predict which takes in the features of a new data point, applies the preprocessing steps, and makes a prediction using the trained model.

**Python**

class PredictPipeline:

def \_\_init\_\_(self):

pass

def predict(self,features):

...

**6. Web Application**

The web application is built using Flask. It has a route / that accepts both GET and POST requests. On a GET request, it renders the home page. On a POST request, it takes the URL from the form, extracts features, makes a prediction, and renders the home page with the prediction result.

**Python**

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

def predict\_datapoint():

...

**7. Logging**

The logging component is responsible for logging information, warnings, and errors during the execution of the program. It logs the messages to a file with the current timestamp in its name.

**Python**

logging.basicConfig(

filename=LOG\_FILE\_PATH,

format="[ %(asctime)s ] %(lineno)d %(name)s - %(levelname)s - %(message)s",

level=logging.INFO

)

**Exception**

The CustomException class is a custom exception class that logs the error message and details about where the error occurred.

**Python**

class CustomException(Exception):

def \_\_init\_\_(self, error\_message, error\_detail:sys):