Used Car Price Prediction System

A MINI-PROJECT REPORT

Submitted by

Nandeeshwaran P–2116220701179

in partial fulfilment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

AUTONOMOUS, CHENNAI

NOV/DEC, 2024

BONAFIDE CERTIFICATE

Certified that this mini project "**Used Car Price Prediction System**"is the bonafide work of" **Nandeeshwaran P (2116220701179)"**who carried out the project work under my supervision.

SIGNATURE

Mrs. JANANEE V,

Assistant Professor,

Computer Science & Engineering

Rajalakshmi Engineering College

Thandalam, Chennai -602105.

Submitted for the End semester practical examination to be held on ____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I express my sincere thanks to my beloved and honourable chairman **MR.S.MEGANATHAN** and the chairperson **DR.M.THANGAM MEGANATHAN** for their timely support and encouragement.

I am greatly indebted to my respected and honourable principal **Dr. S.N.MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by my head of the department **Dr. P. KUMAR**, and my Academic Head **Dr.SABITHA**, for being ever supporting force during my project work.

I also extend my sincere and hearty thanks to my internal guide **Mrs. JANANEE V** for her valuable guidance and motivation during the completion of this project.

My sincere thanks to my family members, friends and other staff members of Computer Science and Engineering

Nandeeshwaran P 2116220701179

ABSTRACT

The price of a new car in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But, due to the increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and it's value in the present day scenario. In fact, seller also has no idea about the car's existing value or the price he should be selling the car at. To overcome this problem we have developed a model which will be highly effective. Machine learning Algorithms are used because they provide us with continuous value as an output and not a categorized value. Because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been developed which acquires input from any user and displays the Price of a car according to user's inputs.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

The used car market has seen significant growth as more people seek affordable vehicles. However, determining the fair price of a used car can be challenging, given the various factors that influence its value, such as brand, model, mileage, year, fuel type, and overall condition. This project aims to address this challenge by developing a machine learning model that accurately predicts the price of a used car based on historical data. By providing accurate price estimates, the model will help buyers make informed decisions, ensuring they get value for their money while purchasing used cars.

1.2 SCOPE OF THE WORK

The project covers the entire machine learning pipeline, from data collection and preprocessing to model development and deployment. It involves exploring the data, applying feature engineering, training various models, and selecting the best-performing one for deployment. The final model is integrated into a web application, allowing users to input car details and receive a price estimate instantly. The scope of the work also includes the deployment of this web application on a cloud platform, making it accessible to a wide range of users. The solution is aimed at used car buyers, sellers, and dealerships, offering them an easy-to-use tool for pricing used vehicles.

1.3 AIM AND OBJECTIVES OF THE PROJECT

The primary aim of the project is to develop a robust machine learning model for predicting the prices of used cars, taking into account various features. The project seeks to build a solution that can provide accurate and reliable predictions, accessible through a user-friendly web interface. The ultimate goal is to create a tool that can assist consumers in making data-driven decisions when purchasing used cars.

Objectives:

1. **Data Collection and Cleaning: t**o gather and preprocess data, ensuring it is clean, consistent, and ready for model training.

2. **Model Selection and Training :** To train and evaluate multiple machine learning models (including Decision Trees, Random Forest, and Gradient Boosting) and choose the best one based on evaluation metrics.

3. **Hyperparameter Tuning:**To optimize model performance by tuning hyperparameters using methods like GridSearchCV.

4. **Deployment and Scalability:**To deploy the app on a cloud platform (AWS Elastic Beanstalk), ensuring scalability and reliability for real-time usage

CHAPTER 2

SYSTEMSPECIFICATIONS

1. HARDWARE SPECIFICATIONS

Processor : Pentium IV OrHigher

Memory Size : 128 GB(Minimum)

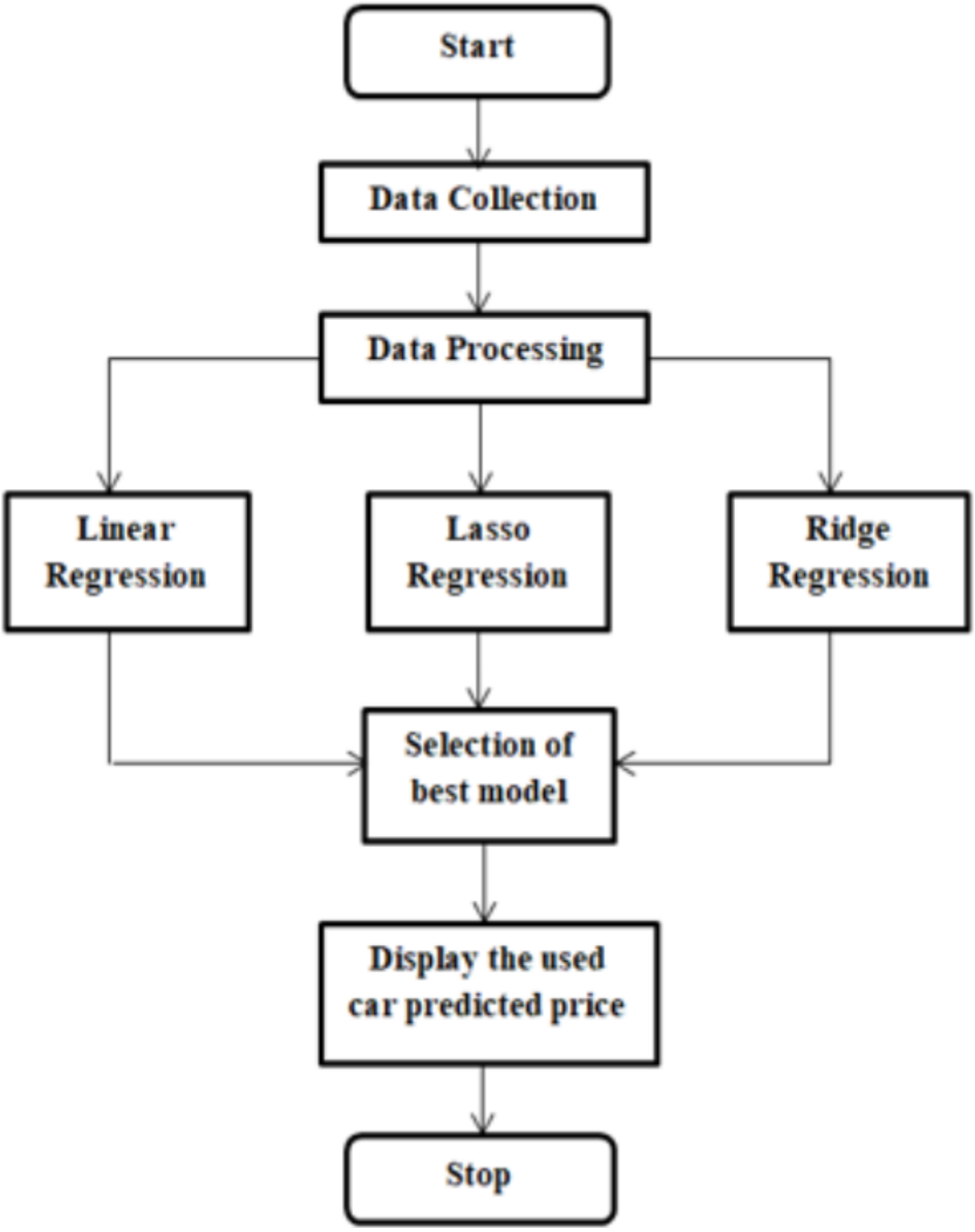
HDD :40 GB (Minimum)

2. SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 7 AND PLUS
Front – End	:	HTML, CSS,
Back – End	:	MYSQL

CHAPTER 3

ARCHITECTURE DIAGRAM



CHAPTER 4

MODULE DESCRIPTION

4.1. User Registration and Login Module:

The User Registration and Login Module for an Electricity Bill Management System plays a critical role in ensuring secure and smooth access for users. During registration, users provide essential information such as their name, email, phone number, address, and password. The system validates these inputs, ensuring that the email format is correct, the password is strong, and the email or phone number is unique. Upon successful registration, users receive a confirmation via email or SMS to verify their identity, ensuring that only verified users can log in.

4.2. Feature Selection and Engineering Module:

The **Feature Selection and Engineering Module** is designed to identify and construct the most relevant features for the prediction task. Based on exploratory data analysis (EDA) and correlation studies, features such as **Brand**, **Model**, **Fuel Type**, and **Kilometers Driven** are considered important. In this module, redundant or irrelevant features, like the "New_Price" column, are removed due to missing values, and continuous features are log-transformed for normal distribution. New features are also created, such as the interaction between **Mileage** and **Kilometers Driven**, which is highly correlated with car prices. This module focuses on improving the model’s accuracy by using only the most important and informative features.

4.3. User Input and Data Preprocessing Module:

The **User Input and Data Preprocessing Module** is the first step in the Used Car Price Prediction system. This module gathers user inputs, including car details such as the brand, model, year of manufacture, kilometers driven, fuel type, transmission type, and ownership history. Once the data is received, it undergoes preprocessing, where features like categorical data (e.g., brand, location) are encoded, and numerical data (e.g., kilometers driven, engine capacity) are transformed. Key preprocessing techniques include **target encoding** for high-cardinality features like brand and location, **log transformation** of continuous features, and creating interaction terms between related features, such as kilometers driven and mileage. This module ensures the input data is clean, structured, and ready for prediction.

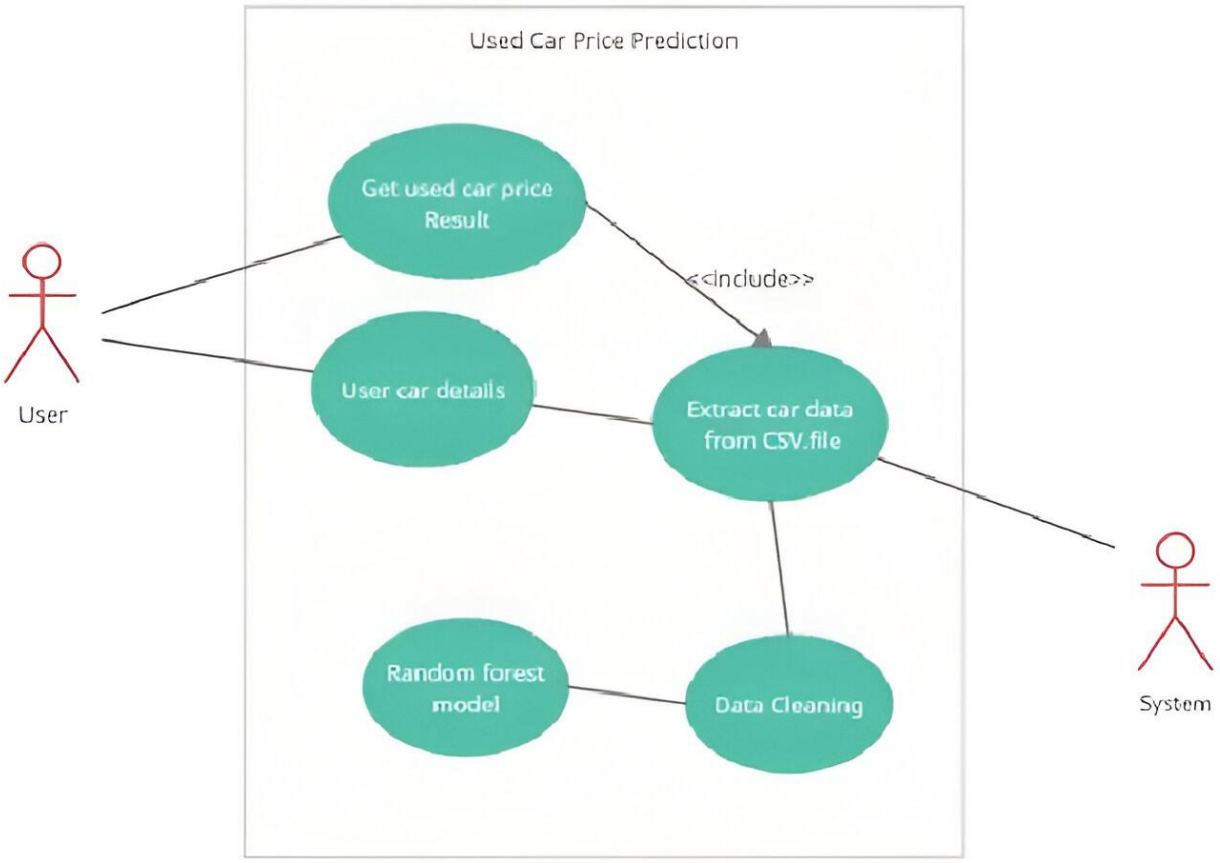
4.4. Price Prediction Module:

The **Price Prediction Module** is the user-facing module of the system. It takes the inputs provided by the user—such as car brand, model, year, kilometers driven, fuel type, and transmission—and processes them using the trained machine learning model to generate a price prediction. The module applies the same feature transformations used during training, including scaling and encoding, to the input data before feeding it to the model. Once the prediction is generated, it is displayed to the user through a web interface. This module enables real-time price predictions, providing users with an accurate estimate of the value of a used car based on current market data.

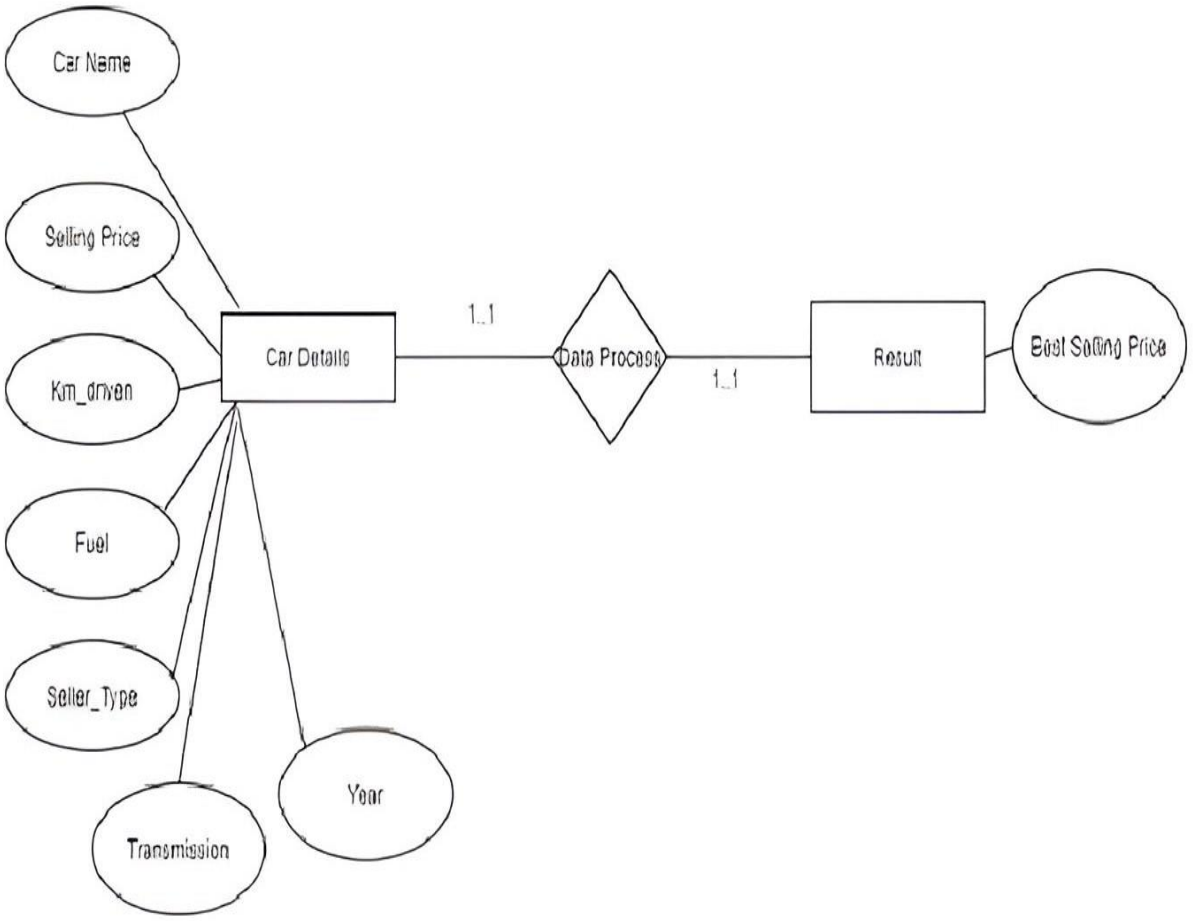
CHAPTER 5

SYSTEM DESIGN

5.1 USE CASE DIAGRAM

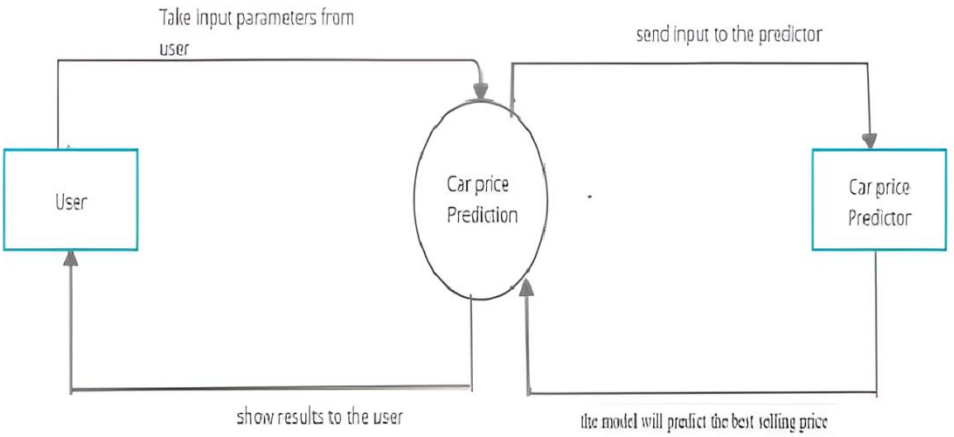


5.2 ER DIAGRAM

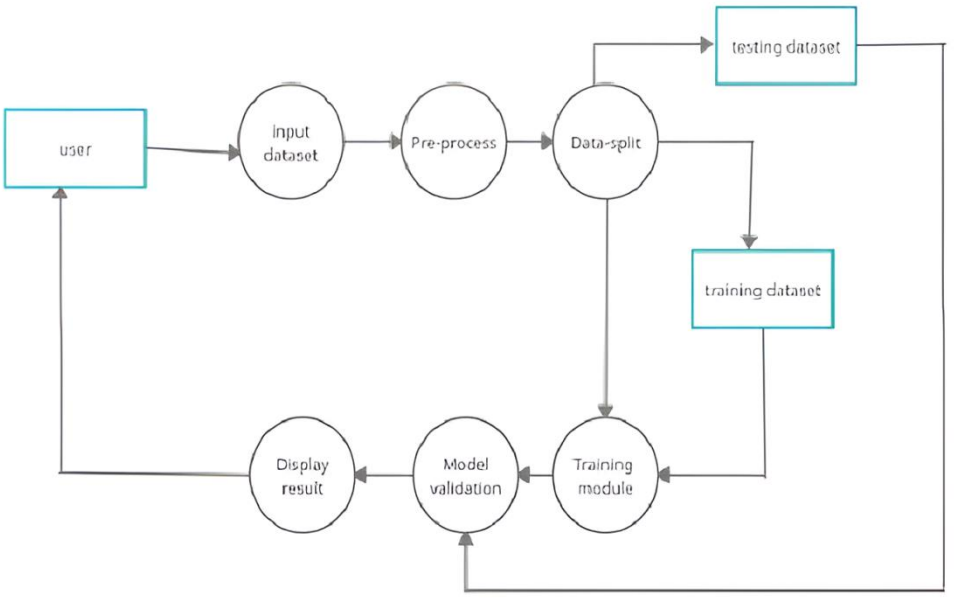


5.3. DFD DIAGRAM

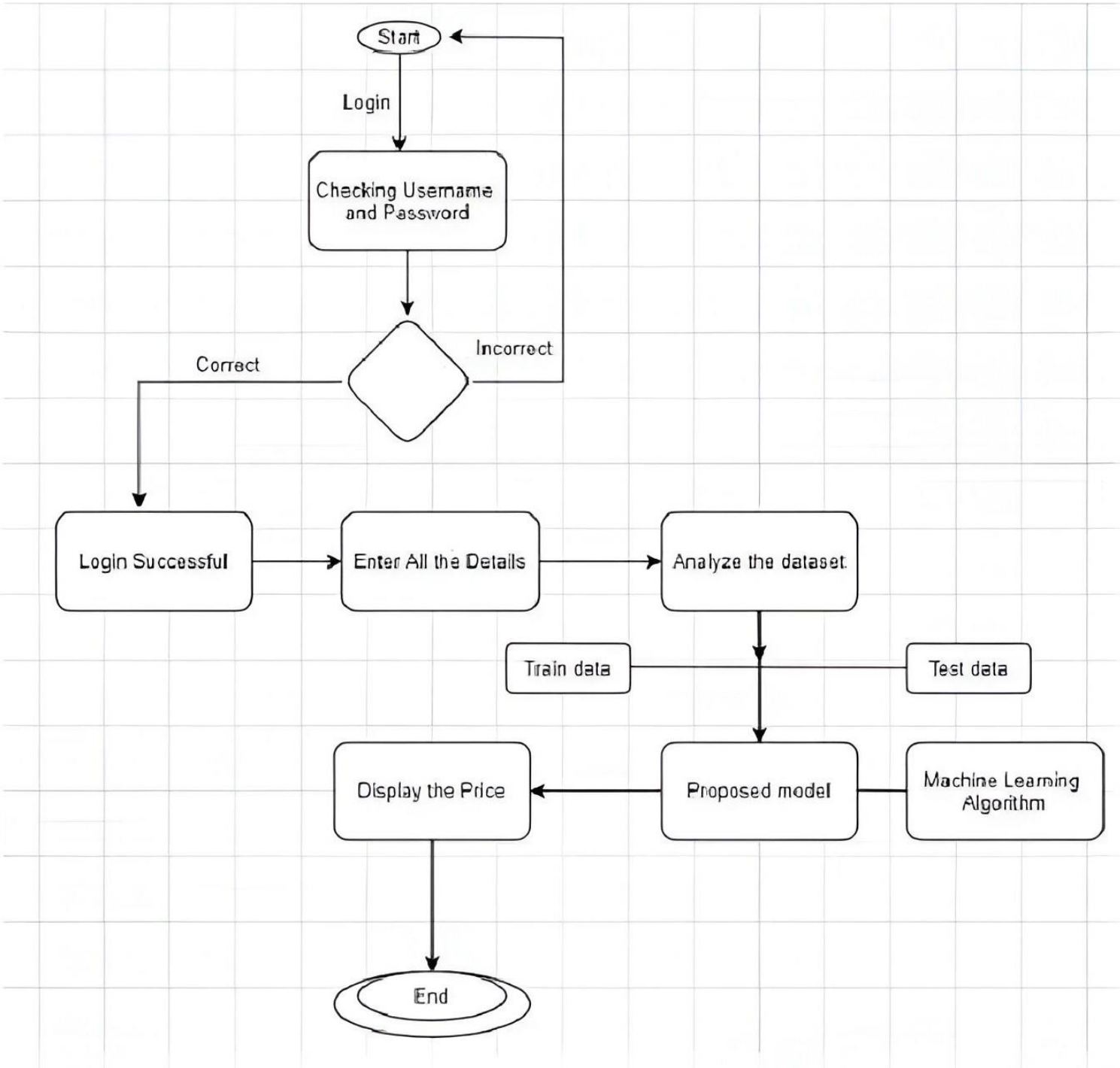
Data flow diagram Level-0



Data Flow diagram Level-1



5.4. ACTIVITY DIAGRAM



CHAPTER 6

SAMPLE CODING

```
#!/usr/bin/env python
from flask import Flask, render_template, flash, request, jsonify, Markup
import logging, io, os, sys
import pandas as pd
import numpy as np
from modules.custom_transformers import *
#from sklearn.ensemble import GradientBoostingRegressor
import scipy
import pickle
import mysql.connector
## eb cli init
#>../aws-elastic-beanstalk-cli-setup/scripts/bundled_installer
#>echo 'export PATH="/home/skumar/.ebcli-virtual-env/executables:$PATH"' >>
~/.bash_profile && source ~/.bash_profile

#Freeing up port with Port no $Port_Number
#sudo fuser -k $Port_Number/tcp

# EB looks for an 'application' callable by default.
application = Flask(__name__)
application.secret_key = 'your_secret_key'
np.set_printoptions(precision=2)

#Model features
gbm_model = None
features = ['Brand', 'Model', 'Location', 'Year', 'Kilometers_Driven',
        'Fuel_Type', 'Transmission', 'Owner_Type', 'Mileage', 'Engine',
        'Power', 'Seats']

@application.before_first_request
def startup():

    global gbm_model, model2brand

    # gbm model
    with open('static/GBM_Regressor_pipeline.pkl', 'rb') as f:
        gbm_model = pickle.load(f)

        # min, max, default values to categories mapping dictionary
    with open('static/Dictionaries.pkl', 'rb') as f:
```

```
        default_dict,min_dict, max_dict, default_dict_mapped = pickle.load(f)

    # Encoded values to categories mapping dictionary
    with open('static/Encoded_dicts.pkl', 'rb') as f:
        le_brands_Encdict,le_models_Encdict,le_locations_Encdict,le_fuel_types_E
ncdict,le_transmissions_Encdict,le_owner_types_Encdict = pickle.load(f)

    with open('static/model2brand.pkl', 'rb') as f:
        model2brand = pickle.load(f)

@application.errorhandler(500)
def server_error(e):
    logging.exception('some eror')
    return """
    And internal error <pre>{}</pre>
    """.format(e), 500

@application.route("/", methods=['POST', 'GET'])
def login():
     # Encoded values to categories mapping dictionary
      # Encoded values to categories mapping dictionary
    return render_template('navbar.html')

@application.route("/about", methods=['POST', 'GET'])
def about():
     # Encoded values to categories mapping dictionary
      # Encoded values to categories mapping dictionary
    return render_template('aboutus.html')

@application.route("/logintry", methods=['POST', 'GET'])
def logintry():
     # Encoded values to categories mapping dictionary
      # Encoded values to categories mapping dictionary
    return render_template('login.html')

@application.route("/index", methods=['POST', 'GET'])
def index():
     # Encoded values to categories mapping dictionary
      # Encoded values to categories mapping dictionary
    with open('static/Encoded_dicts.pkl', 'rb') as f:
        le_brands_Encdict,le_models_Encdict,le_locations_Encdict,le_fuel_types_E
ncdict,le_transmissions_Encdict,le_owner_types_Encdict = pickle.load(f)

            return render_template( 'index.html', model2brand =
model2brand,le_models_Encdict = le_models_Encdict,le_locations_Encdict =
```

```
le_locations_Encdict, le_fuel_types_Encdict = le_fuel_types_Encdict, le_transmissions_Encdict = le_transmissions_Encdict, le_owner_types_Encdict = le_owner_types_Encdict, le_brands_Encdict = le_brands_Encdict,price_prediction = 17.09)

@application.route('/submit', methods=['GET', 'POST'])
def handle_submit():
    if request.method == 'POST':
        # Get the form data
        email = request.form['loginemail']
        password = request.form['loginpassword']
        mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="nandy_car"
        )
        mycursor = mydb.cursor()
        sql = "SELECT * FROM user WHERE email = %s AND password = %s"
        adr = (email, password)
        mycursor.execute(sql, adr)
        myresult = mycursor.fetchall()
        if myresult:
            with open('static/Encoded_dicts.pkl', 'rb') as f:
                le_brands_Encdict,le_models_Encdict,le_locations_Encdict,le_fuel_typ
es_Encdict,le_transmissions_Encdict,le_owner_types_Encdict = pickle.load(f)

                            return render_template( 'index.html', model2brand =
model2brand,le_models_Encdict = le_models_Encdict,le_locations_Encdict =
le_locations_Encdict, le_fuel_types_Encdict = le_fuel_types_Encdict,
le_transmissions_Encdict = le_transmissions_Encdict, le_owner_types_Encdict =
le_owner_types_Encdict, le_brands_Encdict =
le_brands_Encdict,price_prediction = 17.09)
        else:
            print("wrong?")
              flash("Invalid username or password. Please try again.")  # Display an
error message
            return render_template('login.html') # Redirect back to the login page

@application.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        print(request.form)  # Check form data

        signemail = request.form.get('signemail')
```

```
        signpassword = request.form.get('signpassword')
        signuser = request.form.get('signname')
        signphone = request.form.get('signphone')

        # Validate form fields
        if not (signemail and signpassword and signuser and signphone):
            flash("All fields are required!")
            return render_template('register.html')  # Ensure this page exists

        mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="nandy_car"
        )
        mycursor = mydb.cursor()

        # Check if the user already exists
        sql_check = "SELECT * FROM user WHERE email = %s"
        mycursor.execute(sql_check, (signemail,))
        existing_user = mycursor.fetchall()
        if existing_user:
            flash("Email already exists. Please use a different email.")
            return render_template('login.html')  # Redirect back to register

        # Insert new user
        sql = "INSERT INTO `user` (`user`, `phone`, `email`, `password`) VALUES
(%s, %s, %s, %s);"
        adr = (signuser, signphone, signemail, signpassword)

        try:
            mycursor.execute(sql, adr)
            mydb.commit()  # Commit the transaction
            flash("Registration successful!")  # Flash a success message
            return redirect('/index')  # Redirect to index after successful registration
        except Exception as e:
            mydb.rollback()  # Roll back in case of error
            flash("An error occurred: {}".format(e))  # Flash error message
            return render_template('login.html')  # Redirect back to registration form
    else:
        return render_template('login.html')  # GET request for registration page


# accepts either deafult values or user inputs and outputs prediction
@application.route('/background_process', methods=['POST', 'GET'])
```

```
def background_process():
    Brand = request.args.get('Brand')
    Model = request.args.get('Model')
    Location = request.args.get('Location')
    Year = int(request.args.get('Year'))
    Kilometers_Driven = float(request.args.get('Kilometers_Driven'))
    Fuel_Type = request.args.get('Fuel_Type')
    Transmission = request.args.get('Transmission')
    Owner_Type = request.args.get('Owner_Type')
    Mileage = float(request.args.get('Mileage'))
    Engine = float(request.args.get('Engine'))
    Power = float(request.args.get('Power'))
    Seats = float(request.args.get('Seats'))

    # values stroed in list later to be passed as df while prediction
    user_vals = [Brand, Model, Location, Year, Kilometers_Driven,
        Fuel_Type, Transmission, Owner_Type, Mileage, Engine,
        Power, Seats]

    x_test_tmp = pd.DataFrame([user_vals],columns = features)
    float_formatter = "{:.2f}".format

    pred = float_formatter(np.exp(gbm_model.predict(x_test_tmp[features])[0]))
    return jsonify({'price_prediction':pred})

# when running app locally
if __name__ == '__main__':
    application.debug = False
    application.run(host='0.0.0.0')
```

CHAPTER 7

SCREEN SHOTS

Fig. 7.1. Home

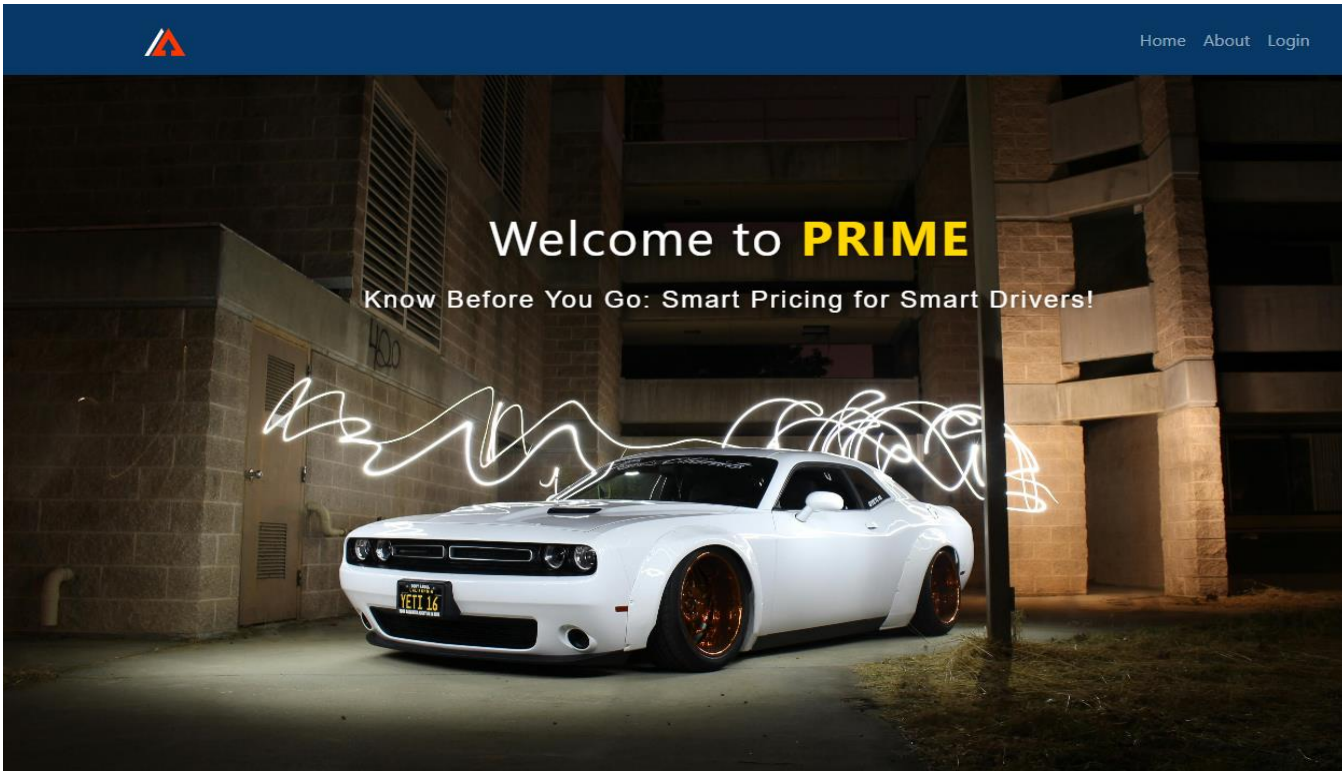


Fig.7.2. Login

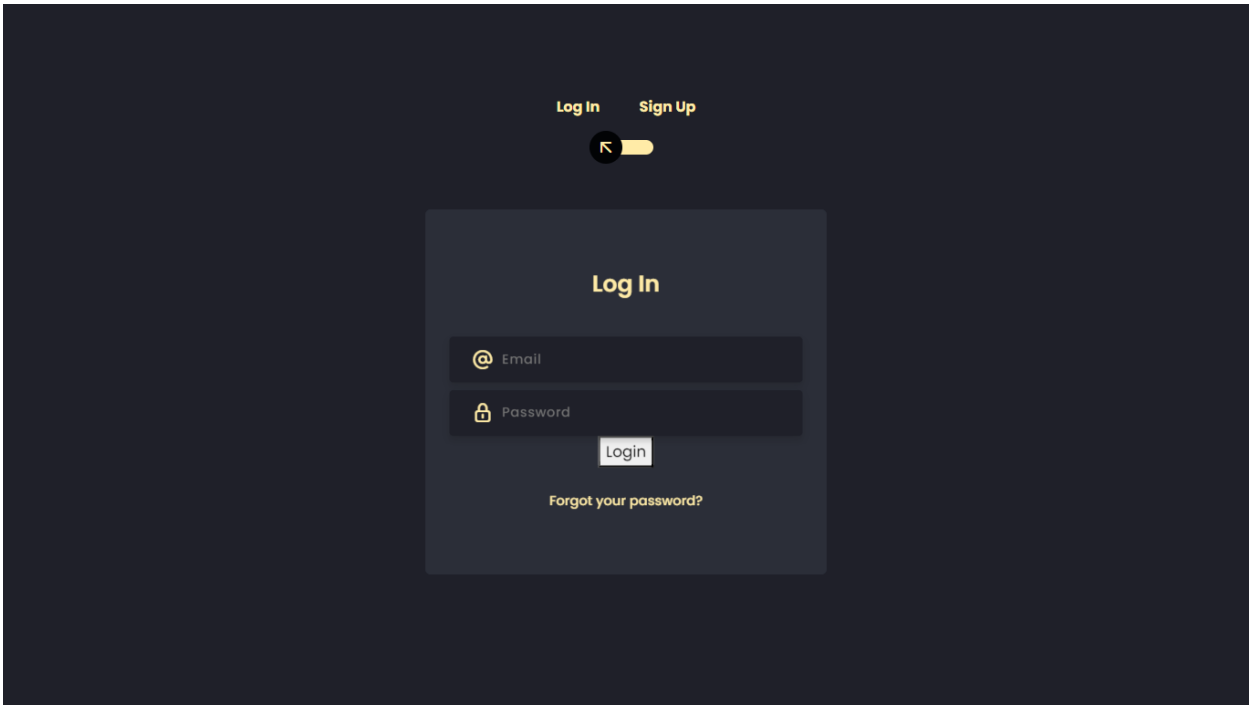


Fig.7.3. Sign Up

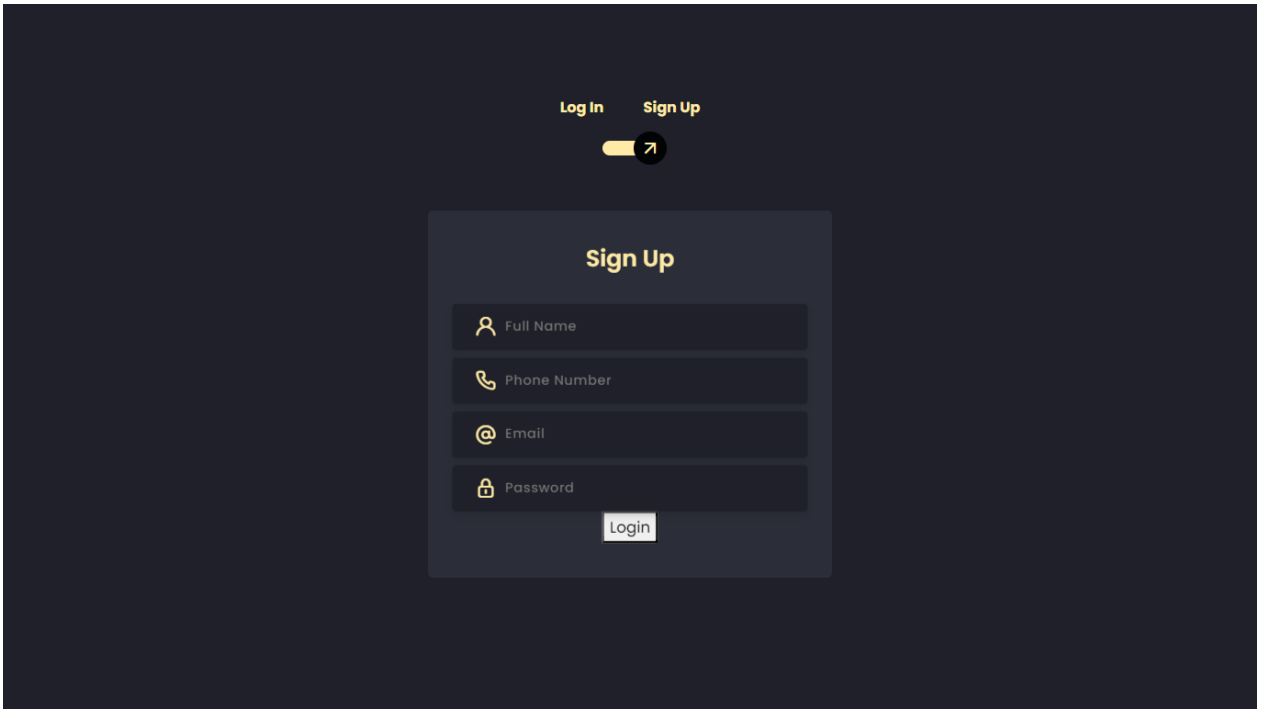
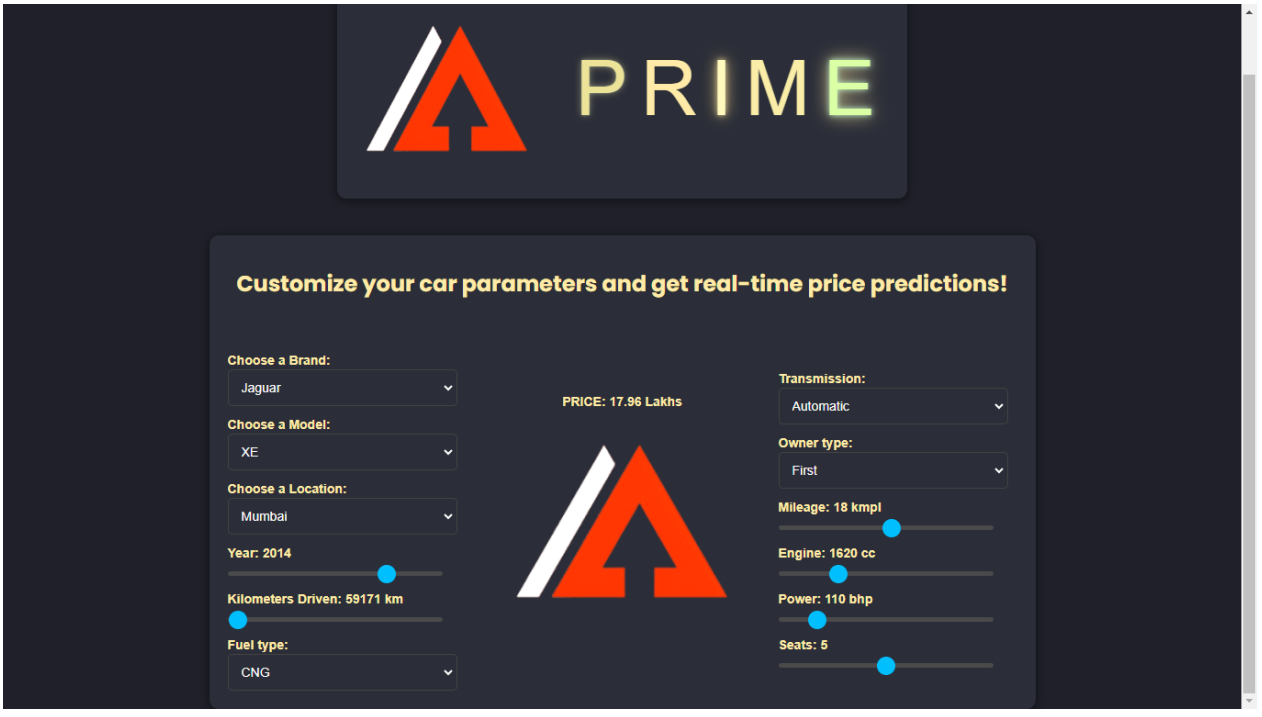


Fig. 7.3 Dashboard



CHAPTER 8

CONCLUSION

In conclusion, the Used Car Price Prediction project has successfully developed a machine learning-based solution that can predict the price of a used car with high accuracy. By leveraging various machine learning techniques and deploying the model on a cloud platform, this project demonstrates the practical application of data science in solving real-world problems. The solution is highly scalable and can be accessed by users through a user-friendly web interface, making it a valuable tool for used car buyers, sellers, and dealerships.

This project has highlighted the importance of data preprocessing, feature engineering, and hyperparameter tuning in building effective machine learning models. Future improvements could include expanding the dataset, refining the model with additional features, and enhancing the web interface for a better user experience. Overall, this project showcases the potential of machine learning in the automotive industry, particularly in enabling data-driven decisions in the used car market.

REFERENCES

1. HTML , CSS , JS – www.w3schools.com
2. MYSQL – www.youtube.com
3. Product Details– www.amazon.in
4. Carousel Slider – www.glidejs.com
5. Font Awesome Icons – www.fontawesome.com
6. PHP Mailer - https://github.com/PHPMailer/PHPMailer
7. SweetAlert2 - https://sweetalert2.github.io/v10.html