

Architecture

FLIGHT FARE PREDICTION APPLICATION

Written By	Pankaj Jaiswal, Omkar Bhatuse Siddhiraj Kolwankar
Document Version	0.1
Last Revised Date	25 – Aug -2021

Document Control

Change Record:

Version	Date	Author	Comments
0.1	21– Aug - 2021	Pankaj	Introduction & Architecture defined
0.2	23– Aug - 2021	Siddhiraj	Technical specifications
0.3	25– Aug - 2021	Omkar	Architecture Description
0.4	30-Aug 2021	Pankaj Omkar Siddhiraj	Deployment
	3-Aug-2021	Pankaj Siddhiraj	Unit Testing

Reviews:

Version	Date	Reviewer	Comments
0.1	26– May - 2021	Omkar,Siddhiraj	Document Content , Version Control and Unit Test Cases to be added

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments
0.1	15-May-2021	Omkar,Siddhiraj	Pankaj,Siddhiraj	

Contents

1. Introduction	1
1.1. What is Low-Level design document?	1
1.2. Scope.....	1
2. Technical specifications	
2.1. Logging	
2.2. Database	
3. Architecture	2
4. Architecture Description	3
4.1. Data Collection.....	3
4.2. Data Cleaning.....	3
4.3. Data Pre-processing	3
4.4. Exploratory Data Analysis.....	4
4.5. Model Building.....	4
4.6 Hyperparameter Tuning:.....	
4.7 Model Dump:.....	4
4.8 User Interface	4
4.9 Data Validation.....	4
4.10 User Data Inserting into Database	4
4.11 Model Call/.pkl file loaded	4
4.12 Deployment	
4.13 WebFlow	
4.14 Model training/validation workflow	
4.15 User I/O Flow	
5. User Inferace of Website	
6. Unit Test Cases	5

1. Introduction

Why this Low-Level Design Document?

The purpose of this document is to present a detailed description of the Deep EHR System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher management for its approval.

The main objective of the project is to predict if a person can get a chronic disease in his/her future based on the EHR. EHR stands for Electronic Health Record, EHR is nothing but a dataset of medical history of the patients.

Scope

This software system will be a Web application. This system will be designed to detect the diseases at earliest for better disease management, improved interventions, and more efficient health-care resource allocation using previous EHR records available. More specifically, Early detection of any preventable diseases is important for better disease management. This system is designed to predict the diseases from patient information such as demographics, disease history, lab results, procedures and medications.

Constraints

We will only be selecting a few of the chronic diseases.

Risk

Document specific risks that have been identified or that should be considered.

Out of Scope

Delineate specific activities, capabilities, and items that are out of scope for the project.

2. Technical Specifications

2.1 Logging

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.

2.2 Database

System needs to store every request into the database and we need to store it in such a way that it is easy to retrain the model as well.

1. The User chooses the disease.
2. The User gives required information.
3. The system stores each and every data given by the user or received on request to the database. Database you can choose your own choice whether MongoDB/ MySQL.

3. Architecture



4 Architecture Description

4.1 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 are given in the comma separated value format (.csv). These data are collected from the Kaggle which contains both the test data and train data.

4.2 Data Cleaning

In the cleaning process, we have cleaned up all the data because data is present in very bad format which was not recognized by machine. So data engineering is done very first.

4.3 Data Pre-processing

Data Pre-processing steps we could use are Null value handling, One hot encoding, columns in an integer format by implementing proper techniques to manage the columnar data.

4.4 Exploratory Data Analysis

In eda we have seen various insights from the data so we have selected which column is most important and dropped some of the columns by observing their spearman rank co-relation and plotting their heatmap from seaborn library also we did outlier removal and null value managed in an efficient manner and also implemented one hot encoding their.

4.5 Model Building:

After cleaning the data and completing the feature engineering, we have done split data into the train data and test data and implemented various regression algorithms like Linear, DecisionTree, SVR, RandomForest, K-NN, AdaBoost, GradientBoost Regression and also calculated their accuracies on test data and train data.

4.6 Hyperparameter Tuning:

In hyperparameter tuning we have implemented various ensemble techniques like random forest regressor, bagging and boosting we also did randomized search cv or grid search cv and from that we also implemented cross validation techniques for that. From that we have chosen best parameters according to hyperparameter tuning and best score from their accuracies so we got 85% accuracy in our random forest regression after hyper parameter tuning.

4.7 Model Dump:

After comparing all accuracies and checked all roc, auc curve we have chosen hyperparameterized random forest regression as our best model by their results so we have dumped these model in a pickle file format with the help of joblib python module.

4.8 User Interface :

In Frontend creation we have made a user interactive page where user can enter their input values to our application. In these frontend page we have made a form which has beautiful styling with css and bootstrap. These html user input data is transferred in json format to backend. Made these html fully in a decoupled format.

4.9 Data from User

Here we will collect users requirement to catch their flight like a number of stops, departure date, destination , Source, Number of hours , Day etc.

4.10 Data Validation

Here Data Validation will be done, given by the user

4.11 Model Call/.pkl file loaded

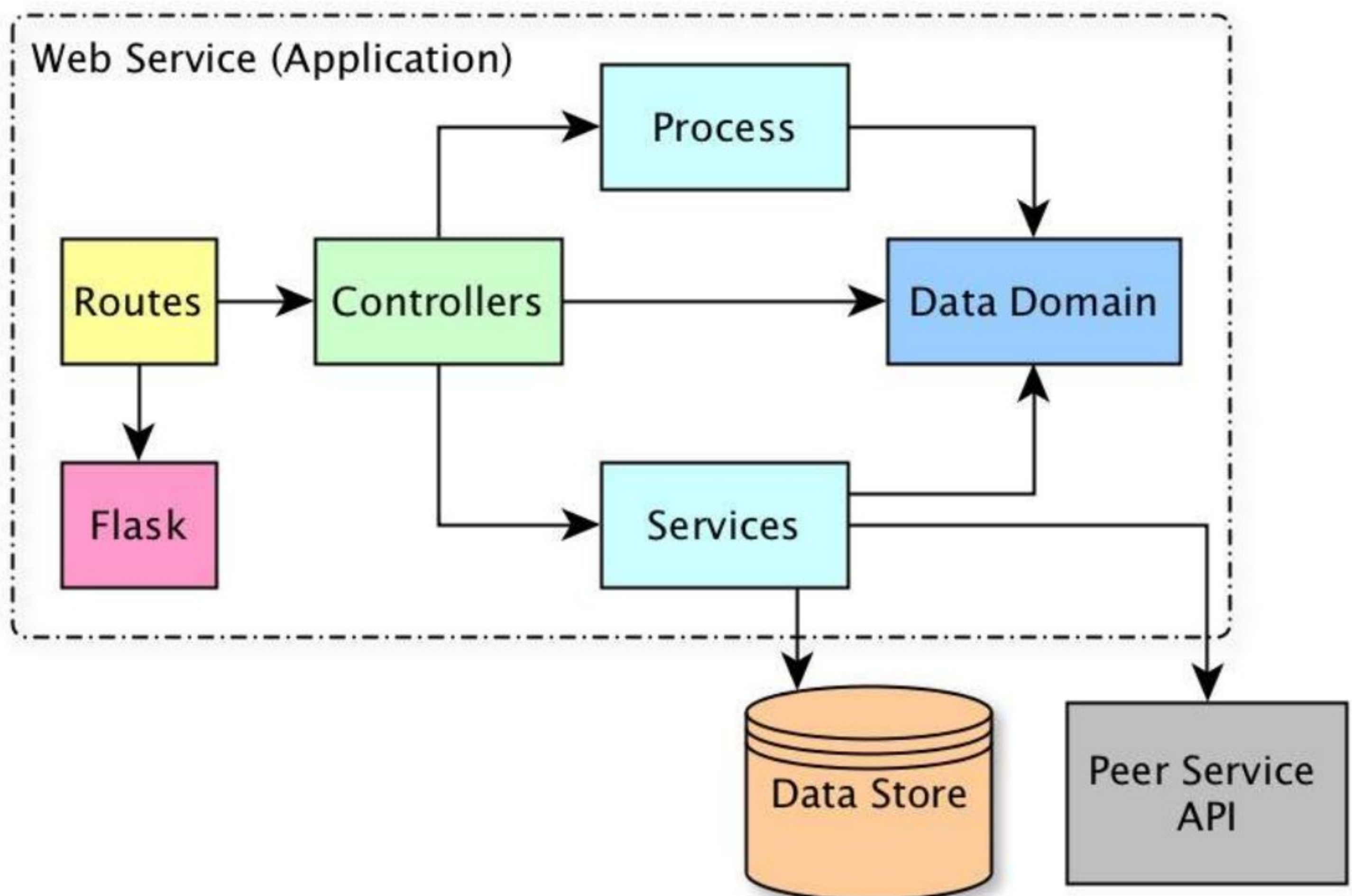
Based on the User input will be throwing to the backend in the dictionary format so our we are loading our pickle file in the backend and predicting some total number of hours as a output and sending to our html page.

4.12 Deployment

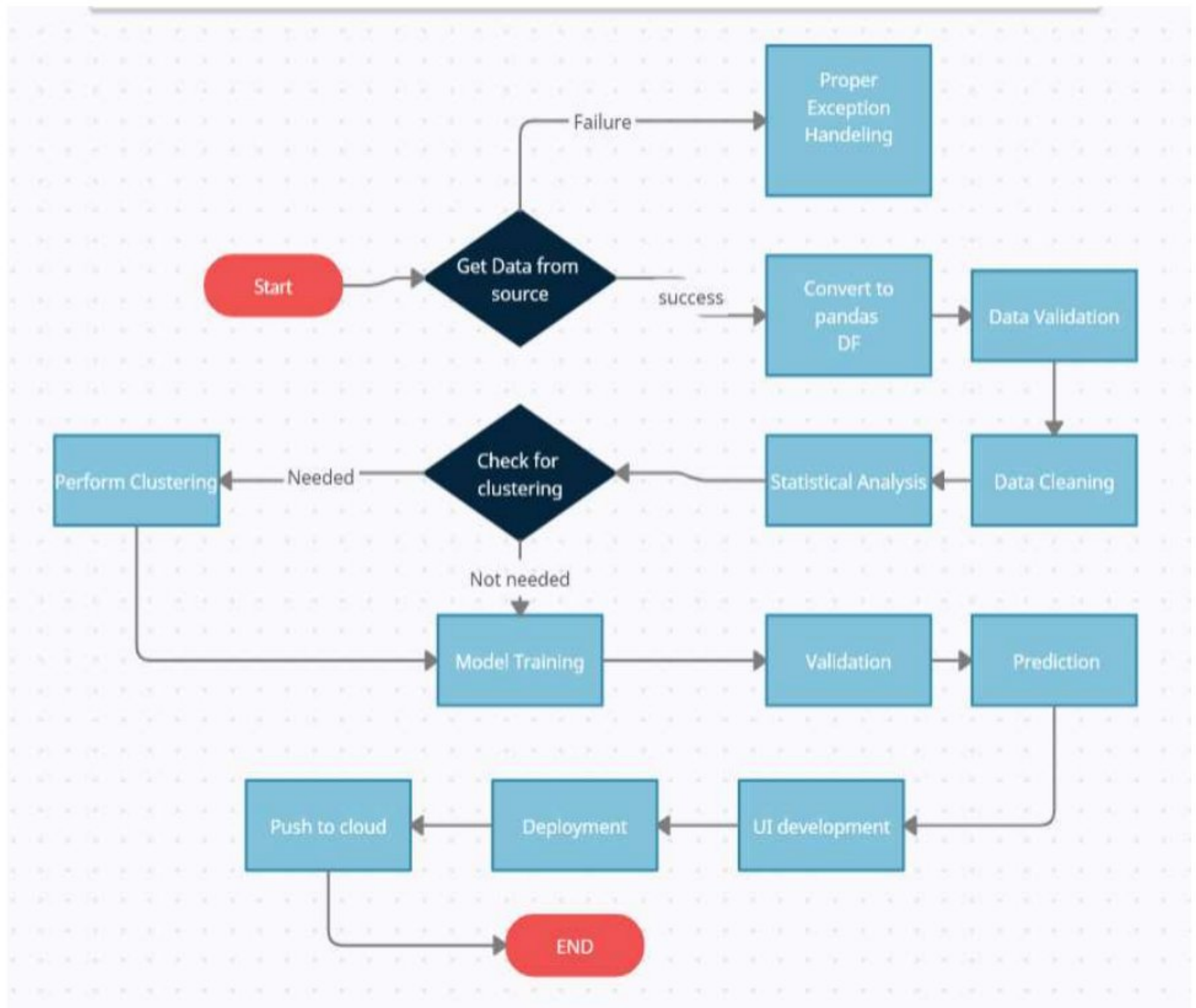
We will be deploying the model to AWS, Heroku ,Azure on these 3 cloud platforms

4.13 Web flow

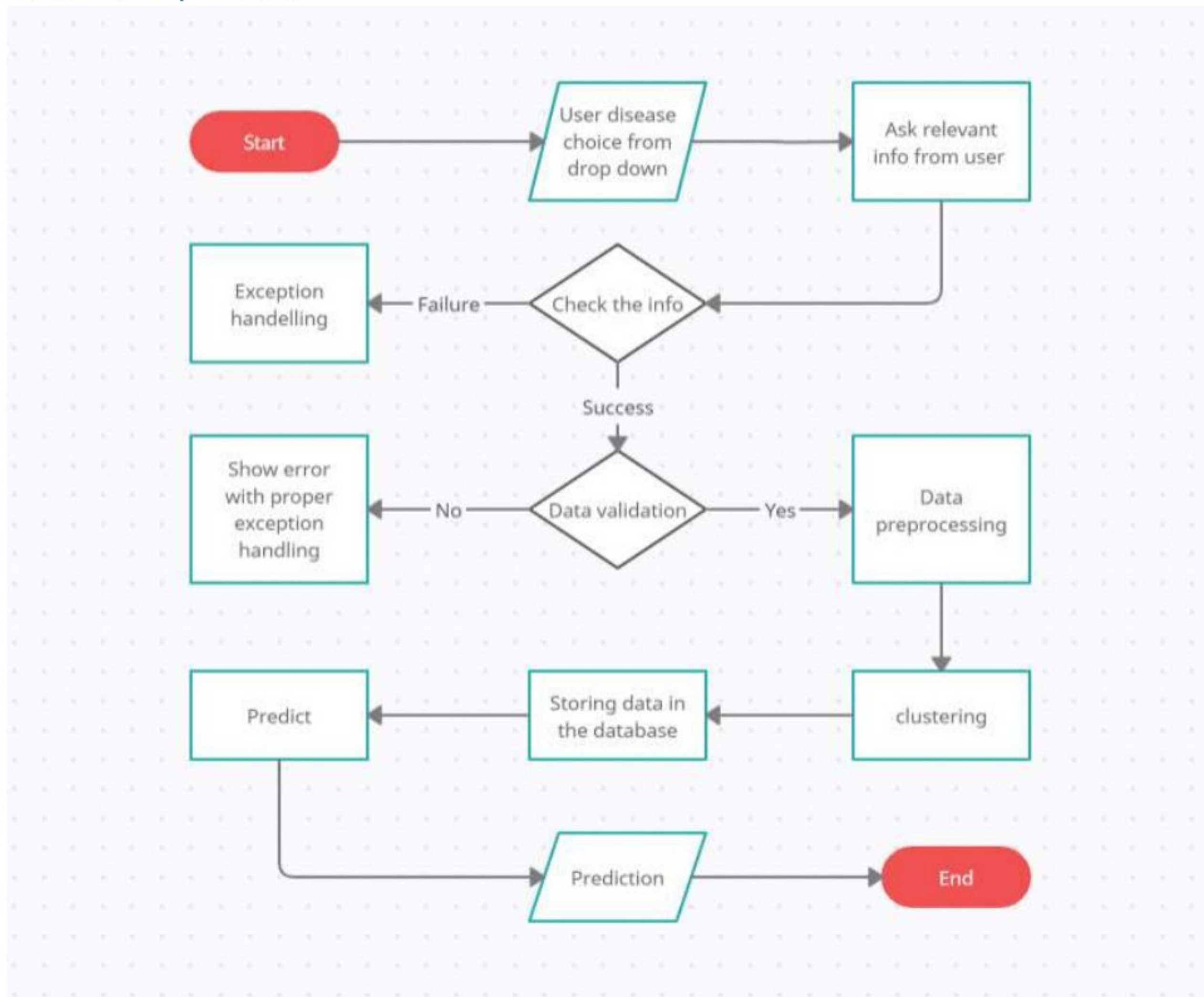
We have used Flask framework for this project so here is the Web Flow based Architecture of this flask project. We have integrated flight fare prediction model in this flask project



4.14 Model Training/Validation WorkFlow



4.15 User I/O Flow



User Interface of Website

Main Page

The screenshot displays a web browser window with the URL `127.0.0.1:5000`. The browser's address bar and tabs are visible at the top. The website, titled "POS ARILINES" with the tagline "Fly over the Sky!", features a "POS FLIGHT PRICE PREDICTION" section. A prominent green button labeled "Calculate your average flight fare !" is centered at the top of the form. Below this, the form is organized into six input fields arranged in a 3x2 grid:

- Departure Date:** A date picker field showing "dd-mm-yyyy --:--".
- Arrival Date:** A date picker field showing "dd-mm-yyyy --:--".
- Source:** A dropdown menu with "Delhi" selected.
- Destination:** A dropdown menu with "Cochin" selected.
- Stopage:** A dropdown menu with "Non-Stop" selected.
- Select Airline:** A dropdown menu with "Jet Airways" selected.

A dark blue "Calculate Fare" button is positioned at the bottom center of the form. The footer of the page includes the text "Pankaj Jaiswal <|> Onkar Bhatuse <|> Siddhiraj Koliwankar" and "© 2021 POS ARILINES". The browser's taskbar at the bottom shows the Windows search bar, system tray icons, and the date/time "15:03 04-09-2021".

Main Page Main Page with filled Information

POS Flight Price Prediction

127.0.0.1:5000

Departure Date

04-09-2021 15:03

Arrival Date

04-09-2021 15:03

Source

Kolkata

Destination

Hyderabad

Stopage

1

Select Airline

Vistara

Activate Windows

Type here to search

30°C Rain

1504

04-09-2021

Main Page After Predicting average Flight Fare

POS ARILINES
Fly over the Sky!

POS FLIGHT PRICE PREDICTION

Your average flight fare will be Rs. 4897.14

Departure Date
dd-mm-yyyy --:--

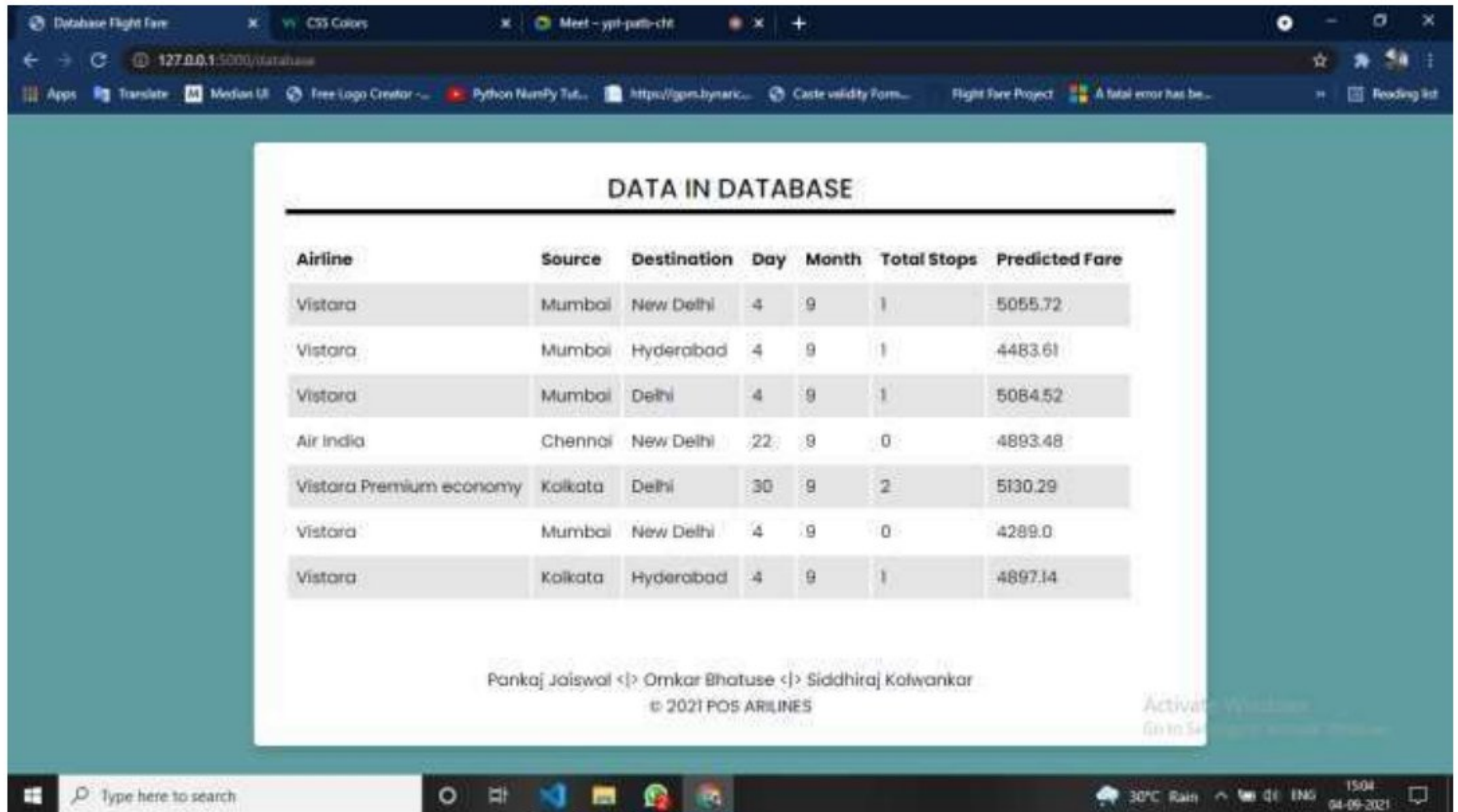
Arrival Date
dd-mm-yyyy --:--

Source

Destination

Windows taskbar: Type here to search, 30°C Rain, 15:04, 04-09-2021

Data in a Database



The screenshot shows a web browser window with a single tab titled 'Database Flight Fare'. The address bar displays '127.0.0.1:5000/database'. The browser's taskbar at the bottom shows various open applications including 'CSS Colors', 'Meet - ypr-path-ctf', and 'Flight Fare Project'. The main content area features a table titled 'DATA IN DATABASE' with the following data:

Airline	Source	Destination	Day	Month	Total Stops	Predicted Fare
Vistara	Mumbai	New Delhi	4	9	1	5055.72
Vistara	Mumbai	Hyderabad	4	9	1	4483.61
Vistara	Mumbai	Delhi	4	9	1	5084.52
Air India	Chennai	New Delhi	22	9	0	4893.48
Vistara Premium economy	Kolkata	Delhi	30	9	2	5130.29
Vistara	Mumbai	New Delhi	4	9	0	4289.0
Vistara	Kolkata	Hyderabad	4	9	1	4897.14

Below the table, the text 'Pankaj Jaiswal <|> Omkar Bhatuse <|> Siddhiraj Kolwankar' and '© 2021 POS AIRLINES' is visible. A Windows watermark 'Activate Windows Go to Settings to activate Windows' is present in the bottom right corner of the browser window.

5 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 Response time of url from backend model.	1. Application is accessible	The latency and accessibility of application is very faster we got in aws ec2 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 Predict Flight Price button to submit the inputs	1. Application is accessible 2. User is logged in to the application	User should get Submit button 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
Verify whether user has options to	1. Application is accessible	User should have options to filter

filter the recommended results as well		the recommended results as well
	3. User is logged in to the application	
Verify whether the KPIs indicate details Of the correct inputs	1. Application is accessible 2. User is logged in to the application	The KPIs should indicate details of the suggested inputs to users.