# FLIGHT FRARE PREDICTION ARCHITECTURE DESIGN

# Contents

# **Abstract**

- 1 Introduction
- 1.1 Why this Low-Level Design Document?
- 1.2 Scope
- 1.3 Constraints
- 1.4 Risks
- 1.5 Out of Scope
- 2 Technical specifications
- 2.1 Predicting Fare
- 2.2 Logging
- 2.3 Database
- 3 Technology stack
- 4 Proposed Solution
- 5 Model training/validation workflow
- 6 User I/O workflow
- 7 Exceptional scenarios

# **Abstract**

In the present day contemporary scenario, Airlines have become popular and affordable domestically at least, and these flight fare is a factor people consider to travel. All these fares aspect varies according to season, weather condition, availability of staff and so on. So here we are going to build ML model that predicts the price of an Airline which would be useful prior to bookings.

Introduction

# 1.1 Why this Low-Level Design Document?

The purpose of this document is to showcase a detailed working on how the flight fare prediction happens. Thus, explaining the features of the system, UI, what it does and so on. This document brings a clear clarity for the higher management and the developers to take further action.

# 1.2 Scope

This system is a web application which is used to predict the flight fare for any traveller. Mainly, this helps in getting to check various flights, departure date, arrival date, stoppages and so on at a go and predicts a fair price.

### 1.3 Constraints

The main constraint of this project is one cannot sort the prices according to their requirement and budget.

# 1.4 Risks

Document specific risks that should be identified.

# 1.5 Out of scope

Delineate specific activities and features that are out of scope.

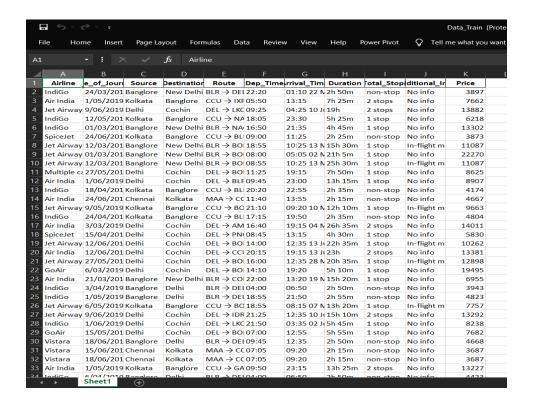
# 2 Technical Specifications

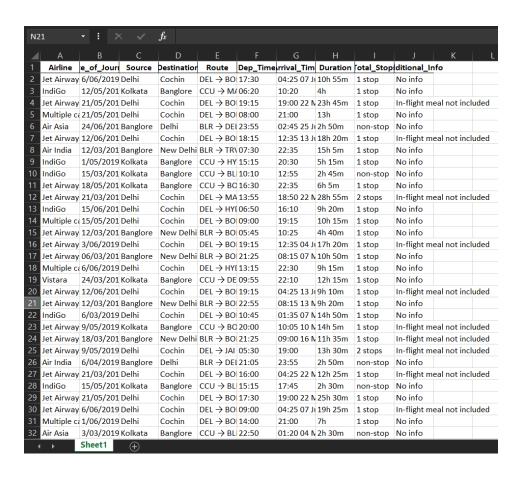
### 2.1Dataset

Feature	Outcome	Source
Price(training data)	yes	Git hub
Price(testing data)	yes	Git hub

# 2.2Dataset Overview

The Dataset contains close to 10600 records of information that has Airway carriers, routes, Source of boarding, destination, and additional info. Some of the columns has less or no information that's useful. It has train and test data in separate file.





# 2.3Predicting Fare

- System displays the features that are to be filled.
- User fills in the required details in the proper format.
- System displays the desired output.

# 2.4Logging

We should be able to log every activity done by the user.

- The System identifies at what step logging required
- The System should be able to log each and every system flow.
- Developers can choose logging methods. You can choose database logging/

File logging as well.

System should not be hung even after using so many loggings. Logging just

because we can easily debug issues so logging is mandatory to do.

# 2.5Database

We create an API for the upload of data into the Cassandra Database, steps performed are:

Connection is made with database

Created a database with name flight fare

Cqlsh command is given for creating data table with required parameters.

Cqlsh command is written for uploading the dataset into the table.

# 3. Technology Stack

Front End	HTML, CSS
Backend	Python Flask
Deployment	Render.com

# 4. Proposed Solution

Based on the trained model, if we were to look at the output we use Random Forest Regressor for the output prediction. Out of all the used and tested models we select the random forest regressor here.

Models used- Random forest, Linear regression, Gradient boosting, XG Boost regressor

Actual model – Random forest regressor

# 5. Model Training

- Importing the necessary Libraries
- read the data. (get insights on it).
- Check for null values. (Drop or use imputation)
- EDA
- Standardize the data set
- Encode the dataset (Label or One-Hot encoding upon requirement)
- Preprocessing
- Model Building
- Deploy using the Flask library
- Start the app (command is "python app.py")
- Input from the user
- Get the Result.

# 6. User I/O workflow

