LOW LEVEL DESIGN (LLD)

FLIGHT FARE ESTIMATOR

|  |  |
| --- | --- |
| **Written By** | **Anmol Vaid** |
| **Document Version** | **1.0** |
| **Last Revised Data** | **11-Dec-2021** |

Document Version Control:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 1.0 | 11-Dec-2021 | Anmol Vaid |  |
|  |  |  |  |

Contents

[1. Introduction 3](#_Toc90220319)

[1.1 What is Low-Level design document? 3](#_Toc90220320)

[1.2 Scope 3](#_Toc90220321)

[2. Architecture 3](#_Toc90220322)

[3. Architecture Description 4](#_Toc90220323)

[3.1 Data Collection 4](#_Toc90220324)

[3.2 Feature selection 4](#_Toc90220325)

[3.3 Model Selection 4](#_Toc90220326)

[3.4 Hyper-parameter Tuning 5](#_Toc90220327)

[3.5 Model Saving 5](#_Toc90220328)

[3.6 Data from User 5](#_Toc90220329)

[3.7 Displaying Fair on web portal 5](#_Toc90220330)

[4. Test Cases 5](#_Toc90220331)

# Introduction

## What is Low-Level design document?

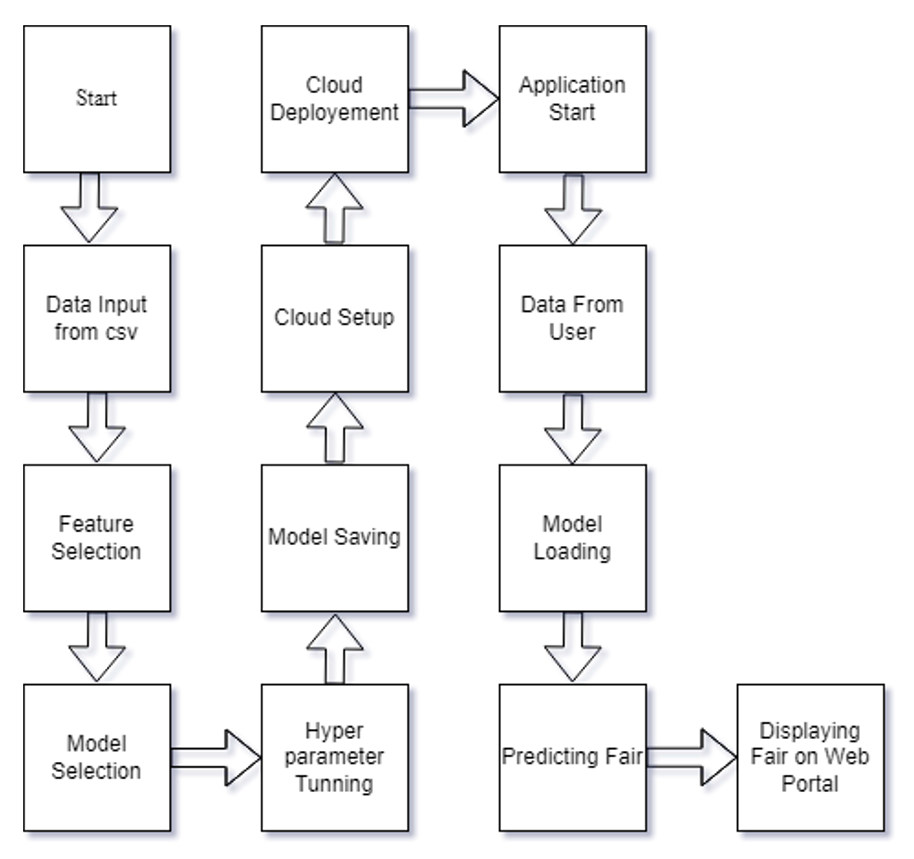
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for flight fare estimation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer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

## 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

Below is architecture of this project.



# Architecture Description

## 3.1 Data Collection

Dataset row columnar data includes the flight service, flight fare, number of stops, total number of duration, departure-arrival date and all. These are given in the comma separated value format (.csv). These data is collected from the Kaggle which contains both the test data and train data.

## 3.2 Feature selection

It is most important step before we begin building our model. In order to avoid false prediction, robust and accurate model we eliminate features with multicollinearity, etc. Selection of those features which are related to for prediction is done.

## 3.3 Model Selection

Model Selection is performed by file model\_training.py which is inside Model\_Training. For model Training we are just choosing XGBRegressor because it is performing best than other models When calculating mean\_absolute\_error XGB always had a very low mean absolute error (mae).

## 3.4 Hyper-parameter Tuning

We are doing hyperparameter tuning only for XGB and because of tuning our model has a low mean\_absolute\_error.

## 3.5 Model Saving

Model Saving is the final step in the training part. We are saving the model in folder Model\_for\_prediction as model.pickle and I am using module pickle for saving the ML model.

## 3.6 Data from User

On Application Starting user will be interacting with a UI which is designed using HTML/CSS. Following details will be provided by user:

* Departure Date
* Arrival date
* Source
* Destination
* Stoppage
* Which Airline you want to travel
* Submit

## 3.7 Displaying Fair on web portal

When user will click on submit button, Fair wll be displayed just below submit button in INR.

# Test Cases

Test cases are given below

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application URL is accessible to the user |    Application URL should be defined | Yes, it is defined. |
| Verify whether the Application loads completely for the user when the URL is accessed |    Application URL is accessible | Yes, it is accessible and deployed |
|    Application is deployed |

# 