**MINI PROJECT REPORT**

**On**

# REAL ESTATE PRICE PREDICTOR

**Submitted as partial fulfillment of**

**MASTER OF COMPUTER APPLICATION**

**DEGREE**

**SESSION 2024-25**

**By**

**ANUSHKA GUPTA**

**2300320140031**

**Under the guidance of**

**MS. SURBHI SHARMA**

**ASSISTANT PROFESSOR (Sr. Scale)**

**ABES ENGINEERING COLLEGE, GHAZIABAD (032)**

**AFFILIATED To Dr. A.P.J ABDUL KALAM TECHNICAL UNIVERSITY,**

**UTTAR PRADESH LUCKNOW**

# STUDENT DECLARATION

I hereby declare that the work being presented in this report entitled "**REAL ESTATE PRICE PREDICTOR"** is an authentic record of my own work carried out under the supervision of **"Mrs. Surbhi Sharma".**

**DATE:** **Signature of Student**

**Anushka Gupta**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**Signature of HOD**
**Prof. (Dr.) Devendra Kumar**
**HOD-MCA**

**signature of Internal**
**Mrs. Surbhi Sharma**
**Assistant Professor**
**(Sr. Scale)**

**Date:**

# ACKNOWLEDGEMENT

A software project is never the effort of an individual alone; it is the culmination of collective ideas, suggestions, reviews, and dedicated teamwork. I take this opportunity to express my heartfelt gratitude to all those who contributed to the successful completion of this project.

Primarily, I extend my sincere thanks to my project guide for her invaluable guidance, unwavering support, and constructive feedback throughout this endeavour. Her mentorship has been instrumental in navigating the complexities of the project and achieving its successful completion.

I am deeply grateful to the faculty and staff of my college for their assistance and encouragement despite their demanding schedules. A special note of appreciation goes **to Ms. Surbhi Sharma, Assistant Professor (Sr. Scale)**, for her insightful guidance, active support, and constant motivation, which have been pivotal to my progress.

Lastly, I express my profound gratitude to **Prof. (Dr.) Devendra Kumar, Head of the Department (MCA)**, for his visionary leadership, dedication to fostering an enriching academic environment, and efforts in instilling a professional mindset throughout my academic journey.

This project would not have been possible without the support, guidance, and encouragement of all those mentioned above, for which I am sincerely thankful.

**Signature of Student**
**ANUSHKA GUPTA**
**2300320140031**
**MCA-III Semester**

# ABSTRACT

**Real estate price predictor** plays a pivotal role in empowering buyers, sellers, and investors to make informed decisions in the dynamic property market. This project introduces a robust solution for accurately predicting real estate prices by considering key factors such as location, property size, number of rooms, amenities, and market trends.

The platform leverages advanced machine learning techniques, utilizing algorithms like Random Forest and Gradient Boosting, to deliver high predictor accuracy. A meticulously curated dataset undergoes pre-processing through normalization and feature engineering, enabling the model to uncover and learn complex relationships within the data.

Seamlessly integrated with a modern web interface, the platform allows users to input property details and obtain instant price predictors. Interactive visualizations further provide insights into pricing patterns and market fluctuations, aiding comprehensive trend analysis.

This project highlights the synergy between machine learning and practical applications, addressing challenges such as data variability, overfitting, and usability. By offering an intuitive and accessible interface, the platform becomes an invaluable tool for stakeholders in the real estate sector, enabling data-driven decision-making with confidence.

# TABLE CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

## 1.1 Overview

The real estate market is highly dynamic, with property prices influenced by factors like location, size, amenities, and market trends. Accurately predicting these prices is challenging, but essential for buyers, sellers, and investors to make informed decisions. This **Real Estate Price Predictor and Market Analysis Platform** leverages advanced machine learning techniques to deliver precise property valuations and actionable market insights, addressing the complexities of the real estate landscape.

Using algorithms like **Random Forest and Gradient Boosting**, the platform analyses curated datasets with preprocessing steps like data normalization and feature engineering to enhance predictor accuracy. A modern, user-friendly interface built with **HTML, CSS, JavaScript, and Bootstrap** allows users to input property details for instant price estimates. Additionally, interactive visualizations provide insights into market trends, enabling stakeholders to assess pricing patterns and make strategic decisions.

Designed for accessibility and reliability, the platform caters to buyers evaluating fair prices, sellers setting competitive rates, and investors analysing market dynamics. By merging advanced technology with real-world real estate needs, it transforms property evaluation into an efficient, accurate, and user-centric process, empowering stakeholders to navigate the market with confidence.

## 1.2 Objective

The primary objective of the Real Estate Price Predictor System is to create a user-friendly, efficient, and accurate platform that empowers users to make informed decisions in the dynamic real estate market. The detailed objectives are:

1. **Accurate Price Predictor:** Leverage machine learning models to analyse property features and provide precise real estate price estimates, aiding buyers, sellers, and investors.

2. **Market Insights**: Offer interactive visualizations to present real-time trends and historical data, enabling users to understand market dynamics.

3. **Accessibility**: Design an intuitive interface that caters to users with varying levels of real estate knowledge, ensuring ease of use and accessibility.

4. **Decision Support:** Equip stakeholders with data-driven tools to evaluate property values, identify trends, and make informed investment or selling decisions.

5. **Scalability:** Establish a foundation for future enhancements, such as incorporating additional property features or integrating APIs for expanded market analysis.

6. **Reliability:** Ensure a secure and efficient platform that delivers accurate predictors and insights, addressing the diverse needs of the real estate industry.

## 1.3 Need of the Project

The dynamic and ever-changing real estate market presents significant challenges for individuals and businesses trying to evaluate property prices accurately. Without reliable, data-driven tools, buyers, sellers, and investors often rely on guesswork or inconsistent information. Key challenges faced by stakeholders include:

1. **Complex Property Valuation:** Accurately estimating property prices involves analysing numerous factors such as location, size, amenities, and market trends, which can be overwhelming without advanced tools.

2. **Inconsistent and Outdated Data:** Market insights are often scattered or outdated, making it difficult for users to make informed decisions.

3. **Lack of Accessible Analytical Tools:** Many real estate analysis tools are either too complex or inaccessible for individuals without technical expertise.

## 1.4 Problem Statement

The real estate market is characterized by its dynamic nature, influenced by factors such as location, property size, amenities, and market trends. Buyers, sellers, and investors often face challenges in accurately estimating property values due to the lack of accessible, dependable, and data-driven tools. This results in:

1. **Market Uncertainty:** Lack of clarity on current market trends, leading to uninformed decisions.
2. **Time-Consuming Analysis**: Manual evaluation of property values and market trends is tedious and prone to inaccuracies.
3. **Overpricing or Underpricing**: Buyer’s risk overpaying for properties, while sellers may undervalue their assets without accurate price predictions.
4. **Limited Accessibility**: Most existing tools are either overly complex or not tailored to local markets, making them inaccessible to general users.

## 1.5 Scope

### 1.5.1 Technical Scope

This project involves the development of a **Real Estate Price Predictor and Analysis Platform** using advanced machine learning techniques and modern web technologies:

1. **Machine Learning Models:** Algorithms like Random Forest and Gradient Boosting are employed for accurate price predictors based on key factors such as property location, size, and amenities.

2. **Web Development Tools:** Technologies like **HTML, CSS, JavaScript, and Bootstrap** are used to create an intuitive and responsive interface.

3. **Data Preprocessing:** Techniques like normalization, feature engineering, and handling missing values ensure the model processes data accurately and efficiently.

4. **Visualization Tools:** Interactive charts and graphs enable users to explore market trends and pricing patterns effectively.

### 1.5.2 Operational Scope

The operational scope focuses on providing a seamless user experience for all stakeholders, including buyers, sellers, and investors. Key operational aspects include:

1. **Accurate Price Predictor:** Allowing users to input property details and receive reliable price estimates.
2. **Market Analysis:** Offering insights into trends and fluctuations through real-time and historical data visualizations.
3. **Scalability and Reliability:** Ensuring the platform can handle increasing user demands by leveraging scalable infrastructure.
4. **User-Friendly Experience:** Simplifying property evaluation for users with varying levels with knowledge.

# 2. Feasibility Study

After conducting a detailed study and analysis of the existing requirements and functionalities for the REAL ESTATE PRICE PREDICTOR System, the next step is to perform a feasibility study. Feasibility studies evaluate all possible ways to provide an effective solution to the given problem. The proposed solution should meet all user requirements and offer flexibility to accommodate future enhancements or changes. The feasibility study for this project includes the following aspects:

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

## 2.1 Technical Feasibility

The **technical feasibility** of the Real Estate Price Predictor and Analysis Platform is crucial for its long-term success and reliability. The platform utilizes a combination of innovative technologies to ensure that it performs efficiently and offers users a seamless experience.

- **Machine Learning Algorithms**: The platform uses state-of-the-art machine learning models like **Random Forest** and **Gradient Boosting**. These algorithms are chosen for their robustness and ability to handle complex, nonlinear data, which is typical in real estate pricing. The **Random Forest** algorithm is effective in handling large datasets with high variance, while **Gradient Boosting** offers high predictor accuracy by focusing on model errors. Both algorithms are well-suited for regression tasks, ensuring accurate property price predictors.

- **Web Technologies**: The frontend of the platform is designed using widely adopted web technologies such as **HTML**, **CSS**, **JavaScript**, and **Bootstrap**. HTML and CSS are used to structure and style the content, ensuring a clean and organized layout. **JavaScript** enables dynamic content interaction, making the platform user-friendly and responsive. **Bootstrap**, a popular CSS framework, ensures the platform's interface is both **responsive** and **mobile-friendly**, adapting to various screen sizes and devices. This guarantees that users have a smooth experience whether they access the platform from a desktop, tablet, or smartphone.

- **Data Visualization Tools**: To help users interpret market trends and property price predictors, the platform integrates powerful data visualization libraries like **Matplotlib**. These tools provide interactive, intuitive visualizations that make complex data easier to understand.

- **Cloud Infrastructure**: Hosting the platform on **cloud services** like **AWS** or **Azure** ensures scalability, reliability, and security. Cloud hosting enables the platform to scale resources dynamically based on user demand. These services provide robust **data storage**, **processing power**, and **high availability**, minimizing the risk of downtime and ensuring fast access to the platform's features. Additionally, both AWS and Azure offer powerful **AI and ML services**, which can be integrated into the platform for future enhancements.

## 2.2 Operational Feasibility

The **operational feasibility** of the platform is assessed by evaluating its ability to meet user requirements, maintain consistent performance, and scale as demand increases.

- **Accessibility**: One of the core strengths of the platform is its **user-friendly interface**, which is designed to cater to a wide range of users, including property buyers, sellers, investors, and real estate agents. The platform's intuitive design allows users with limited technical knowledge to easily input data, interpret predictors, and make decisions based on the analysis. Additionally, the platform supports multiple languages and currencies, ensuring that users from different regions can use it effectively.

- **Scalability**: The platform is built with scalability in mind, ensuring that as the user base grows, the platform can accommodate increased traffic and data processing without compromising performance. By leveraging cloud hosting, the platform can automatically scale its resources, such as server capacity and database storage, based on real-time demand. This ensures smooth operation, even during peak usage times.

- **Practical Utility**: The platform provides practical value by offering accurate property price predictors and in-depth market trend analysis. These features help real estate stakeholders make informed decisions, reducing uncertainty in property transactions. The platform's ability to predict property values with high accuracy based on historical data, market trends, and location-specific factors enhances its utility for decision-making.

## 2.3 Economic Feasibility

The **economic feasibility** of the platform is assessed in terms of development costs, potential revenue streams, and its position in the market.

- **Cost Efficiency**: The platform benefits from utilizing open-source technologies like Python for machine learning and **JavaScript frameworks** for the frontend, which significantly reduces initial development costs. The use of **cloud services** further minimizes infrastructure and maintenance costs, as cloud providers offer pay-as-you-go pricing models, allowing the platform to scale its operations efficiently without incurring unnecessary expenses.

- **Revenue Opportunities**: The platform has several potential revenue streams, which include Premium Features, Subscription Model, Advertisements and Partnerships.

- **Market Potential**: The global market for real estate technology is expanding rapidly, with stakeholders seeking innovative solutions to make more informed decisions. By offering a data-centric platform that combines predictive analytics, trend analysis, and visualization, the project addresses a significant demand for reliable, technology-driven insights. The growing reliance on **big data** and **artificial intelligence** in real estate positions the platform to tap into a lucrative market that is expected to continue growing in the coming years.

# 3.Software Requirement Specifications (SRS)

## 3.1 Introduction

### 3.1.1 Purpose of the SRS Document

The purpose of this document is to outline the requirements, functionalities, and specifications for the Real Estate Price Predictor system. This system aims to provide an accurate, efficient, and user-friendly platform for predicting property prices based on various influential factors such as location, size, amenities, and market trends. This SRS serves as a foundational guide for the development and deployment of the system, ensuring that it aligns with its objectives and delivers optimal performance.

### 3.1.2 Scope of the Project

The Real Estate Price Predictor project is designed to assist buyers, sellers, and investors in the real estate market by offering accurate price estimations and insights into market trends. By leveraging advanced machine learning algorithms and data analytics, the platform simplifies the complexities of property valuation and empowers users to make informed decisions. This web-based project includes features such as property price predictor, trend visualization, and user-friendly interactions. The system is intended for local deployment and offers a scalable foundation for future enhancements, such as cloud integration and expanded datasets.

### 3.1.3 Overview of the Real Estate Price Predictor System

The system enables users to input property details such as location, size, number of rooms, and other relevant features through an intuitive web interface. The backend processes this data using pre-trained machine learning models to provide price predictors. Additionally, the frontend displays interactive charts and insights about market trends, helping users understand pricing patterns over time. The platform is designed to be accessible to users with varying levels of technical expertise, ensuring usability across a broad audience.

## 3.2 System Overview

### 3.2.1 System Architecture

The system follows a client-server architecture:

- **Client-Side**: Provides a modern, responsive user interface for data input, displaying predictors, and visualizing market trends.
- **Server-Side**: Handles data preprocessing, predictor generation, and serves static content, ensuring secure and efficient operations.

### 3.2.2 High-Level Functionality and Features

- **Data Input**: Users can input property details, including location, size, number of rooms, amenities, and other features.
- **Data Preprocessing**: The system preprocesses user-provided data, ensuring it is clean and formatted for predictor.
- **Feature Selection**: Identifies and utilizes the most relevant features for accurate predictors.
- **Predictor Generation**: Employs machine learning models like Random Forest and Gradient Boosting to deliver precise property price estimates.
- **Result Visualization**: Displays predictors along with graphical insights into market trends and pricing patterns.
- **Market Analysis**: Provides users with real-time updates and historical data visualizations to understand market dynamics.

### 3.2.3 Assumptions and Constraints

**Assumptions:**

- Users provide accurate and complete property details.
- The pre-trained machine learning models are well-trained and optimized for real estate datasets.

**Constraints:**

- The system operates locally, relying on the user’s computational resources.
- Predictors are subject to the quality and comprehensiveness of the input data.
- No external database is utilized; data is processed in real-time without storage.

## 3.3 Functional Requirements

### 3.3.1 Data Input

- The system must allow users to input key property details such as location, size, number of rooms, and amenities through an intuitive interface.
- Input fields must include validations to ensure data accuracy and completeness.

### 3.3.2 Data Preprocessing

- User-provided data should be cleaned and formatted for compatibility with the predictor model.
- Missing or inconsistent data should be handled effectively, using default values or omission strategies.

### 3.3.3 Feature Selection

- The system must utilize relevant property features such as location, size, and amenities for generating predictors.
- Feature selection ensures the inclusion of impactful data points to enhance model accuracy.

### 3.3.4 Predictor Generation

- A pre-trained machine learning model must analyse user input and predict property prices with high accuracy.
- Predictors should account for dynamic factors, including current market trends and location-specific data.

### 3.3.5 Result Visualization

- The system should display clear and actionable predictor results, including a detailed price estimate.
- Visual aids such as bar graphs, line charts, and trend curves should be used to present market insights.

## 3.4 Non-Functional Requirements

### 3.4.1 Performance Requirements

- **Response Time**: The system must generate predictors within 2-3 seconds of data input.
- **Scalability**: The platform should be designed to handle an increasing number of users and data points effectively.
- **Accuracy**: The machine learning models must maintain a predictor accuracy of at least 85%.

### 3.4.2 Security and Privacy Requirements

- **Data Confidentiality**: User data should be processed locally, with no external sharing or storage.
- **Secure Processing**: Ensure that all data is handled securely, minimizing the risk of unauthorized access.

### 3.4.3 Reliability and Availability Requirements

- **System Reliability**: The system must be robust, ensuring minimal errors during usage.
- **Fault Tolerance**: Clear feedback should be provided for invalid inputs, allowing users to correct errors without restarting.
- **System Uptime**: The system must remain operational whenever the local deployment is active.

## Real-World Context

In real-world scenarios, property buyers often struggle to make decisions due to inconsistent and incomplete data. Sellers face challenges in determining competitive prices, while investors require tools to analyse market trends effectively. By providing accurate predictors and real-time insights, this system bridges the gap between data and decision-making. It serves as a reliable resource for understanding pricing dynamics, aiding all stakeholders in navigating the complexities of the real estate market with confidence.

## 3.5 Hardware Requirements

### 3.5.1 User-Side Hardware Requirements

- **Device**: A computer, tablet, or smartphone with a modern web browser.
- **Internet Connection**: Stable connectivity for accessing the application.
- **Screen**: Minimum 10-inch display with 1024x768 resolution or higher.

### 3.5.2 Developer-Side Hardware Requirements

- **Development Machine**:
    - **Processor**: Dual-core or higher (e.g., Intel Core i5).
    - **Memory**: At least 8 GB of RAM.
    - **Storage**: 20 GB of free space.
- **Operating System**: Linux (e.g., Ubuntu), Windows 10/11, or macOS.
- **Tools**: IDE (e.g., VS Code), terminal for managing dependencies, and a local web server (Flask) for testing.

## 3.6 Software Requirements

### 3.6.1 User-Side Software Requirements

- **Web Browser**:
  Users need a web browser such as Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari to access the Real Estate Price Predictor platform. The browser must support modern web standards like HTML5, CSS3, and JavaScript.
- **Cross-Device Compatibility**:
  The user interface should function seamlessly on desktop and mobile browsers, ensuring accessibility on laptops, tablets, and smartphones.
- **Input Capability**:
  Users must have access to input devices, such as a keyboard, mouse, or touchscreen, to interact with the application and enter property details.

### 3.6.2 Developer-Side Software Requirements

- **Programming Language**:
  The application backend is developed using Python due to its robust ecosystem for data science and web development.
- **Web Framework**:
  Flask is used as the web framework for its lightweight nature, making it ideal for developing a modular and efficient application.
- **Python Libraries**:
  Several Python libraries are utilized to support functionality:
  - **pandas**: For efficient data manipulation and preprocessing tasks.
  - **NumPy**: To perform numerical operations essential for machine learning workflows.
  - **scikit-learn**: For integrating machine learning models like Random Forest and Gradient Boosting.
  - **matplotlib/seaborn**: For creating optional visualizations of market trends and data analysis.
- **Frontend Technologies**:
  The user interface is built using:
  - HTML and CSS for static content and styling.
  - JavaScript and Bootstrap for dynamic elements and responsive design.

- **Integrated Development Environment (IDE)**:
  Developers are recommended to use tools like:
  - PyCharm or Visual Studio Code for debugging and code development.
  - Jupyter Notebook for prototyping machine learning models.
- **Version Control System**:
  Git is employed for version tracking, collaborative development, and managing various stages of the project.

# 4. DESIGN

**Platform Overview**

The Real Estate Price Predictor platform is designed to provide an intuitive, user-friendly experience, featuring a modern, responsive interface. The homepage acts as a centralized dashboard, offering:

- **Real-time Price Predictors**: Instant and accurate property price estimates based on user inputs.
- **Market Trends Insights**: Key trends and fluctuations in the real estate market for informed decision-making.
- **Top Locations**: Highlights of high-demand areas with current pricing trends.

**Key Features**

1. **Price Estimator**
   - Users can input property details such as location, size, and amenities to receive accurate price predictors instantly.
2. **Interactive Market Analysis**
   - Data Visualizations: Powered by libraries like Chart.js, users can explore historical market data and trends over customizable timeframes (daily, monthly, yearly).
   - **Tools to compare property prices across various locations and property types.**
3. **Trend Analysis & Insights**
   - Displays regional property trends, including high-growth areas and price fluctuations, enabling better investment planning.

**Navigation and Usability**

The platform is designed for seamless usability with:

- **Streamlined Navigation:** Clearly defined sections to guide users effortlessly.
- **Intuitive Data Input:** Simple forms with tooltips to assist users in entering property details correctly.
- **Real-time Updates:** Ensures all predictors and market data reflect the latest information.

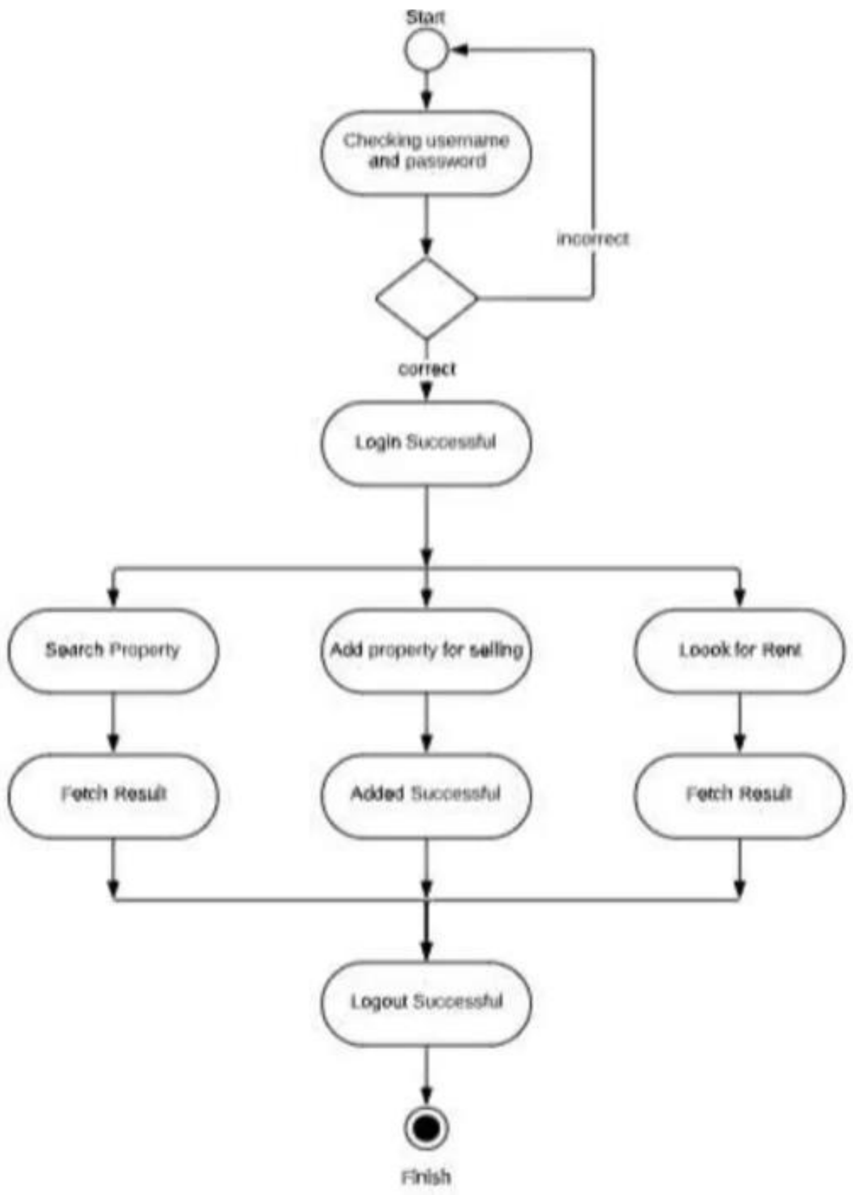
**Security and Performance**

- **HTTPS Encryption:** Secures all data exchanges between users and the platform.
- **Input Validation:** Ensures accurate data entry while protecting against malicious inputs.
- **Cloud Hosting (AWS/Netlify):** Provides a scalable, reliable backend capable of handling.

## 4.1 Flowchart

A flowchart is a visual representation of a process or algorithm using various shapes, symbols, and arrows to illustrate the sequence of steps or actions. It provides a clear and concise way to depict the flow of information, decision points, and the order of operations within a system or procedure. Flowcharts are widely used in various fields such as software development, business processes, project planning, and problem-solving. They serve as a valuable tool for understanding, documenting, and communicating complex processes in an easily understandable and systematic manner.

## Flowchart of Real Estate Price Predictor

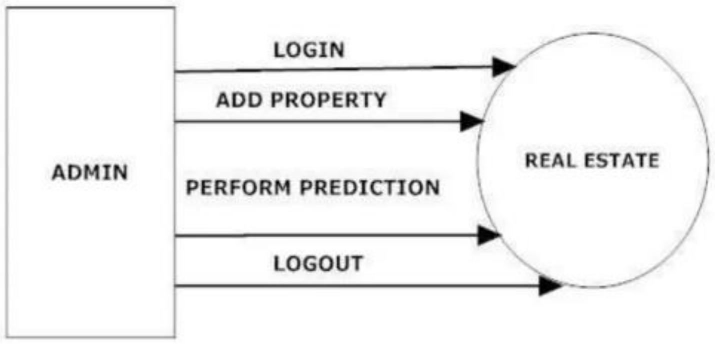


## 4.2 DATA FLOW DIAGRAM

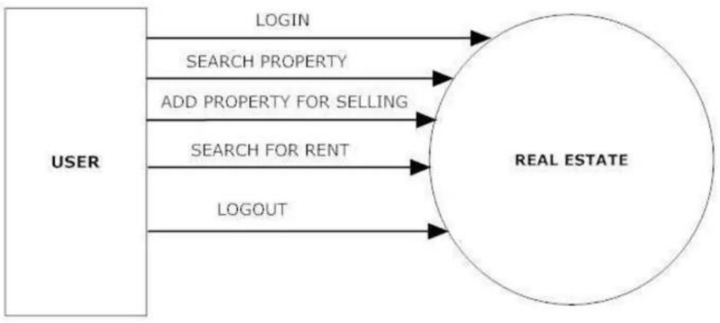
A data flow diagram (DFD) is a diagram that describes the flow of data and the process that change dat11a throughout a system. It’s a structured analysis and design tool that can be used for flow charting in place of or in association with information. The DFD reviews the current system, prepares input and output specification, specifies the implementation plan etc. Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

### Context Level Diagram (0 Level)

A Level 0 Data Flow Diagram (DFD), also known as a Context Diagram, provides a high-level view of a system by illustrating the interactions between the system and external entities. It represents the system as a single process and showcases the flow of data between the system and its external entities. The primary purpose of a Level 0 DFD is to offer a concise and easily understandable overview of the system's scope and boundaries.



DFD level 0(admin)

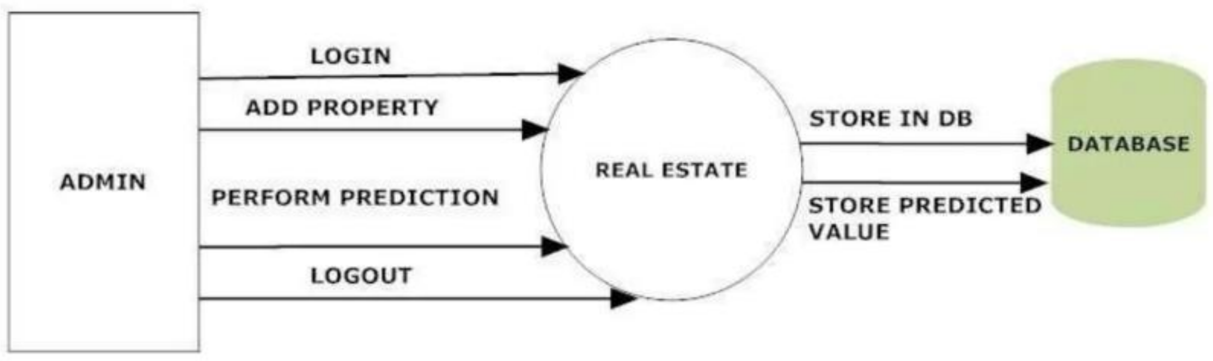


DFD level 0(user)

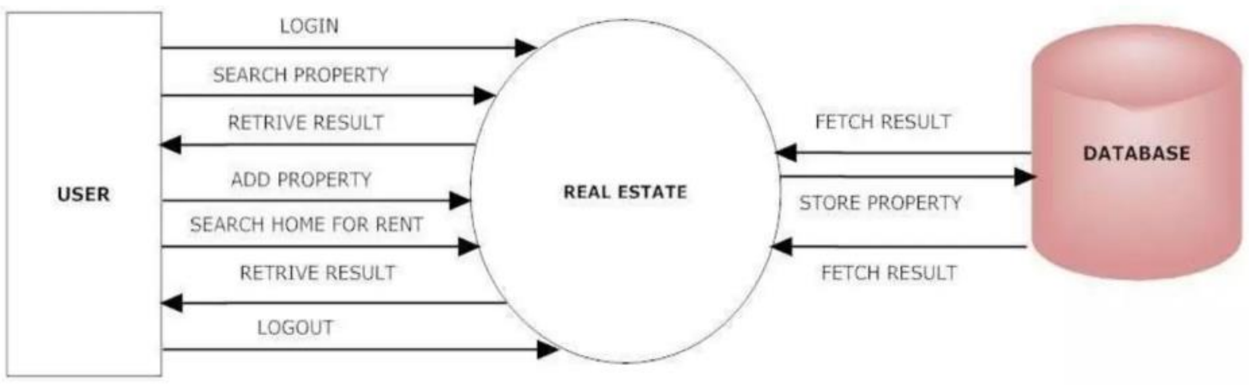
## Data Flow Diagram (1 Level)

A Level 1 Data Flow Diagram (DFD) is a visual representation that provides a more detailed view of the processes and data flows within a system compared to a Level 0 DFD. At Level 1, the diagram expands on the processes identified in the Level 0 DFD and decomposes them into sub processes, offering a more granular depiction of the system's functionality.

The Level 1 DFD acts as an intermediary step between the high-level overview of the Level 0 DFD and the more detailed representations that might follow in subsequent levels. It helps in breaking down complex processes into manageable components, making it easier for analysts and stakeholders to understand the system's inner workings.



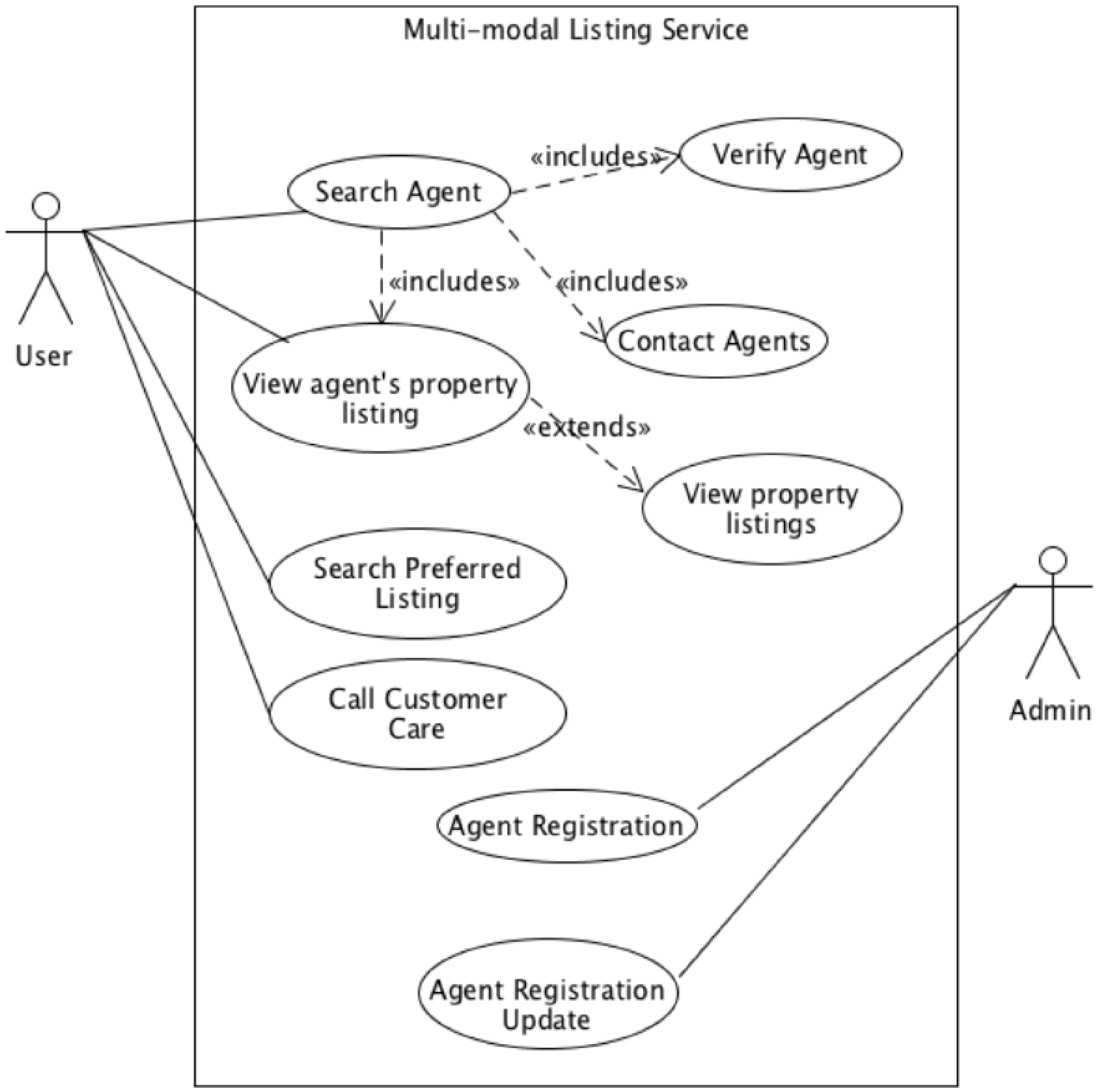
DFD level 1(admin)



DFD level 1(user)

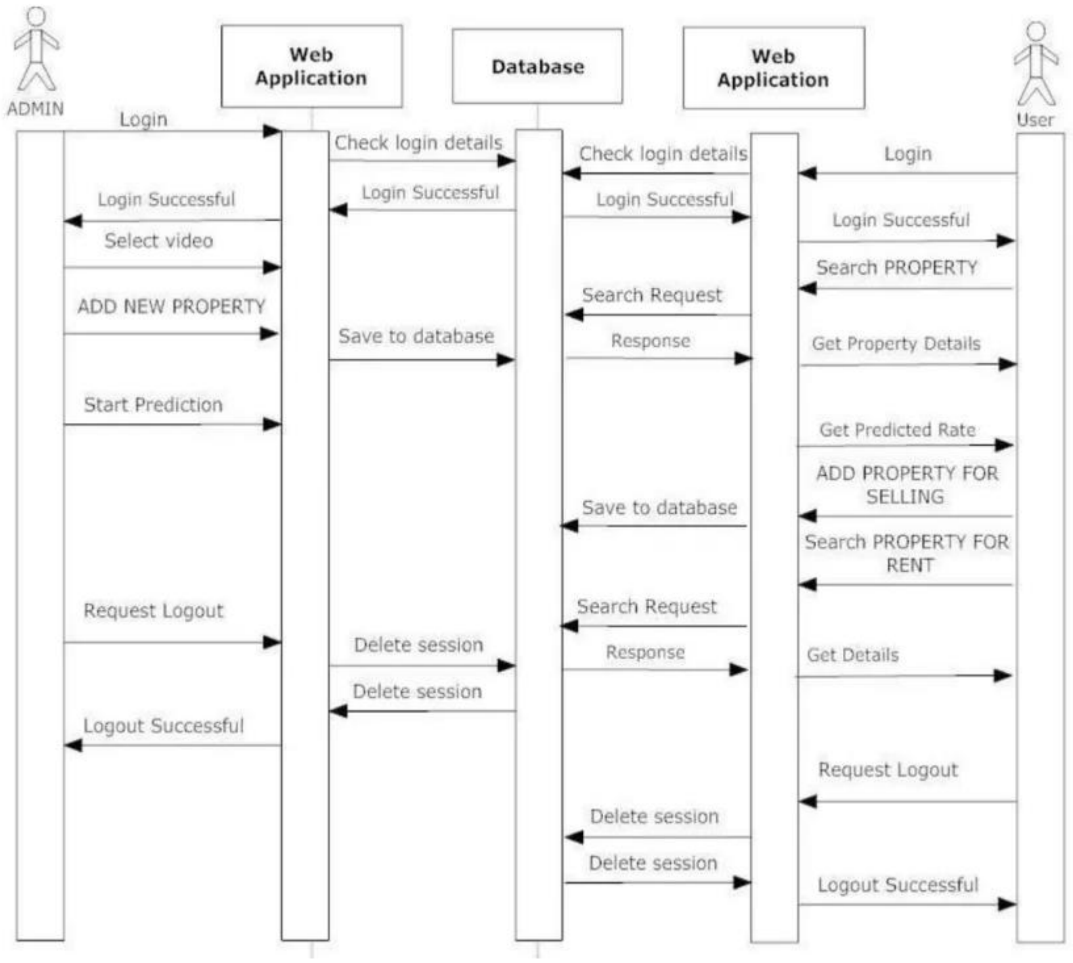
## 4.3 USE CASE DIAGRAM

A case diagram is a graphical representation that illustrates how a system interacts with external entities (actors) and showcases the various ways these entities can interact with the system. The primary purpose of a case diagram is to capture and communicate the functional requirements of a system in a clear and visual manner.

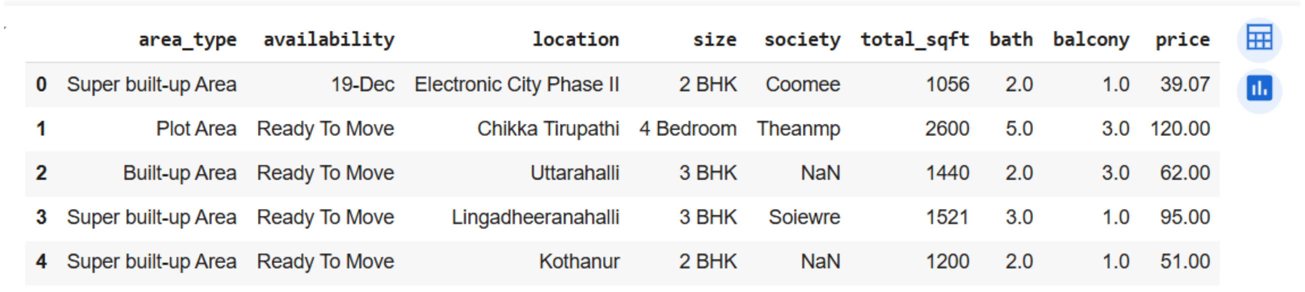


## 4.4 SEQUENCE DIAGRAM

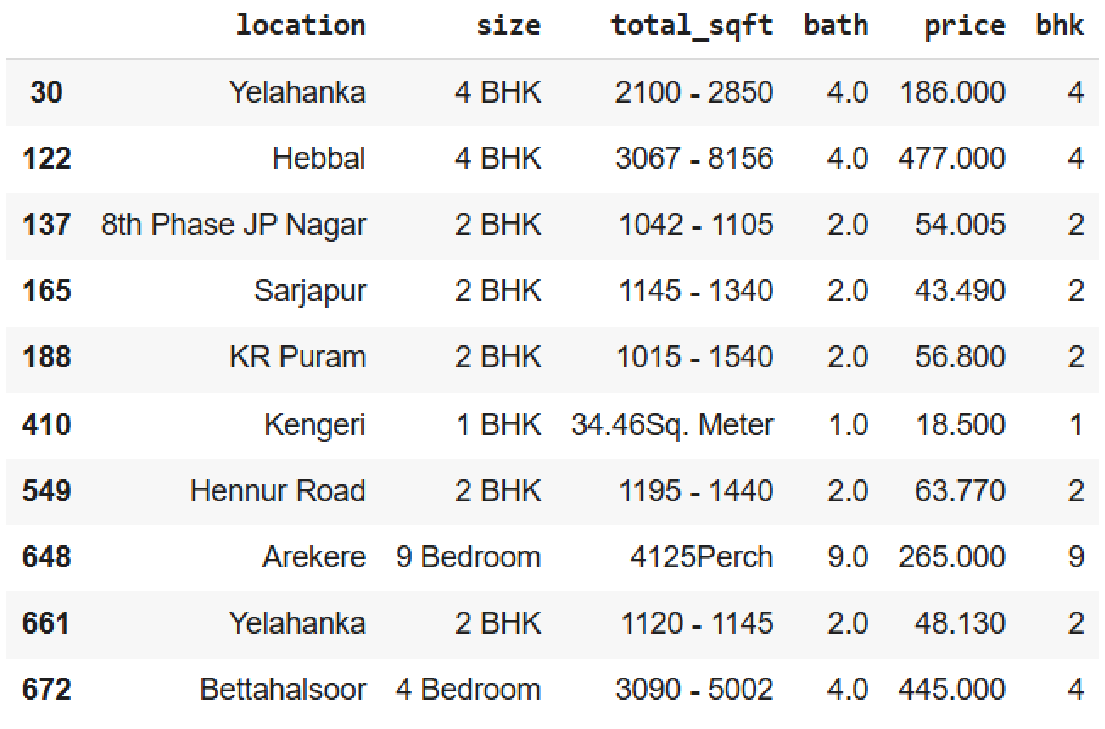
A sequence diagram is a type of interaction diagram that depicts how objects or entities in a system interact with each other over time to accomplish a particular process or workflow. It visually represents the sequence of messages or events exchanged between participants (such as actors, objects, or components) in a linear time-based order.



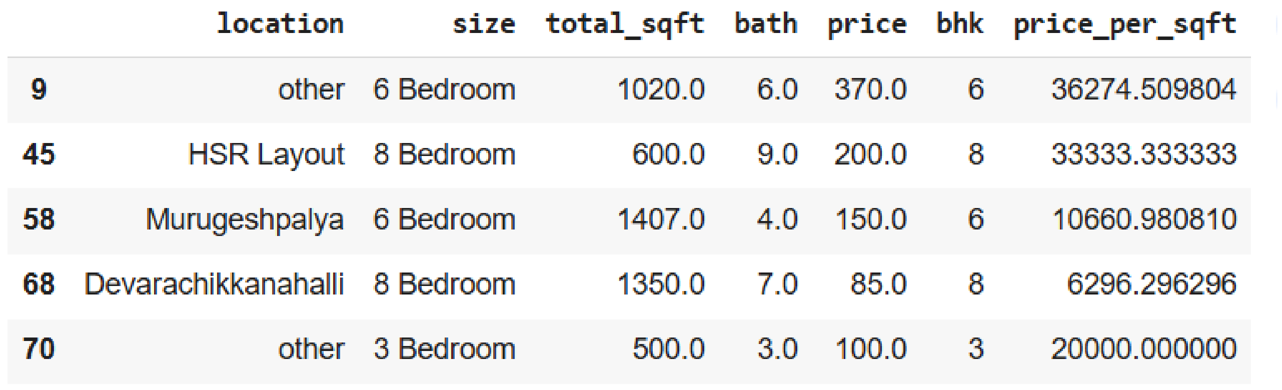
## 4.5 DATA



| | area_type | availability | location | size | society | total_sqft | bath | balcony | price |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Super built-up Area | 19-Dec | Electronic City Phase II | 2 BHK | Coomee | 1056 | 2.0 | 1.0 | 39.07 |
| 1 | Plot Area | Ready To Move | Chikka Tirupathi | 4 Bedroom | Theanmp | 2600 | 5.0 | 3.0 | 120.00 |
| 2 | Built-up Area | Ready To Move | Uttarahalli | 3 BHK | NaN | 1440 | 2.0 | 3.0 | 62.00 |
| 3 | Super built-up Area | Ready To Move | Lingadheeranahalli | 3 BHK | Soiewre | 1521 | 3.0 | 1.0 | 95.00 |
| 4 | Super built-up Area | Ready To Move | Kothanur | 2 BHK | NaN | 1200 | 2.0 | 1.0 | 51.00 |

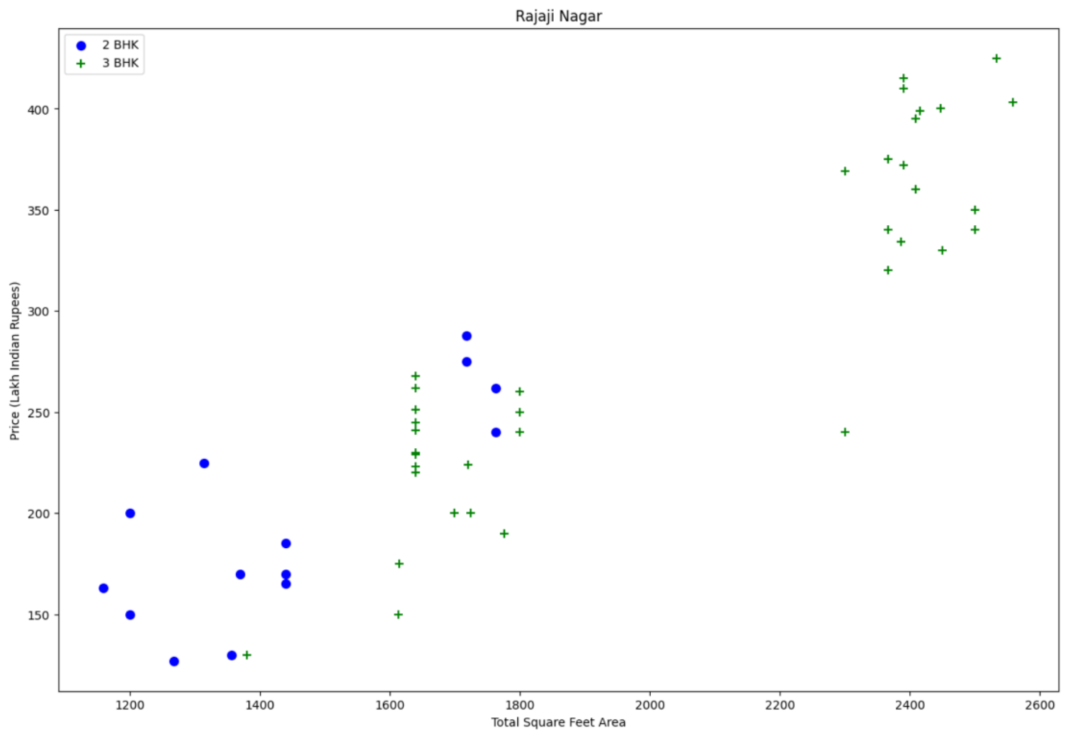


| | location | size | total_sqft | bath | price | bhk |
|---|---|---|---|---|---|---|
| 30 | Yelahanka | 4 BHK | 2100 - 2850 | 4.0 | 186.000 | 4 |
| 122 | Hebbal | 4 BHK | 3067 - 8156 | 4.0 | 477.000 | 4 |
| 137 | 8th Phase JP Nagar | 2 BHK | 1042 - 1105 | 2.0 | 54.005 | 2 |
| 165 | Sarjapur | 2 BHK | 1145 - 1340 | 2.0 | 43.490 | 2 |
| 188 | KR Puram | 2 BHK | 1015 - 1540 | 2.0 | 56.800 | 2 |
| 410 | Kengeri | 1 BHK | 34.46Sq. Meter | 1.0 | 18.500 | 1 |
| 549 | Hennur Road | 2 BHK | 1195 - 1440 | 2.0 | 63.770 | 2 |
| 648 | Arekere | 9 Bedroom | 4125Perch | 9.0 | 265.000 | 9 |
| 661 | Yelahanka | 2 BHK | 1120 - 1145 | 2.0 | 48.130 | 2 |
| 672 | Bettahalsoor | 4 Bedroom | 3090 - 5002 | 4.0 | 445.000 | 4 |

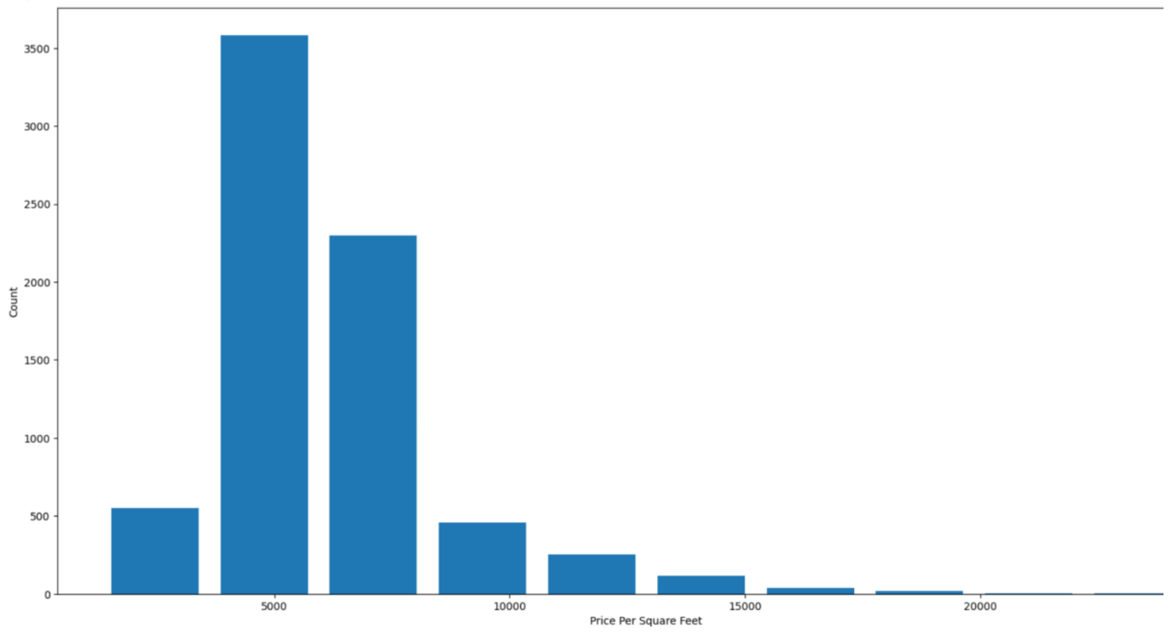


| | location | size | total_sqft | bath | price | bhk | price_per_sqft |
|---|---|---|---|---|---|---|---|
| 9 | other | 6 Bedroom | 1020.0 | 6.0 | 370.0 | 6 | 36274.509804 |
| 45 | HSR Layout | 8 Bedroom | 600.0 | 9.0 | 200.0 | 8 | 33333.333333 |
| 58 | Murugeshpalya | 6 Bedroom | 1407.0 | 4.0 | 150.0 | 6 | 10660.980810 |
| 68 | Devarachikkanahalli | 8 Bedroom | 1350.0 | 7.0 | 85.0 | 8 | 6296.296296 |
| 70 | other | 3 Bedroom | 500.0 | 3.0 | 100.0 | 3 | 20000.000000 |

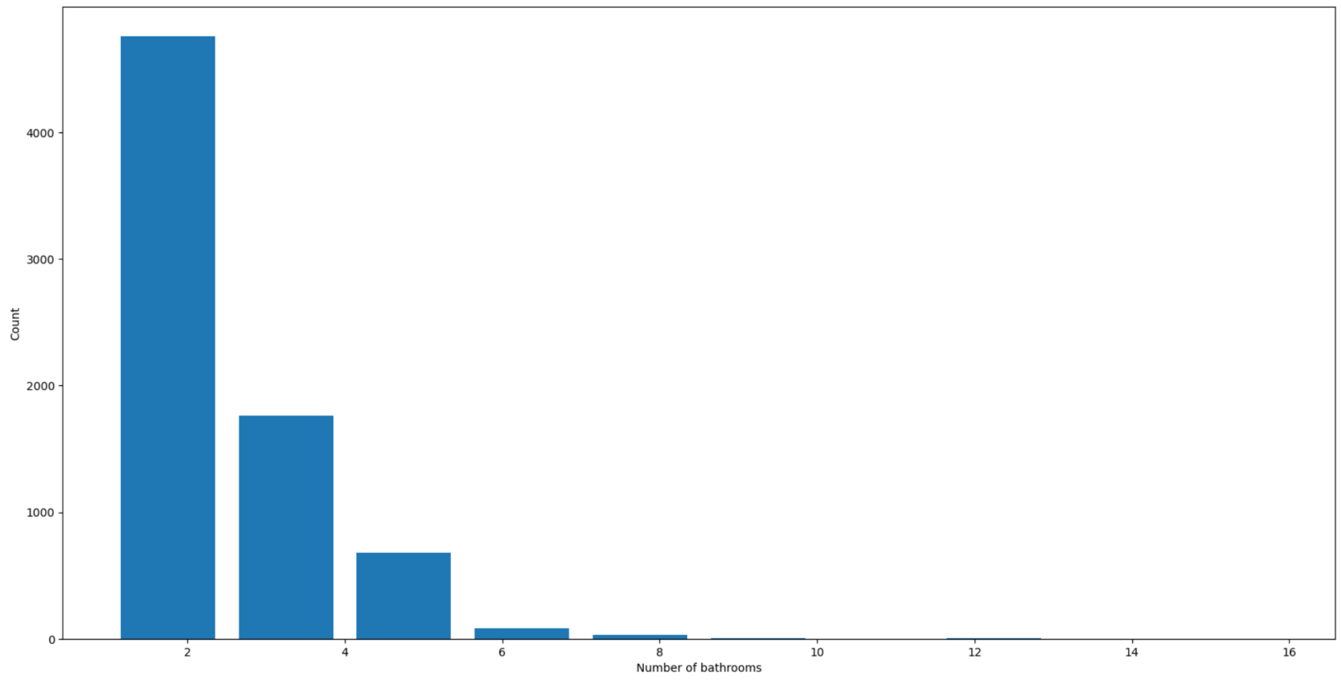
|   | location | size | total_sqft | bath | price | bhk | price_per_sqft |
|---|---|---|---|---|---|---|---|
| **0** | Electronic City Phase II | 2 BHK | 1056.0 | 2.0 | 39.07 | 2 | 3699.810606 |
| **1** | Chikka Tirupathi | 4 Bedroom | 2600.0 | 5.0 | 120.00 | 4 | 4615.384615 |
| **2** | Uttarahalli | 3 BHK | 1440.0 | 2.0 | 62.00 | 3 | 4305.555556 |
| **3** | Lingadheeranahalli | 3 BHK | 1521.0 | 3.0 | 95.00 | 3 | 6245.890861 |
| **4** | Kothanur | 2 BHK | 1200.0 | 2.0 | 51.00 | 2 | 4250.000000 |
| **5** | Whitefield | 2 BHK | 1170.0 | 2.0 | 38.00 | 2 | 3247.863248 |
| **6** | Old Airport Road | 4 BHK | 2732.0 | 4.0 | 204.00 | 4 | 7467.057101 |
| **7** | Rajaji Nagar | 4 BHK | 3300.0 | 4.0 | 600.00 | 4 | 18181.818182 |
| **8** | Marathahalli | 3 BHK | 1310.0 | 3.0 | 63.25 | 3 | 4828.244275 |
| **9** | other | 6 Bedroom | 1020.0 | 6.0 | 370.00 | 6 | 36274.509804 |



```
Text(0, 0.5, 'Count')
```



```
Text(0, 0.5, 'Count')
```



# 5. GUI

A Graphical User Interface (GUI) enables users to interact with systems, applications, or devices through visual and interactive components rather than relying on text-based commands.

## Key Features of a GUI:

1. **Graphical Elements:**

    - **Homepage Dashboard:** Displays key features like real-time property prices, market trends, and search options.
    - **Interactive Charts:** Visualizes price trends and historical data using sliders and drop-down menus for customization.
    - **Search and Filter Options:** Easy-to-use text boxes and filters to refine property searches by location, price range, or property type.
    - **Buttons and Icons:** Clearly labelled icons and buttons for navigating between features, such as predictors, visualizations, and settings.

2. **Point-and-Click Interaction:** Users can easily interact with the platform using a mouse, keyboard, or touch screen to view predictors, select filters, or analyse data.
3. **Visual Feedback:** Instant visual responses, such as highlighting selected buttons, zoom effects on charts, or loading indicators for real-time data updates.
4. **Ease of Use:** Designed for users with diverse technical expertise, the interface prioritizes simplicity and clarity through tooltips, labels, and intuitive navigation.
5. **Multitasking:** Multiple features, like comparing properties and viewing charts, can be accessed simultaneously using separate panels or tabs.

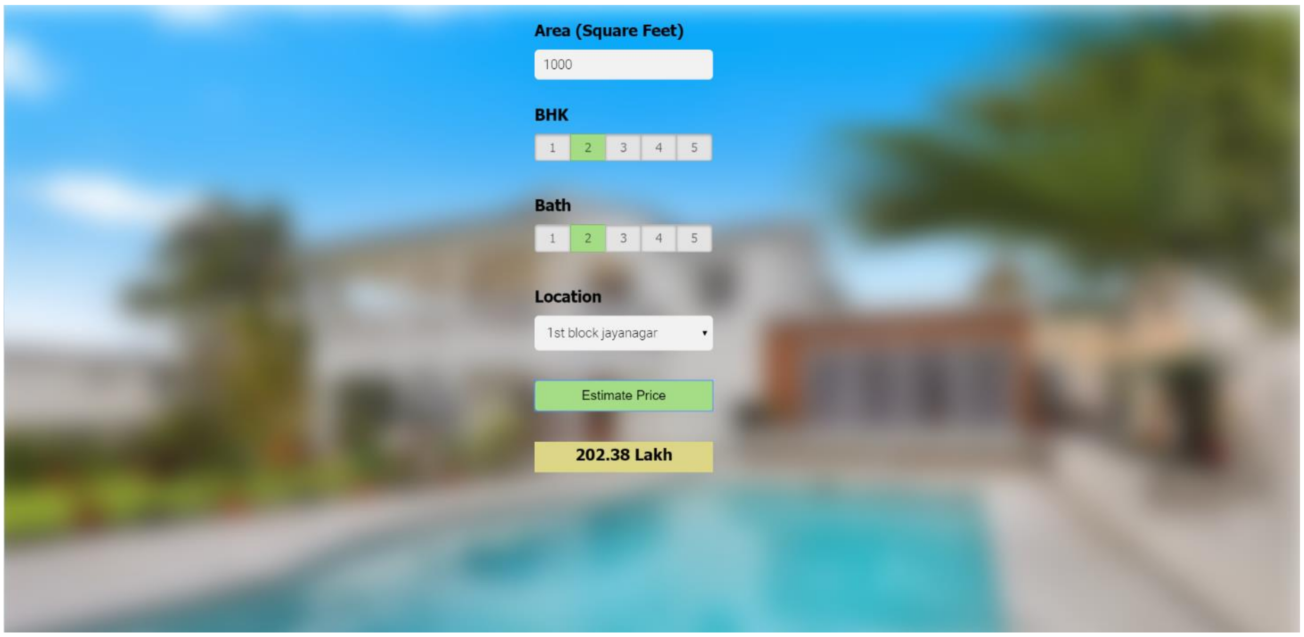
### Real-World Context and Benefits

The GUI simplifies the complex processes of property evaluation and market analysis by focusing on user experience. For instance:

- **Buyers** can explore market trends, input their desired property features, and receive price estimates to ensure informed decisions.
- **Sellers** can analyse market conditions and compare their property's valuation with related listings.
- **Investors** benefit from detailed trend visualizations and filtering tools, helping them identify lucrative opportunities or regions.

By combining functionality, aesthetics, and accessibility, the GUI empowers users with actionable insights while maintaining a seamless and enjoyable interface experience.

## 5.1 Modules Screenshot



Area (Square Feet)

1000

BHK

1 [✓ 2] 3 4 5

Bath

1 [✓ 2] 3 4 5

Location

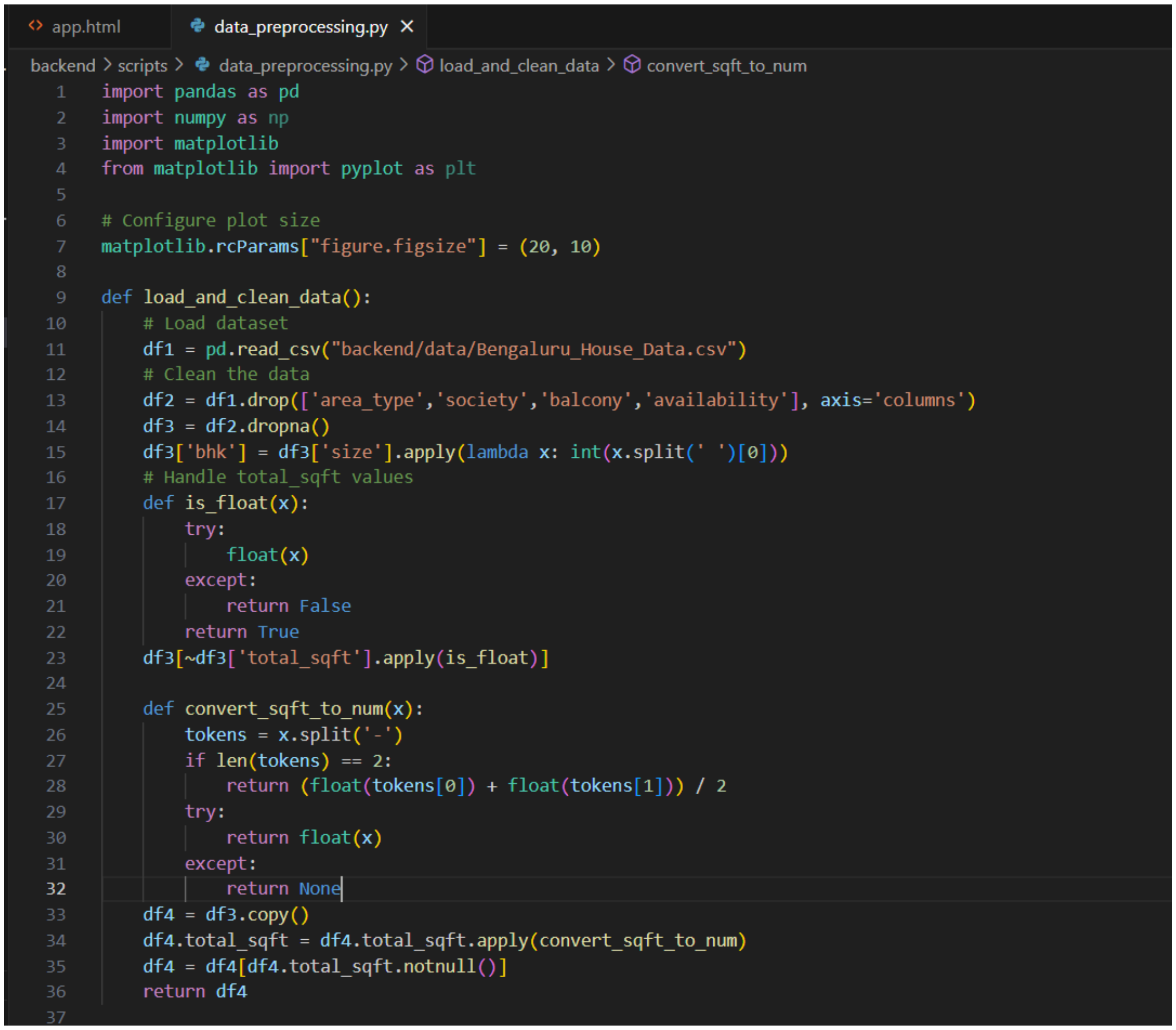
1st block jayanagar

202.38 Lakh

## 6. Coding:

### BACKEND

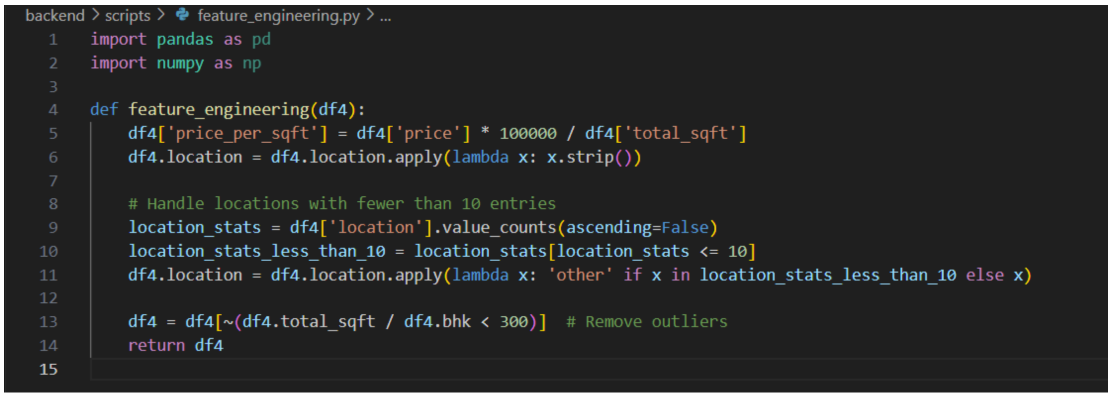
### 6.1 Data preprocessing



app.html
data_preprocessing.py
backend > scripts > data_preprocessing.py > load_and_clean_data > convert_sqft_to_num

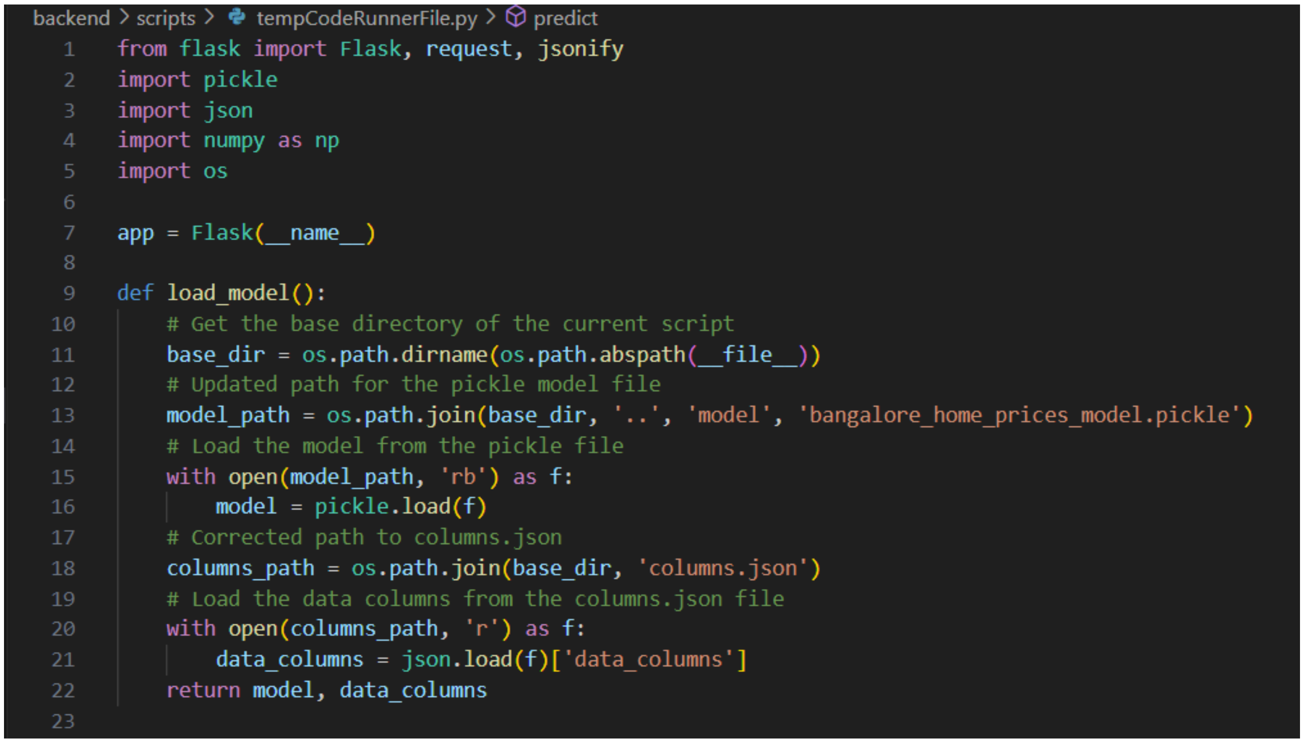
```python
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt

# Configure plot size
matplotlib.rcParams["figure.figsize"] = (20, 10)

def load_and_clean_data():
    # Load dataset
    df1 = pd.read_csv("backend/data/Bengaluru_House_Data.csv")
    # Clean the data
    df2 = df1.drop(['area_type','society','balcony','availability'], axis='columns')
    df3 = df2.dropna()
    df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
    # Handle total_sqft values
    def is_float(x):
        try:
            float(x)
        except:
            return False
        return True
    df3[~df3['total_sqft'].apply(is_float)]

    def convert_sqft_to_num(x):
        tokens = x.split('-')
        if len(tokens) == 2:
            return (float(tokens[0]) + float(tokens[1])) / 2
        try:
            return float(x)
        except:
            return None
    df4 = df3.copy()
    df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
    df4 = df4[df4.total_sqft.notnull()]
    return df4

```

### 6.2 Feature Engineering



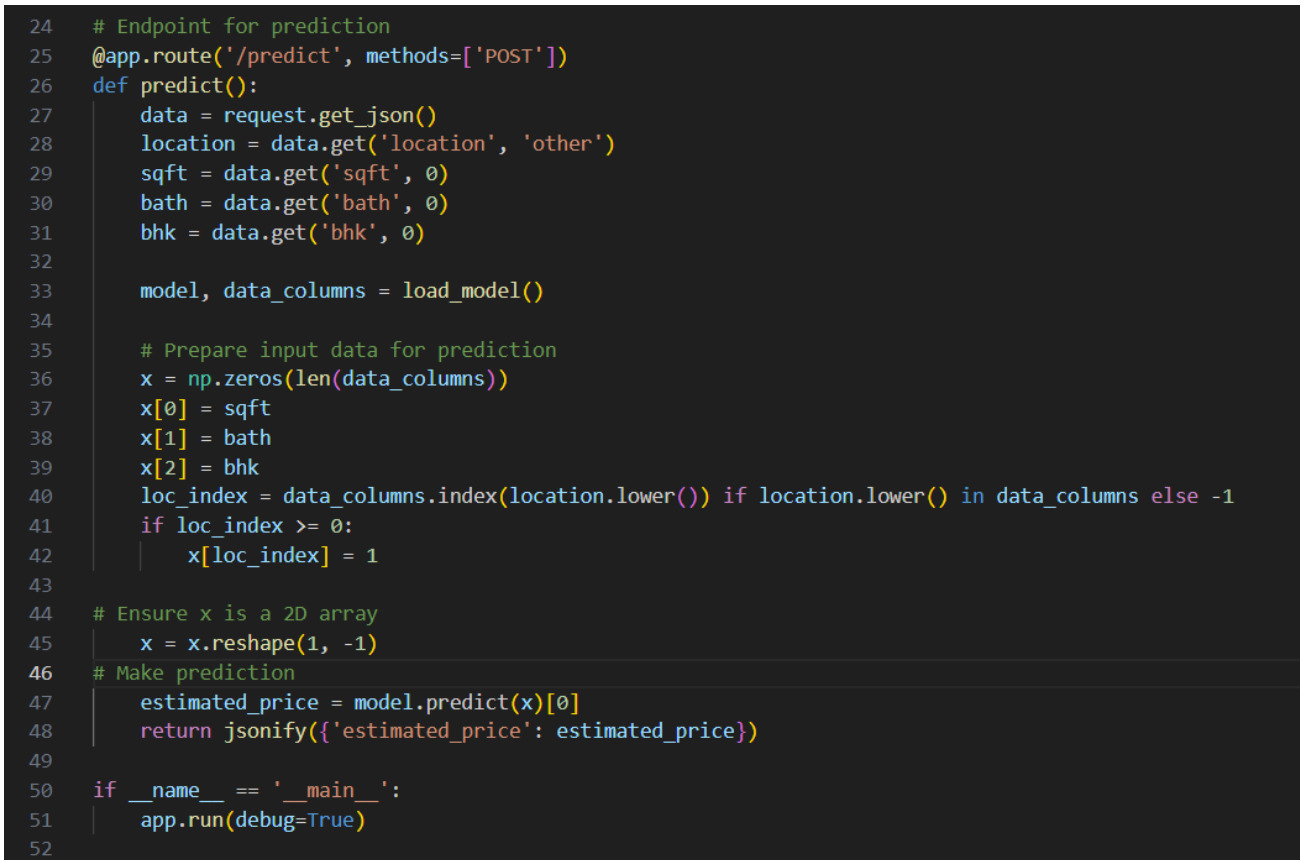
```python
import pandas as pd
import numpy as np

def feature_engineering(df4):
    df4['price_per_sqft'] = df4['price'] * 100000 / df4['total_sqft']
    df4.location = df4.location.apply(lambda x: x.strip())

    # Handle locations with fewer than 10 entries
    location_stats = df4['location'].value_counts(ascending=False)
    location_stats_less_than_10 = location_stats[location_stats <= 10]
    df4.location = df4.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)

    df4 = df4[~(df4.total_sqft / df4.bhk < 300)]  # Remove outliers
    return df4

```

## 6.3 Temp code Runner



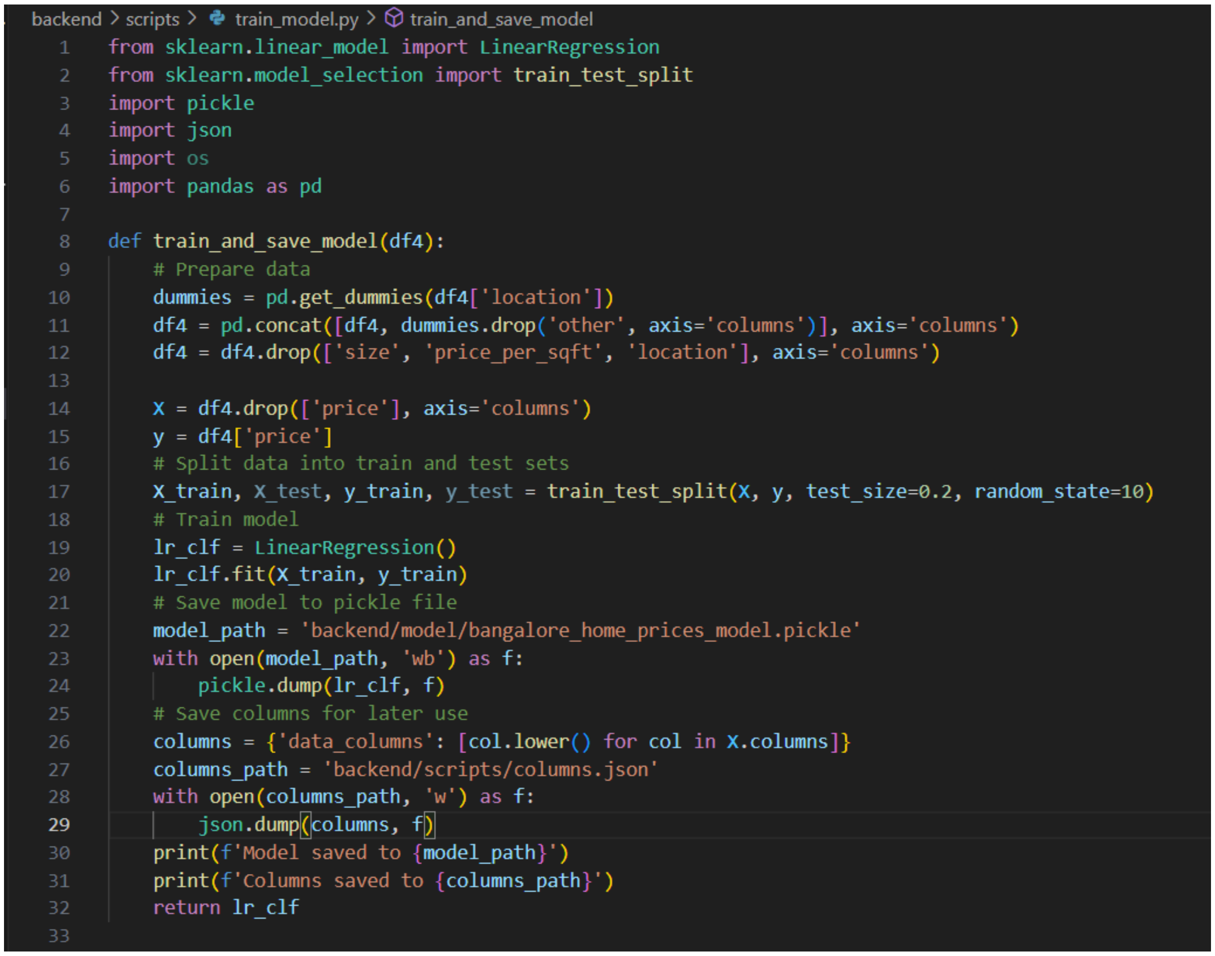
backend > scripts > tempCodeRunnerFile.py > predict

```python
from flask import Flask, request, jsonify
import pickle
import json
import numpy as np
import os

app = Flask(__name__)

def load_model():
    # Get the base directory of the current script
    base_dir = os.path.dirname(os.path.abspath(__file__))
    # Updated path for the pickle model file
    model_path = os.path.join(base_dir, '..', 'model', 'bangalore_home_prices_model.pickle')
    # Load the model from the pickle file
    with open(model_path, 'rb') as f:
        model = pickle.load(f)
    # Corrected path to columns.json
    columns_path = os.path.join(base_dir, 'columns.json')
    # Load the data columns from the columns.json file
    with open(columns_path, 'r') as f:
        data_columns = json.load(f)['data_columns']
    return model, data_columns

```

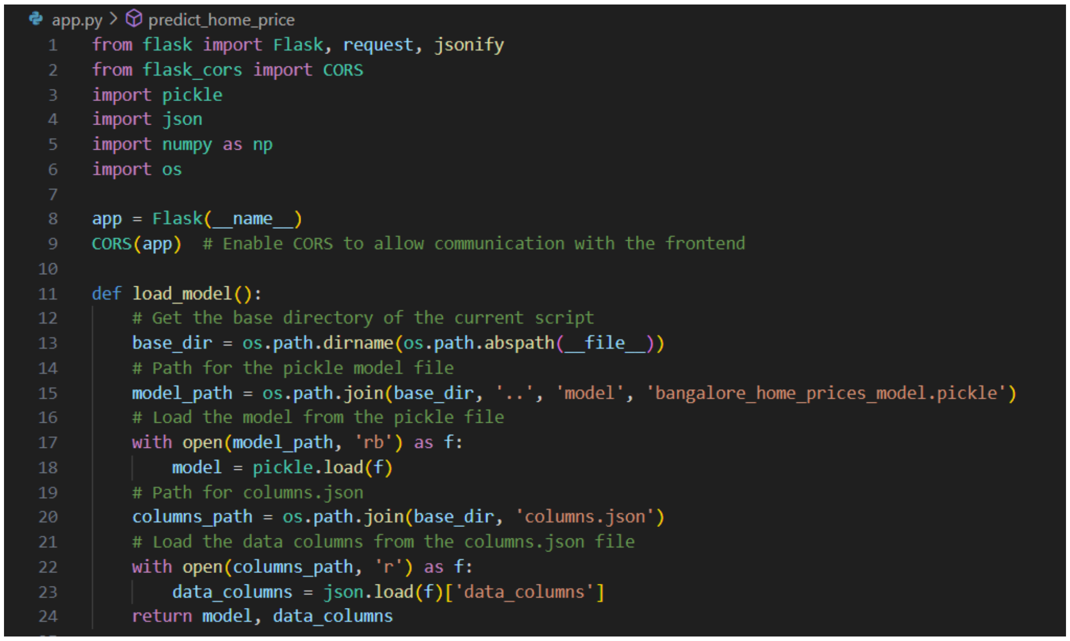


```python
# Endpoint for prediction
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    location = data.get('location', 'other')
    sqft = data.get('sqft', 0)
    bath = data.get('bath', 0)
    bhk = data.get('bhk', 0)

    model, data_columns = load_model()

    # Prepare input data for prediction
    x = np.zeros(len(data_columns))
    x[0] = sqft
    x[1] = bath
    x[2] = bhk
    loc_index = data_columns.index(location.lower()) if location.lower() in data_columns else -1
    if loc_index >= 0:
        x[loc_index] = 1

# Ensure x is a 2D array
    x = x.reshape(1, -1)
# Make prediction
    estimated_price = model.predict(x)[0]
    return jsonify({'estimated_price': estimated_price})

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

```

## 6.4 Train Model

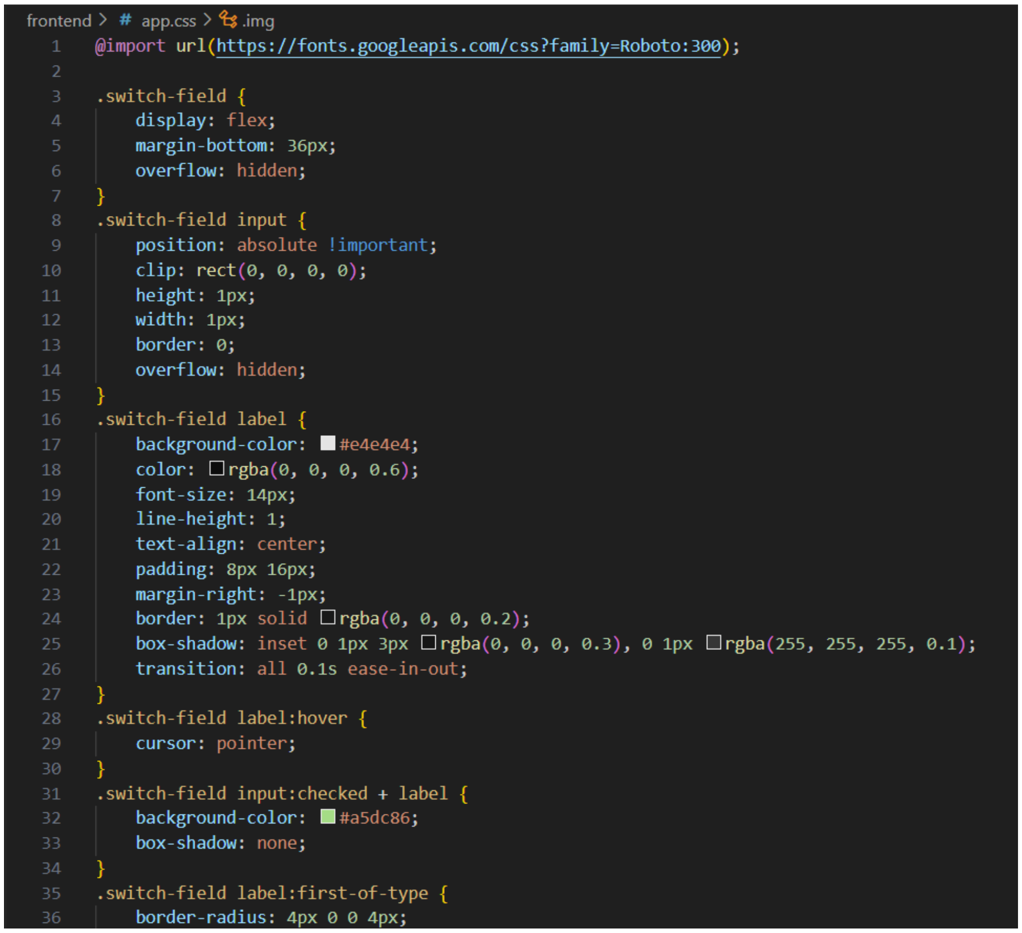

backend > scripts > train_model.py > train_and_save_model
```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pickle
import json
import os
import pandas as pd

def train_and_save_model(df4):
    # Prepare data
    dummies = pd.get_dummies(df4['location'])
    df4 = pd.concat([df4, dummies.drop('other', axis='columns')], axis='columns')
    df4 = df4.drop(['size', 'price_per_sqft', 'location'], axis='columns')

    X = df4.drop(['price'], axis='columns')
    y = df4['price']
    # Split data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)
    # Train model
    lr_clf = LinearRegression()
    lr_clf.fit(X_train, y_train)
    # Save model to pickle file
    model_path = 'backend/model/bangalore_home_prices_model.pickle'
    with open(model_path, 'wb') as f:
        pickle.dump(lr_clf, f)
    # Save columns for later use
    columns = {'data_columns': [col.lower() for col in X.columns]}
    columns_path = 'backend/scripts/columns.json'
    with open(columns_path, 'w') as f:
        json.dump(columns, f)
    print(f'Model saved to {model_path}')
    print(f'Columns saved to {columns_path}')
    return lr_clf

```

## 6.5 APP.py


app.py > predict_home_price
```python
from flask import Flask, request, jsonify
from flask_cors import CORS
import pickle
import json
import numpy as np
import os

app = Flask(__name__)
CORS(app)  # Enable CORS to allow communication with the frontend

def load_model():
    # Get the base directory of the current script
    base_dir = os.path.dirname(os.path.abspath(__file__))
    # Path for the pickle model file
    model_path = os.path.join(base_dir, '..', 'model', 'bangalore_home_prices_model.pickle')
    # Load the model from the pickle file
    with open(model_path, 'rb') as f:
        model = pickle.load(f)
    # Path for columns.json
    columns_path = os.path.join(base_dir, 'columns.json')
    # Load the data columns from the columns.json file
    with open(columns_path, 'r') as f:
        data_columns = json.load(f)['data_columns']
    return model, data_columns
```

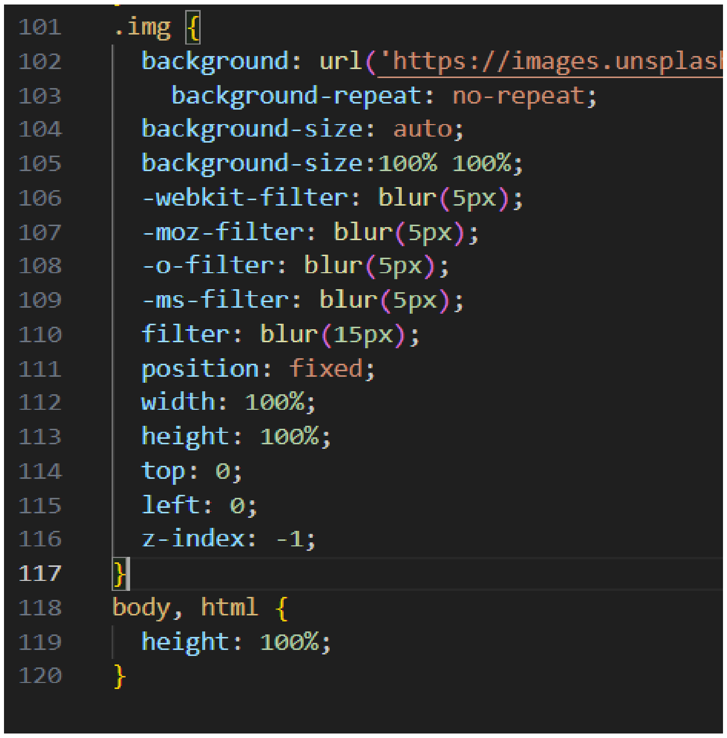
**FRONTEND**

## 6.6 APP.css

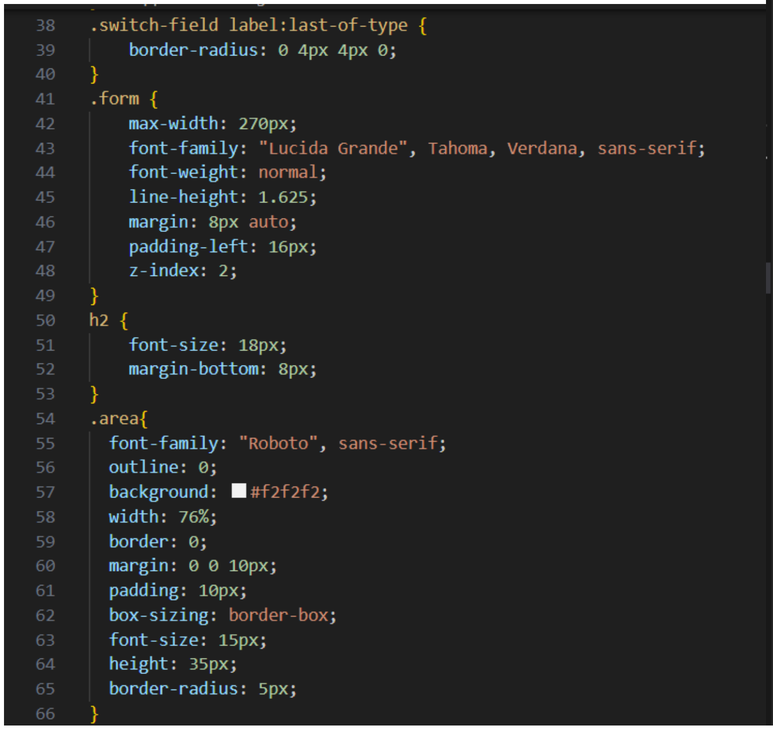


frontend > # app.css > .img

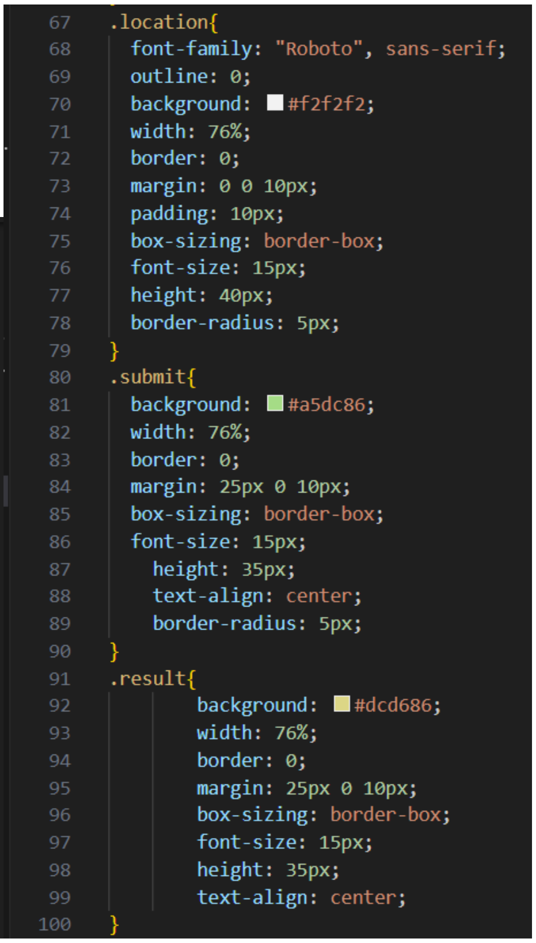
```css
@import url(https://fonts.googleapis.com/css?family=Roboto:300);

.switch-field {
    display: flex;
    margin-bottom: 36px;
    overflow: hidden;
}
.switch-field input {
    position: absolute !important;
    clip: rect(0, 0, 0, 0);
    height: 1px;
    width: 1px;
    border: 0;
    overflow: hidden;
}
.switch-field label {
    background-color: #e4e4e4;
    color: rgba(0, 0, 0, 0.6);
    font-size: 14px;
    line-height: 1;
    text-align: center;
    padding: 8px 16px;
    margin-right: -1px;
    border: 1px solid rgba(0, 0, 0, 0.2);
    box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255, 255, 0.1);
    transition: all 0.1s ease-in-out;
}
.switch-field label:hover {
    cursor: pointer;
}
.switch-field input:checked + label {
    background-color: #a5dc86;
    box-shadow: none;
}
.switch-field label:first-of-type {
    border-radius: 4px 0 0 4px;
```



```css
.img {
    background: url('https://images.unsplas
      background-repeat: no-repeat;
    background-size: auto;
    background-size:100% 100%;
    -webkit-filter: blur(5px);
    -moz-filter: blur(5px);
    -o-filter: blur(5px);
    -ms-filter: blur(5px);
    filter: blur(15px);
    position: fixed;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    z-index: -1;
}
body, html {
    height: 100%;
}
```

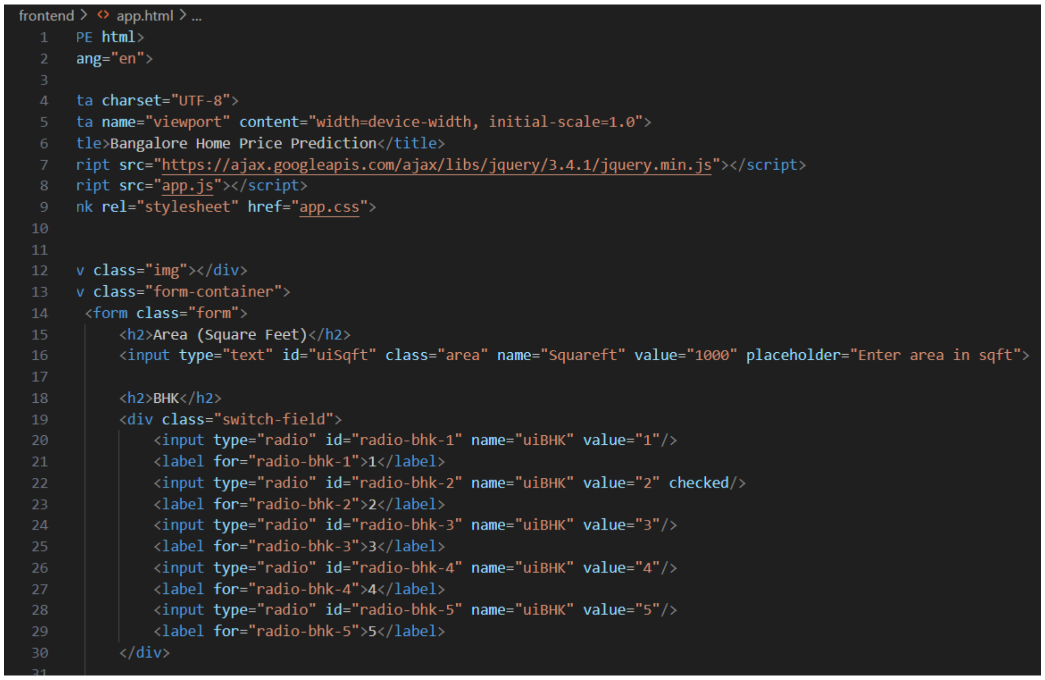


```css
.switch-field label:last-of-type {
    border-radius: 0 4px 4px 0;
}
.form {
    max-width: 270px;
    font-family: "Lucida Grande", Tahoma, Verdana, sans-serif;
    font-weight: normal;
    line-height: 1.625;
    margin: 8px auto;
    padding-left: 16px;
    z-index: 2;
}
h2 {
    font-size: 18px;
    margin-bottom: 8px;
}
.area{
  font-family: "Roboto", sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 76%;
  border: 0;
  margin: 0 0 10px;
  padding: 10px;
  box-sizing: border-box;
  font-size: 15px;
  height: 35px;
  border-radius: 5px;
}
```



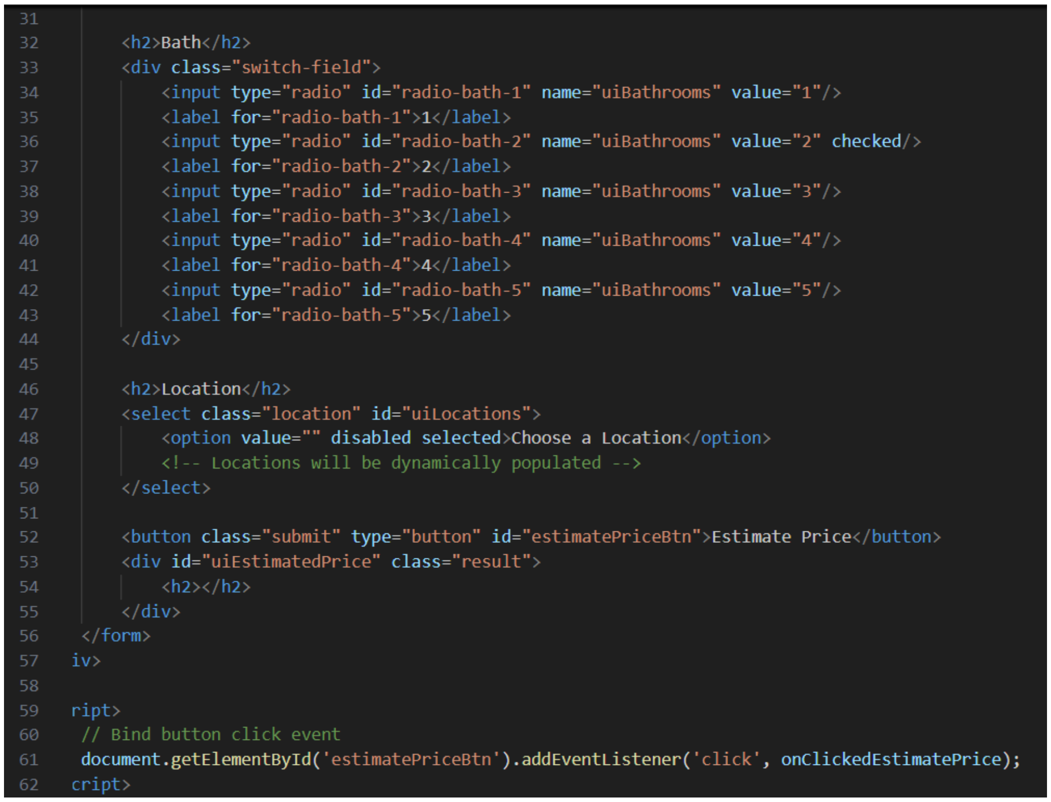
```css
.location{
  font-family: "Roboto", sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 76%;
  border: 0;
  margin: 0 0 10px;
  padding: 10px;
  box-sizing: border-box;
  font-size: 15px;
  height: 40px;
  border-radius: 5px;
}
.submit{
  background: #a5dc86;
  width: 76%;
  border: 0;
  margin: 25px 0 10px;
  box-sizing: border-box;
  font-size: 15px;
    height: 35px;
    text-align: center;
    border-radius: 5px;
}
.result{
        background: #dcd686;
        width: 76%;
        border: 0;
        margin: 25px 0 10px;
        box-sizing: border-box;
        font-size: 15px;
        height: 35px;
        text-align: center;
}
```

## 6.7 App.html



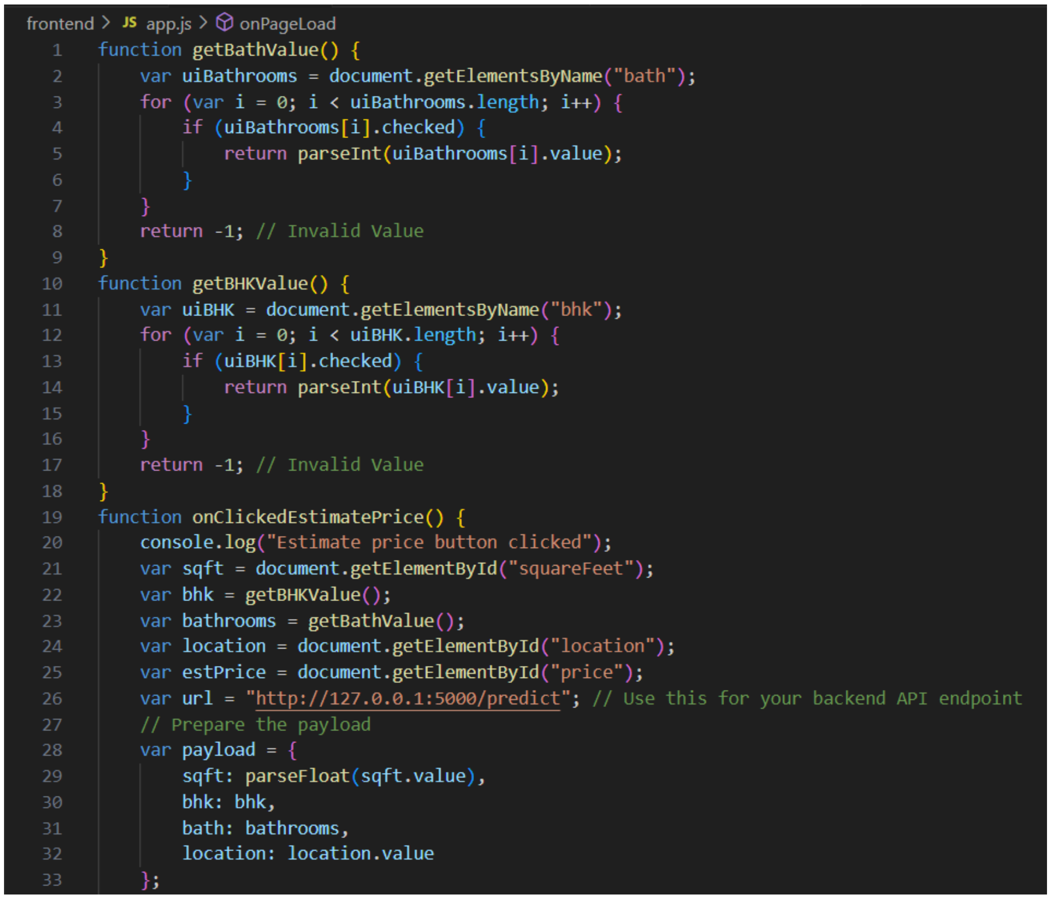
frontend > <> app.html > ...

```html
PE html>
ang="en">

ta charset="UTF-8">
ta name="viewport" content="width=device-width, initial-scale=1.0">
tle>Bangalore Home Price Prediction</title>
ript src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
ript src="app.js"></script>
nk rel="stylesheet" href="app.css">


v class="img"></div>
v class="form-container">
 <form class="form">
     <h2>Area (Square Feet)</h2>
     <input type="text" id="uiSqft" class="area" name="Squareft" value="1000" placeholder="Enter area in sqft">

     <h2>BHK</h2>
     <div class="switch-field">
         <input type="radio" id="radio-bhk-1" name="uiBHK" value="1"/>
         <label for="radio-bhk-1">1</label>
         <input type="radio" id="radio-bhk-2" name="uiBHK" value="2" checked/>
         <label for="radio-bhk-2">2</label>
         <input type="radio" id="radio-bhk-3" name="uiBHK" value="3"/>
         <label for="radio-bhk-3">3</label>
         <input type="radio" id="radio-bhk-4" name="uiBHK" value="4"/>
         <label for="radio-bhk-4">4</label>
         <input type="radio" id="radio-bhk-5" name="uiBHK" value="5"/>
         <label for="radio-bhk-5">5</label>
     </div>
```

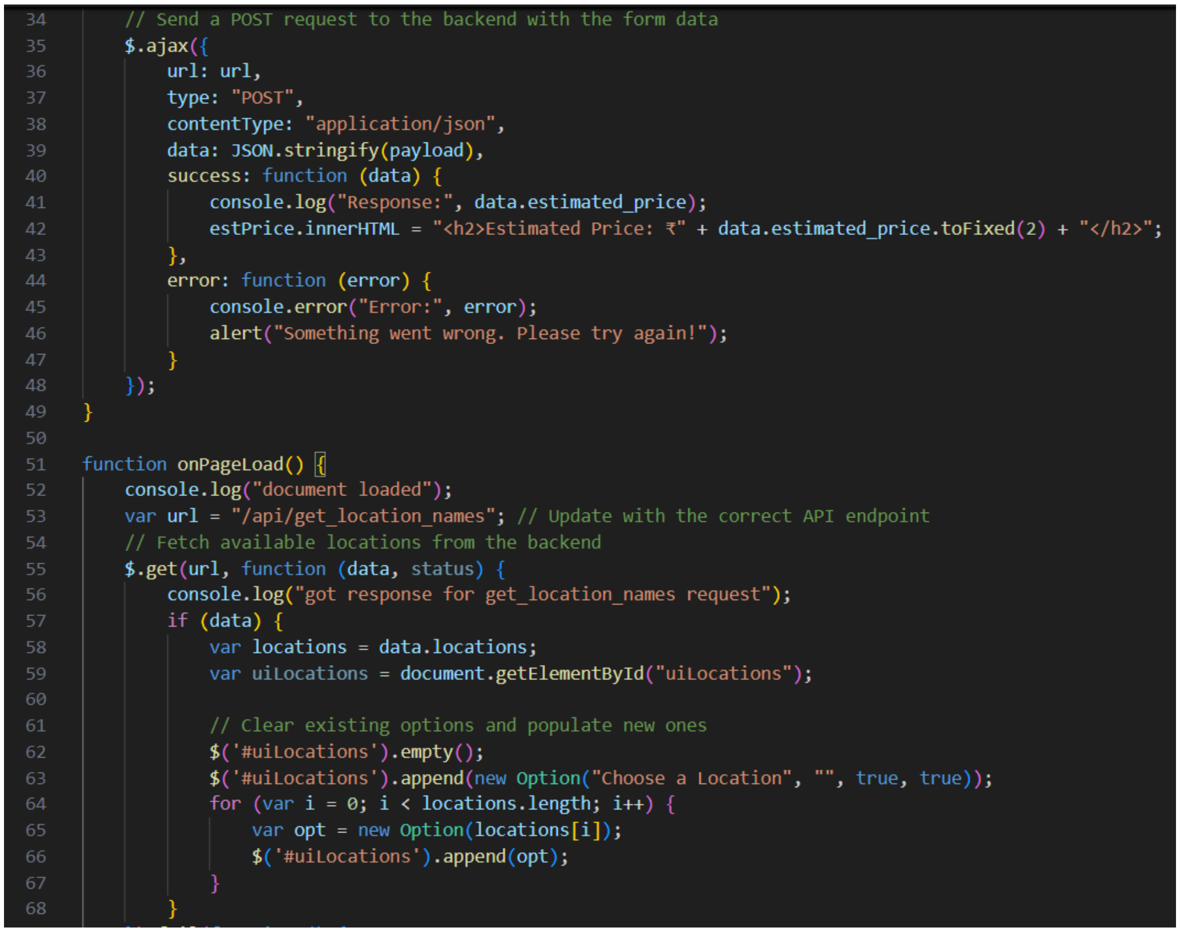


```
     <h2>Bath</h2>
     <div class="switch-field">
         <input type="radio" id="radio-bath-1" name="uiBathrooms" value="1"/>
         <label for="radio-bath-1">1</label>
         <input type="radio" id="radio-bath-2" name="uiBathrooms" value="2" checked/>
         <label for="radio-bath-2">2</label>
         <input type="radio" id="radio-bath-3" name="uiBathrooms" value="3"/>
         <label for="radio-bath-3">3</label>
         <input type="radio" id="radio-bath-4" name="uiBathrooms" value="4"/>
         <label for="radio-bath-4">4</label>
         <input type="radio" id="radio-bath-5" name="uiBathrooms" value="5"/>
         <label for="radio-bath-5">5</label>
     </div>

     <h2>Location</h2>
     <select class="location" id="uiLocations">
         <option value="" disabled selected>Choose a Location</option>
         <!-- Locations will be dynamically populated -->
     </select>

     <button class="submit" type="button" id="estimatePriceBtn">Estimate Price</button>
     <div id="uiEstimatedPrice" class="result">
         <h2></h2>
     </div>
 </form>
iv>

ript>
 // Bind button click event
 document.getElementById('estimatePriceBtn').addEventListener('click', onClickedEstimatePrice);
cript>
```

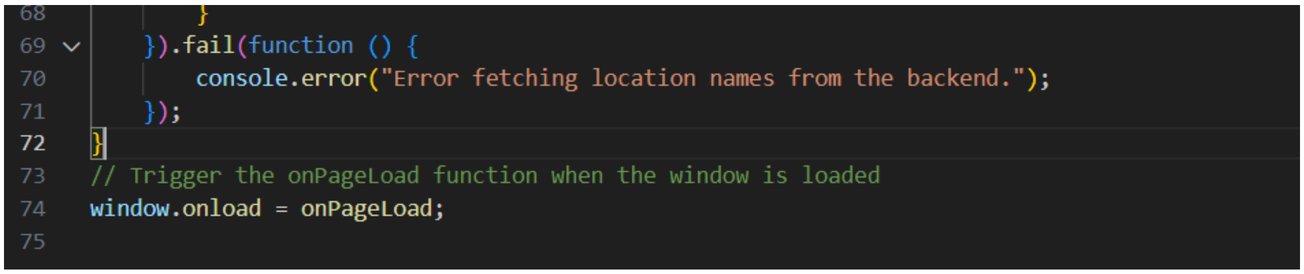
## 6.8 APP.js



```
function getBathValue() {
    var uiBathrooms = document.getElementsByName("bath");
    for (var i = 0; i < uiBathrooms.length; i++) {
        if (uiBathrooms[i].checked) {
            return parseInt(uiBathrooms[i].value);
        }
    }
    return -1; // Invalid Value
}
function getBHKValue() {
    var uiBHK = document.getElementsByName("bhk");
    for (var i = 0; i < uiBHK.length; i++) {
        if (uiBHK[i].checked) {
            return parseInt(uiBHK[i].value);
        }
    }
    return -1; // Invalid Value
}
function onClickedEstimatePrice() {
    console.log("Estimate price button clicked");
    var sqft = document.getElementById("squareFeet");
    var bhk = getBHKValue();
    var bathrooms = getBathValue();
    var location = document.getElementById("location");
    var estPrice = document.getElementById("price");
    var url = "http://127.0.0.1:5000/predict"; // Use this for your backend API endpoint
    // Prepare the payload
    var payload = {
        sqft: parseFloat(sqft.value),
        bhk: bhk,
        bath: bathrooms,
        location: location.value
    };
```

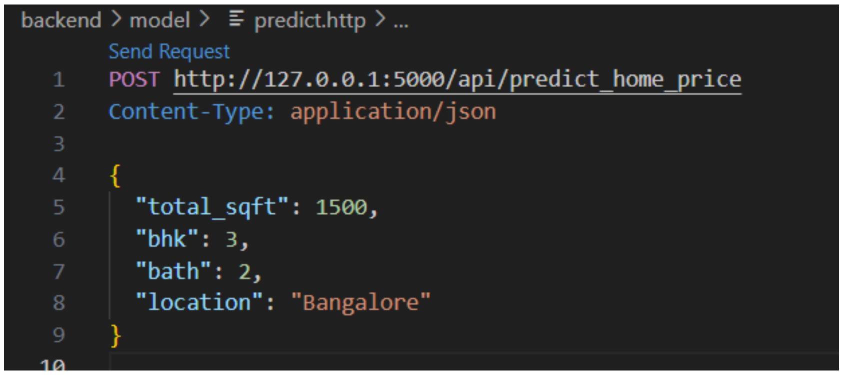


```
    // Send a POST request to the backend with the form data
    $.ajax({
        url: url,
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify(payload),
        success: function (data) {
            console.log("Response:", data.estimated_price);
            estPrice.innerHTML = "<h2>Estimated Price: ₹" + data.estimated_price.toFixed(2) + "</h2>";
        },
        error: function (error) {
            console.error("Error:", error);
            alert("Something went wrong. Please try again!");
        }
    });
}

function onPageLoad() {
    console.log("document loaded");
    var url = "/api/get_location_names"; // Update with the correct API endpoint
    // Fetch available locations from the backend
    $.get(url, function (data, status) {
        console.log("got response for get_location_names request");
        if (data) {
            var locations = data.locations;
            var uiLocations = document.getElementById("uiLocations");

            // Clear existing options and populate new ones
            $('#uiLocations').empty();
            $('#uiLocations').append(new Option("Choose a Location", "", true, true));
            for (var i = 0; i < locations.length; i++) {
                var opt = new Option(locations[i]);
                $('#uiLocations').append(opt);
            }
        }
```



```
        }
    }).fail(function () {
        console.error("Error fetching location names from the backend.");
    });
}
// Trigger the onPageLoad function when the window is loaded
window.onload = onPageLoad;
```

## 6.9 predict.http



backend > model > predict.http > ...

Send Request

```
POST http://127.0.0.1:5000/api/predict_home_price
Content-Type: application/json

{
  "total_sqft": 1500,
  "bhk": 3,
  "bath": 2,
  "location": "Bangalore"
}
```

# 7. Future Scope

The Real Estate Price Predictor Platform holds substantial potential for future advancements and enhancements to provide users with a more robust, user-friendly, and insightful experience. Key areas for future development include:

- **Advanced Features**: Introducing features like neighbourhood-specific analysis, AI-driven investment recommendations, and advanced filtering options to tailor predictors and insights based on user preferences. Real-time alerts for market changes and price fluctuations can further empower user decision-making.
- **Multi-Language Support:** Expanding accessibility by supporting multiple languages, enabling users across different regions to leverage the platform effectively.
- **Mobile Application:** Developing a dedicated mobile app for increased accessibility, providing users with the ability to interact with the platform seamlessly while on the go. Mobile-specific features, such as instant price notifications, augmented reality property views, and voice commands, can add value to the user experience.
- **Expanded Data Integration**: Incorporating additional APIs for comprehensive property evaluations, including mortgage rates, rental trends, crime rates, and proximity to schools, hospitals, and other amenities.
- **Educational Resources:** Adding blogs, video tutorials, and infographics about real estate investment, property valuation, and market trends to educate users. Hosting live webinars or Q&A sessions with industry experts could also foster a deeper understanding of the real estate market.
- **Enhanced Security Measures**: Implementing measures like two-factor authentication (2FA), encryption protocols, and regular audits to ensure user data privacy and system integrity.
- **Community Features**: Adding forums, chatrooms, and property discussion groups to facilitate interaction among buyers, sellers, and investors. User-generated reviews and ratings for properties and agents can create a trusted ecosystem.
- **Partnerships and Collaborations**: Collaborating with real estate firms, banks, and financial

institutions to provide integrated services such as verified property listings, loan calculators, and financial advisory tools.

- **Sustainability Insights**: Including features like energy efficiency scores and environmental impact assessments to cater to eco-conscious users.

# 8.Conclusion

The Real Estate Price Predictor Platform represents a significant step forward in simplifying the complex process of property valuation and market analysis. Designed with user-centric features, the platform addresses the challenges faced by property buyers, sellers, and investors by offering accurate predictors, interactive tools, and a seamless user experience. By integrating core functionalities such as property price estimation, market trend analysis, and location-based filtering, the platform ensures a comprehensive solution for users navigating the real estate market.

The use of advanced technologies like Flask for the backend, pre-trained machine learning models for accurate predictors, and interactive visualizations powered by libraries like Chart.js underscores the platform's technical sophistication. Its responsive web design guarantees accessibility across devices, catering to a diverse audience of users, from first-time homebuyers to seasoned investors.

The platform's architecture emphasizes scalability, security, and reliability. Local deployment ensures quick and efficient processing, while secure input handling and real-time data analysis safeguard user information and ensure accurate results. Its intuitive interface makes the system accessible to users with varying levels of technical expertise, fostering greater engagement and confidence in decision-making.

In conclusion, this platform effectively bridges the gap between the complexities of real estate market analysis and the needs of its users. By combining advanced predictive technology, user-friendly design, and a focus on addressing market challenges, it establishes itself as a valuable tool for navigating the dynamic real estate landscape. The project not only meets current demands but also lays a solid foundation for future enhancements, making it a scalable and impactful solution for the real estate industry.

## 9. References

- Aakash Chauhan, Aditya Jain, Purushottam Sharma, Vikas Deep
  "REAL ESTATE PRICE PREDICTOR using Evolutionary Rule Learning."
  *International Conference on Computational Intelligence and Communication Technology (CICT 2018).*
  This paper discusses evolutionary rule learning techniques for real estate price predictor, providing insights into advanced predictive modelling approaches.
- Senthilkumar Mohan, Chandrasekar Thirumalai, Gautam Srivastava
  "Effective REAL ESTATE PRICE PREDICTOR Using Hybrid Machine Learning Techniques."
  *IEEE Access (Volume: 7), 2019.*
  This study explores the use of hybrid machine learning techniques, which could provide useful methodologies for improving the accuracy of price predictors in your project.
- K. Prasanna Lakshmi, Dr. C.R.K. Reddy
  "Fast Rule-Based REAL ESTATE PRICE PREDICTOR using Associative Classification Mining."
  *International Conference on Computer, Communication, and Control (IC4), 2015.*
  This work highlights rule-based predictive systems, offering practical insights for the efficient predictor of real estate prices.
- Dangare Chaitrali S, Sulabha S Apte
  "Improved study of REAL ESTATE PRICE PREDICTOR system using data mining classification techniques."
  *International Journal of Computer Applications, 47.10 (2012): 44-8.*
  This paper discusses the application of data mining techniques in real estate price predictor, relevant to the machine learning focus of your project.
- Chen A H, Huang S Y, Hong P S, Cheng C H, Lin E J
  "HDPS: REAL ESTATE PRICE PREDICTOR system."
  *In 2011 Computing in Cardiology (pp. 557-60). IEEE.*
  Although primarily for medical data systems, the methodologies described can inspire structured data handling and predictive system design in real estate price predictor.
- **Bengaluru House Price Data**

The primary dataset for training the real estate price predictor model. This dataset contains crucial features like location, size, number of bedrooms, and price, which were pre-processed and used to train the machine learning model.

Dataset Link: Bengaluru House Price Data

- **Flask Documentation**

  The official documentation for Flask, a lightweight Python web framework used to develop the backend of the project. Flask facilitated the creation of RESTful APIs to serve predictors and integrate machine learning models with the frontend interface.

  Documentation Link: Flask Documentation

- **scikit-learn Documentation**

  The library utilized for implementing the machine learning pipeline, including data preprocessing, feature selection, and model training. It provides tools for regression, classification, and model evaluation.

  Documentation Link: scikit-learn Documentation.

- **pandas Documentation**

  A powerful Python library used for data manipulation and analysis. It was essential for cleaning and structuring the Bengaluru House Price dataset.

  Documentation Link: pandas Documentation.

- **NumPy Documentation**

  A Python library for numerical computing, which was used to manage array-based data processing during feature engineering and model input preparation.

  Documentation Link: NumPy Documentation

- **Matplotlib Documentation**

  This library was used for creating visualizations to analyse data trends and present predictor results. Graphical representations like price trends and comparisons were generated with Matplotlib.

  Documentation Link: Matplotlib Documentation

- **Bootstrap Documentation**

  The frontend interface was styled using Bootstrap to create a responsive and visually appealing design for the web application.