HOUSE PRICE PREDICTION

**Submitted by**

|  |  |
| --- | --- |
| **SHANMUGAVELAN M** | **(61781921106097)** |

#### Of

SONA COLLEGE OF TECHNOLOGY

#### SALEM - 636006

A PROJECT REPORT

#### Submitted to

**U19IT604 SOFTWARE DEVELOPMENT AND TESTING LABORATORY**

#### in partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

IN

## INFORMATION TECHNOLOGY

**(Autonomous)**

**ANNA UNIVERSITY: CHENNAI 600 025**

2022-2023

# BONAFIDE CERTIFICATE

Certified that this project report **“HOUSE PRICE PREDICTION”** is the bonafide work of **“SHANMUGAVELAN M (61781921106097)”** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **LAB- IN-CHARGE** | **HEAD OF THE DEPARTMENT** |
|  |  |

Submitted for End Semester Practical examination held on .....................................

|  |  |
| --- | --- |
| **INTERNAL EXAMINER** | **EXTERNAL EXAMINER** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **LIST OF FIGURES** | iv |
| **1** | **PROBLEM STATEMENT** | **3** |
| **2** | **PROBLEM SCOPE** | **4** |
| **3** | **SOFTWARE REQUIREMENTS SPECIFICATION** | **5** |
| **4** | **SYSTEM ANALYSIS** | 14 |
|  | 4.1 ATTRIBUTE MATRIX | 14 |
|  | 4.2 TRACEABILITY MATRIX | 16 |
|  | 4.3 TRACEABILITY TREE | 17 |
| **5** | **SYSTEM DESIGN** | 19 |
|  | 5.1 USE CASE DIAGRAM | 19 |
|  | 5.2 ACTIVITY DIAGRAM | 20 |
|  | 5.3 CLASS DIAGRAM | 21 |
|  | 5.4 SEQUENCE DIAGRAM | 22 |
|  | 5.5 COMPONENT DIAGRAM | 23 |
|  | 5.6 COLLABORATION DIAGRAM | 24 |
|  | 5.7 DEPLOYMENT DIAGRAM | 25 |
| **6** | **SYSTEM PROTOTYPING** | **26** |
|  | 6.1 LOGIN PAGE | 26 |
|  | 6.2 SIGN UP I PAGE | 27 |
|  | 6.3 SIGN UP II PAGE | 27 |
|  | 6.4 WORKING PAGE | 28 |
|  | 6.5 OUTPUT PAGE | 29 |
| **7** | **SYSTEM IMPLEMENTATION** | **30** |
| **8** | **SYSTEM TESTING** | **34** |
| **9** | **CONCLUSION AND FUTURE ENHANCEMENT** | **37** |
| **APPENDIX 1** | **SAMPLE CODE** | **39** |
|  | **SCREENSHOTS OF IMPLEMENTATION** | **36** |

|  |  |  |
| --- | --- | --- |
| FIGURE NO | TITLE | PAGE NO |
| 2.1.1 | Product perspective Usecase Diagram | 6 |
| 4.1.1 | Stakeholder Requirement Matrix | 14 |
| 4.1.2 | Use case Requirement Matrix | 15 |
| 4.2.1 | Traceability Matrix | 16 |
| 4.2.2 | Use Case Vs Use Case Requirement | 16 |
| 4.2.3 | Use Case Vs Stack Holder Requirement | 17 |
| 4.2.4 | Traceability tree |  |
| 5.1.1 | House price prediction Use Case Diagram | 18 |
| 5.2.1 | Activity Diagram | 19 |
| 5.3.1 | Class Diagram | 20 |
| 5.4.1 | Sequence Diagram | 21 |
| 5.5.1 | Component Diagram | 22 |
| 5.6.1 | Collaboration Diagram | 23 |
| 5.7.1 | Deployment Diagram | 24 |
| 6.1 | Login Page | 26 |
| 6.2 | Sign up I Page | 27 |
| 6.3 | Sign up II Page | 27 |
| 6.4 | Working Page | 28 |
| 6.5 | Output Page | 29 |
| 9.1.1 | House price prediction user input | 41 |
| 9.1.2 | Predicted output | 42 |

LIST OF FIGURES

# CHAPTER-1

## PROBLEM STATEMENT

* Develop a machine learning model to predict house prices based on features such as square footage, number of bedrooms, location, and other relevant factors. The goal is to create an accurate and reliable prediction tool that can assist potential buyers, sellers, and real estate professionals in estimating property values.
* The solution should undergo rigorous testing and validation to ensure its accuracy, and regular updates may be required to incorporate new data and adapt to changing market trends. Ultimately, the goal is to create a valuable tool for real estate professionals, buyers, and sellers, facilitating more informed and strategic decision-making in the dynamic and competitive real estate market.

# CHAPTER-2

## PROJECT SCOPE

**Problem Statement Definition:**

The system aims to predict house prices accurately based on various features like size, location, number of bedrooms, etc., to assist buyers, sellers, and real estate professionals in making informed decisions.

**Data Collection:**

Gather comprehensive datasets containing information about houses, including features such as size, number of bedrooms, bathrooms, location, amenities, sale prices, etc., from reliable sources like real estate listings, property databases, and government records.

# CHAPTER-3

**SOFTWARE REQUIREMENT SPECIFICATION**

1. **INTRODUCTION:**
   1. **Purpose:**

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the House Price Prediction system, its intended functionality, and constraints. This will serve as a reference for developers during implementation and testing phases and also help stakeholders understand the features and capabilities of the proposed system.

* 1. **Scope:**

This SRS defines the requirements for a web-based application that predicts residential property prices based on various factors such as location, size, number of bedrooms, bathrooms, car spaces, etc. The primary objective is to enable users to obtain an estimated value for their properties or prospective purchases by inputting relevant details through a user-friendly interface.

* 1. **Definition and Abbreviations:**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SRS | Software requirement specification |
| UI | User Interface |
| DBMS | Data Base Management System |
| MAE | Mean Absolute Error |
| RMSE | Root Mean Square Error |
| R2 SCORE | Coefficient Of Determination |
| API | Application Programming Interface |
| TLS | Transport Layer Security |

* 1. **References:**

This web application has been prepared on the basis of discussion with Team members, faculty members and also taken information from following books & website.

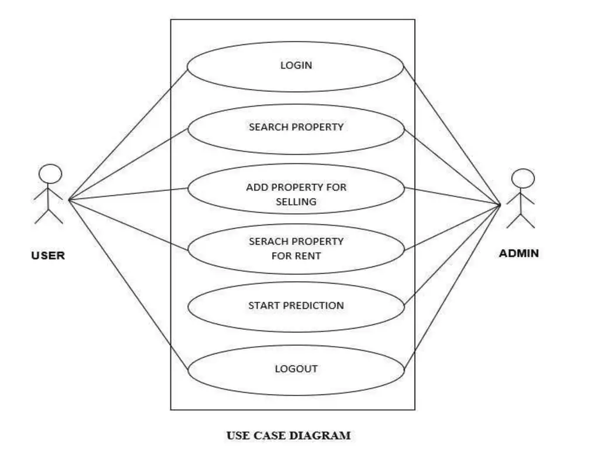
[**www.google.com**](http://www.google.com)

[**www.wikipedia.com**](http://www.wikipedia.com)

[**www.github.com**](http://www.github.com)

1. **General Descriptions:**
   1. **Product Perspective:**

The House Price Prediction System represents a digital solution intended to streamline and improve real estate transactions by offering reliable price estimations derived from sophisticated machine learning algorithms. These models consider numerous factors influencing property values, thereby empowering users with knowledgeable insights when making crucial decisions related to buying, selling, or investing in residential properties.



**Figure 2.1.1 product perspective usecase diagram**

* 1. **Functionalities:**
     1. **Data collection and aggregation:**

Collecting and processing comprehensive data sets from multiple sources, such as public records, MLS listings, census data, satellite imagery, and economic indicators, to inform the ML model.

* + 1. **Feature engineering:**

Transforming raw data into meaningful variables that capture important aspects of the properties being analyzed, such as proximity to amenities, school districts, or transportation hubs.

* + 1. **Model training and evaluation:**

Training and testing various ML algorithms to find the most accurate and efficient one, comparing performance metrics such as mean squared error, root mean squared error, R-squared, and adjusted R-squared.

* + 1. **Price estimation:**

Computing accurate and reliable price estimates for individual properties based on their unique attributes and contextual factors.

* + 1. **Uncertainty quantification:**

Providing uncertainty measures around each price estimate, reflecting the variability and reliability of the underlying data and modeling assumptions.

* + 1. **Visualization and reporting:**

Presenting the output in an intuitive and engaging format, such as interactive maps, charts, graphs, or tables, allowing users to quickly understand the implications of the analysis.

* + 1. **Integration with business processes:**

Incorporating the pricing engine seamlessly within larger real estate ecosystems, such as brokerages, mortgage lenders, insurers, or investment firms, enabling automation, standardization, and scale.

* 1. **User characteristics:**

There are various kinds of users for the product. Usually web products are visited by various users for different reasons.

The users include :

* + 1. **Real estate professionals:**

Real estate professionals who seek advanced functionality, customizability, and integration capabilities

* + 1. **Individual home owners and buyers:**

Individual home owners and buyers who desire simplicity, clarity, ease-of-use, and personalized guidance. Understanding both audiences is crucial for creating impactful products that cater to their distinct needs.

* 1. **Generals Constraints:**
     1. **Data quality:**

The accuracy and relevance of your data will greatly impact the performance of your model. Make sure to clean and preprocess your data to remove any outliers or irrelevant features.

* + 1. **Feature selection:**

Not all features may be relevant or useful for predicting house prices. It's important to carefully select the most informative features to include in your model.

* + 1. **Model complexity:**

A complex model with many parameters can often fit the training data better, but it may also overfit and perform poorly on new, unseen data. You'll need to find a balance between model complexity and generalization ability.

* 1. **Assumption and Dependencies:**
  + Linearity, independence, and stationarity assumptions of the data
  + Impact of outliers, sample size, missing values, multicollinearity, and domain knowledge
  + Importance of handling outliers, sufficient sample size, dealing with missing values, regularization techniques, and collaboration with domain experts.
  + Choose right algorithm for data and problem , Tune hyper parameters properly
  + Interpret model outcomes when they are necessary.

1. **Specific Requirements:**

The functionality and non-functional requirements for the house price prediction system will shape its interaction with the environment and provide a solid foundation for the design phase. Clear and accurate definition of these requirements is crucial for the effectiveness of the project.

* 1. **Functional Requirements:**

**Input of House Features**: Users can input various features of a house such as area, number of bedrooms, bathrooms, location, etc.

**Model Training**: Machine learning models will be trained using historical housing data to learn patterns and relationships between house features and prices.

**Price Prediction**: The system will predict house prices based on the input features provided by users and the trained machine learning models.

**Presentation of Predictions**: Predicted house prices will be presented to users through a user-friendly interface, facilitating informed decision-making in buying or selling houses.

**3.1.2 Non-Functional Requirements:**

**Accuracy**: The prediction accuracy should be above 90%, ensuring reliable estimations for users.

**Performance**: The system should respond to user queries within seconds, providing a seamless user experience.

**Scalability**: The system should be able to handle a large volume of data and user requests as the user base grows.

**Usability:** The interface should be intuitive and easy to use, catering to users of varying technical expertise.

**Security**: User data and predictions should be securely stored and protected from unauthorized access to maintain user privacy and confidentiality.

**3.1.3 Process Specification:**

All the processes involved in the house price prediction system are described as follows:

**Customer Login:**

Each customer will have their own account ID and password. This page will require both attributes for customers to access their accounts.

**System Features:**

Not every visitor to the system will be a customer. Some may be interested in reading about the features the system provides. The main page should provide basic features and benefits to these users.

**Order for House Price Prediction:**

New visitors to the system may be interested in predicting the price of a house. They should be provided an easy path to access the prediction functionality.

**Input House Features:**

New users should fill out a form to register with the system. If the inputted values are logically correct, their details will be sent to the system administration block; otherwise, they will be prompted to input the values again.

**Welcome Page:**

After logging in, users will be provided with an interface offering various tasks related to house price prediction. They can choose a task to proceed with their work.

**Staff Login:**

A staff login link will be provided on the main page for system staff to access their accounts. The type of staff will be recognized, and they will be directed to the appropriate module.

**Check House Price Prediction:**

After logging in, users can check the predicted price of a house by clicking the appropriate link. The system will display the predicted price based on the input features.

**Transfer Balance:**

If users want to transfer money to another account, this module will provide them with the opportunity. The system will check the balance and execute the transfer if sufficient funds are available.

**House Details:**

If users visit the system physically, they can provide house details to system staff who will provide information about the predicted price. Users can perform tasks at the physical location similar to those available online.

**Order Prediction Report:**

If users want to order a prediction report, they can do so through this module.

* 1. **External Interface Requirements:**

These requirements are discussed under the following categorization.

* + 1. **User Interface:**

The application will be accessed through a browser interface. The interface would be optimized for viewing at resolutions of 1024 x 768 and 800 x 600 pixels. The software would be fully compatible with major web browsers such as Chrome, Firefox, and Safari.

* + 1. **Hardware Interface:**
       1. **Server Side:**

Operating System: Linux or Windows Server.

Processor: Intel Xeon or equivalent.

RAM: 4 GB or more.

Hard Drive: 100 GB or more.

* + - 1. **Client Side:**

Operating System: Windows, macOS, or Linux.

Processor: Intel Core i5 or equivalent.

RAM: 4 GB or more.

* + 1. **Software Interface:**
       1. **Client Side:**

Web Browser: Chrome, Firefox, Safari.

Operating System: Windows, macOS, Linux.

* + - 1. **Web Server:**

Web Server Software: Apache, Nginx.

Operating System: Linux.

* + 1. **Communication Interface:**

Users must connect to the internet to access the system:

Internet Connection: Broadband or Dial-up.

Browser Compatibility: Chrome, Firefox, Safari.

* 1. **Non-Functional Requirements:**

Non-functional requirements are characteristics of the system that are not directly related to its functionalities but are essential for its overall performance and quality. The house price prediction system must adhere to the following non-functional requirements to ensure its completeness and effectiveness:

1. **Conformance to Specific Standards:**

The system must adhere to industry standards and best practices in machine learning and data science to ensure accuracy and reliability in house price predictions.

**b) Performance Constraints:**

The system should be optimized for performance, utilizing memory efficiently and ensuring fast and reliable access for users. Memory management should be carefully implemented to avoid wastage.

**c) Hardware Limitations:**

The system should be designed to be platform-independent and compatible with a wide range of hardware configurations. It should be able to function effectively even on low-specification hardware.

**d) Maintainable:**

Each module of the system should be designed in a modular and extensible manner, allowing for easy integration of new features and updates.

**e) Reliable:**

The system should be reliable, providing accurate predictions consistently over time.

**f) Testable:**

The system should be designed with testability in mind, allowing for thorough testing to ensure functionality and performance under various conditions.

* 1. **Other Requirements:**

**Software Quality Attributes:**

The quality of the system is crucial for user satisfaction. The following software quality attributes are assumed to be maintained:

**a) Accurate and Reliable:**

The system should provide accurate and reliable house price predictions to users, building trust and confidence in its capabilities.

1. **Secured:**

The system should prioritize security measures to protect user data and ensure privacy and confidentiality.

**d) Fast Speed:**

The system should deliver fast and responsive performance, providing users with timely predictions and a smooth user experience.

**e) Compatibility:**

The system should be compatible with various platforms and devices, allowing users to access it seamlessly from different environments.

**4.Possible Product Evolution:**

No system remains static, and it's essential to consider future enhancements and updates. Here are some potential areas for evolution for the house price prediction system:

**4.1 Enhanced Feature Set:**

As user needs evolve, additional features such as neighborhood analysis, property valuation trends, and predictive analytics could be incorporated into the system to provide more comprehensive insights.

**4.2 Improved User Interface:**

Periodic updates to the user interface, incorporating modern design trends and user feedback, can enhance the user experience and keep the system visually appealing and intuitive.

**4.3 Technology Advancements:**

As technology evolves, the system may need to adopt newer technologies and frameworks that offer improved performance, security, and scalability. This could involve transitioning to advanced machine learning algorithms, cloud-based infrastructure, or distributed computing platforms to meet growing demands and ensure the system's longevity.

**CHAPTER-4**

**SYSTEM ANALYSIS**

The next workflow in the RUP is the Analysis of the requirements which have been specified in the SRS. The Analysis is done with the help of Rational Requisite Pro. The three views or reports which form the basis for analysis are

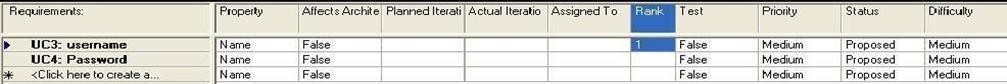
1. Attribute Matrix

2. Traceability Matrix

3. Traceability Tree

**4.1 ATTRIBUTE MATRIX**

The Attribute Matrix view is a spreadsheet like display that lists the requirements of a specific requirements type and their attributes. Requirements are arranged in rows, listed by tag number and followed by requirement name. Attributes are arranged in columns.



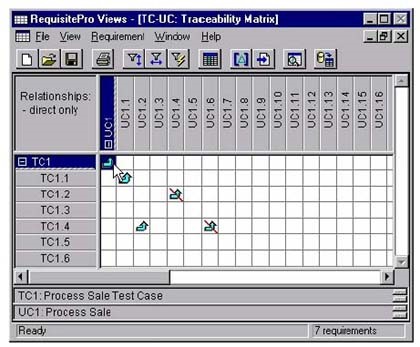
**Figure 4.1.1 Stakeholder Requirement Matrix**

### 

**Figure 4.1.2 Use case Requirement Matrix**

**4.2 TRACEBILITY MATRIX:**

Traceability Matrix is a view that illustrates the relationships between requirements of the same or different types. We can use this matrix to create, modify and delete traceability relationships and view indirect relationships and view direct relationships and traceability relationships with a suspect state. We can also use the traceability matrix to filter and sort the requirements and columns requirements separately.



**Figure 4.2.1 TRACEBILITY MATRIX**

### Use Case Vs Use Case Requirement

### Figure 4.2.2 Use Case Vs Use Case Requirement

### Use Case Vs Stack Holder Requirement

### 

### Figure 4.2.3 Use Case Vs Stack Holder Requirement

### 4.3 TRACEABILITY TREE

### A view that displays all internal and external requirements traced to or from a requirement. The traceability tree only displays the first level project traceability.

### Stack Holder Requirement Traceability Tree

### IN

### 

### Stack Holder Requirement Traceability Tree

### OUT

### 

**CHAPTER-5**

**SYSTEM DESIGN**

## USECASE DIAGRAM

A Use Case Diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication association between the actors and the use cases and generalization among the use cases. A use case corresponds to a sequence of transactions, in which each transaction is invoked from outside the system and engages internal objects to interact with each other. An actor is anything that interacts with the use case.

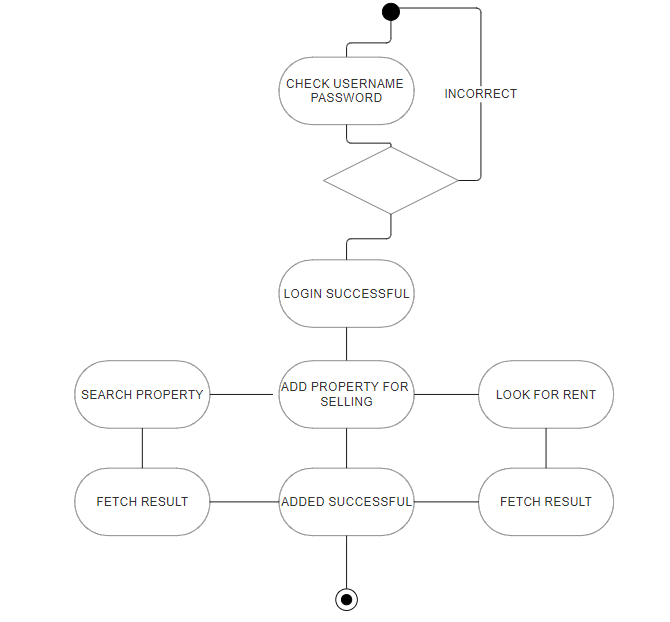
##### HOUSE PRICE PREDICTION TOOL

##### 

**Figure 5.1.1 House Price Prediction Use Case Diagram**

## ACTIVITY DIAGRAM

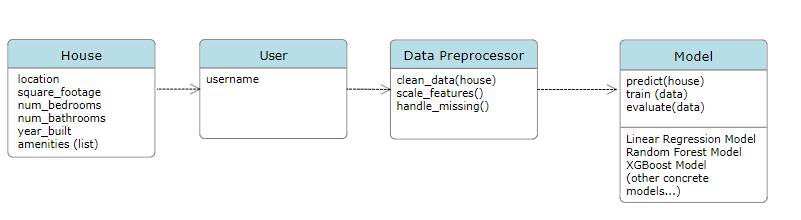
Activity Diagrams illustrate the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

****

**Figure 5.2.1 Activity Diagram**

## CLASS DIAGRAM

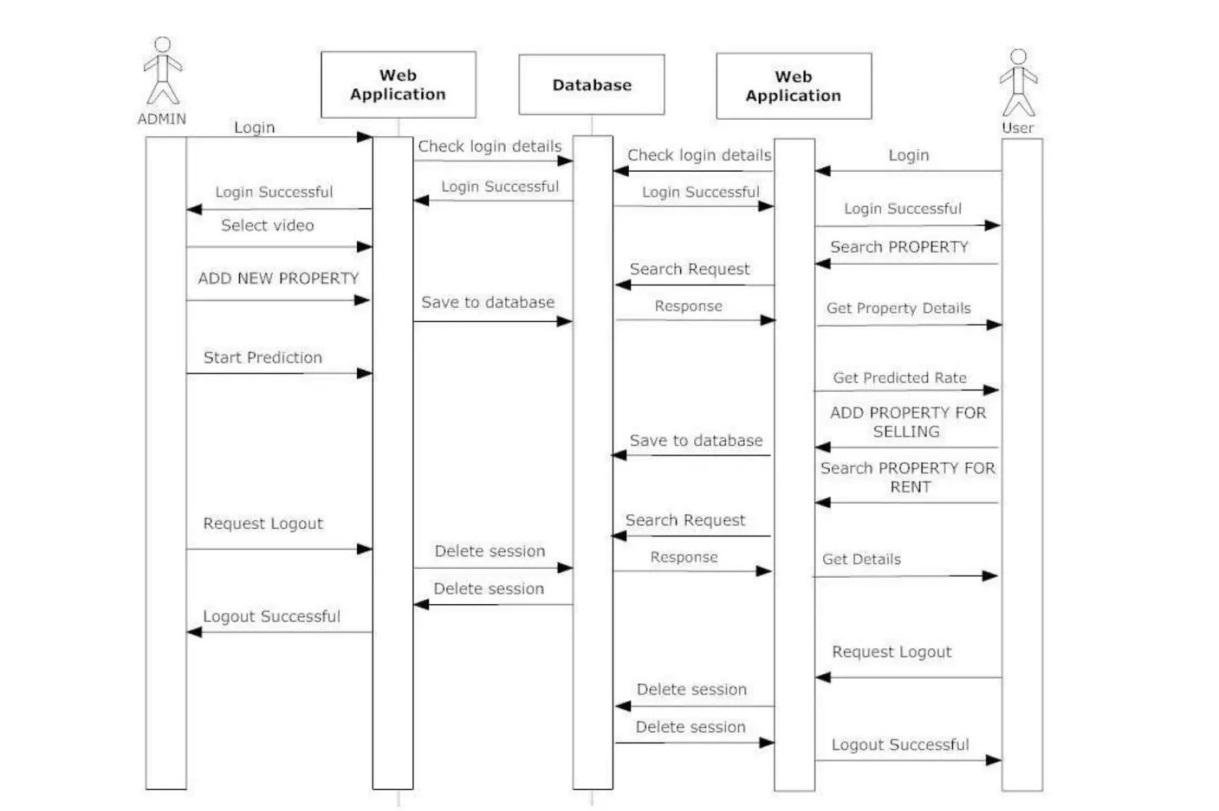
A Class Diagram is a collection of static modeling elements such as classes and their relationships, connected as a graph to each other and to their contents. These diagrams show the static structures of the model.



**Figure 5.3.1 Class Diagram**

## SEQUENCE DIAGRAM

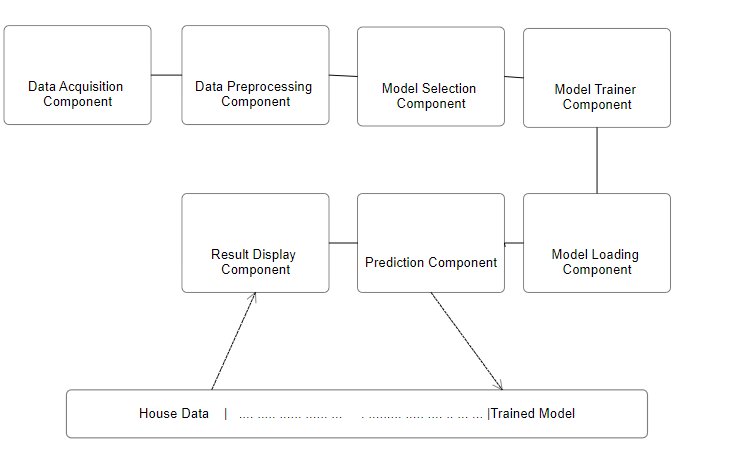
Sequence Diagrams are easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and its environment. A sequence diagram shows the interaction arranged in a time sequence. It shows the object participating by their lifelines and the messages they exchange, arranged in a time sequence.



**Figure 5.4.1 Sequence Diagram**

## COMPONENT DIAGRAM

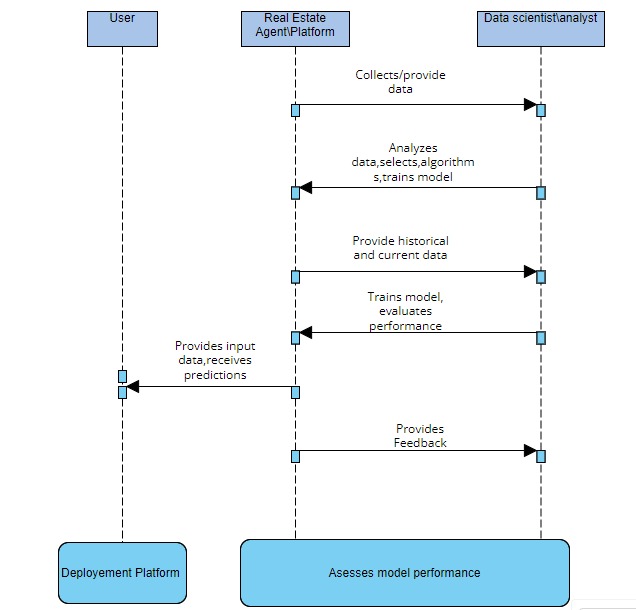
Component diagram models the physical components such as source codes, executable programs, user interface in a design. These high level physical components may or may not be equivalent to many smaller components we use in creation of application.



**Figure 5.5.1 Component Diagram**

## COLLABORATION DIAGRAM

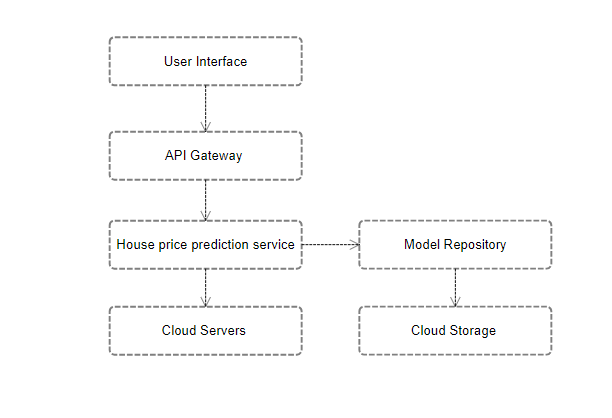
A collaboration diagram represents a collaboration, which is a set of objects related in a particular context, and interaction, which is a set of messages exchanged among the objects within the collaboration to achieve a desired outcome.



**Figure 5.6.1 Login Collaboration Diagram**

## DEPLOYMENT DIAGRAM

Deployment diagram shows the configuration of run time processing elements and the software components, processes and objects that live in them.



**Figure 5.7.1 Deployment Diagram**

# CHAPTER-6

# SYSTEM PROTOTYPING

Figure 6.1 LOGIN PAGE

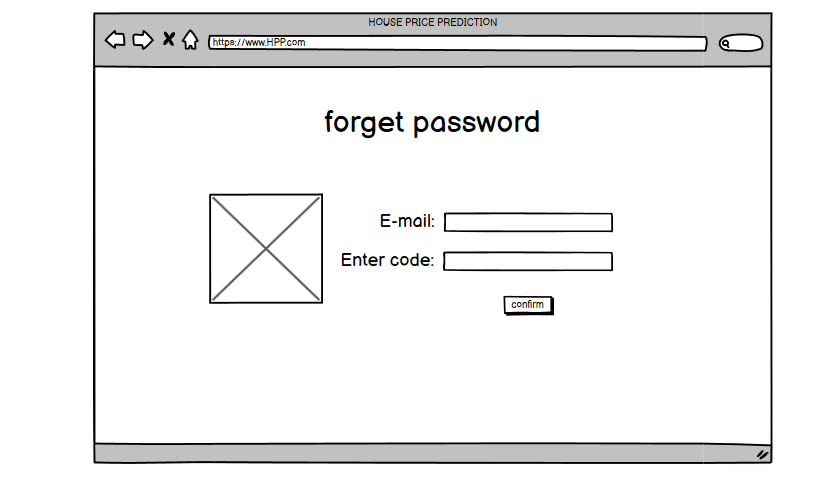


Figure 6.2 SIGNUP I PAGE

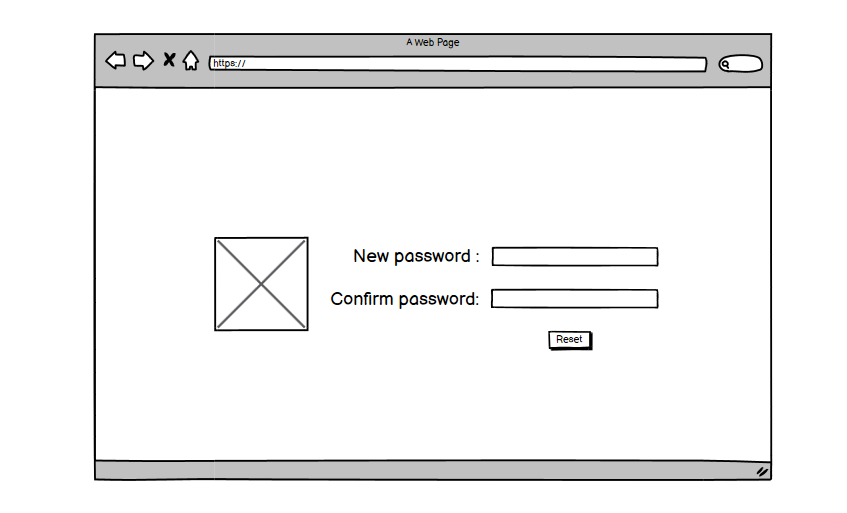


Figure 6.3 SIGN UP II PAGE

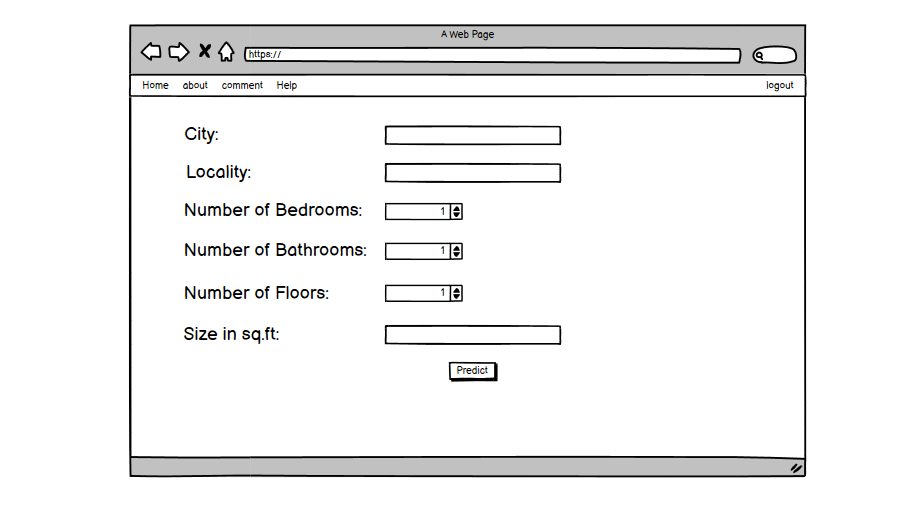
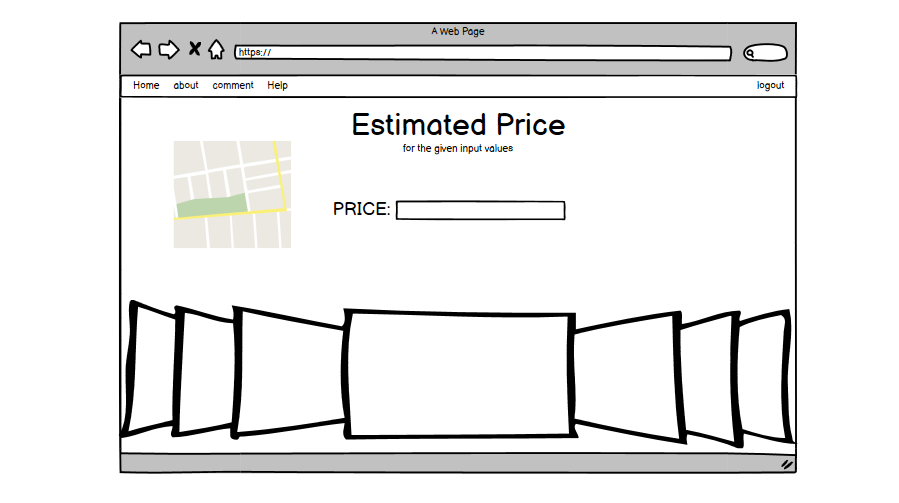


Figure 6.4 WORKING PAGE

Figure 6.5 OUTPUT PAGE

# CHAPTER-7

**SYSTEM IMPLEMENTATION**

**System Implementation for House Price Prediction:**

**1. Setting Up Development Environment:**

- Choose appropriate programming languages and libraries for implementation (e.g., Python, scikit-learn, TensorFlow, etc.).

- Install necessary development tools and packages.

**2. Data Preprocessing:**

- Load the collected dataset into the development environment.

- Preprocess the data by handling missing values, encoding categorical variables, and performing feature scaling or normalization.

**3. Model Development:**

- Select a suitable machine learning algorithm for regression tasks (e.g., Linear Regression, Decision Trees, Random Forest, etc.).

- Split the dataset into training and testing sets.

- Train the selected model on the training data.

**4. Model Evaluation:**

- Evaluate the trained model's performance using appropriate evaluation metrics (e.g., Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, etc.).

- Validate the model's effectiveness using cross-validation techniques to ensure robustness.

**5. Fine-Tuning:**

- Fine-tune hyperparameters of the model using techniques like grid search or random search to optimize performance further.

**6. Deployment:**

- Once the model is trained and evaluated satisfactorily, deploy it into a production environment.

- Choose deployment options based on the system's requirements, such as deploying as a web service, integrating into existing applications, or packaging as a standalone executable.

- Ensure scalability, reliability, and efficiency of the deployed model.

**7. User Interface (UI):**

- Develop a user-friendly interface for interacting with the prediction system.

- Design the UI to allow users to input house features and receive price predictions conveniently.

- Implement visualizations to display insights from the data and model predictions effectively.

**8. Integration:**

- Integrate the prediction system with other systems or platforms if necessary, such as real estate websites, mobile applications, or property management tools.

- Ensure seamless data exchange and communication between the prediction system and integrated components.

**9. Testing:**

- Conduct thorough testing of the implemented system to ensure functionality, reliability, and accuracy.

- Perform unit tests, integration tests, and end-to-end tests to validate different aspects of the system's behavior.

**10. Documentation:**

- Document the implementation process, including setup instructions, code documentation, and system architecture diagrams.

- Provide user manuals or guides for utilizing the prediction system effectively.

**11. Training and Support:**

- Provide training sessions or materials for users, administrators, and stakeholders on using the prediction system.

- Offer ongoing support and troubleshooting assistance to address any issues or questions that arise during system usage.

**12. Security and Compliance:**

- Implement security measures to protect sensitive data and prevent unauthorized access to the system.

- Ensure compliance with data protection regulations and industry standards related to privacy and security.

**13. Monitoring and Maintenance:**

- Set up monitoring mechanisms to track the performance and usage of the prediction system in production.

- Establish procedures for periodic maintenance, updates, and enhancements to keep the system functional and up-to-date.

**14. Feedback and Improvement:**

- Gather feedback from users and stakeholders to identify areas for improvement and enhancements.

- Incorporate feedback into future iterations of the prediction system to enhance its effectiveness and usability.

**15. Deployment and Rollout:**

- Deploy the implemented prediction system to the target environment.

- Monitor the deployment process and ensure a smooth rollout to users and stakeholders.

- Communicate changes and updates effectively to users and provide support during the transition period.

# CHAPTER 8

# SYSTEM TESTING:

# 8.1 TEST PLAN:

# 8.1.1 PROJECT DESCRIPTION:

# The project involves the development of a house price prediction system. The basic steps in the project are as follows:

# Data Collection: Gather data on various factors that affect house prices, such as location, size, number of rooms, amenities, etc.

# Data Preprocessing: Clean the data, handle missing values, normalize features, and perform feature engineering if necessary.

# Model Building: Develop machine learning models to predict house prices based on the collected and preprocessed data.

# Model Evaluation: Evaluate the performance of the models using appropriate evaluation metrics.

# 8.1.2 TESTING STRATEGY:

# Unit testing and Functional testing will be used in each phase of testing to ensure the quality and reliability of the system.

# 8.1.3 TESTING LEVEL PLAN:

# White Box Testing will be given higher priority than Black Box Testing to thoroughly examine the internal logic of the system.

# 8.1.4 UNIT TEST:

# Unit testing will be conducted by the person responsible for writing the code. White box testing will be primarily used at this level to identify errors in logic.

# 8.1.5 MODULE TEST:

# Module testing will be performed by a single programmer or a small group of programmers who write units that work together in a single module. Test cases will be utilized for module testing, especially if the module is self-contained.

# 8.1.6 INTEGRATION TEST:

# Integration tests will be handled by the development team. Incremental integration testing will be preferred, adding one module at a time to isolate errors.

# 8.1.8 ALPHA TEST:

# In this test, "Internal" testers will run live data through the system to identify bugs not found in integration tests. The customer may wish to observe the alpha test or provide some "Real" data.

# 8.1.9 BETA TEST:

# Beta testing is the next step, where the program is released to the customer with the understanding that it is still being tested. The customer agrees to stress the application and report any discovered bugs or problems to the development team. The team agrees to be a "Friendly" user but to really put the system through its paces, trying to break it.

# 8.1.10 REGRESSION TEST:

# After the product is released, errors may be found, or enhancements suggested by the customers in the field. As these are corrected or implemented, the rest of the system must also be tested again to ensure that new fixes did not break any of the old code. Regression testing is usually performed using an automated script that runs a set of cases known to exercise the entire system.

# 8.1.11 USABILITY TEST:

# Usability testing is a special form of testing that focuses on the layout and utility of the user interface rather than the functionality of the program. This step is often a prototype before the actual system code is written, making it easy to modify if needed.

# CHAPTER-9

**CONCLUSION AND FUTURE ENHANCEMENTS**

In conclusion, the House Price Prediction System represents a crucial tool for both buyers and sellers in the real estate market, facilitating informed decision-making and contributing to overall market efficiency. By leveraging machine learning algorithms, this system accurately predicts house prices based on various features, empowering users with valuable insights.

However, there are opportunities for future enhancements and advancements in the system:

1. **Enhanced Feature Engineering:** The system could benefit from more sophisticated feature engineering techniques to capture nuanced characteristics of properties, such as neighborhood amenities, architectural styles, or historical market trends.

2. **Incorporation of External Data Sources:** Integrating additional data sources, such as economic indicators, demographic trends, or environmental factors, could enrich the predictive models and improve their accuracy and robustness.

3. **Personalized Recommendations**: Tailoring price predictions based on individual preferences and requirements could enhance user experience and satisfaction, providing more relevant and actionable insights.

4. **Interactive Visualization Tools:** Developing interactive visualization tools could empower users to explore and analyze the data intuitively, gaining deeper insights into market trends and property dynamics.

5**. Integration with Real Estate Platforms**: Seamless integration with real estate platforms and applications would streamline the process of accessing and utilizing price predictions, enhancing accessibility and usability for stakeholders.

Overall, the House Price Prediction System holds significant potential to revolutionize the real estate industry, driving efficiency, transparency, and informed decision-making. Continued research and development efforts will further enhance its capabilities and impact in the market.

**APPENDIX 1**

**SAMPLE CODE**

from flask import Flask, render\_template, request

import joblib

import pandas as pd

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.model\_selection import train\_test\_split

app = Flask(\_\_name\_\_)

train\_data = pd.read\_csv('train.csv')

X = train\_data.drop(['TARGET(PRICE\_IN\_LACS)'], axis=1)

y = train\_data['TARGET(PRICE\_IN\_LACS)']

X\_train, \_, \_, \_ = train\_test\_split(X, y, test\_size=0.1, random\_state=42)

n\_features = X\_train.shape[1]

try:

loaded\_model = joblib.load('E:\house price prediction\gradient\_boosting\_model (1).pkl')

except FileNotFoundError:

print("Model file not found. Please check the file path.")

exit()

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

@app.route('/predict', methods=['POST'])

def predict():

if request.method == 'POST':

try:

user\_input = {

'POSTED\_BY': 1 if request.form['POSTED\_BY'] == 'Owner' else 0,

'UNDER\_CONSTRUCTION': int(request.form['UNDER\_CONSTRUCTION']),

'RERA': int(request.form['RERA']),

'BHK\_NO.': int(request.form['BHK\_NO.']),

'BHK\_OR\_RK': 1 if request.form['BHK\_OR\_RK'] == 'BHK' else 0,

'SQUARE\_FT': int(request.form['SQUARE\_FT']),

'READY\_TO\_MOVE': int(request.form['READY\_TO\_MOVE']),

'RESALE': int(request.form['RESALE']),

}

except ValueError:

return render\_template('error.html', message='Invalid input. Please enter valid values.')

try:

prediction = loaded\_model.predict(pd.DataFrame(user\_input, index=[0]))[0]

# Round the prediction to the nearest integer and convert to lacs

prediction\_in\_lacs = round(prediction) / 100000

except Exception as e:

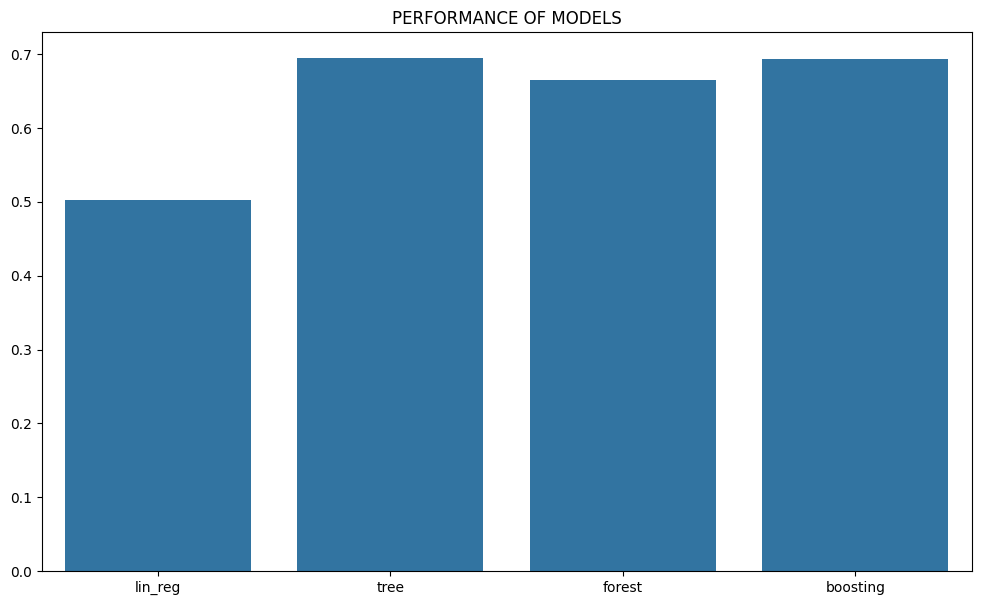
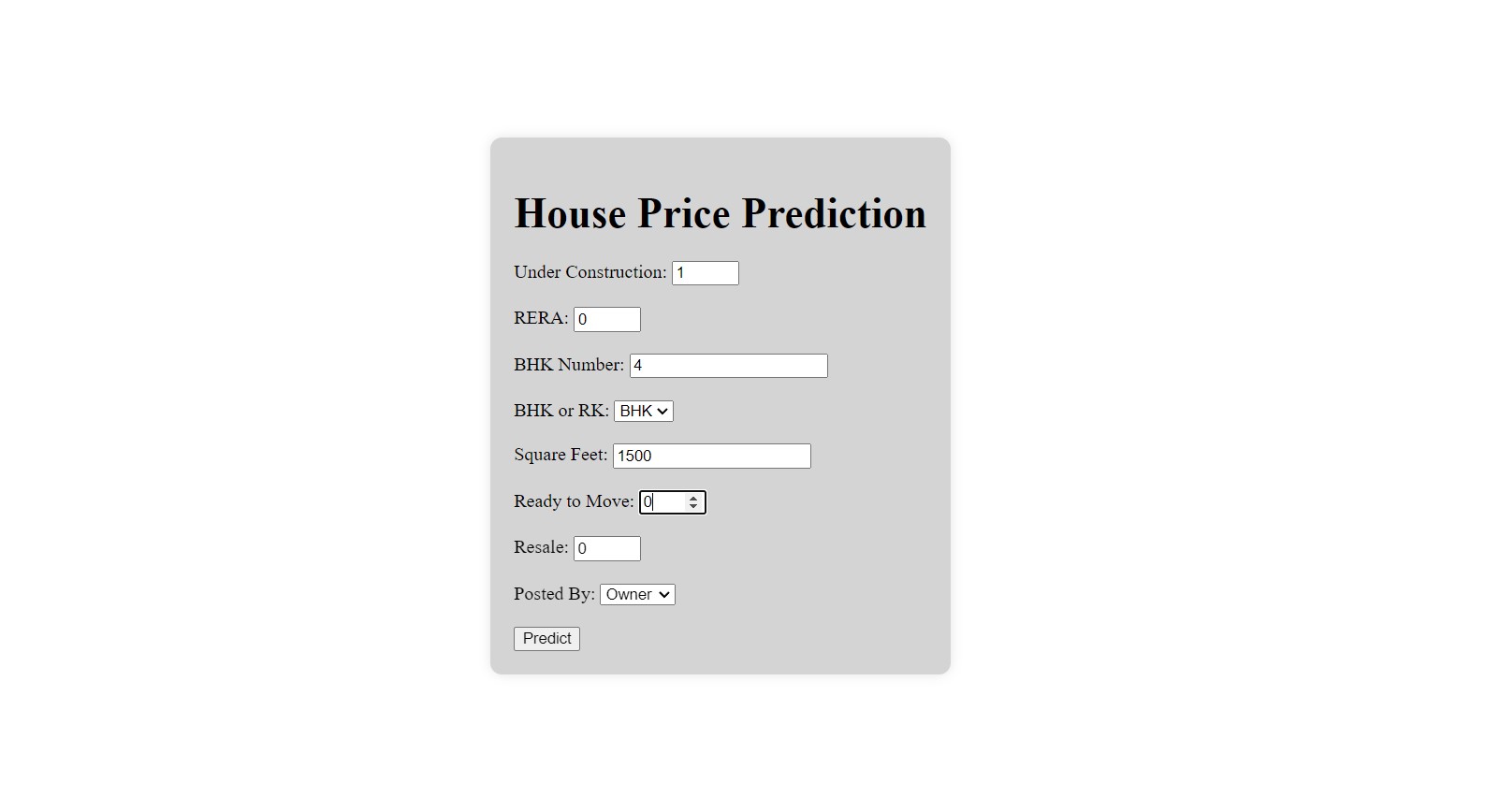
return render\_template('error.html', message='An error occurred while making the prediction. {}'.format(str(e)))

return render\_template('result.html', prediction=prediction\_in\_lacs)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

## SCREENSHOTS OF THE IMPLEMENTATION



**Figure 9.1.1 House price prediction user input**

**Figure 9.1.2 Predicted output**