LOW LEVEL DESIGN (LLD)

i

# LOW LEVEL DESIGN

Thyroid Disease Detection System

Made by – Vinayak Verma

LOW LEVEL DESIGN (LLD)

**Document Version Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **28-07-2023** | Vinayak verma | **Architecture** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

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

**Approval Status:**

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

LOW LEVEL DESIGN (LLD)

## Contents

1. Architecture
2. Architecture Description
   1. Data Description
   2. Export Data from DB to CSV for training
   3. Data Preprocessing
   4. Data Classification

2.5Model saving

* 1. Cloud setup
  2. Push App to cloud
  3. Data from client side for prediction

2.8 Prediction

LOW LEVEL DESIGN (LLD)

### 1. Introduction

**1.What is Low-Level design document?**

A Low-Level Design Document (LLD) is a detailed technical document that delineates the internal logical architecture of a software program or system. Specifically tailored for the Thyroid Disease Detection System, the LLD focuses on providing a comprehensive design for the actual program code. Its primary objective is to furnish a blueprint for software implementation, detailing components, classes, methods, and their interrelationships. This document includes specific program specifications, enabling programmers to directly translate the design into executable code with the LLD as a guiding reference

#### 1.2 Scope

The scope of Low-Level Design (LLD) encompasses a component-level design process characterized by a systematic refinement approach. During this phase, designers concentrate on defining data structures, specifying software architecture, writing source code, and optimizing performance algorithms. While data organization may have been outlined during the requirement analysis phase, LLD refines and solidifies these designs, ensuring clarity and precision in the data design work. Ultimately, LLD serves a pivotal role in guiding the development team, facilitating a methodical and efficient coding process for the Thyroid Disease Detection System.

03

LOW LEVEL DESIGN (LLD)

### 2. Architecture

2

**Data**

**Preprocessing**

**Data classi**

**-**

**fication**

**Export data from**

**database**

**to**

**csv**

**for**

**training**

**Start**

**Data from**

**client to be**

**predicted**

**Application**

**Start**

**Prediction**

**Data**

**Preprocessing**

**Data classi**

**-**

**fication**

**Model saving**

**Cloud setup**

**Hyper**

**-**

**parameter**

**tuning**

**End**

**Pushing app**

**to**

**cloud**

**Export data**

**from**

**database**

**to**

**csv**

**to prediction**

**Architecture Description: Thyroid Disease Prediction**

**3.1 Data Description:**  
We will utilize the Thyroid Disease Data Set sourced from the UCI Machine Learning Repository, comprising 7200 instances distributed across different data batches.

**3.2 Export Data from Database to CSV for Training:**  
All data batches from the database will be consolidated into a single CSV file for subsequent model training.

**3.3 Data Preprocessing:**  
During this phase, we will explore the dataset and conduct exploratory data analysis (EDA) as needed. Based on our findings, data preprocessing tasks such as handling null values and dropping unnecessary columns will be performed. We will develop separate modules for these preprocessing steps, applicable for both training and prediction phases.

**3.4 Data Classification:**  
We will employ the Random Forest technique for data classification, utilizing an ensemble of decision trees to categorize data into predefined classes or groups. This approach involves building multiple decision trees on various data subsets and aggregating their predictions to achieve the final classification. It enhances accuracy, mitigates overfitting, and effectively handles complex datasets. The trained random forest model will be saved for future predictions.

**3.5 Hyperparameter Tuning:**  
Post classification, we will conduct hyperparameter tuning to optimize model performance for accurate predictions.

**3.6 Model Saving:**  
After classification and tuning, all trained models will be saved to facilitate future predictions.

**3.7 Cloud Setup:**  
We will establish a cloud environment for deploying our models, involving the creation of a Streamlit app with a user interface to interact with the backend models.

**3.8 Push App to Cloud:**  
Once the local testing of the application is complete, we will deploy the entire application to the cloud, making it publicly accessible.

**3.9 Client-Side Data for Prediction:**  
With the application deployed on the cloud, we will begin receiving prediction data from clients. This data will undergo the same data cleansing process as the training data using previously developed modules. It will then proceed through data preprocessing, classification, and be fed into the appropriate saved models for generating predictions.

**3.10 Display Prediction Results:**  
Upon completing all the aforementioned steps, the final thyroid disease prediction results will be displayed through the application interface.

### 1. User Interface

1. For prediction, we will make a separate UI which will take all inputs from a single user and give back the prediction there only.

#### 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 | | | 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 2. User is signed up to 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.  2. | Application is accessible 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.  2. | Application is accessible 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.  2. | Application is accessible 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. |
| Verify whether user have option to download their result or not. | 1.  2. | Application is accessible User is signed up to the application 3.User is logged in to the application | | User should have option to download their output result. |