LOW LEVEL DOCUMENT

FLIGHT FARE PREDICTOR

|  |  |
| --- | --- |
| Written By | Pushpam Kumar Jha |
| Document Version | 1.0 |
| Last Revised Data | 8th July 2023 |

# 1 Document Control:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 8th July 2023 | Pushpam Kumar Jha |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 2 Reviews:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Reviewer | Comments |
| 1.0 | 8th July 2023 | Pushpam Kumar Jha |  |
|  |  |  |  |
|  |  |  |  |

# 3 Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Review Date | Reviewed By | Approved By | Comments |
| 1.0 | 8th July 2023 | Pushpam Kumar Jha | Pushpam Kumar Jha |  |

Table of Contents

[1 Document Control: 2](#_Toc89379022)

[2 Reviews: 2](#_Toc89379023)

[3 Approval Status: 2](#_Toc89379024)

[1.Introduction 5](#_Toc89379025)

[1.1 What is Low-Level design document? 5](#_Toc89379026)

[1.2. Scope 5](#_Toc89379027)

[2. Architecture 6](#_Toc89379028)

[3. Architecture Description 7](#_Toc89379029)

[3.1 Data Collection 7](#_Toc89379030)

[3.2 Data Validation 7](#_Toc89379031)

[3.3 Data Preprocessing 7](#_Toc89379034)

[3.4 Model Selection 8](#_Toc89379035)

[3.5 Hyperparameter Tuning 8](#_Toc89379036)

[3.6 Model Saving 8](#_Toc89379037)

[3.7 Data from User 8](#_Toc89379038)

[3.8 Data Validation 8](#_Toc89379039)

[3.9 Data Preprocessing 9](#_Toc89379040)

[3.10 Model Loading 9](#_Toc89379041)

[3.11 Model.predict 9](#_Toc89379042)

[3.12 Saving the output file 9](#_Toc89379043)

[4 Test Cases 9](#_Toc89379044)

# 1.Introduction

## 1.1 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.

## 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, the data organization may be defined during requirement analysis and then refined during data design work

# 2. Architecture

# 3. Architecture Description

## 3.1 Data Collection

We have 14k Dataset row columnar data includes the flight service, flight fare, number of stops, total number of duration, departure-arrival date and all. These is 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 Data Validation

In Data Validation part we basically check whether files inside the folder provide are having same columns or not with the data type of each column each by each file. If the file is found to matching the name of columns and dtype of the columns then we are supposed to go to next step otherwise stop and log the error.

## 3.3 Data Preprocessing

Data Preprocessing step is very necessary for model creation, We have mainly categorical data into our csv file so we need to convert that categorical data into numerical format. Data Preprocessing is getting done by the file data\_transformation.py

## 3.4 Model Selection

Model Selection is performed by file model\_training.py which is inside Model\_Training. For model Training we are just choosing Random Forest Regressor because it is performing best than other models When calculating mean\_squared\_error Random Forest Regressor always had a very low mse.

## 3.5 Hyperparameter Tuning

We are doing hyperparameter tuning only for Random Forest Regressor and because of tuning our model has a low mean\_squared\_error.

## 3.6 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.

## 3.7 Data from User

On Application Starting user will be interacting with a UI which is designed using HTML/CSS.

Which will take the required data from user like Source, Destination, Date of Journey, Total Stops, Arrival Time, Departure Time, Additional Information.Adddddps

## 

## 3.8 Data Preprocessing

In data preprocessing I will be handling categorical columns plus some other columns as well as like Duration, Arrival\_Time column.

## 3.9 Model Loading

Model loading is actually very simple we will be calling the model which we saved as modle.pickle inside artifacts folder. For loading the model I am using pickle library.

## 3.10 Model.predict

After loading the model everything is very simple you just have to predict the preprocessed data.

## 3.11 Showing the output

The output is shown on the webpage.

# 4 Test Cases

Test cases are given below

|  |  |  |
| --- | --- | --- |
| **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 Response time of url from backend model. | 1. Application is accessible | The latency and accessibility of application is very faster we got in heroku service. |
| Verify whether user is giving standard input. | 1. Handeled test cases at backends. | User should be able to see successfully valid results. |
| Verify whether user is able to see input fields on logging in | 1. Application is accessible 2. 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 logged in to the application | User should be able to edit all input fields |
| Verify whether user gets Custom File Predict, Default File Predict button to submit the inputs | 1. Application is accessible 2. User is logged in to the application | User should get both buttons to submit the inputs |
| Verify whether user is presented with recommended results on clicking submit | 1. Application is accessible 2. User is logged in to the application | User should be presented with recommended results on clicking submit |
| Verify whether the recommended results are in accordance to the selections user made | 1. Application is accessible 2. User is logged in   to the application and database | The recommended results should be in accordance to the selections user made |