Low Level Design (LLD)

Thyroid Disease Detection System

Written By	Komal Madiwal
Document Version	1.0
Last Revised Date	

Document Version Control

Change Record:

Version	Date	Author	Comments
1.0	20-01-2024	Komal Madiwal	Introduction & Architecture defined

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

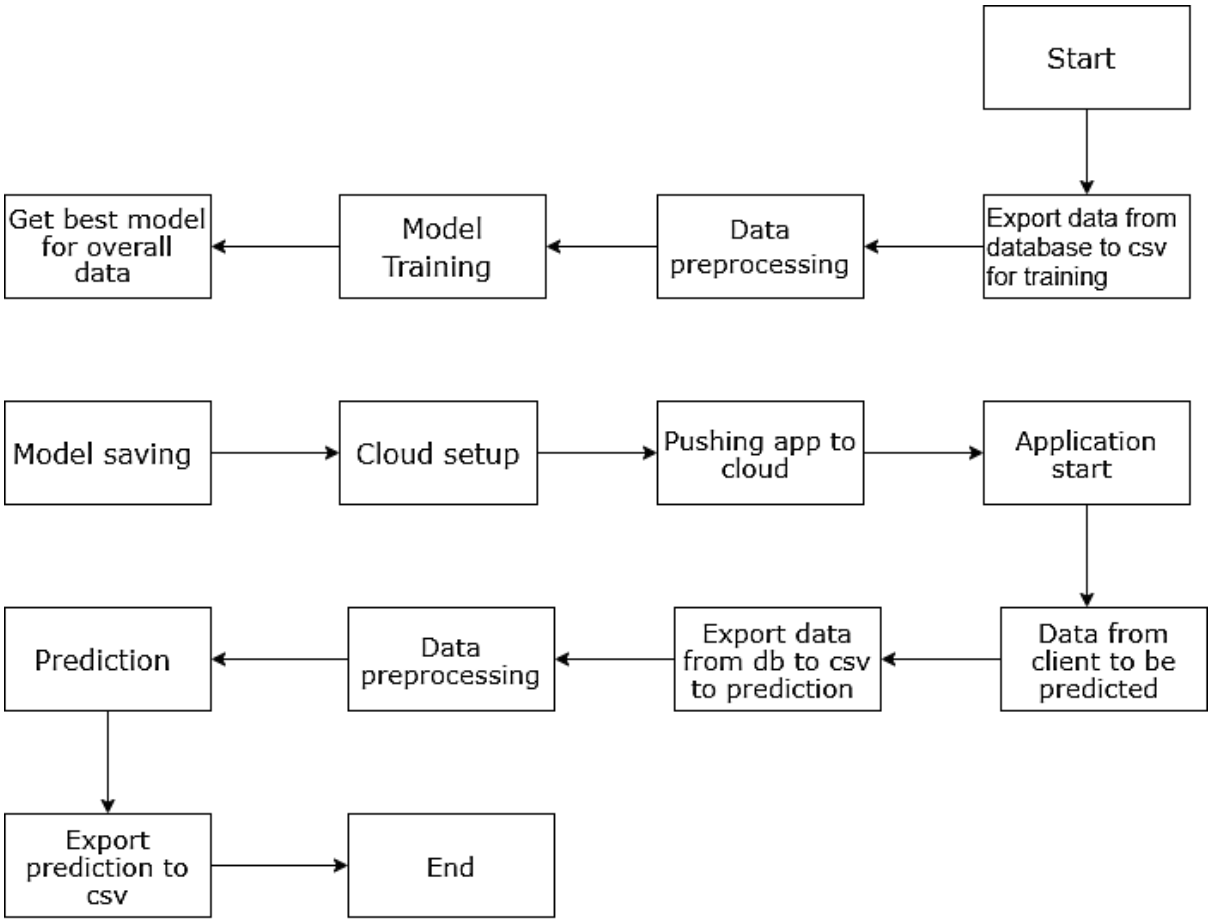
1.1 What is Low-Level design document?

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 describes the modules so that 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, thedata organization may be defined during requirement analysis and then refined during data design work.

2.Architecture



3.Architecture Description

3.1Data 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.

3.2Export Data from database to CSV for Training

Here we will be exporting all batches of data from database into one csv filefor training.

3.3Data 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.

3.4 Model Training

Various machine learning models will be trained on the preprocessed data for subsequent use in prediction.

3.5 Get Best Model for Overall Data

Different models will be evaluated, and the best-performing model for the overall dataset will be selected.

3.6 Model Saving

After identifying the best model, we'll save it for future use in prediction.

3.7 Cloud Setup

Cloud setup will be performed for model deployment. Additionally, a Flask app and user interface will be created, integrating the model with the Flask app and UI.

3.8 Push App to Cloud

After local testing, the application will be pushed to the cloud to make it accessible for users.

3.9 Data from Client Side for Prediction

Prediction data from clients will be exported from the database and undergo the same data cleansing process as the training data. This involves data preprocessing using modules created for training data.

3.10 Export Prediction to CSV

Once predictions for client data are complete, the final step is to export the predictions to a CSV file for delivery to the client.

3.11 User Interface

For single user input prediction, we will make a separate UI which will take all inputs from a single user and give back the prediction there only. So that any single user can also use this system for single prediction.

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 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 into 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