i

Low Level Design

Thyroid Disease Detection System

|  |  |
| --- | --- |
| Written By | Sushant Bisht |
| Document Version | 1.0 |
| Last Revised Date | 28-07-2022 |

**Document Version Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **28-07-2022** | **Sushant Bisht** | **User Interface defined** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

# Contents

Document Version control 2

1 Introduction 4 1.1 What is Low-Level Design Document 4

1.2 Scope 4 2 Architecture 5

1. Architecture Description 6
   1. Data Description 6
   2. Export Data from DB to CSV for training 6
   3. Data Preprocessing 6
   4. Hyperparameter Tuning 6
   5. Model saving 7
   6. Cloud setup 7
   7. Push App to cloud 7 3.8 Data from client side for prediction 7
   8. User Interface 7 Unit Test Case 8

# Introduction

## What is Low-Level design document?

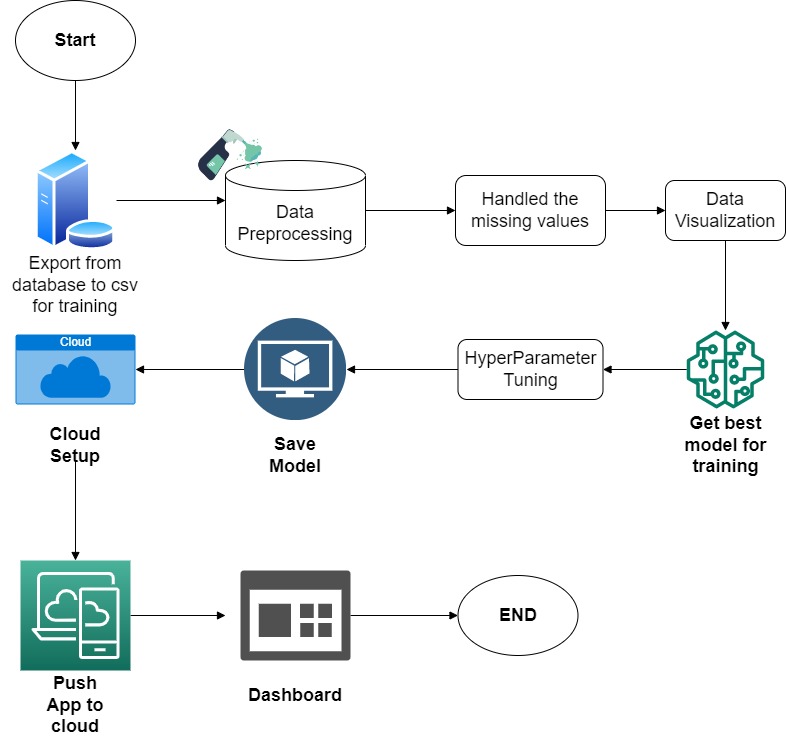
The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program code for Thyroid Disease Detection System. LLD describe the class diagrams with the methods and relations between classes and program specs. It describe the modules so that the programmer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

## 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.

## Architecture

2

****

1. **Architecture Description**

3

* 1. **Data Description**

We will be using Thyroid Disease Data Set present in UCI Machine Learning Repository. This Data set is satisfying our data requirement. Total 7200 instances present in different batches of data. Data is consists of categorical and numerical attributes.

## Export Data from database to CSV for Training

All those batch file which after passing through data validation step get stored in Database. And here we will be exporting all those CSV files in a single CSV file for training purpose. For this we need to write a separate module.

## Data Preprocessing

We will be exploring our data set here and do EDA if required and perform data preprocessing depending on the data set. We first explore our data set in Jupyter Notebook and decide what pre-processing and Validation we have to do such as imputation of null values, dropping some column, etc and then we have to write separate modules according to our analysis, so that we can implement that for training as well as prediction data.

## Get best model

Here we will train various model on training set which we will obtain in Data

Preprocessing step, and then will try to get best model for training set.

## Hyperparameter Tuning

After selecting best model, we will do hyperparameter tuning for each selected model, and try to increase performance of the models. And finally save model.

## Model Saving

After performing hyperparameter tuning for models, we will save our models as per best accuracy. so that we can use them for prediction purpose and accordingly we will call model for making prediction over client data.

## Cloud Setup

Here We will do cloud setup for model deployment. Creating all required files needed for cloud deployment of app. Here we also create our streamlit app and user interface and integrate our model with streamlit

## Push app to cloud

After doing cloud setup and checking app locally, we will push our app to cloud to start the application. We will be deploying the model to Heroku.

## Data from client side for prediction purpose

Now our application on cloud is ready for doing prediction. We will get the inputs from the user through our dashboard, and our model will predict the outcome.

## User Interface

**Type of User Interface.**

1. For single user input prediction, we will make a UI which will take all inputs from a single user and give back the prediction there only. So that user can also use this system for single prediction.
2. **Unit Test Cases**

5

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application URL is accessible to the user | 1. Application URL should be  defined | Application URL should be accessible to the user |
| Verify whether the Application loads completely for the user when the URL is accessed | 1.Application URL is accessible 2.Application  is deployed | The Application should load completely for the user when the URL is accessed |
| Verify whether the User is able to sign up in the application | 1. Application is accessible | The User should be able to sign up in the application |
| Verify whether user is able to successfully login to the application | 1. Application is accessible 2. User is signed up to the application | User should be able to successfully login to the application |
| Verify whether user is able to see input fields on logging in | 1. Application is   accessible   1. User is signed up to the application 3.User is logged in to the application | User should be able to see input fields on logging in |
| Verify whether user is able to edit all input fields | 1. Application is accessible 2. User is signed up to the application 3.User is logged in to the application | User should be able to edit all input fields |
| Verify whether user gets Submit button to submit the inputs | 1. Application is accessible 2. User is signed up to the application 3.User is logged in to the application | User should get Submit button to submit the inputs |
| Verify whether user get prediction/output back after submitting the inputs | 1. Application is accessible 2. User is signed up to the application 3.User is logged in to the application | User should get their output after submitting the inputs. |
| Verify whether the output which user get is accordance to inputs user made. | 1. Application is accessible 2. User is signed up to the application 3.User is logged in to the application | The output should be in accordance with the inputs user made. |
|  |  |  |