"AutoValuator: Car Price Prediction Model"

A Minor Project Report Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya

Towards Partial Fulfillment for the Award of

Bachelor of Technology in Computer Science and Engineering

Submitted by:

Priyanshi Goyal (0827CS221204)

Rahul Sharma (0827CS221217)

Rachit Shivhare (0827CS221213)

Pranay Jain (0827CS221195)

Guided by

Prof. Krupi Saraf

Computer Science and Engineering

Acropolis Institute of Technology & Research, Indore

Jan-June 2025

EXAMINER APPROVAL

The Minor Project entitled ***"AutoValuator: Car Price Prediction Model"*** submitted by **Priyanshi Goyal** (0827CS221204)**, Rahul Sharma**(0827CS221217)**, Rachit Shivhare** (0827CS221213)**, Pranay Jain** (0827CS221195) has been examined and is hereby approved towards partial fulfillment for the award of ***Bachelor of Technology in Computer Science and Engineering*** discipline, for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner) **(External Examiner)**

Date: **Date:**

RECOMMENDATION

This is to certify that the work embodied in this minor project entitled "***AutoValuator: Car Price Prediction Model***" submitted by **Priyanshi Goyal** (0827CS221204)**, Rahul Sharma**(0827CS221217)**, Rachit Shivhare** (0827CS221213)**, Pranay Jain** (0827CS221195) is a satisfactory account of the bonafide work done under the supervision of ***Prof. Krupi Saraf,*** is recommended towards partial fulfillment for the award of the Bachelor of Technology (Computer Science Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal.

(Project Guide)

(Project Coordinator)

(Dean Academics)

STUDENTS UNDERTAKING

This is to certify that the minor project entitled ***"AutoValuator: Car Price Prediction Model*** has developed by us under the supervision of ***Prof. Krupi Saraf***. The whole responsibility of the work done in this project is ours. The sole intension of this work is only for practical learning and research.

We further declare that to the best of our knowledge; this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work found then we are liable for explanation to this.

Priyanshi Goyal (0827CS221204)

Rahul Sharma (0827CS221217)

Rachit Shivhare (0827CS221213)

Pranay Jain (0827CS221195)

Acknowledgement

We thank the almighty Lord for giving me the strength and courage to sail out through the tough and reach on shore safely.

There are number of people without whom this project would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentor **Prof. Krupi Saraf**, AITR, Indore for her motivation, sagacious guidance, constant encouragement, vigilant supervision, and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Prof. Krupi Saraf**, AITR, Indore for his support, suggestion, and inspiration for carrying out this project. I am very much thankful to other faculty and staff members of the department for providing me all support, help and advice during the project. We would be failing in our duty if do not acknowledge the support and guidance received from **Prof. Narendra Pal**, AITR, Indore whenever needed. We take opportunity to convey my regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing me all necessary facilities for project to achieve our objectives.

We are grateful to **our parent** and **family members** who have always loved and supported us unconditionally. To all of them, we want to say “Thank you”, for being the best family that one could ever have and without whom none of this would have been possible.

Priyanshi Goyal (0827CS221204) Rahul Sharma (0827CS221217)
Rachit Shivhare (0827CS221213) Pranay Jain (0827CS221195)

Executive Summary

AutoValuator: Car Price Prediction Model

Submitted to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (MP), India, for the partial fulfillment of a Bachelor of Technology in Computer Science and Engineering, under the guidance of Prof. Krupi Saraf.

Car Price Predictor is a smart web application that predicts the resale value of a car based on various input features like brand, year of purchase, fuel type, kilometers driven, transmission type, etc. Traditionally, users need to browse multiple sites or consult experts to estimate used car prices. This system simplifies that process using Machine Learning.
The application uses a modern tech stack with a React front end and a Python-Flask backend, integrated with a trained ML model. It is helpful for individuals, car dealers, and online resale platforms by providing accurate, instant price predictions. The system reduces manual effort, brings consistency, and enhances decision-making in the used car market.

Keywords: Car Price Prediction, Machine Learning, Smart Valuation, Web Application

“If your vision is for a year,

plant seeds.

If your vision is for a decade,

plant trees.

If your vision is for a lifetime,

nurture people.”

- Oriental Saying

List of Figures

List of Tables

List of Abbreviations

Abbr1: ML – Machine Learning

Abbr2: AI – Artificial Intelligence

Abbr3: UI – User Interface

Abbr4: CSV- Comma Separated Values

Abbr5: RMSE – Root Mean Square Error

Abbr6: API – Application Programming Interface

Abbr7: MAE – Mean Absolute Error

Abbr8: MSE – Mean Squared Error

Abbr9: R^2 – Coefficient Of Determination

Abbr10: RF – Random Forest

Abbr11: LR – Linear Regression

Abbr12: GB– Gradient Boosting

Table of Contents

Chapter 1. Introduction

AutoValuator is a smart and easy-to-use website that helps users predict the price of a car based on its details like brand, model, year, fuel type, kilometers driven, and more. Many people face confusion when buying or selling a car because they don't know its correct market value. Our project solves this problem using machine learning. By entering basic information about the car, users can get an estimated price within seconds. This saves time, removes guesswork, and helps users make better decisions. AutoValuator aims to make car buying and selling easier, faster, and more trustworthy for everyone.

1.1 Overview

AutoValuator: Car Price Prediction Model is a machine learning-based solution designed to estimate the selling price of a car based on various features like brand, model, year of purchase, fuel type, transmission, and kilometers driven. The aim of this project is to assist buyers and sellers in making informed decisions by providing accurate price predictions, reducing market uncertainty and negotiation gaps.

This project uses a dataset of used cars to train the model, allowing it to learn the patterns and relationships between car attributes and their prices. By deploying the model through a user-friendly interface, users can input car details and get instant price estimates. The overall goal is to bring transparency, convenience, and efficiency into the car resale market through data-driven technology.

1.2 Background and Motivation

The second-hand automobile market is expanding rapidly, but determining the correct price of a used car remains a challenge for both buyers and sellers. Traditional methods rely on guesswork or dealer opinion, which often leads to inaccurate pricing and lack of trust.

The motivation behind **AutoValuator** was to develop a smart and reliable system that can predict car prices using machine learning. By analyzing historical data and understanding the impact of different car features on pricing, the project aims to bring accuracy, fairness, and confidence to the used car pricing process.

1.3 Problem Statement and Objectives

Problem Statement:

In the used car market, price estimation is often subjective and varies widely due to a lack of standardized methods. Buyers fear overpaying, and sellers worry about underselling. This creates a trust gap and inefficiency in the market.

Objective:

The main objective of **AutoValuator** is to build a machine learning model that accurately predicts the resale value of a car based on its features. The goal is to assist users in making data-driven pricing decisions, ensuring fairness, reducing negotiation time, and enhancing transparency in the buying/selling process.

1.4 Scope of the Project

The **AutoValuator** project focuses on predicting the prices of used cars based on specific input features like brand, year, fuel type, transmission, kilometres driven, and more. It is designed for users including individual buyers, sellers, and car dealers who want accurate price estimations.

The scope includes:

- Collecting and preprocessing a relevant car dataset.
- Training a machine learning model to predict car prices.
- Evaluating the model for accuracy and reliability.
- Deploying the model through a simple user interface.

In future, the model can be expanded to include more parameters (like insurance, location, car condition), and even integrate with online resale platforms.

1.5 Team Organization

The **AutoValuator** project was developed by a team of four dedicated members, each contributing to different aspects of the project based on their strengths and interests. The responsibilities were divided as follows:

- **Priyanshi Goyal:** Responsible for designing and implementing the user interface (UI) of the application. Ensured that the frontend is interactive, user-friendly, and provides an intuitive experience for car price predictions.
- **Rahul Sharma:** Focused on front-end development, ensuring that the web application is responsive and aesthetically appealing. Worked on integrating the frontend with backend services for smooth data exchange.
- **Rachit Shivhare:** Handled backend development, building the API services for data retrieval and model interaction. Additionally, worked on training and testing the machine learning model to predict car prices accurately based on input features.
- **Pranay Jain:** Contributed to data preprocessing and model development, ensuring the machine learning model performs optimally. Worked on integrating the model into the backend system and handling server-side logic.

1.6 Report Structure

The **Autovaluator: Car Price Prediction Model** is a machine learning-based system designed to predict the price of a used car based on various factors such as brand, model, year, and mileage. This report outlines the process of building the model, its implementation, and the outcomes of this project.

Chapter 1: Introduction – This chapter provides an overview of the **AutoValuator: Car Price Prediction Model** project, including its significance, goals, and the problem it addresses. It explains the motivation behind the project and outlines the objectives that the project aims to

achieve. The chapter also introduces the basic concepts related to car price prediction and the technologies used.

Chapter 2: Review of Literature – This chapter reviews existing research and literature relevant to the project. It discusses previous work in the domain of car price prediction, machine learning applications in pricing models, and how other systems approach price estimation. The review highlights the gaps in existing methods, which the **Autovaluator** model aims to fill.

Chapter 3: Proposed System – This chapter outlines the design and architecture of the proposed system. It explains the machine learning algorithms and methodologies chosen for the price prediction model. The system's workflow, features, and how it addresses the problem statement are discussed in detail. This chapter also includes the technologies used (such as programming languages, frameworks, and libraries) and the expected outcome of the system.

Chapter 4: Implementation – In this chapter, the actual implementation of the **AutoValuator** system is described. It covers the steps taken to build both the frontend and backend of the application. Key aspects of model training, data preprocessing, integration of the machine learning model with the web interface, and any challenges faced during development are discussed here. Code snippets, diagrams, and other relevant details may be included to illustrate the implementation process.

Chapter 5: Conclusion – This chapter summarizes the key findings and results of the project. It discusses how well the system meets the objectives outlined earlier and any potential areas for improvement. The limitations of the model, future work, and how the system can be enhanced or scaled further are also covered. The conclusion provides a final assessment of the project's success and its impact on the used car market.

Chapter 2. Review of Literature

The review of literature presents an overview of the various studies and techniques that have been employed in the field of car price prediction and machine learning. Many researchers have explored different methodologies such as regression analysis, decision trees, and neural networks to estimate the prices of used cars. These approaches have their strengths and limitations, which are critical in understanding the evolution of price prediction models. This chapter will delve into these studies, evaluate their findings, and identify the gaps that the **AutoValuator** model aims to address, ultimately providing a foundation for the proposed system.

2.1 Preliminary Investigation

2.1.1 Current System

This survey explores various research studies on second-hand car price prediction using machine learning techniques. Each study utilizes different models, datasets, and preprocessing techniques to improve accuracy. The following sections summarize five key research papers relevant to the AutoValuator project.

1. **Second-hand car price prediction based on Multiple Linear Regression And Random Forest**

- **Author**: Jiaying Gao
- **ML Techniques Used**: Multiple Linear Regression , Random Forest
- **Dataset**: Kaggle: Old Car Dataset (Updated by Milan Vaddoriya).
- **Overview**

  This study compares the performance of Multiple Linear Regression and Random Forest models for predicting second-hand car prices. The dataset used was preprocessed by handling missing values, removing outliers, and selecting relevant features. The study found that while MLR provided basic insights, it struggled with non-linearity in data, making it less effective for real-world applications. In contrast, Random Forest performed significantly better as it could handle complex relationships between variables, leading to more accurate predictions

2. Predicting the price of used cars using machine learning-based Regression model

- **Authors**: G. SelvaKumar, S. Sruthi, M. Surya, A. Tamilselvi, R. Kavya
- **ML Techniques Used**: Linear Regression, Decision Tree, Random Forest, XGBoost
- **Dataset**: Cardekho Used Car dataset from Kaggle
- **Overview**

This research examines multiple regression models for second-hand car price prediction. The study uses various preprocessing techniques such as data cleaning, feature selection, and outlier removal. Decision Tree and Random Forest showed better performance than Linear Regression, but XGBoost emerged as the most accurate model due to its ability to handle non-linear relationships. The research emphasizes the importance of choosing the right regression model, with XGBoost being the most effective due to its boosting mechanism that reduces errors.

3. Vehicle Price Prediction System using Machine Learning Techniques

- **Authors**: Kafeel Noor, Shahbaz Jan
- **ML Techniques Used**: Multiple Linear Regression
- **Dataset**: Data collected from pakweheels.com (Pakistan's largest used car website)
- **Overview**

This study focuses on predicting car prices using Multiple Linear Regression. The dataset was scraped from PakWheels.com and underwent preprocessing to remove duplicate records and missing values. The model achieved 98% accuracy, showing that MLR can work effectively when the dataset is well-structured. However, the study was conducted on Pakistani market data, which may limit its applicability to other regions. The research underlines that while MLR can yield high accuracy, it may not generalize well across different datasets and vehicle markets.

4. Prediction of Used Car Prices Using Machine Learning Techniques Based on Vehicle Characteristics and Details

- **Authors**: Enis Gegic, Becir Isakovic, Dino Keco, Darko Kreso, Dijana Asceric
- **ML Techniques Used**: Support Vector Machine (SVM), Random Forest (RF), Artificial Neural Network (ANN)
- **Dataset**: Scraped from https://autopijac.rs/ (Bosnian car marketplace)

- **Overview**

This paper evaluates different machine learning techniques for car price prediction, comparing SVM, Random Forest, and Artificial Neural Networks. The dataset was scraped from an online marketplace and underwent preprocessing to remove inconsistencies. Random Forest emerged as the best-performing model due to its ability to handle high-dimensional data and complex relationships. ANN showed promising results but required substantial computational power. The study suggests that Random Forest is a reliable choice for real-world applications, while ANN can be explored for further advancements.

5. **Used Cars Price Prediction using Supervised Learning Techniques**

- **Authors**: V Pattabiraman, M Ganesh
- **ML Techniques Used**: Linear Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest (RF)
- **Dataset**: Kaggle dataset sourced from Craigslist ("Used Cars Price Prediction")
- **Overview**

This study investigates the effectiveness of different supervised learning models for predicting used car prices. Data preprocessing steps included handling missing values, scaling features, and feature selection. Among the tested models, Random Forest provided the highest accuracy making it suitable for real-world application.

K-Nearest Neighbors (KNN) performed well but required extensive fine-tuning for optimal results. SVM was effective but computationally expensive. The research highlights that Random Forest remains the best-performing model, while KNN and SVM serve as alternative approaches with potential for optimization.

2.2 Limitations of Current System

The major limitations of the existing platforms include:

- **Data Inconsistency and Quality**: Many models rely on incomplete, outdated, or inconsistent data. The accuracy of predictions is often compromised due to gaps in the dataset or errors in data collection.
- **Limited Feature Consideration:** Current systems may fail to consider all relevant factors that influence car pricing, such as regional pricing differences, economic conditions, or changes in demand and supply.

- **Model Overfitting:** Some machine learning models tend to overfit the training data, leading to poor generalization to new, unseen data. This results in predictions that may be inaccurate when applied to real-world scenarios.
- **Lack of Real-Time Updates:** Most models are static and do not incorporate real-time market trends, making them less responsive to sudden shifts in the market, such as price fluctuations caused by new trends or economic factors.
- **Complexity and Usability:** Many existing systems are complex, requiring technical expertise to operate or are not user-friendly, limiting their accessibility for average users, such as individual buyers and sellers.

2.3 Requirement Identification and Analysis for Project

In the **Autovaluator** project, the primary goal is to develop a machine learning model that accurately predicts the price of used cars. To achieve this, a thorough identification and analysis of both functional and non-functional requirements were essential.

Functional Requirements:

1. **Data Collection and Preprocessing**: The system needs to gather a large dataset with relevant car features such as make, model, year, mileage, fuel type, and others. Proper data preprocessing, including handling missing values, encoding categorical features, and scaling numerical values, is crucial.
2. **Model Development**: The core functionality of the system is the development of a machine learning model capable of predicting the car's price based on the input features. The model should be trained, tested, and evaluated to ensure high accuracy.

3. **Frontend Interface**: A user-friendly interface where users can input car details and receive the predicted price. The interface must be simple, responsive, and intuitive for users with varying technical expertise.
4. **Backend Integration**: The backend system must handle user requests, interact with the machine learning model, and send predictions to the frontend. It should be capable of processing requests in real-time.

Non-Functional Requirements:

1. **Performance**: The system must provide predictions quickly, ensuring minimal response time when users input car details.
2. **Scalability**: The system should be able to handle an increasing number of users and more extensive datasets as the project scales.
3. **Security**: User data should be securely handled, and the system should implement necessary measures to prevent unauthorized access.
4. **Usability**: The system should be easy to use for both technical and non-technical users, with an intuitive interface and minimal learning curve.
5. **Accuracy**: The machine learning model must provide highly accurate predictions to ensure the system's reliability.

2.3.1 Conclusion

This reviewes the existing literature on car price prediction models, highlighting the methodologies and technologies that have been used to tackle this problem. While significant progress has been made using techniques like regression analysis, decision trees, and neural networks, several limitations remain, such as data inconsistency, limited feature sets, overfitting, and complexity. These challenges offer valuable insights into the improvements needed for more accurate and user-friendly models. The **Autovaluator** model aims to address these gaps by leveraging a more comprehensive approach to data collection, model selection, and user interface design. The next chapters will detail the proposed system and its implementation, building on the foundation laid by existing research.

Chapter 3. Proposed System

3.1 The Proposal

The **AutoValuator: Car Price Prediction Model** proposes an innovative approach to estimating the price of used cars through machine learning algorithms. The system aims to provide accurate, real-time car price predictions based on several key features such as brand, model, year of manufacture, mileage, fuel type, and others. By leveraging modern machine learning techniques and a user-friendly interface, the system will enable users to make informed decisions when buying or selling used cars.

The proposed system consists of two primary components: the **frontend**, where users can input car details, and the **backend**, which processes the data and returns the predicted price using a trained machine learning model. The integration of these components ensures seamless interaction between the user and the predictive model.

3.2 Benefits of the Proposed System

The **AutoValuator: Car Price Prediction Model** offers several key benefits over existing car pricing systems:

1. **Accuracy and Reliability**: By using advanced machine learning algorithms, the system provides more accurate car price predictions based on various features, minimizing errors compared to traditional manual methods.
2. **Real-Time Predictions**: The model offers real-time price predictions, allowing users to make quick decisions when buying or selling a car, which is crucial in the fast-paced used car market.
3. **User-Friendly Interface**: The system is designed to be intuitive and easy to use, requiring minimal technical knowledge. Users can simply input relevant car details and receive a predicted price within seconds.
4. **Comprehensive Dataset**: The model leverages a diverse dataset that takes into account a wide range of factors, such as car make, model, year, mileage, and fuel type, resulting in more accurate and comprehensive predictions.
5. **Scalability**: The system is built to scale, meaning it can handle a growing number of users and larger datasets without compromising performance.

6. **Cost and Time Efficiency**: By automating the price prediction process, the system saves time and resources for both buyers and sellers, eliminating the need for manual price comparisons and evaluations.
7. **Data-Driven Insights**: The system provides data-driven insights, allowing users to better understand the factors influencing car prices and make more informed decisions.
8. **Market Relevance**: The model is built to adapt to changes in the market, ensuring that the predictions reflect current trends and price fluctuations, which many traditional methods fail to do.

3.3 Block Diagram

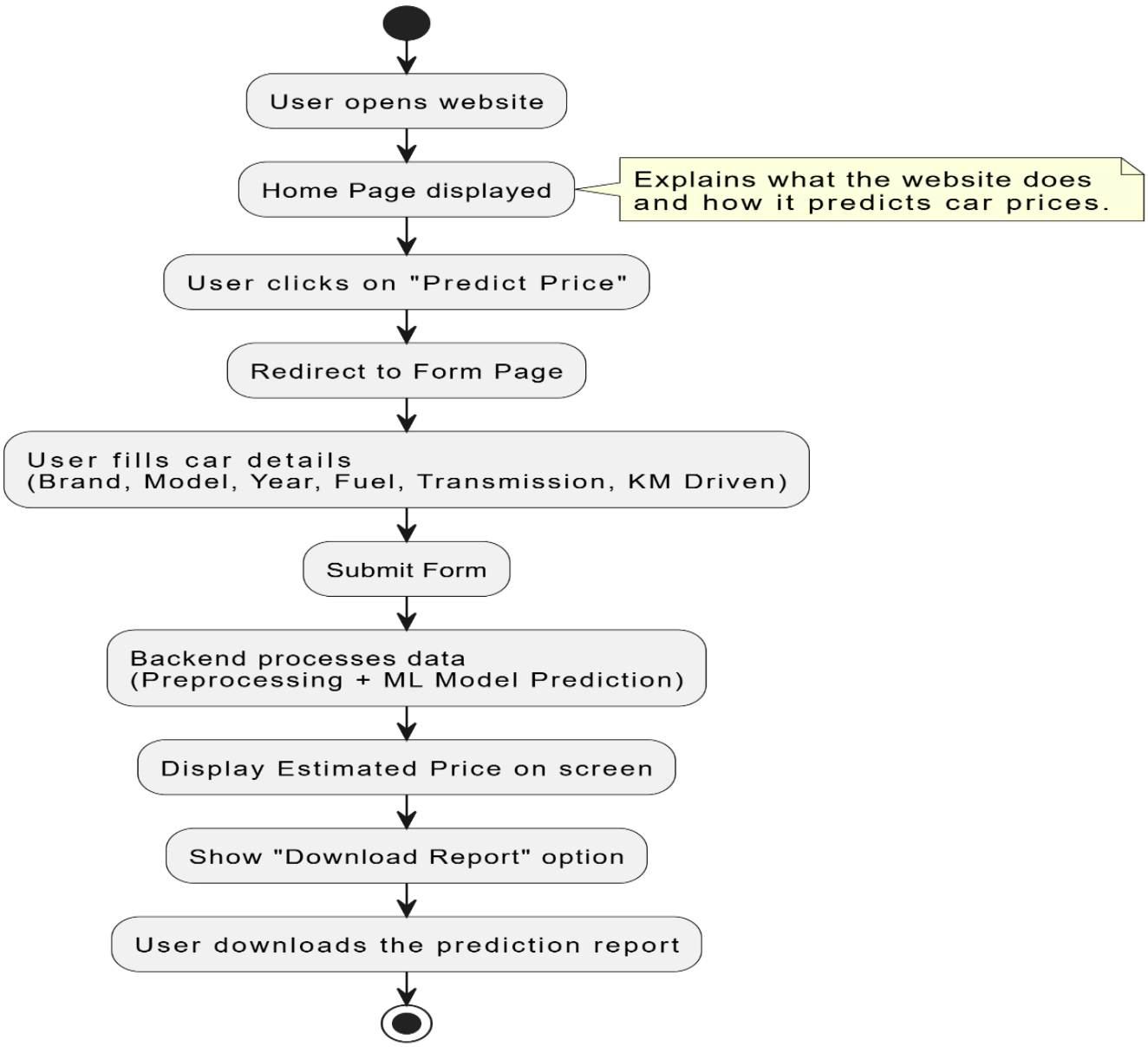


Figure 3.1: Block Diagram

3.4 Feasibility Study

A feasibility study evaluates the practicality and potential of a system by analyzing its technical, economic, and operational factors. For *AutoValuator*, this study ensures that the system is viable, effective, and worth implementing.

3.4.1 Technical Feasibility

This aspect examines whether the system can be developed using the current technology and tools available.

1. **Frontend Development:**

   The frontend is built using **React.js**, a popular JavaScript framework known for creating dynamic and responsive user interfaces. React's component-based architecture allows for efficient development and smooth interaction between the user and the system.

Figure 3.2 – Frontend Languages

2. **Backend Development:**
   - The backend is powered by **Python**, which provide a robust and scalable platform for handling API requests, managing user interactions, and integrating the machine learning model. These technologies are commonly used for building fast and efficient backend services..

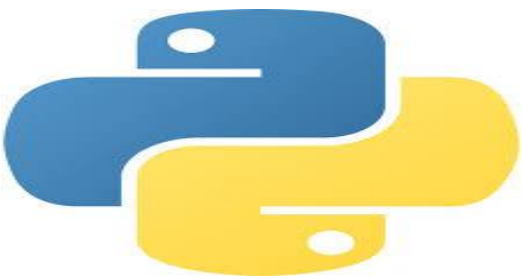


Figure 3.3 – Backend Languages

3. Machine Learning Algorithms:

The project uses well-established machine learning algorithms such as regression models, decision trees, and random forests, all of which are widely supported by Python libraries like scikit-learn and TensorFlow. These algorithms are known for their accuracy and scalability in prediction tasks.

- Linear Regression
- Support Vector Regression (SVR)
- Decision Tree Regressor
- Extra Trees Regressor
- Random Forest Regressor (Final choice)

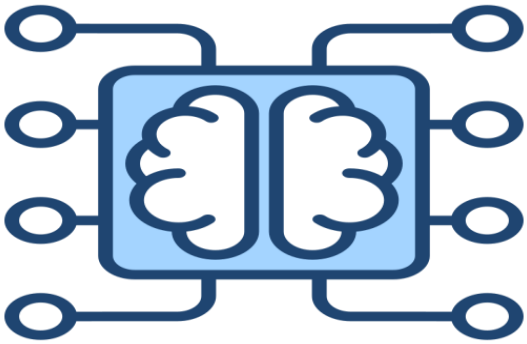


Figure 3.4 – Machine Learning

4. Data Preprocessing and Feature Engineering

Several preprocessing techniques were applied to clean and prepare the data for training:

- Unnecessary columns such as max_power, car_name, seller_type, and Unnamed: 0 were dropped.
- Column names were renamed for better readability.
- The Engine-Capacity values were converted from cc to liters.
- Missing or incorrect values (e.g., cars with 0 seats) were fixed by replacing them with appropriate values.
- Duplicate entries were removed to avoid data redundancy.
- One-hot encoding was applied to categorical features like brand, fuel type, and transmission to convert them into numeric format

Feature Selection: To determine which features were most important in predicting the car price, an ExtraTreesRegressor model was used. This model provided feature importance scores, and a bar chart was plotted to visualize the top influencing features.

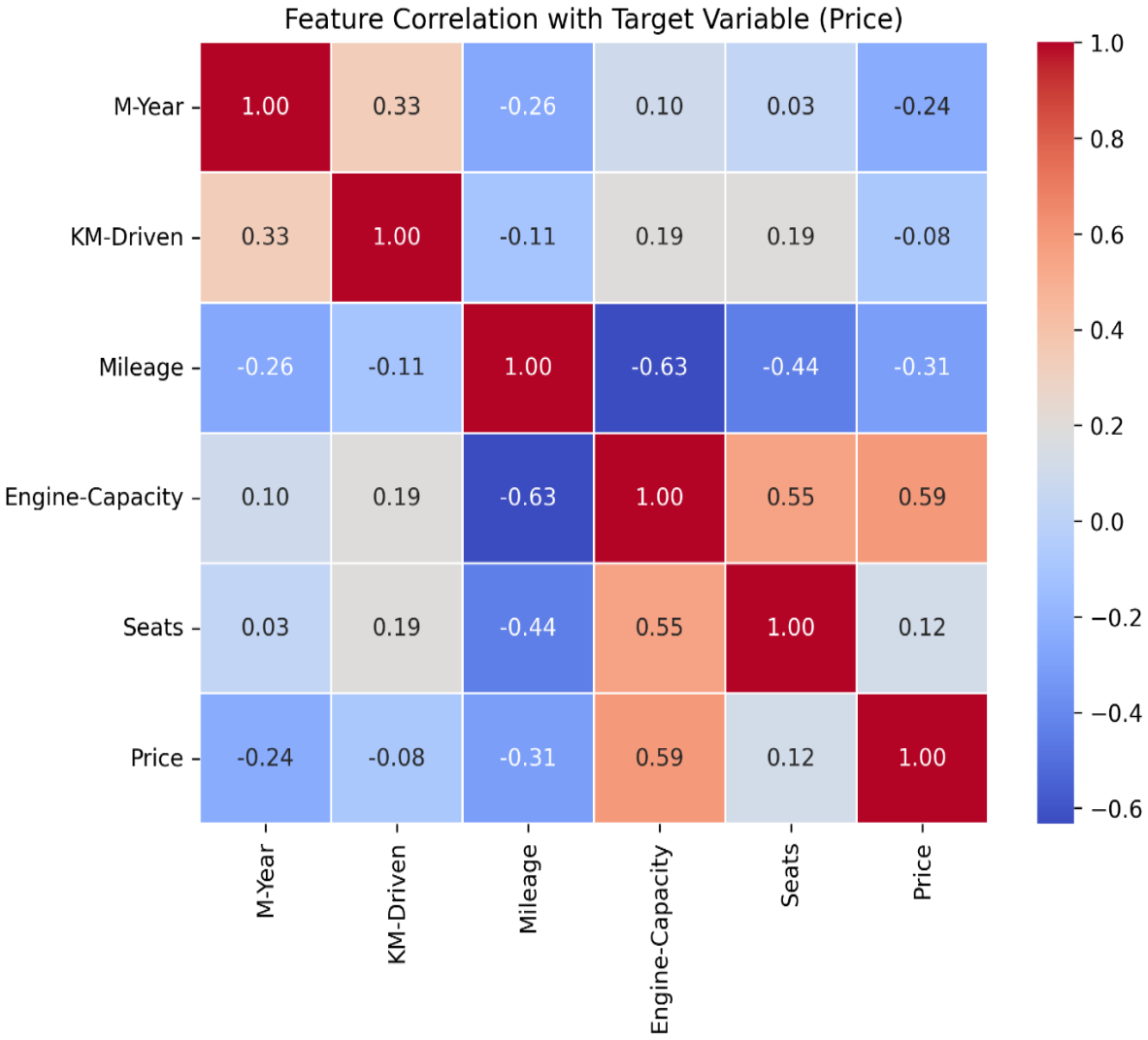


Fig 3.5: Feature Corelation

5. **Deployment:**

   The system can be deployed on cloud platforms such as **AWS**, **Heroku**, or **Google Cloud**, which offer scalable resources and machine learning support, ensuring that the model can handle a growing number of users and requests.

   With these technologies, the system will meet the technical requirements for successful development and operation.

3.4.2 Economic Feasibility

This aspect determines whether the system is cost-effective and affordable.

- **Low Initial Investment**: The project uses free, open-source technologies like React.js, Node.js, and scikit-learn, minimizing development costs without the need for expensive software.
- **Affordable Hosting:** Cloud platforms like Heroku or AWS offer low-cost hosting options, with pay-as-you-go models that keep the initial deployment cost minimal.
- **Revenue Generation:** Potential revenue can be earned through premium features, advertisements, and subscription models for users or businesses needing frequent car price predictions
- **Scalability:** The system is designed to scale efficiently without significant increases in cost, making it adaptable to growing user traffic and data.
- **Cost Efficiency:** The automated car price prediction reduces manual effort and time, saving both users and businesses valuable resources.

3.4.3 Operational Feasibility

This aspect assesses whether the system can operate smoothly and fulfill its intended purpose.

- **Ease of Use:** The system offers a user-friendly interface where users can easily input car details and get instant price predictions without needing any technical knowledge.
- **Smooth Workflow:** From data input to prediction, every step is streamlined, ensuring a smooth and efficient user experience for both buyers and sellers.
- **Maintainability:** The system is built with modular and scalable code, making it easy to maintain, update, or enhance as user requirements evolve.
- **Integration Ready:** It can be easily integrated into existing car dealer platforms or online marketplaces, enhancing its practical use in real-world environments.
- **User Satisfaction:** By providing fast, accurate, and reliable price predictions, the system ensures user satisfaction and encourages repeat usage.

1.5 Design Representation

Frontend Design Representation of our project is:

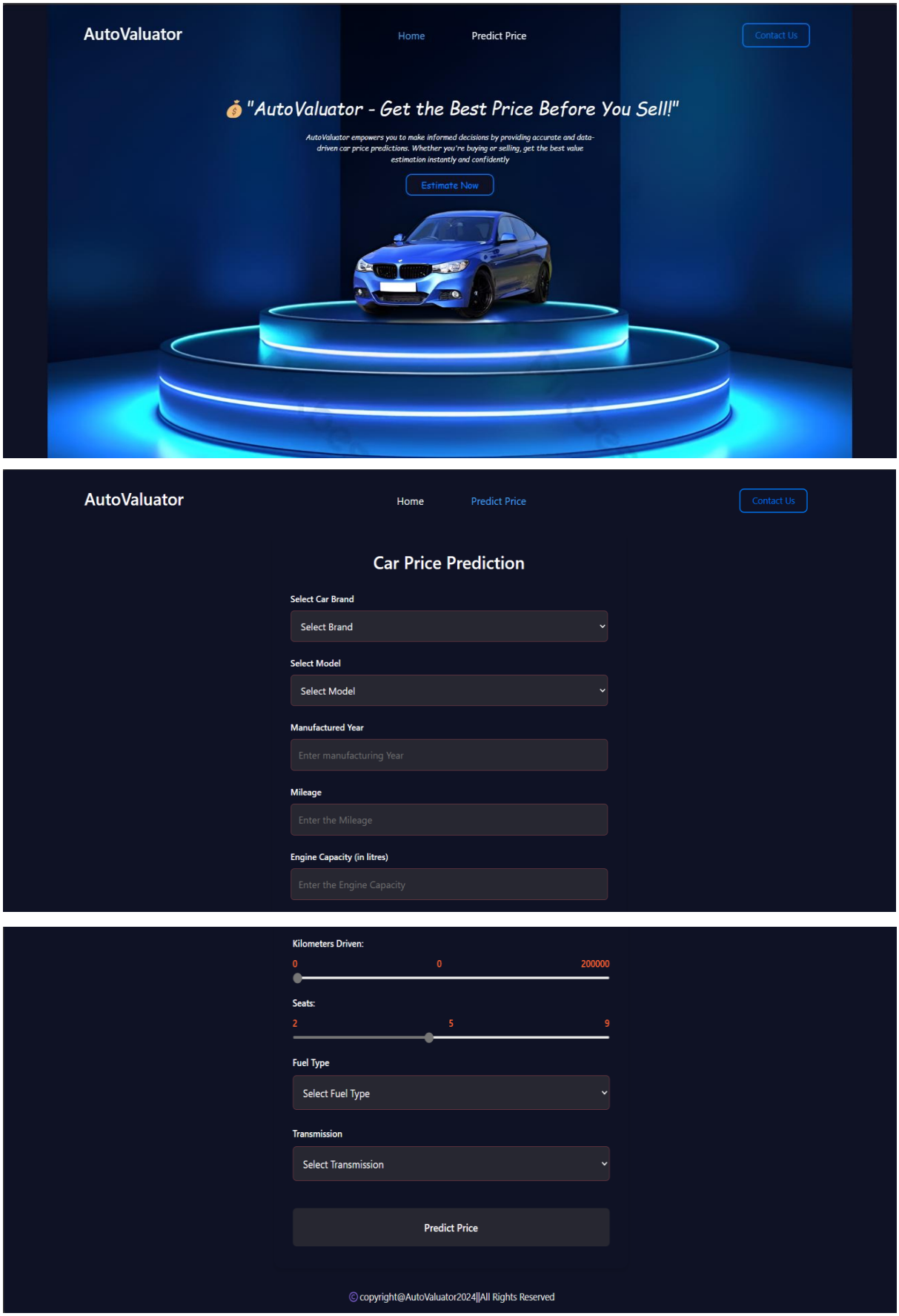


Figure 3.6 – Design Representation

1.5.1 Design Diagrams

- **Activity Diagram**

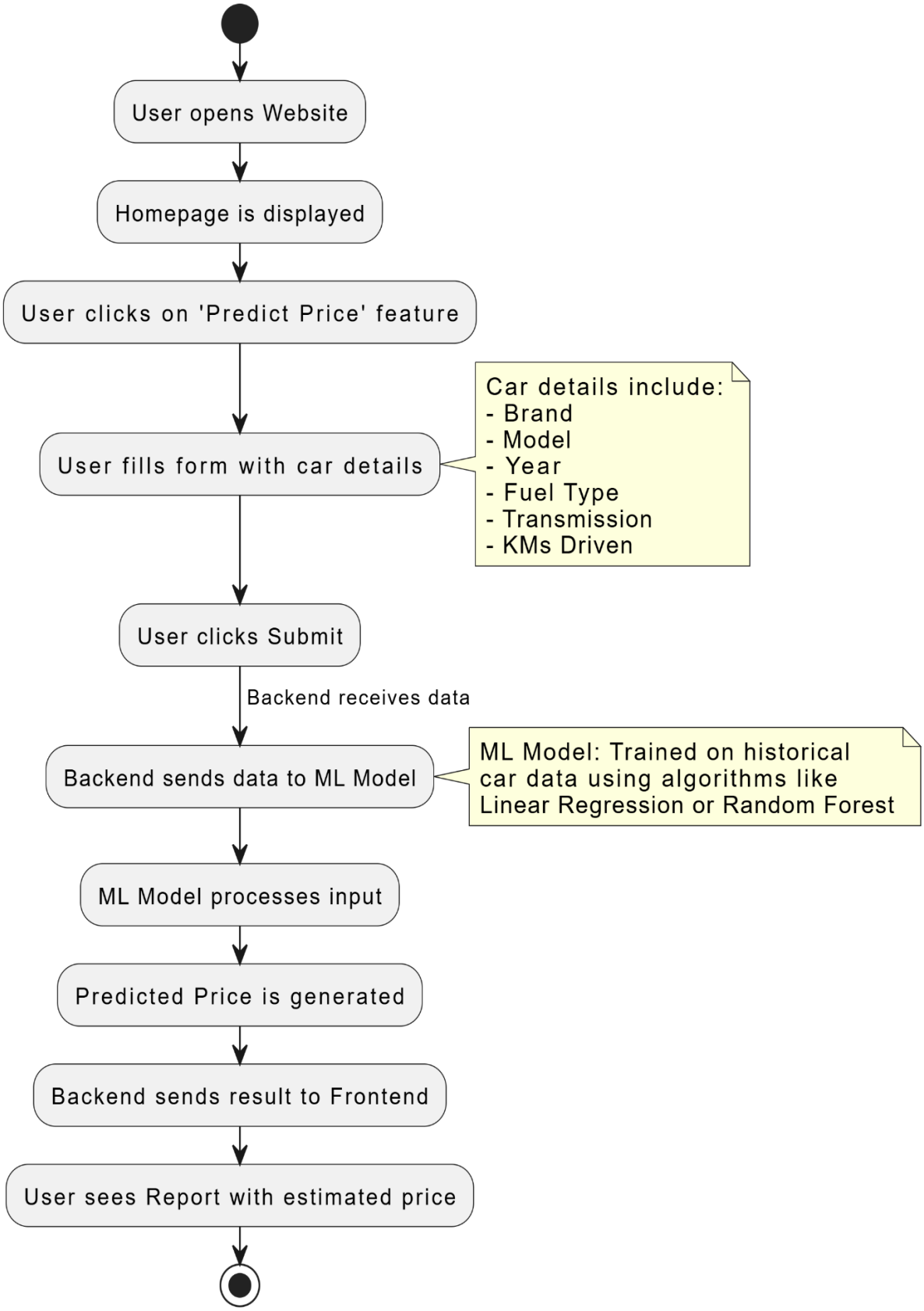


Figure 3.7 : Activity Diagram

- **Use Case Diagram**

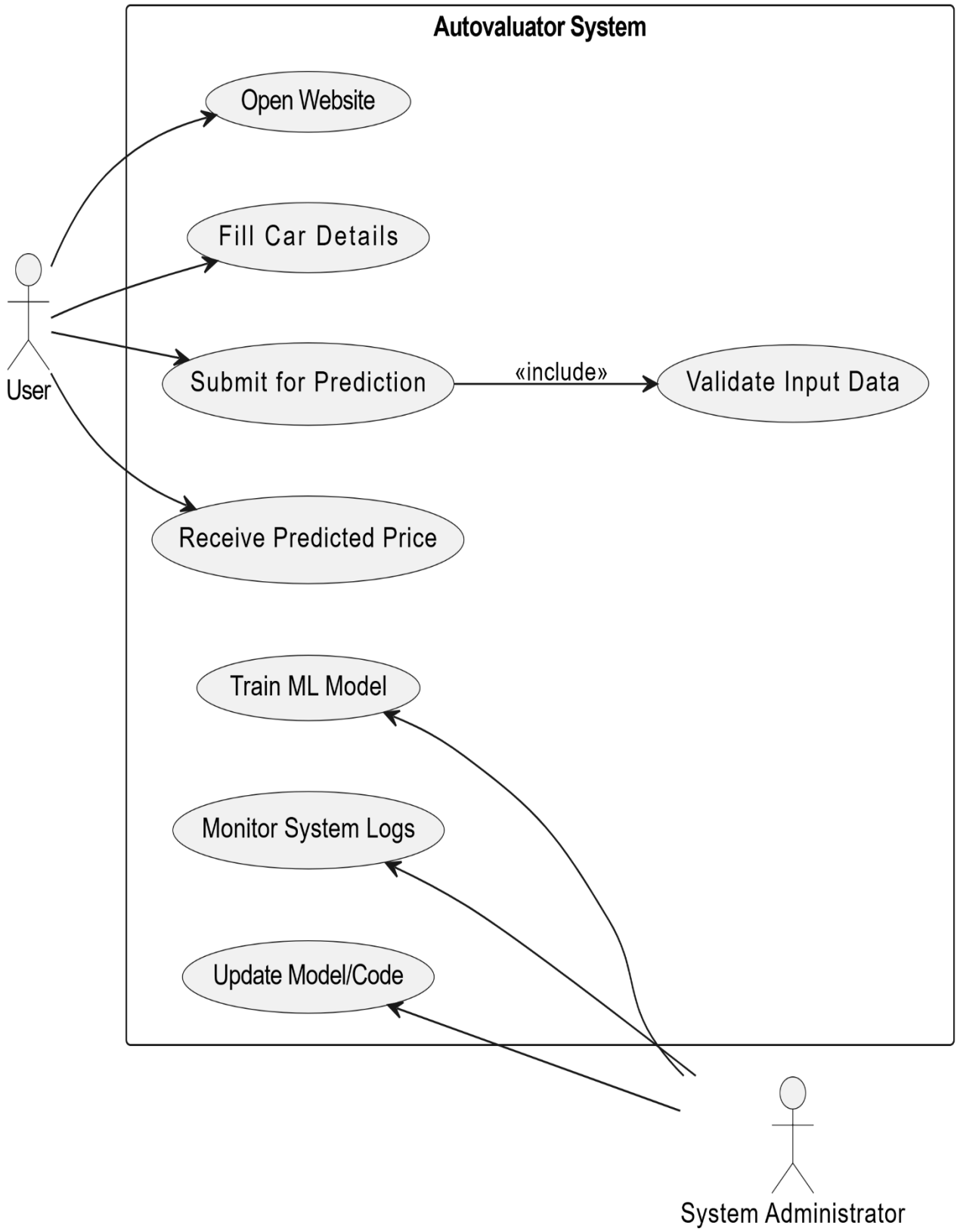


Figure 3.8 : Use Case Diagram

- **ER Diagram**

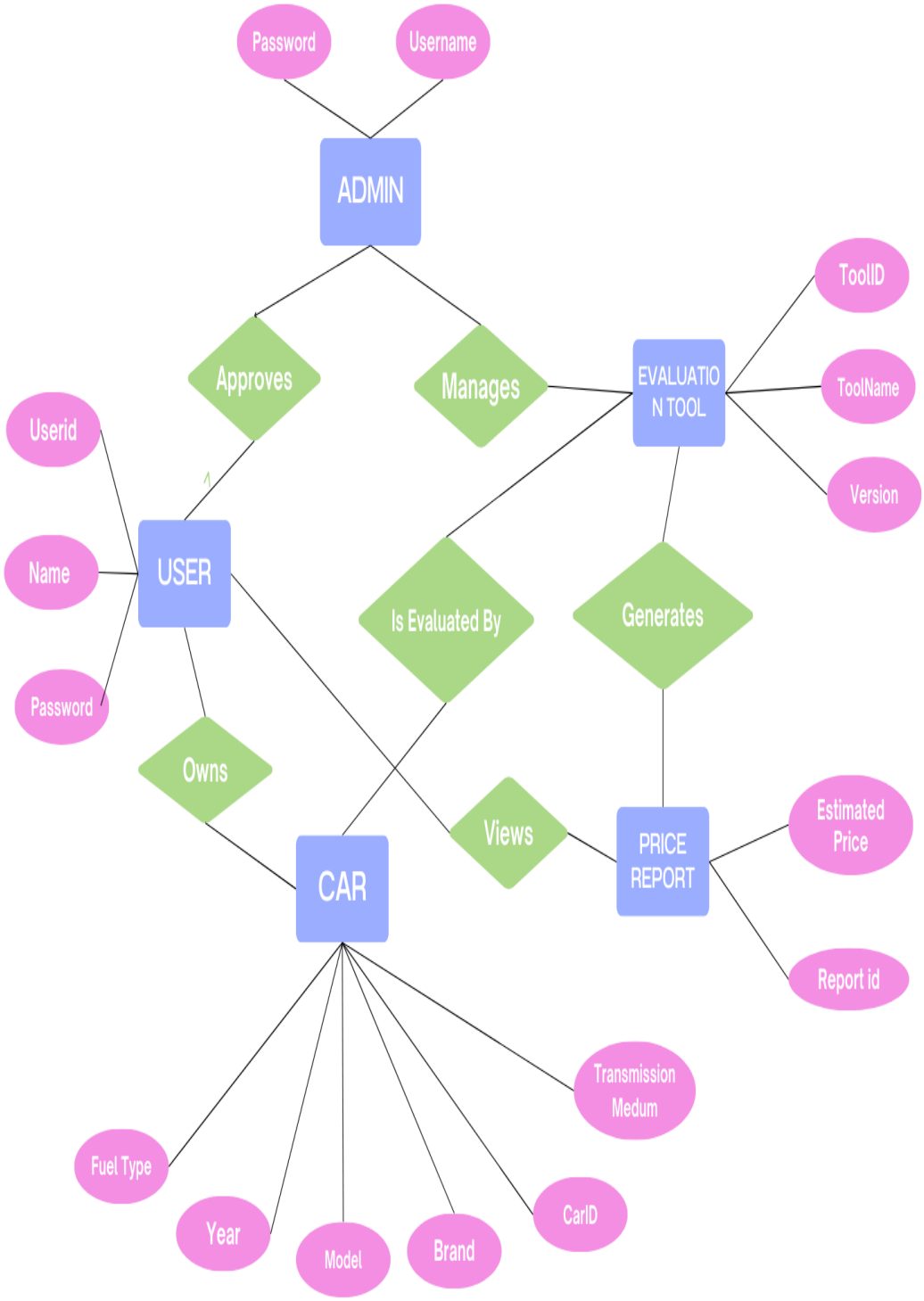


Figure 3.9: ER Diagram

- **Flowchart**

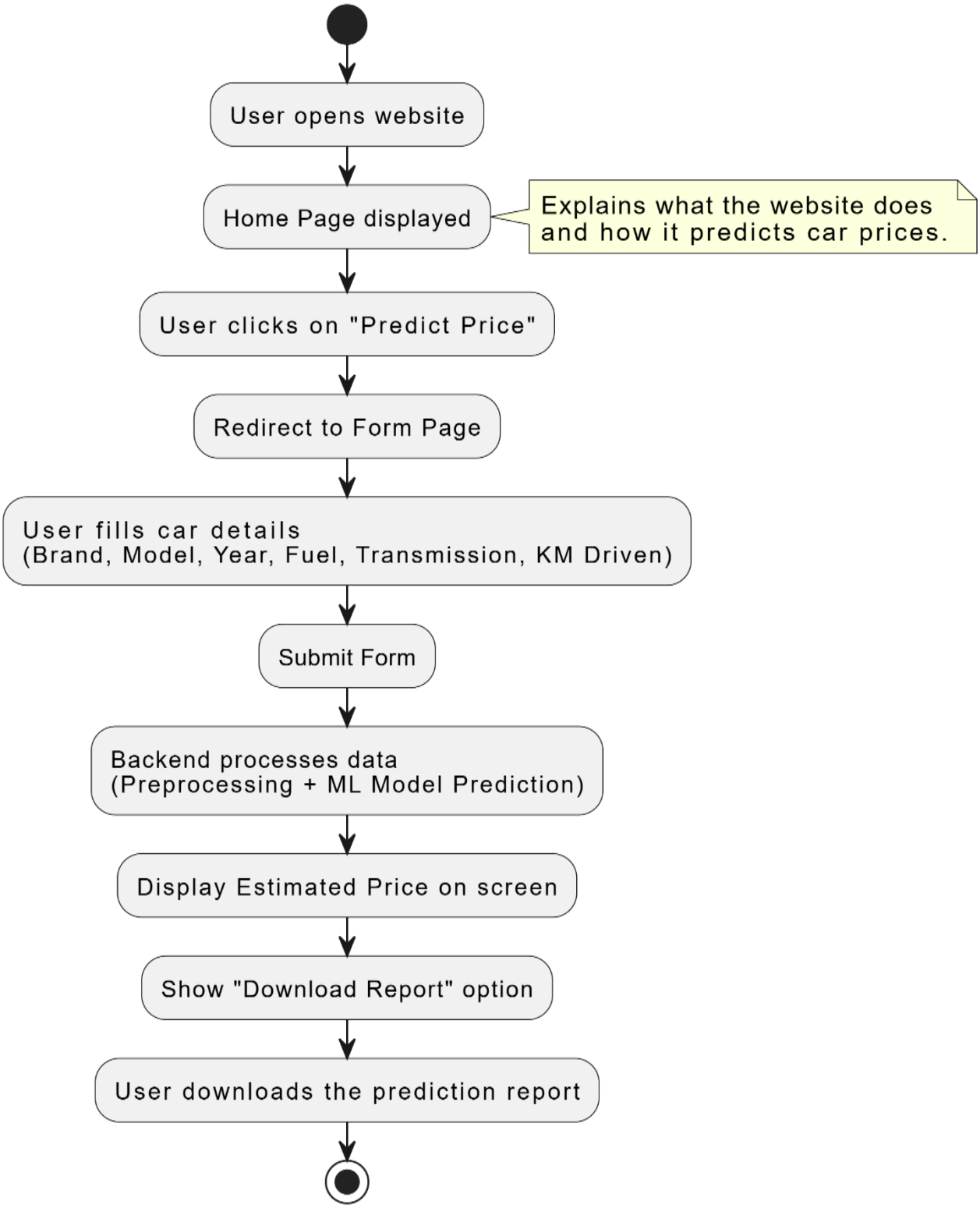


Figure 3.10: Flowchart

- **Sequence Diagram**

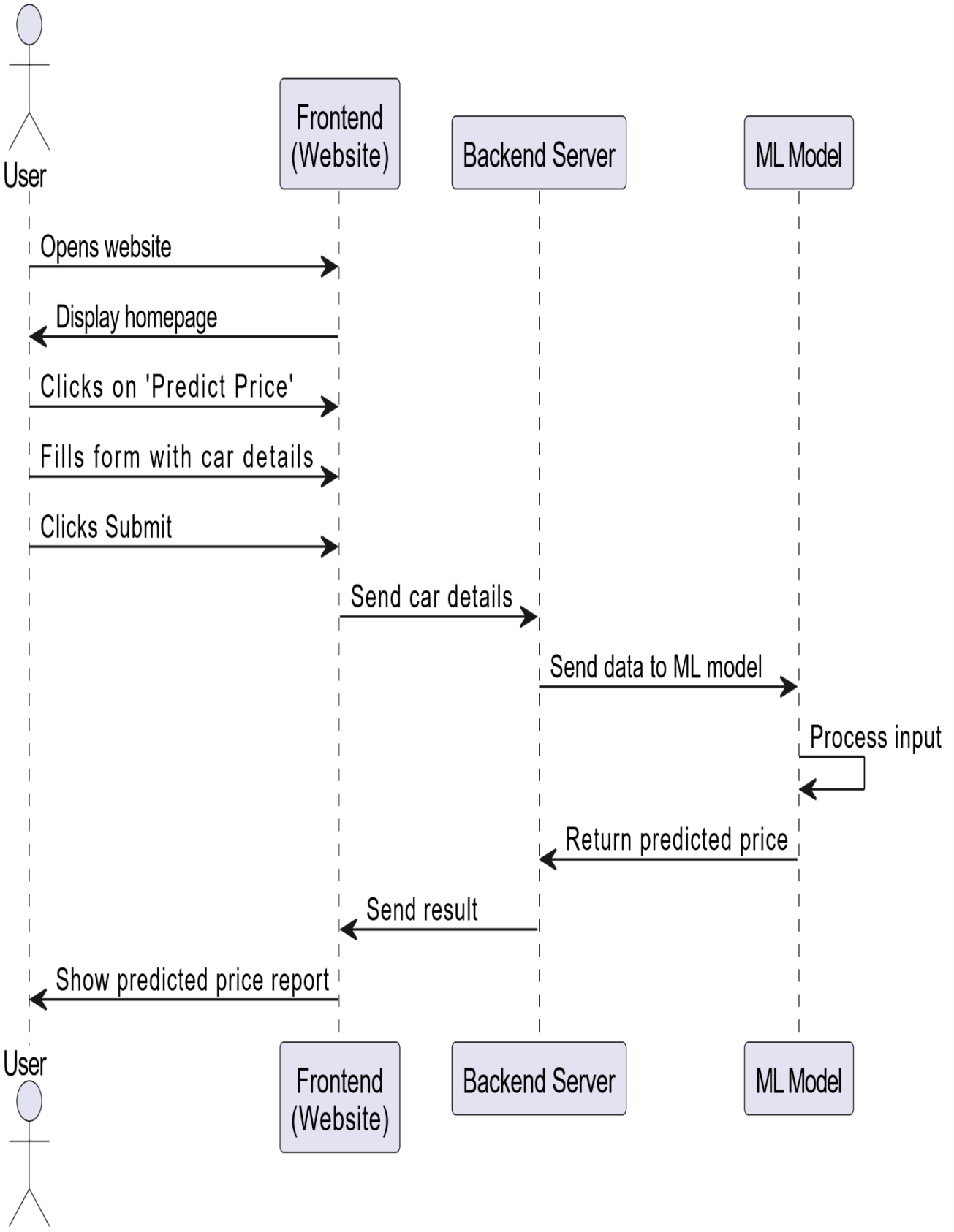


Figure 3.11: Sequence Diagram

3.6 Deployment Requirements

There are various requirements (hardware, software and services) to successfully deploy the system. These are mentioned below :

3.6.1 Hardware

Component	Specification
Processor	Minimum Intel i5 or AMD Ryzen 5 or above
RAM	Minimum 8 GB (Recommended: 16 GB)
Storage	At least 500 GB HDD or 256 GB SSD
GPU (optional)	NVIDIA GPU (for faster ML model training)
Network	Stable internet connection (for web hosting)

Table 3.1: Hardware Requirements

3.6.2 Software

Component	Specification / Tools Used
Operating System	Windows 10/11, Linux (Ubuntu), or macOS
Programming Language	Python (for ML and backend logic)
ML Libraries	scikit-learn, pandas, NumPy, joblib
Web Framework	Flask or Django (Python)
Frontend Technologies	HTML, CSS, JavaScript (or React if used)
Database (optional)	SQLite / MySQL / MongoDB (if storing records)
IDE/Code Editor	VS Code / Jupyter Notebook / PyCharm
Web Browser	Chrome / Firefox (for testing frontend)
Deployment Platform	Heroku / Render / Localhost / AWS (depending on usage)

Table 2: Software Requirement

Chapter 4 . Implementation

The implementation phase focuses on bringing the proposed system to life through proper coding, model training, and integration of all components. This chapter explains how various modules of the system were developed and connected to create a fully functional car price prediction application.

4.1 Technique Used

4.1.1 React.js for Frontend

React.js is used for developing the frontend of **AutoValuator**. React is a popular JavaScript library for building user interfaces. It allows us to create reusable UI components, enabling a modular and dynamic design. With features like state management, hooks, and component-based architecture, React provides flexibility and efficiency.

React was used for the following purposes:

- **Frontend Development:**
  - Created a user-friendly interface to display live data such as the count of students and vehicles entering the campus.
  - Designed an admin dashboard that visualizes the data in charts and tables.
- **Component-Based Design:**
  - Modularized the interface using reusable React components for live data, camera feed display, and analytics.
- **State Management:**
  - Managed application state using React's useState and useEffect hooks to ensure real-time updates.

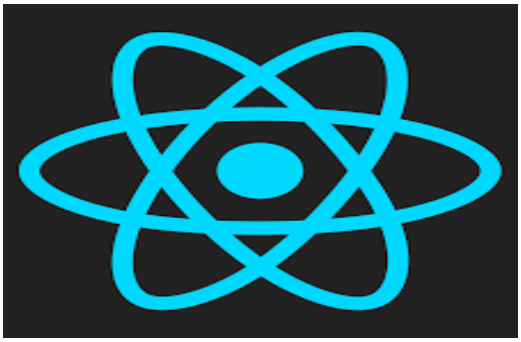


Figure 4.1 - ReactJS

React enables the following:

- Creation of an interactive dashboard for users to access tools.
- Smooth navigation between tools.
- Efficient state management for a seamless user experience.

Frontend of our project looks like:

Home Page:

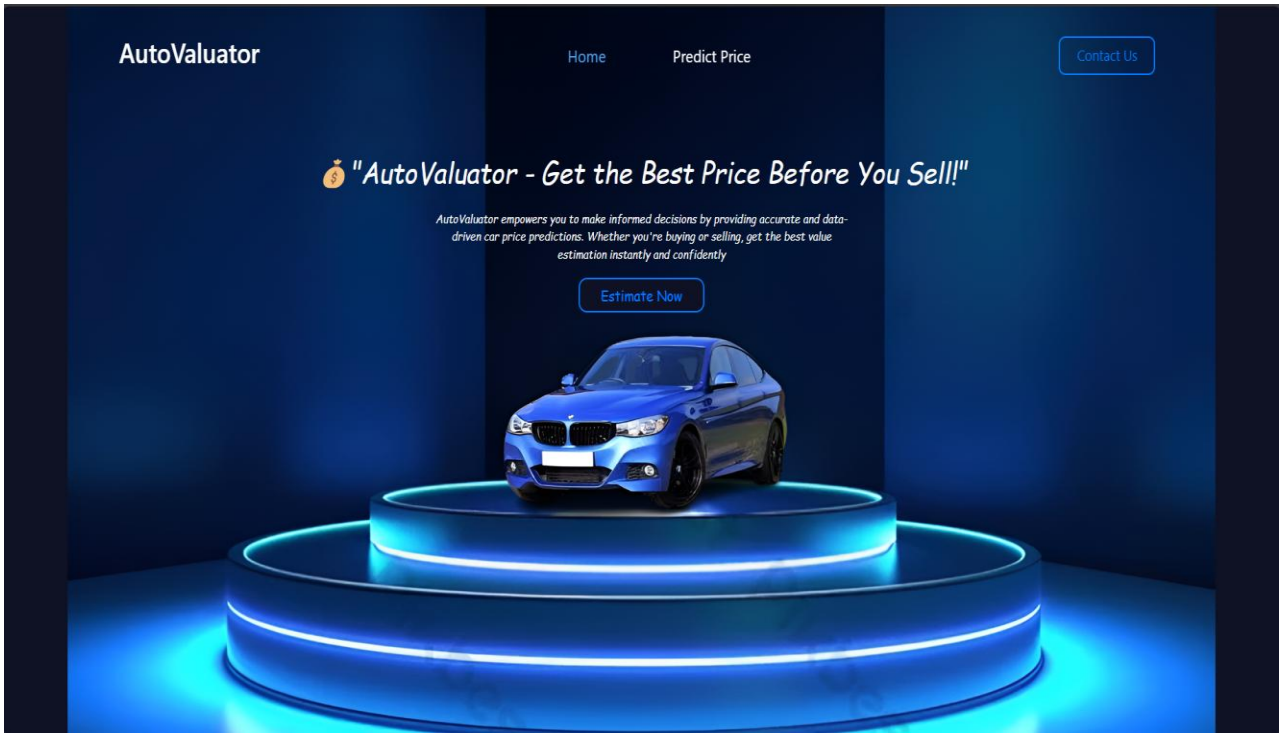


Figure 4.2 – FrontPage Design

Predict Price Design Looks Like:

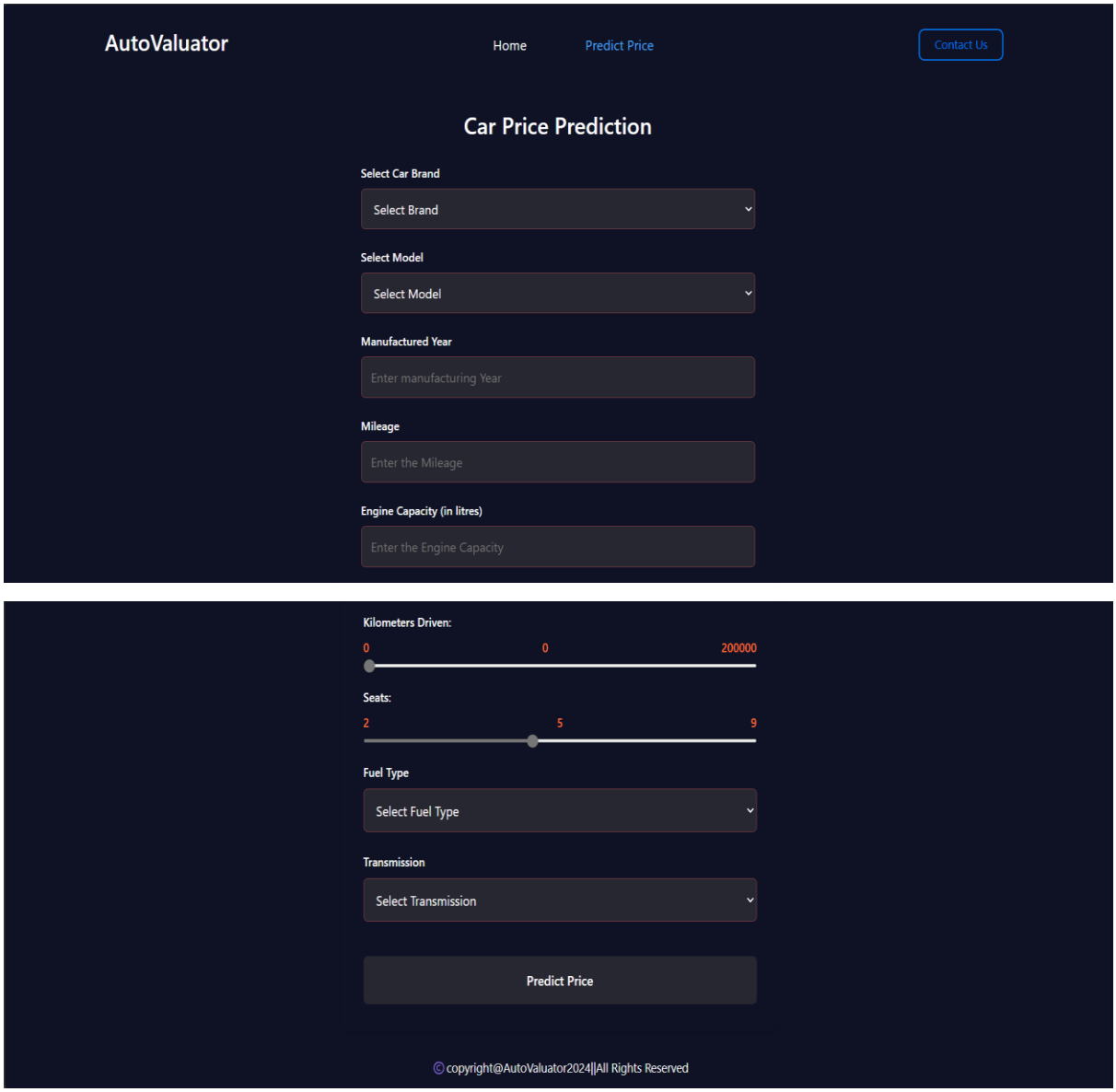


Figure 4.3 Dashboard Design

Result Page looks like:

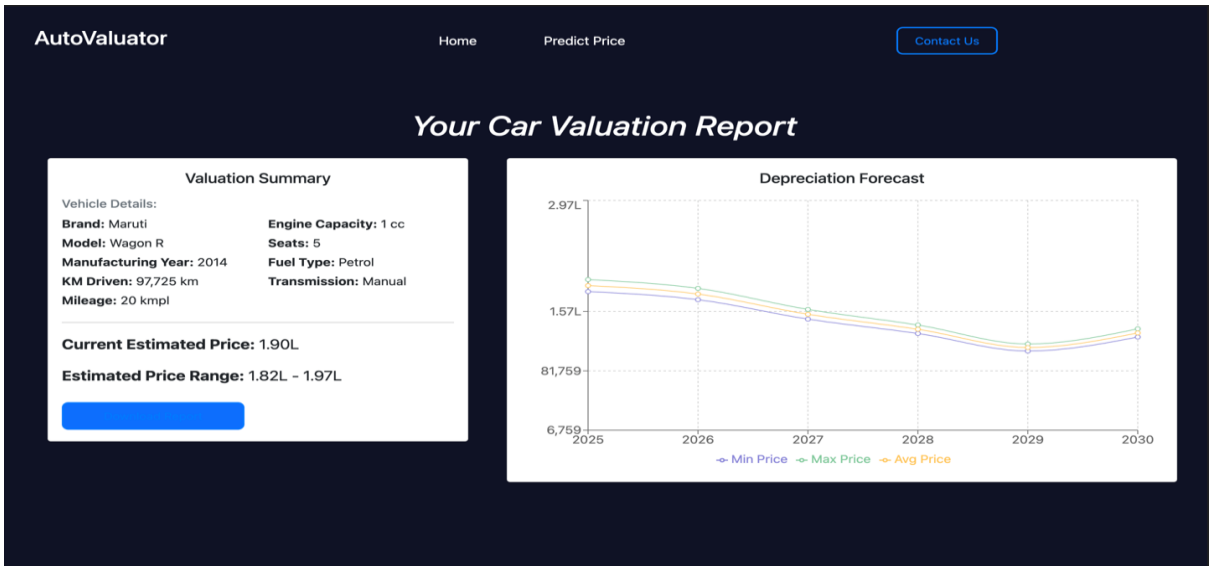


Figure 4.4 – Result Page Design

4.1.2 Python for Backend

- **Python:** Acts as the main backend processing language, managing logic and connecting the ML model with the frontend.
- **Machine Learning Libraries:** Used to train, evaluate, and deploy the ML model for car price prediction.

These are the list of libraries used in our project:

- numpy
- pandas
- os
- matplotlib.pyplot
- sklearn.ensemble
- sklearn.linear_model
- sklearn.svm
- sklearn.tree
- sklearn.model_selection
- sklearn.metrics
- kagglehub
- pickle

How Processing is done:

1. **Data Collection**
   The dataset used in this project was imported from Kaggle using the KaggleHub API. It contains various features of used cars like brand, model, manufacturing year, kilometers driven, fuel type, transmission, engine capacity, mileage, number of seats, and the target variable: selling price.

2. **Data Preprocessing**
   Several preprocessing techniques were applied to clean and prepare the data for training:

- Unnecessary columns such as max_power, car_name, seller_type, and Unnamed: 0 were dropped.
- Column names were renamed for better readability.
- The Engine-Capacity values were converted from cc to liters.
- Missing or incorrect values (e.g., cars with 0 seats) were fixed by replacing them with appropriate values.
- Duplicate entries were removed to avoid data redundancy.
- One-hot encoding was applied to categorical features like brand, fuel type, and transmission to convert them into numeric format.

3. Feature Selection

To determine which features were most important in predicting the car price, an ExtraTreesRegressor model was used. This model provided feature importance scores, and a bar chart was plotted to visualize the top influencing features.

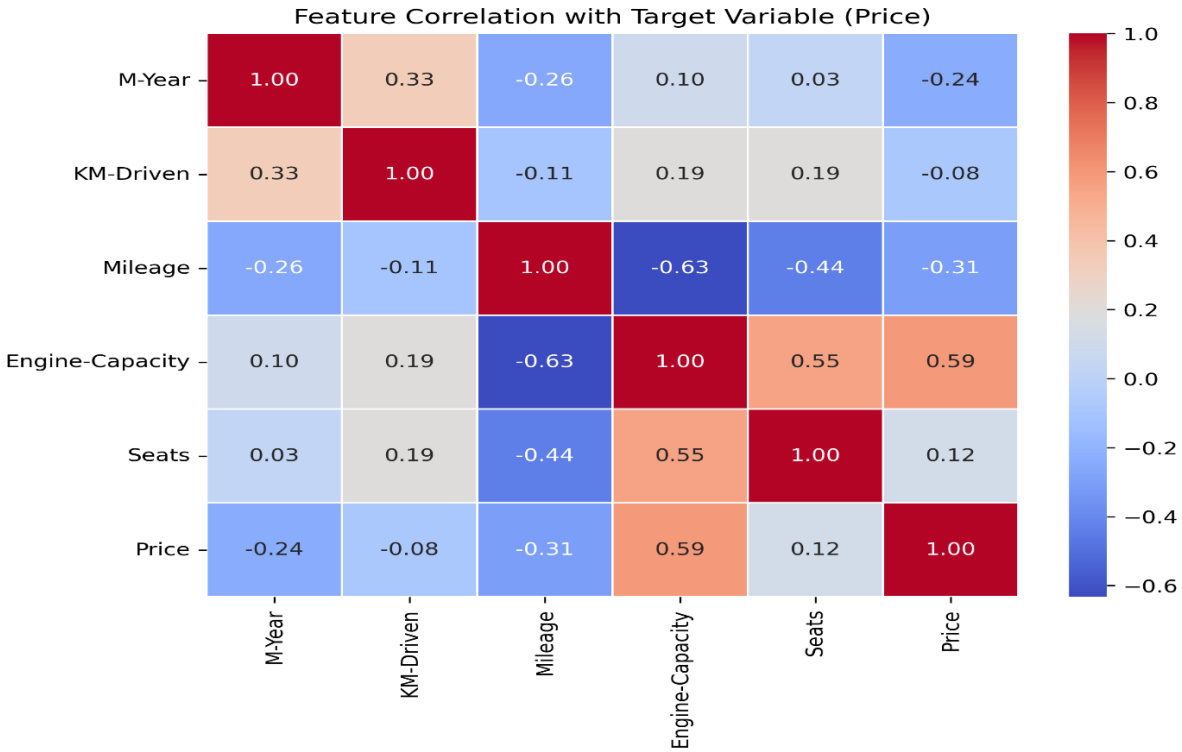


Figure 4.5: Feature Selection

4. Data Splitting

The dataset was split into training (80%) and testing (20%) sets using the train_test_split function from scikit-learn. This ensures that the model is evaluated on unseen data to check its real performance.

5. Model Building

Multiple machine learning regression algorithms were considered, including:

- Linear Regression
- Support Vector Regression (SVR)
- Decision Tree Regressor
- Extra Trees Regressor
- Random Forest Regressor (Final choice)

After testing different models, the Random Forest Regressor was selected due to its better performance in terms of accuracy and lower error values.

6. Model Evaluation

The model was evaluated using the following metrics:

- Mean Squared Error (MSE)
- R-squared (R^2) Score

These metrics were used to assess how well the model predicted car prices on the test data. A higher R^2 score indicates better performance.

7. Visualization

Important features influencing the price were visualized using bar graphs. This helped in understanding which attributes (like mileage, engine capacity, etc.) have more impact on the final price.

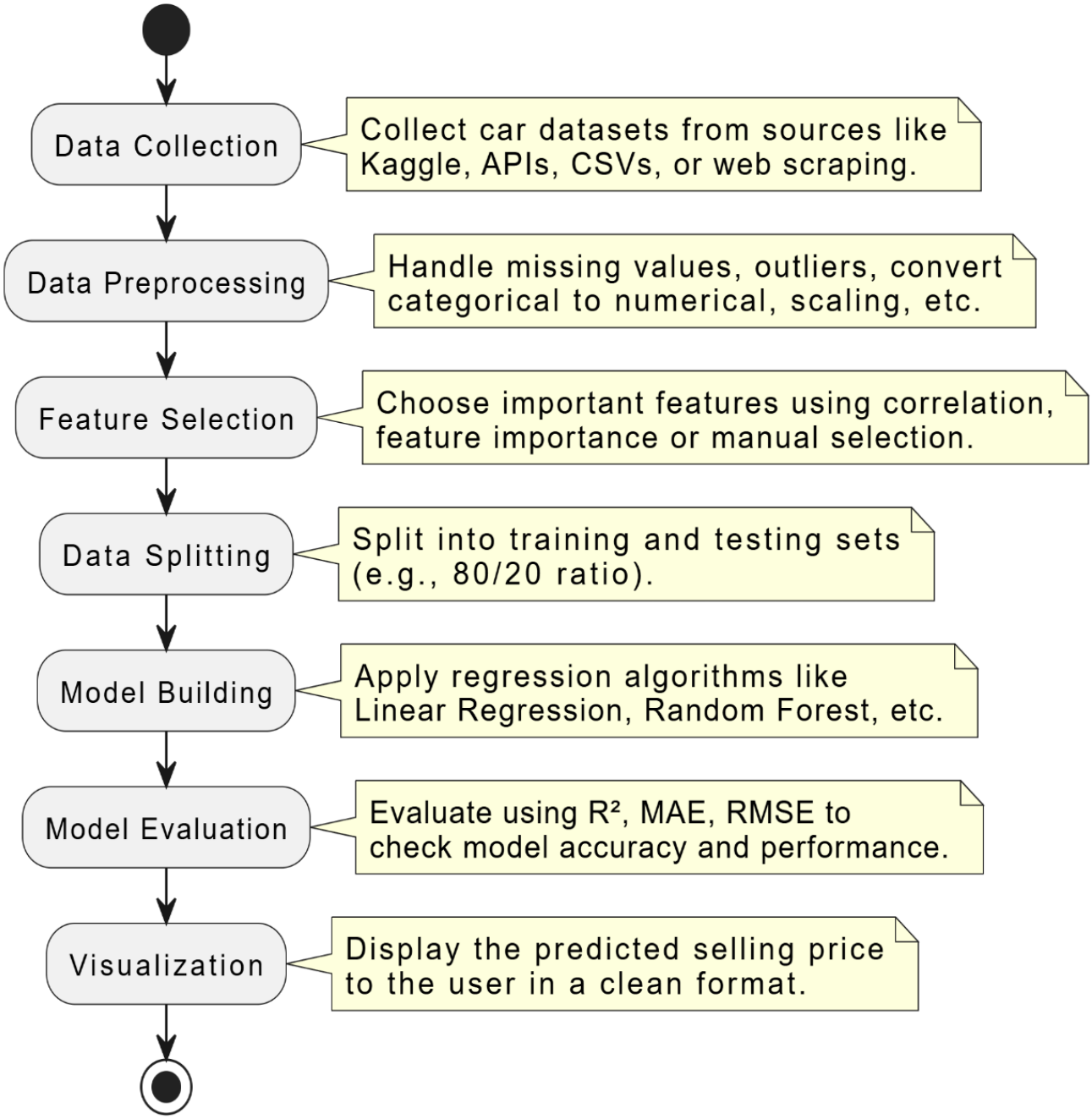


Figure 4.6: Processing

4.2 Tools Used

1. **Python**

   Used as the primary programming language for data preprocessing, machine learning model building, training, and evaluation. Python's simplicity and rich ecosystem make it ideal for data science projects.

2. **JavaScript**

   JavaScript serves as the backbone of the project, offering dynamic interactivity and enabling both client-side and server-side programming.

- **Frontend (React):**

  React.js, a JavaScript library, is used to build the user interface. It helps create reusable components, making the application modular, responsive, and efficient. React's state management and hooks simplify the handling of dynamic content, ensuring a smooth user experience.

Fig 4.7: Tools Used

4.3 Language Used

The project uses a combination of modern and versatile programming languages to ensure seamless functionality, interactive user experience, and efficient backend operations. Here's a detailed overview:

1. Pandas

A powerful Python library used for data manipulation and cleaning. It allows easy handling of structured data.

2. NumPy

Used for performing numerical operations and working with arrays, especially during preprocessing steps.

3. Matplotlib / Seaborn

Used for creating data visualizations such as histograms, scatter plots, and heatmaps to analyze relationships and distributions.

4. Scikit-learn

A machine learning library used for data preprocessing, splitting, model training (e.g., Linear Regression, Random Forest), and evaluation metrics like MAE and R^2 score.

6. Jupyter Notebook

Interactive environments used for writing, testing, and visualizing Python code, especially useful for data analysis workflows.

7. VS Code

A code editor used to write and organize project files locally if not using online notebooks.

8. Flask

Used to create a simple and interactive web app interface for users to input car details and get predicted prices in real-time.

9. Git & GitHub

Used for version control and collaborative development. GitHub also serves as a platform to showcase the project.

10. Vercel

Cloud platforms used to deploy the web application and make it accessible via a public link.

Fig 4.8: Languages Used

4.5 Testing

Testing ensures that the system performs as expected. The functionality and reliability of each tool were tested using various test cases.

4.5.1 Testing Approach

To ensure that the AutoValuator platform performs accurately and smoothly for users, we'll follow a structured testing approach. Here's the plan:

1. **Unit Testing:**
   - **What it is:** We will test individual components of the project separately — such as the form input fields, price prediction logic, and the ML model.
   - **Why it's important:** This helps us identify any issues in small units before integrating them into the complete system.

2. **Integration Testing:**
   - **What it is:** We'll check how different parts of the system work together — for example, whether user inputs from the frontend are correctly sent to the backend and whether the predicted price is properly shown on the frontend.
   - **Why it's important:** Ensures seamless communication between UI and backend, and validates proper flow.

3. **User Testing:**
   - **What it is:** Real users will test the platform by entering car details and predicting prices to see if the UI is intuitive and the results are understandable.
   - **Why it's important:** Users give feedback from a practical perspective, helping to improve usability and fix hidden issues.

4. **Performance Testing:**
   - **What it is:** We'll test the platform's speed and responsiveness when multiple users use it simultaneously or when complex data is entered.
   - **Why it's important:** Ensures the platform performs well under load and provides results in real-time without delays.

5. **Security Testing:**
   - **What it is:** We'll test for secure data handling — especially ensuring that the user inputs are protected from common threats like injection attacks or data breaches.
   - **Why it's important:** Builds trust by ensuring that the website is safe and secure for public use.

6. **Bug Fixing and Retesting:**
   - **What it is:** After discovering any bugs, we'll fix them and retest the full workflow — from user input to report download — to make sure everything works correctly.
   - **Why it's important:** Fixes don't cause new issues, and the overall system remains stable.

4.5.2 Test Cases

Test Case 1: Toyoto Car Price Prediction:

Test Case ID	TC001
Test Case	Verify that car price prediction is correct or not.
Input Parameters	• **Car Brand**: Toyota • **Model**: Corolla • **Year**: 2022 • **Fuel Type**: Petrol • **Transmission**: Automatic • **Mileage (km)**: 5,000 • **Engine (CC)**: 1800
Expected Outcome	Predicted price should fall within ₹12,00,000 to ₹15,00,000 range.
Actual Outcome	₹13,45,000
Status	Pass

Test Case 2: Maruti Suzuki Car Price Prediction

Test Case ID	TC002
Test Case	Verify that car price prediction is correct or not.
Input Parameters	• **Car Brand**: Maruti Suzuki • **Model**: Swift • **Year**: 2012 • **Fuel Type**: Diesel • **Transmission**: Manual • **Mileage (km)**: 1,20,000 • **Engine (CC)**: 1200
Expected Outcome	Predicted price should fall within ₹1,50,000 to ₹2,50,000 range
Actual Outcome	₹2,10,000
Status	Pass

Test Case Outputs

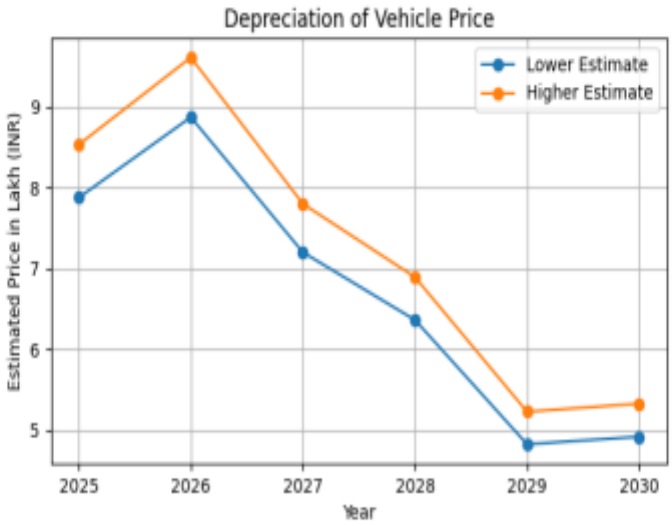
Test Case Output 1: Toyoto Car Price Prediction

Car Price Report

Attribute	Value
Brand	Toyota
Model	Corolla
M-Year	2019
KM-Driven	5559
Mileage	20
Engine-Capacity	1.8
Seats	5
Fuel-Type	Petrol
Transmission	Automatic

Predicted Price Range: INR 7.88 L - INR 8.53 L

Depreciation Graph:



Report generated on 10 May 2025

Note: This valuation is an estimate based on current market trends.

Figure 4.12 – Toyoto Car Prediction

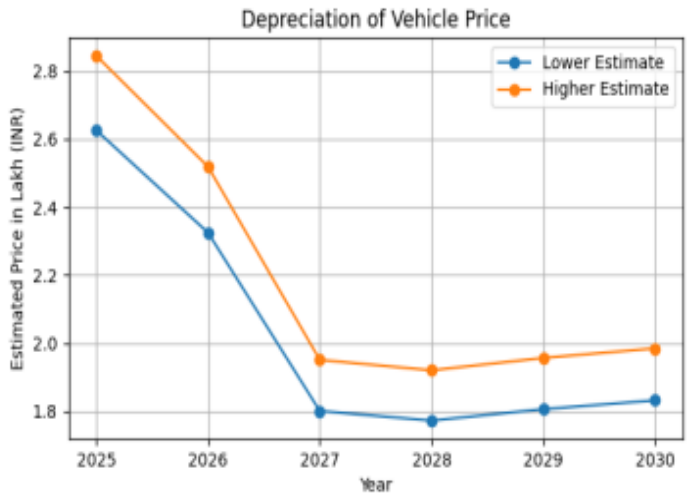
Test Case Output 2: Maruti Suzuki Car Price Prediction

Car Price Report

Attribute	Value
Brand	Maruti
Model	Swift
M-Year	2012
KM-Driven	120313
Mileage	20
Engine-Capacity	1.2
Seats	5
Fuel-Type	Diesel
Transmission	Manual

Predicted Price Range: INR 2.63 L - INR 2.84 L

Depreciation Graph:



Report generated on 10 May 2025

Note: This valuation is an estimate based on current market trends.

Figure 4.13 – Maruti Suzuki Price Prediction

This concludes the implementation of **ToolFusionX**, ensuring all components are functional and meet the project requirements.

.

Chapter 5.Conclusion

5.1 Conclusion

In conclusion, this project aims to provide a simple and effective solution for users to predict accurate car prices based on key inputs like brand, model, fuel type, and more. With a clean, user-friendly interface and machine learning on the backend, the platform ensures that users can easily get an estimated price for their vehicle in just a few steps.

While there are some limitations, such as dependency on existing data and lack of real-time market updates, these can be improved with future enhancements. Overall, AutoValuator helps users make informed decisions, making the car-selling process faster, smarter, and more convenient.

5.2 Limitations of the Work

1. **Limited Dataset:**

   The model's accuracy is dependent on the dataset used for training. A small or unrepresentative dataset may lead to a model that does not generalize well, especially for car models or features that were underrepresented in the training data.

2. **Omission of Certain Factors:**

   The model does not consider external factors like geographic location, market trends, or car condition (e.g., accident history, service history). These factors can significantly influence car prices but were not included due to data limitations.

3. **Overfitting/Underfitting Risks:**

   There is a risk of the model either overfitting to the training data or underfitting by not capturing the complex relationships between features. Despite efforts to optimize the model, achieving the perfect balance between bias and variance remains a challenge.

4. **Lack of Real-Time Market Data:**

   The model does not account for real-time fluctuations in car prices caused by market conditions, supply-demand changes, or seasonal trends, which can affect the accuracy of predictions over time.

5. **Model Complexity:**

   The machine learning models used (e.g., Linear Regression, Decision Trees) may not capture the more complex, nonlinear relationships in the data. More advanced techniques such as neural networks or ensemble methods were not explored due to time constraints.

5.3 Suggestion and Recommendations for Future Work

1. Expanding the Dataset:

- **Recommendation:** To improve model performance, future work should focus on collecting a larger and more diverse dataset. Including additional data points like car condition, accident history, service records, and regional price differences could enhance the accuracy and reliability of predictions.
- **Future Work:** Consider integrating data from multiple sources or APIs to provide real-time car prices and more up-to-date information.

2. Incorporating More Features:

- **Recommendation:** Future work could focus on adding additional features such as car color, number of owners, location (for regional price differences), or even customer reviews. This could lead to a more robust model with better prediction capabilities.
- **Future Work:** Feature engineering and the inclusion of domain-specific features could improve the accuracy of the model, potentially capturing hidden patterns.

3. Advanced Machine Learning Models:

- **Recommendation:** The current models (e.g., Linear Regression, Decision Trees) can be enhanced by using more complex machine learning algorithms such as Random Forests, Gradient Boosting, or Neural Networks. These models can handle non-linear relationships and improve prediction accuracy.
- **Future Work:** Implementing ensemble methods like XGBoost or exploring deep learning models could improve model performance, especially for handling large and complex datasets.

4. Real-Time Price Prediction:

- **Recommendation:** Implementing a real-time data fetching mechanism and predicting car prices dynamically as per changing market conditions would significantly increase the tool's practical application.
- **Future Work:** Integrating with real-time market data sources and APIs could help adjust predictions based on real-time trends and external factors like demand fluctuations or economic conditions.

5. Improving User Interface (UI):

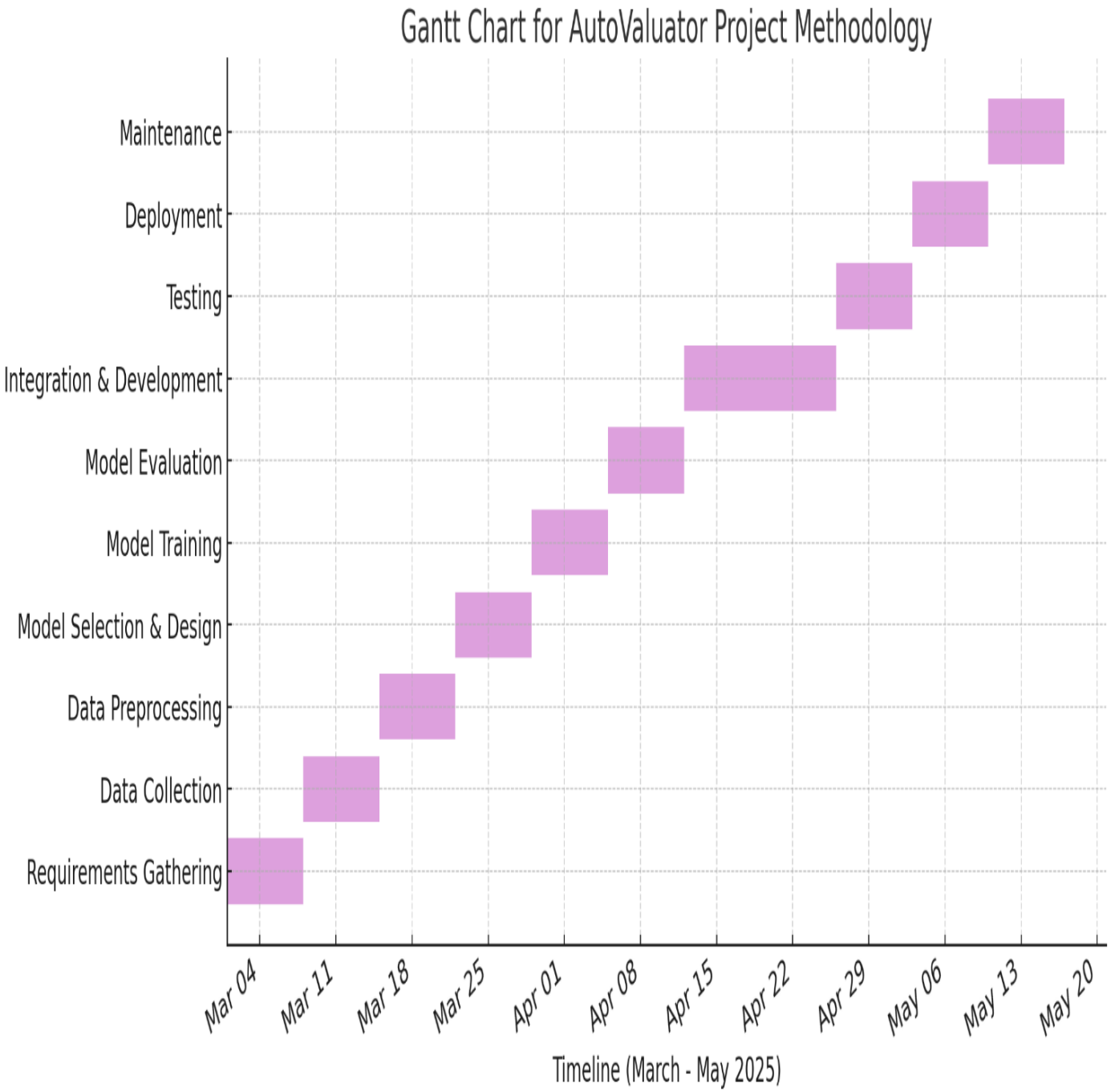
- **Recommendation:** The user interface could be further improved to be more interactive, intuitive, and mobile-friendly. Enhancing the design with better visualization of the prediction results and user input features would make the tool more user-friendly.
- **Future Work:** Include more visualization features like price trends, graphical comparisons, and user-specific recommendations based on the input data.

Bibliography

1. Gegic, E., Isakovic, B., Keco, D., Masetic, Z., & Kevric, J. (2019). Car price prediction using machine learning techniques. *TEM Journal*, *8*(1), 113.
2. Viswanatha, V., Ramachandra, A. C., Vachan, H. V., & Sourav, S. S. (2023, October). Predicting the price of used cars using machine learning. In *2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT)* (pp. 1-6). IEEE.
3. Noor, K., & Jan, S. (2017). Vehicle price prediction system using machine learning techniques. *International Journal of Computer Applications*, *167*(9), 27-31.
4. Das Adhikary, D. R., Sahu, R., & Pragyna Panda, S. (2022). Prediction of used car prices using machine learning. In *Biologically Inspired Techniques in Many Criteria Decision Making: Proceedings of BITMDM 2021* (pp. 131-140). Singapore: Springer Nature Singapore.
5. Venkatasubbu, P., & Ganesh, M. (2019). Used cars price prediction using supervised learning techniques. *Int. J. Eng. Adv. Technol.(IJEAT)*, *9*(1S3).

Project Plan

Gantt Chart



Guide Interaction Sheet

Date	Discussion	Action Plan
12/02/2025	Discussed about title of the project.	AutoValuator: Car Price Prediction Model was decided as title of the project.
19/02/2025	Discussion on the technology to be for implementing the project.	MERN as Frontend and Python as Backend was decided as technology to be used in creation of the project.
26/02/2025	Discussion of the creation of synopsis of the project.	Gathering of information for synopsis creation.
05/03/2025	Synopsis was created successfully for the project.	Synopsis was submitted successfully.
12/03/2025	Discussed on presentation to be given of the project.	Presentation was prepared successfully using MS Powerpoint.
19/03/2025	Synopsis Presentation to be given.	Presentation given successfully.
26/03/2025	Discussion on the implementation of the project.	React , Python , ML libraries were used for implementing the project.
02/04/2025	Design Phase : Module wise UI Form design (ERD , DFD, Use Case and other UML Diagrams)	Design diagrams prepared successfully and submitted.
09/04/2025	Implementation Demo and report discussion	Implementation Demo shown successfully.
16/04/2025	Discussion On Report of the project	Report containing details of the project was prepared.
30/04/2025	Discussion on Research paper of project and its publishing.	Research paper prepared and published successfully.
14/05/2025	Discussion on Video and Technical Poster.	Video and Technical Poster prepared and shown successfully.
21/05/2025	Discussion on Final Project Submission including presentation , implementation and report.	Final Project Submitted Succesfully.

Source Code

Model Implementation:

```
from flask import Flask, request, jsonify, send_file, make_response
import joblib
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from datetime import datetime
from reportlab.platypus import SimpleDocTemplate, Paragraph, Table, TableStyle, Spacer, Image
from reportlab.lib.pagesizes import letter
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib import colors
from io import BytesIO
import json
import base64
from flask_cors import CORS


current_year = datetime.now().year
matplotlib.use('Agg')

def process_car_data(brand, model, year, km_driven, mileage, engine_capacity, seats, fuel_type, transmission):
    model = joblib.load('Predict-Car-V2.pkl')
    print("CURRENT_YEAR : ", current_year - year)
    df = pd.DataFrame({
            'Brand': [brand],
            'Model': [model],
            'M-Year': [current_year - year],
            'KM-Driven': [km_driven],
            'Mileage': [mileage],
            'Engine-Capacity': [engine_capacity],
            'Seats': [seats],
            'Fuel-Type': [fuel_type],
            'Transmission': [transmission]
        })

    df_encoded = pd.get_dummies(df)
    training_columns = []

    with open('column_names.txt', 'r') as f:
        for line in f:
            training_columns.append(line.strip())

    df_encoded = df_encoded.reindex(columns=training_columns, fill_value=0)
    predictions = model.predict(df_encoded)
    price = int(predictions[0])
```

```
    l = price - ( price * 0.04 )
    h = price + ( price * 0.04 )
    return [price, l, h]

def generate_depreciation_graph(model_year, price_range):
    years = np.arange(current_year, current_year + len(price_range))
    lower_values = [price_range[i][1]/100000 for i in range(len(price_range))]
    higher_values = [price_range[i][2]/100000 for i in range(len(price_range))]

    plt.figure(figsize=(7, 4))
    plt.plot(years, lower_values, label='Lower Estimate', marker='o')
    plt.plot(years, higher_values, label='Higher Estimate', marker='o')
    plt.xlabel('Year')
    plt.ylabel('Estimated Price in Lakh (INR)')
    plt.title('Depreciation of Vehicle Price')
    plt.legend()
    plt.grid()

    img_buffer = BytesIO()
    plt.savefig(img_buffer, format='png')
    plt.close()
    img_buffer.seek(0)
    return img_buffer

def generate_pdf_report(input_data, price_range, filename="Car_Price_Report.pdf"):
    pdf_buffer = BytesIO()
    doc = SimpleDocTemplate(pdf_buffer, pagesize=letter)
    styles = getSampleStyleSheet()
    elements = []

    elements.append(Paragraph("Car Price Report", styles["Title"]))
    table_data = [["Attribute", "Value"]] + [[k, str(v)] for k, v in input_data.items()]
    table = Table(table_data, colWidths=[150, 250])
    table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
        ('ALIGN', (0, 0), (-1, -1), 'LEFT'),
        ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
        ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
        ('BACKGROUND', (0, 1), (-1, -1), colors.beige),
        ('GRID', (0, 0), (-1, -1), 1, colors.black)
    ]))
    elements.append(table)
    lower_price = price_range[0][1]
    higher_price =  price_range[0][2]

                elements.append(Paragraph(f"Predicted Price Range: INR
{(lower_price/100000):.2f} L - INR {(higher_price/100000):.2f} L",
styles["Heading2"]))
```

```
    elements.append(Paragraph("Depreciation Graph:", styles["Heading2"]))

    img_buffer = generate_depreciation_graph(input_data["M-Year"], price_range)

    image = Image(img_buffer, width=400, height=230)
    image.hAlign = 'CENTER'
    elements.append(image)

    elements.append(Spacer(1, 24))
    elements.append(Paragraph(f"Report generated on {datetime.now().strftime('%d %B %Y')}", styles["Italic"]))
     elements.append(Paragraph("Note: This valuation is an estimate based on current market trends.", styles["Italic"]))

    doc.build(elements)
    pdf_buffer.seek(0)
    return pdf_buffer

def generate_combined_response(input_data):
    """Generate both JSON data and PDF report"""
    try:
        # Generate predictions for current year and next 5 years
        price_predictions = []
        for i in range(0, 6):
            price_predictions.append(process_car_data(
                input_data['Brand'], input_data['Model'], input_data['M-Year'] - i,
                        input_data['KM-Driven'],  input_data['Mileage'],  input_data['Engine-Capacity'],
                input_data['Seats'], input_data['Fuel-Type'], input_data['Transmission']
            ))

        # Prepare JSON response data
        json_data = {
            "status": "success",
            "data": {
                "current_price": price_predictions[0][0],
                "price_range": [price_predictions[0][1], price_predictions[0][2]],
                "depreciation_forecast": [
                    {
                        "year": current_year + i,
                        "price_range": [price_predictions[i][1], price_predictions[i][2]]
                    } for i in range(len(price_predictions))
                ]
            }
        }

        # Generate PDF report
        pdf_buffer = generate_pdf_report(input_data, price_predictions)

        return json_data, pdf_buffer
```

```
    except Exception as e:
        raise e

app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": ["http://localhost:5173"]}})

@app.route('/process_car', methods=['POST'])
def process_car():
    try:
        data = request.json
          required_fields = ['Brand', 'Model', 'M-Year', 'KM-Driven', 'Mileage', 'Engine-
Capacity', 'Seats', 'Fuel-Type', 'Transmission']

        if not all(field in data for field in required_fields):
            return jsonify({
                "status": "error",
                "message": "Missing required fields",
                "required_fields": required_fields
                }), 400

    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```