RealValuator: Real Estate Price Predictor

Coding And Testing

for

Project Work -1

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

BY

Harshita Chouhan (EN22CS301406)
Harshita Mantri (EN22CS301407)
Harshita Parmar (EN22CS301408)

Under the Guidance of

Prof. Shivani Patnaha
Prof. Dhiraj Kumar

Department of Computer Science & Engineering
Faculty of Engineering
MEDICAPS UNIVERSITY, INDORE- 453331

Aug -Dec 2025

1. Coding Phase

1.1 Introduction

The coding phase of the ***RealValuator*** project involves converting the system design into a fully functional machine learning–powered web application that predicts property prices in Bangalore. The system integrates a trained Linear Regression model, a Flask backend, and an interactive frontend for users to input property details such as square footage, number of bedrooms, bathrooms, and location.

The frontend is developed using **HTML, CSS, and JavaScript**, while the backend is implemented in **Flask (Python)**. Machine learning components are built using **Pandas, NumPy, and scikit-learn**. The trained model and encoded column metadata are stored as artifacts for efficient loading during runtime. The system follows a modular architecture separating UI, prediction services, and data handling layers.

1.2 Coding Objectives

- Build a clean, modular, and user-friendly prediction interface.
- Integrate machine learning model artifacts for real-time price estimation.
- Implement robust backend endpoints for location retrieval and prediction.
- Ensure seamless communication between frontend forms and backend APIs.
- Provide meaningful error messages and user input validation.
- Serve static pages and assets through the Flask backend.

1.3 Tools and Technologies Used

Technology	**Purpose**
HTML / CSS / JavaScript	Frontend structure, styles, and dynamic UI behaviour
Flask (Python)	Backend server, API routing, static file hosting
Pandas / NumPy	Data processing and numerical operations
scikit-learn (Linear Regression)	ML model training and prediction
Pickle / JSON	Model & metadata serialization
Chrome DevTools	Debugging, network monitoring, performance inspection
VS Code	Code development and debugging

1.4 Code Description

The project codebase is structured into separate frontend, backend, and model artifact modules.

- **Frontend Modules**

  - **Landing Page (real.html):**
    Displays project overview, navigation sidebar, hero section, and feature cards.

  - **Price Predictor UI (app.html):**
    Contains input fields for sqft, BHK, bathrooms, and location dropdown.

  - **Frontend Logic (app.js):**

    ○ Fetches available locations from backend ○ Validates user inputs ○ Sends prediction request ○ Displays result or errors

  - **Styling (app.css):**
    Handles layout, background visuals, form design, and responsive styling.

- **Backend Modules**

  - **Flask Server (server.py):**
    Hosts APIs / api/get_location_names and /api/predict_home_price, serves static files, and handles routing.

  - **Prediction Utilities (util.py):**
    Loads ML model, loads JSON column metadata, and performs numerical prediction.

  - **Artifacts Loader:**
    Manages loading of:

    - columns.json

2. Testing Phase

2.1 Introduction

Testing ensures that the *RealValuator* system functions correctly and meets the expected requirements. A combination of **unit testing, integration testing, system testing, and user acceptance testing (UAT)** was carried out to verify prediction accuracy, UI flow, backend behavior, and ML model reliability.

2.2 Testing Objectives

- Validate correct functionality of prediction API.
- Ensure proper integration of frontend form with backend endpoints.

- Verify model accuracy and correct feature input mapping.
- Detect and fix UI/UX issues early.
- Ensure the system handles invalid inputs gracefully.

2.3 Types of Testing

Type	Description	Tools / Method Used
Unit Testing	Testing standalone components (e.g., util.py functions).	Python Unit Tests
Integration Testing	Checking frontend–backend API communication.	Postman, browser tests
System Testing	Testing the full prediction workflow end-to-end.	Manual browser validation
User Acceptance Testing (UAT)	Testing usability, clarity, and responsiveness.	External test users
Performance Testing	Evaluating prediction speed and page loading.	Chrome DevTools

2.4 Test Cases

Test Case ID	Module	Description	Expected Output	Result
TC01	Location API	Frontend request's available locations	List of city locations returned	Pass
TC02	Prediction API	User submits valid sqft + BHK + bath + location	Predicted price returned in lakhs	Pass

TC03	Input Validation	User submits empty or invalid fields	Error message displayed	Pass
TC04	UI Rendering	Predictor form loads correctly	All fields visible and clickable	Pass
TC05	Model Loading	Backend loads ML artifacts on startup	Model and columns loaded without errors	Pass
TC06	Page Navigation	User moves from landing page to predictor	Smooth navigation without reload errors	Pass

2.5 Testing Results

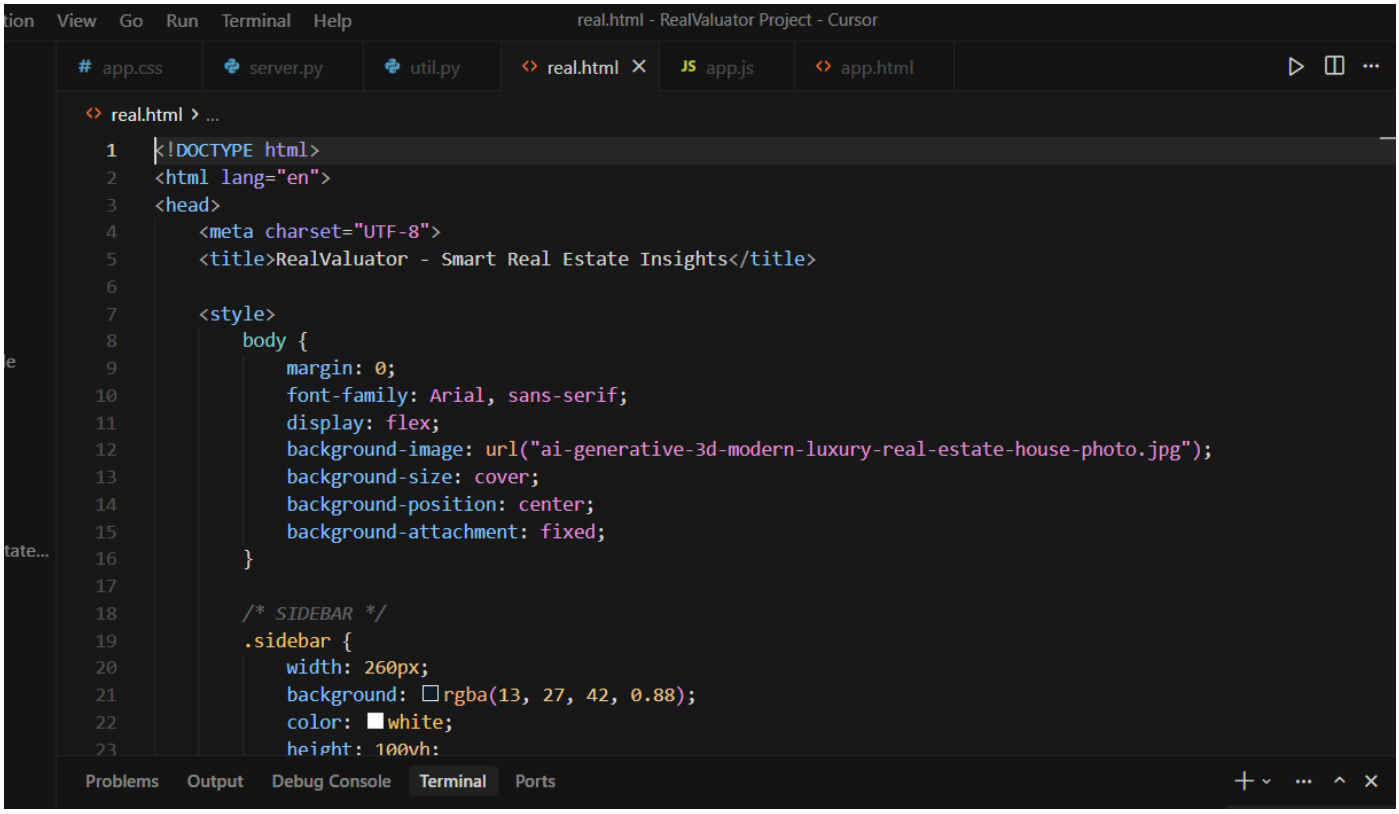
All modules were tested successfully. The backend API responded correctly to all requests, and the prediction model produced the expected price outputs. Minor UI alignment issues and invalid input cases were corrected during integration testing.
The model loading process and API response times were optimized for smooth user experience.

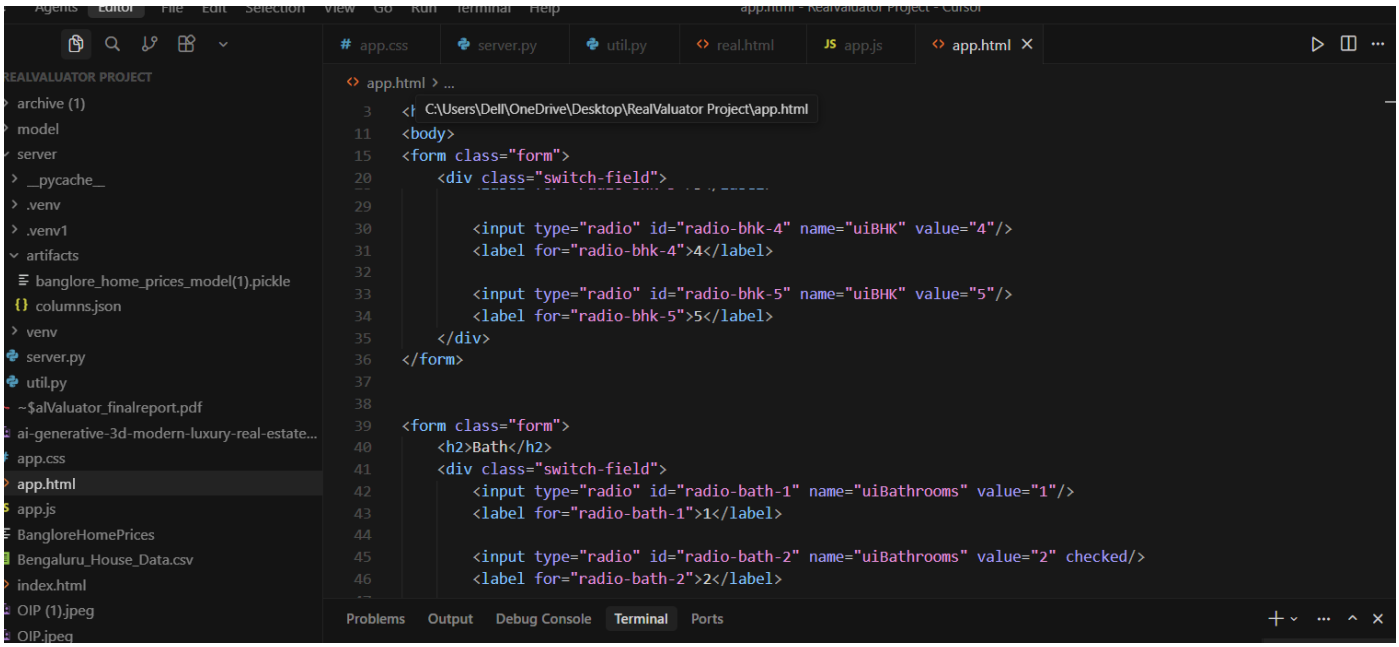
2.6 Conclusion

The *RealValuator* project successfully completed the coding and testing phase. All core functionalities—including location fetching, prediction generation, error handling, and UI rendering—operate as intended.
With a stable backend service, accurate ML predictions, and a polished frontend interface, the application is now ready for deployment and further enhancements.

Maintenance ensures that the RealValuator system continues to function efficiently after deployment by addressing errors, updating data, retraining models, and enhancing system features. Since real estate markets evolve over time, regular maintenance is essential to keep the prediction model accurate and the system reliable. Maintenance activities include monitoring system performance, fixing bugs, updating datasets, improving model accuracy, and ensuring smooth user experience. The goal of maintenance is to ensure longterm usability, adaptability, and security of the system.

3. Code Screenshots





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

/* Background Image */
.img {
  background-image: url("ai-generative-3d-modern-luxury-real-estate-house-photo.jpg");
  background-repeat: no-repeat;
  background-size: cover;
  filter: blur(12px);
  position: fixed;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  z-index: -1;
}


/* Radio Button Group */
.switch-field {
    display: flex;
    margin-bottom: 36px;
}
```

```
Output  Debug Console  Terminal  Ports
powershell
Cursor (cd "...
Cursor (cd "...
Cursor (cd "...
```

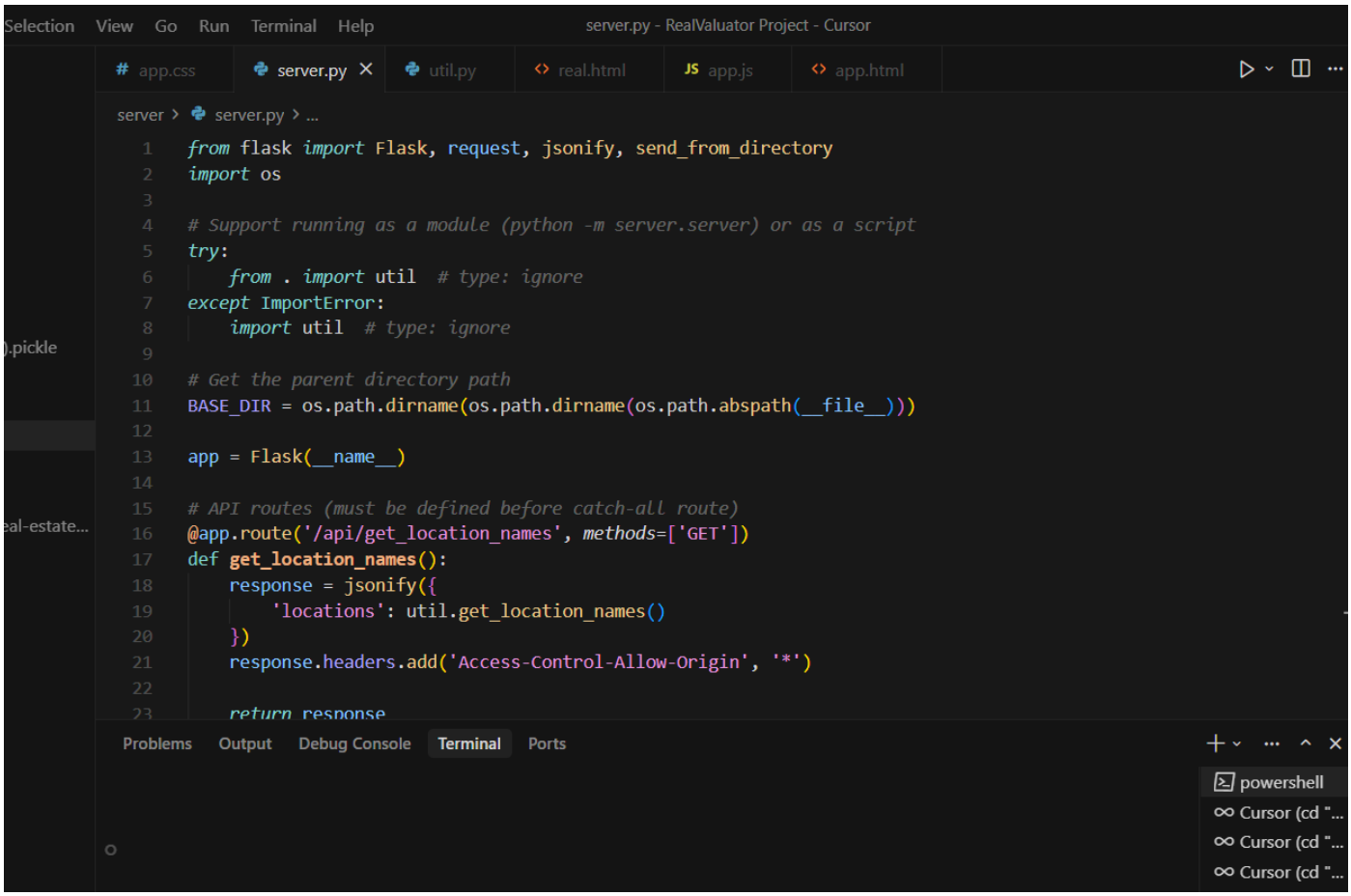
```
Selection  View  Go  Run  Terminal  Help        app.js - RealValuator Project - Cursor
app.css  server.py  util.py  real.html  app.js  app.html
app.js > onClickedEstimatePrice
```

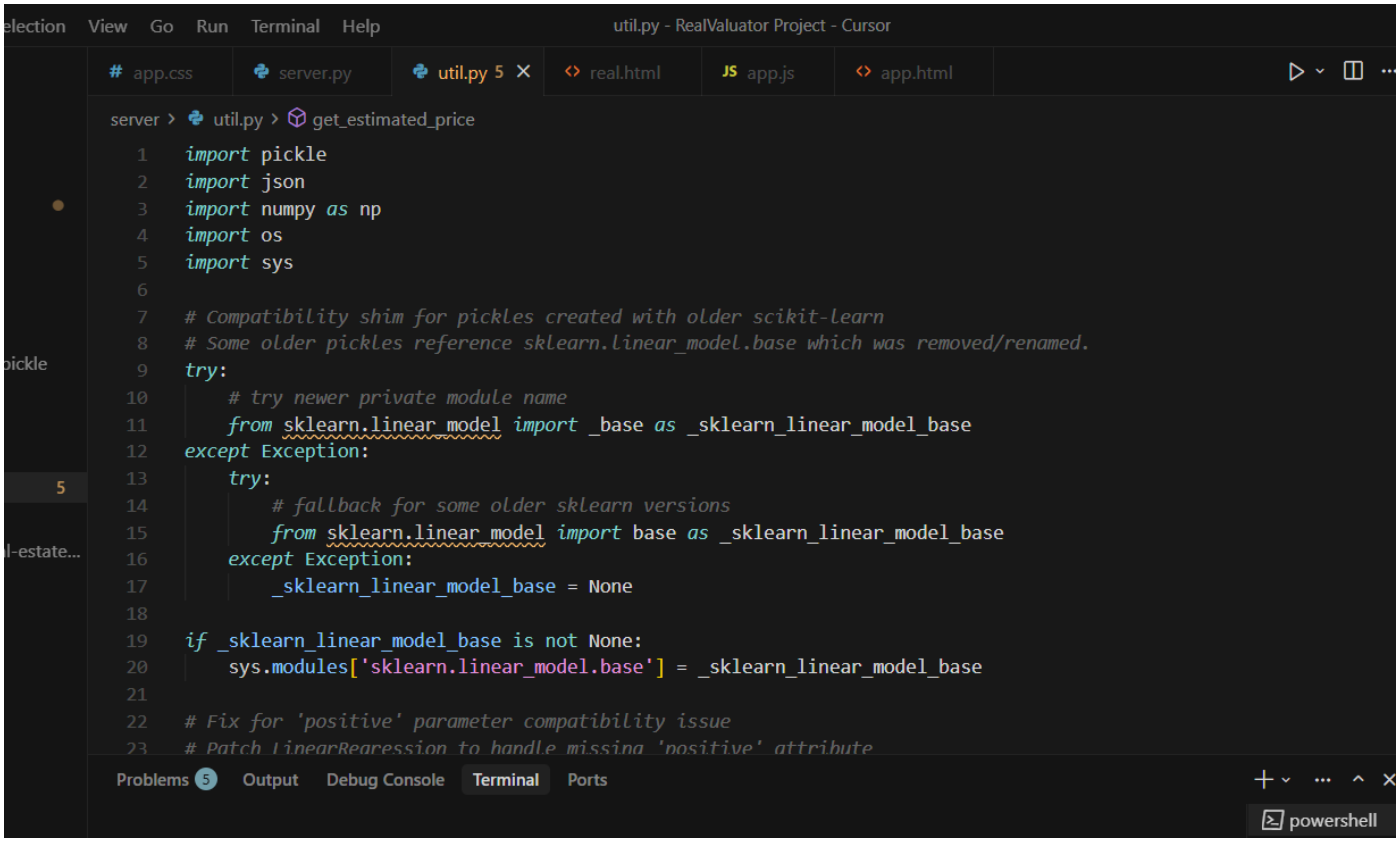
```js
function getBathValue() {
  var uiBathrooms = document.getElementsByName("uiBathrooms");
  for (var i = 0; i < uiBathrooms.length; i++) {
    if (uiBathrooms[i].checked) {
      return parseInt(uiBathrooms[i].value);
    }
  }
  return -1;
}

function getBHKValue() {
  var uiBHK = document.getElementsByName("uiBHK");
  for (var i = 0; i < uiBHK.length; i++) {
    if (uiBHK[i].checked) {
      return parseInt(uiBHK[i].value);
    }
  }
  return -1;
}

function onClickedEstimatePrice() {
  var sqft = document.getElementById("uiSqft");
  var bhk = getBHKValue();
```

```
Problems  Output  Debug Console  Terminal  Ports
powershell
Cursor (cd
Cursor (cd
Cursor (cd
```


Selection View Go Run Terminal Help
server.py - RealValuator Project - Cursor
app.css
server.py
util.py
real.html
app.js
app.html
server > server.py > ...
from flask import Flask, request, jsonify, send_from_directory
import os
# Support running as a module (python -m server.server) or as a script
try:
from . import util # type: ignore
except ImportError:
import util # type: ignore
# Get the parent directory path
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
app = Flask(__name__)
# API routes (must be defined before catch-all route)
@app.route('/api/get_location_names', methods=['GET'])
def get_location_names():
response = jsonify({
'locations': util.get_location_names()
})
response.headers.add('Access-Control-Allow-Origin', '*')
return response
Problems Output Debug Console Terminal Ports
powershell
Cursor (cd "...
Cursor (cd "...
Cursor (cd "...


election View Go Run Terminal Help
util.py - RealValuator Project - Cursor
app.css
server.py
util.py 5
real.html
app.js
app.html
server > util.py > get_estimated_price
import pickle
import json
import numpy as np
import os
import sys
# Compatibility shim for pickles created with older scikit-learn
# Some older pickles reference sklearn.linear_model.base which was removed/renamed.
try:
# try newer private module name
from sklearn.linear_model import _base as _sklearn_linear_model_base
except Exception:
try:
# fallback for some older sklearn versions
from sklearn.linear_model import base as _sklearn_linear_model_base
except Exception:
_sklearn_linear_model_base = None
if _sklearn_linear_model_base is not None:
sys.modules['sklearn.linear_model.base'] = _sklearn_linear_model_base
Fix for 'positive' parameter compatibility issue
Problems 5 Output Debug Console Terminal Ports
powershell