

**REPORT**

**ON**

**PROPERTY DEKHO**

## **Table Of Contents :-**

<b>S.No.</b>	<b>Topics</b>
<b>1.</b>	<b>Introduction</b>
<b>2.</b>	<b>Requirement Analysis</b>
	a) Scope b) Objective c) Limitations
<b>3.</b>	<b>Analysis Phase</b>
	d) Requirements
<b>4.</b>	<b>Design Phase</b>
	e) Flow Chart f) Use Case Diagram
<b>5.</b>	<b>Development Phase</b>
	g) Source Code
<b>6.</b>	<b>Testing Phase</b>
	h) System Screenshots
<b>7.</b>	<b>Facing Problems</b>
<b>8.</b>	<b>Conclusion</b>

## **Introduction :-**

Property Dekho Project is a web-based application. It is a dynamic website. Having a house is every ones need and finding a house in a new city is very difficult. This website provide you with information about your desired property at any city you want. You don't have to go outside and search for the property that you want. You just need to site on your system and make us do our job we will help you to find your best suitable property and you will get each and every information about the property like- price, area size, location, amenities, furnishings, owner of property and some pictures of the property as well. And if you are the owner and a broker who want to sell or rent his/her property then you can post your property on our website in no matter of time your property is visible to all who are looking for the similar one and contact you. The proposed project maintain a centralized repository of all property details. We are working hard to further add some functionality in this website for more greater experience. And we are also working on our mobile application for future.

# **1. Requirement Analysis**

## **Scope:**

Property Dekho project is an web-based application. The main aim of this project is to help users to find their desired property for rent or sell and see full details of the property by sitting at home and can make enquiry about the property by directly contacting the owner or broker of the property. The broker or owner can post their respected property easily and can come across the wide range of property buyers.

## **Objective :**

The Property Dekho Project maintains a centralized repository of all related information. The Objective of this project is to develop a project that automates the process and activities of property details.

The purpose is to design a project using which can perform all operations related to properties.

## **Limitations :**

This project only gives you the information about the property and the owner and broker of the respected property, you can buy or rent any property on our website, you can only get the desired information.

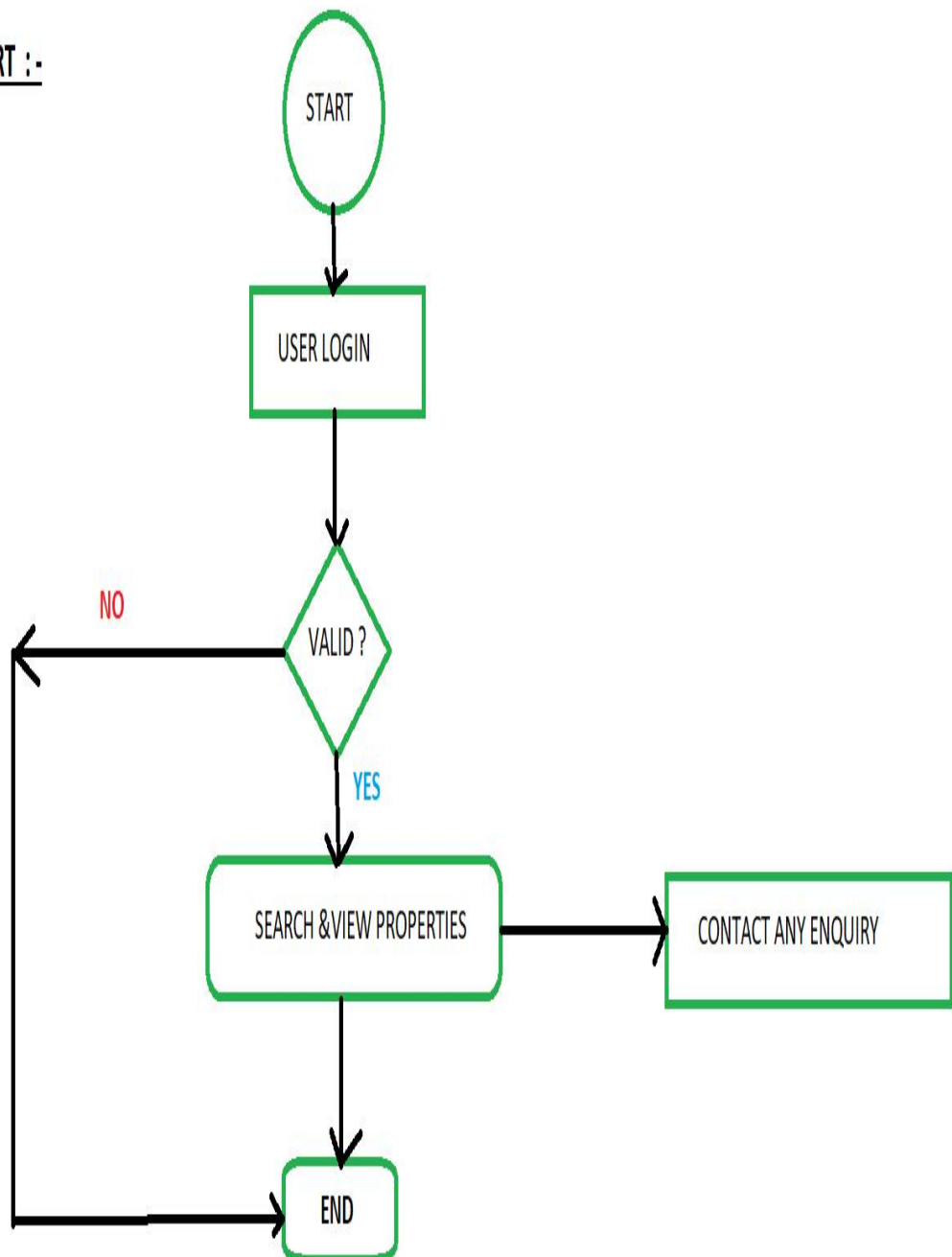
## 2. Analysis Phase :-

### Requirements :-

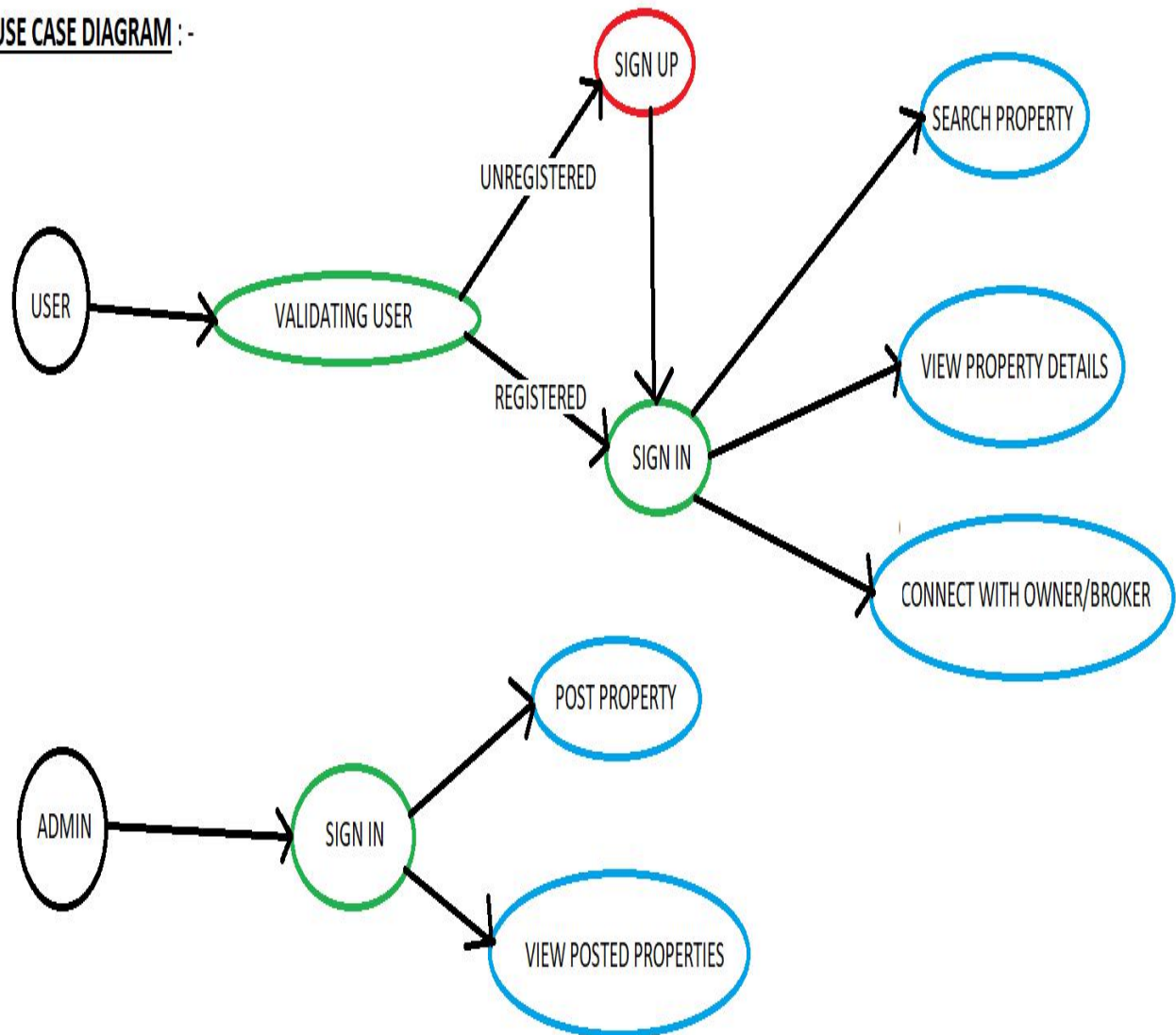
- **Operating System** : Windows
- **Framework** : React js
- **Front End** : HTML 5, CSS 3, BOOTSTRAP 4, JAVASCRIPT, JQUERY
- **Back End** : Node js, Express js
- **Database** : Mongo DB
- **Sharing** : Git, Github
- **Editor** : Visual Studio Code
- **Web Browser** : Google Chrome

### **3. Design Phase :-**

FLOW CHART :-



USE CASE DIAGRAM :-



**4. Development Phase :-**



## App.js :-

```
JS App.js X
myhome > frontend > src > JS App.js > App > useEffect() callback
1  import './App.css';
2  import Header from './Components/Header';
3  import {BrowserRouter as Router} from 'react-router-dom';
4  import {useEffect} from 'react';
5  import {useDispatch} from 'react-redux';
6  import axios from 'axios';
7
8  function App() {
9
10     const dispatch = useDispatch();
11
12     useEffect(() => {
13         if(localStorage.getItem("LOGIN_ID") == 'no'){
14         }else{
15             var id = localStorage.getItem("LOGIN_ID");
16             dispatch({type: "LOGIN_TRUE"});
17             dispatch({type: "LOGGEDIN",payload: id });
18             axios.get('http://localhost:3000/getUser?id='+id).then((res)=>{
19                 if(res.data.data[0].iam == "buyer"){
20                     dispatch({type: "BUYER"});
21                 }
22             });
23         }
24     }, [])
25
26     return (
27         <Router>
28         <div>
29             <Header/>
30         </div>
31         </Router>
32     );
33 }
34
35 export default App;
36
```

## Backend Code :-

```
JS index.js X
myhome > frontend > public > Details > backend > JS index.js > app.post('/postProperty') callback > upload() callback

1  var express = require('express');    File is a CommonJS module; it may be converted to an ES6 module.
2  var cors = require('cors');
3  var bodyParser = require('body-parser');
4  var MongoClient = require('mongodb').MongoClient;
5  var ObjectId = require('mongodb').ObjectId;
6  var path = require('path');
7  var upload = require('./multerConfig');
8  var fs = require('fs');
9
10 var app = express();
11 app.use(cors());
12
13 app.use(express.static(path.join(__dirname, 'userUploads')));
14
15 var client = new MongoClient('mongodb+srv://myhome:myhome@cluster0.zj3m5.mongodb.net/myhome?retryWrites=true&w=majority',{useNewUrlParser:
16 var connection;
17 client.connect((err, db)=>{
18     if(!err){
19         connection = db;
20         console.log("Database Connected Successfully");
21     }
22     else{
23         console.log(err);
24     }
25 });
26
27 app.post('/postProperty', bodyParser.json(),(req,res)=>{    'bodyParser' is deprecated.
28     upload(req,res,(err)=>{
29         if(err){
30             console.log("Error Occured during upload ");
31             console.log(err);
32             res.send({status:"failed", data:err});
33         }
34         else{
35             var propertyCollection = connection.db('myhome').collection('property');
36             var images = req.files.property.map(p=>p.filename);
37             propertyCollection.insert({PropertyDetails:req.body,PropertyImages:images},(err,result)=>{    'result' is declared but its va
38                 if(!err){
```

```
JS index.js X
myhome > frontend > public > Details > backend > JS index.js > app.post(/postProperty) callback > upload() callback

48
49 app.get('/listProperty', (req, res) => { 'req' is declared but its value is never read.
50   var propertyCollection = connection.db('myhome').collection('property');
51   propertyCollection.find({}).toArray((err, docs) => {
52     if (!err) {
53       res.send({ status: "ok", data: docs });
54     }
55     else {
56       res.send({ status: "failed", data: err });
57     }
58   });
59 });
60
61 app.get('/detailProperty', (req, res) => {
62   var propertyCollection = connection.db('myhome').collection('property');
63   propertyCollection.find({ _id: ObjectID(req.query.id) }).toArray((err, docs) => {
64     if (!err) {
65       res.send({ status: "ok", data: docs });
66     }
67     else {
68       res.send({ status: "failed", data: err });
69     }
70   });
71 });
72
73 app.get('/searchProperty', (req, res) => { 'req' is declared but its value is never read.
74   var propertyCollection = connection.db('myhome').collection('property');
75   propertyCollection.find({ city: "", budget: "" }).toArray((err, docs) => {
76     if (!err) {
77       res.send({ status: "ok", data: docs });
78     }
79     else {
80       res.send({ status: "failed", data: err });
81     }
82   });
83 });
84
```

```
JS index.js X
myhome > frontend > public > Details > backend > JS index.js > app.post(/postProperty) callback > upload() callback

119 });
120
121 app.get('/loginUser', (req, res) => {
122   var userCollection = connection.db('myhome').collection('user');
123   userCollection.find({ email: req.query.email, password: req.query.password }).toArray((err, docs) => {
124     if (!err) {
125       res.send({ status: "ok", data: docs });
126     }
127     else {
128       res.send({ status: "failed", data: err });
129     }
130   });
131 });
132
133 app.post('/uploadDP', bodyParser.json(), (req, res) => { 'bodyParser' is deprecated.
134   upload(req, res, (err) => {
135     if (err) {
136       console.log("Error Occured during upload ");
137       console.log(err);
138       res.send({ status: "failed", data: err });
139     }
140     else {
141       var userCollection = connection.db('myhome').collection('user');
142       var image = req.files.profile[0].filename;
143       userCollection.update({ _id: ObjectID(req.body.id) }, { $set: { dp: image } }, (err, result) => { 'result' is declared but its value is
144         if (!err) {
145           res.send({ status: "ok", data: "File Uploaded Succesfully" });
146         }
147         else {
148           res.send({ status: "failed", data: err });
149         }
150       });
151     }
152   });
153 });
154
155 app.post('/removedP', bodyParser.json(), (req, res) => { 'bodyParser' is deprecated.
156   var userCollection = connection.db('myhome').collection('user');
```

```

JS index.js X
myhome > frontend > public > Details > backend > JS index.js > app.post('/postProperty') callback > upload() callback
159     fs.unlinkSync('userUploads/'+req.body.dp);
160     res.send({status:"ok",data:"DP Removed Succesfully"});
161   }
162   else{
163     res.send({status:"failed",data:err});
164   }
165   });
166 });
167
168 app.post('/updateName' , bodyParser.json(),(req,res)=>{ 'bodyParser' is deprecated.
169   var userCollection = connection.db('myhome').collection('user');
170   userCollection.update({_id:ObjectID(req.body.id)},{ $set:{name:(req.body.name)}},(err,result)=>{ 'result' is declared but its value is never read.
171     if(!err){
172       res.send({status:"ok",data:"Changes Made Succesfully"});
173     }
174     else{
175       res.send({status:"failed",data:err});
176     }
177   });
178 });
179
180 app.post('/updateNumber' , bodyParser.json(),(req,res)=>{ 'bodyParser' is deprecated.
181   var userCollection = connection.db('myhome').collection('user');
182   userCollection.update({_id:ObjectID(req.body.id)},{ $set:{number:(req.body.number)}},(err,result)=>{ 'result' is declared but its value is never read.
183     if(!err){
184       res.send({status:"ok",data:"Changes Made Succesfully"});
185     }
186     else{
187       res.send({status:"failed",data:err});
188     }
189   });
190 });
191
192 app.listen(3000,()=>{
193   console.log("Server is listing at port 3000");
194 });

```

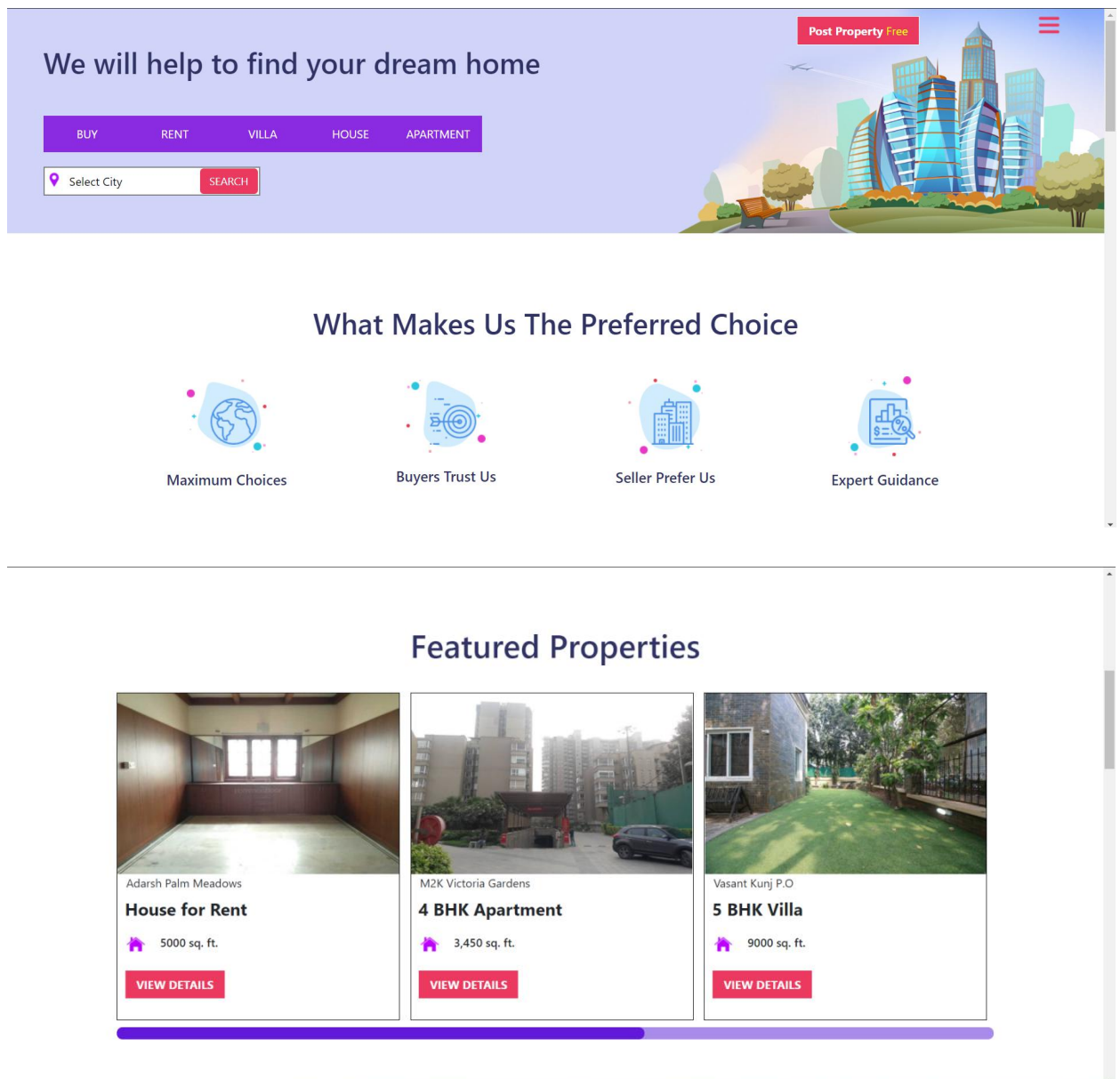
```

JS multerConfig.js X
myhome > frontend > public > Details > backend > JS multerConfig.js > multerOptions > fileFilter
1  var path = require('path'); File is a CommonJS module; it may be converted to an ES6 module.
2  var multer = require('multer');
3
4  var storage = multer.diskStorage(
5    {
6      destination: function(req,file,cd){ 'req' is declared but its value is never read.
7        cd(null,'userUploads');
8      },
9      filename: function(req,file,cd){ 'req' is declared but its value is never read.
10         var ext = path.extname(file.originalname);
11         cd(null,file.fieldname+'-'+Date.now()+'.'+ext);
12       }
13     }
14 )
15
16 var multerOptions = {
17   storage: storage,
18   filefilter: function(req,file,callback){ 'req' is declared but its value is never read.
19     var ext = path.extname(file.originalname);
20     var fieldName = file.fieldname;
21
22     if(fieldName == "property"){
23       if(ext !== '.png' && ext !== '.jpg' && ext !== '.jpeg') {
24         return callback(new Error('Only images are allowed [ png , jpg & jpeg ]'));
25       }
26       callback(null, true);
27     }
28     if(fieldName=="profile"){
29       if(ext !== '.png' && ext !== '.jpg' && ext !== '.jpeg') {
30         return callback(new Error('Only images are allowed for profile'));
31       }
32       callback(null, true);
33     }
34   }
35 }
36
37 var upload = multer(multerOptions).fields([ {name:"profile", maxCount:1},{name:'property' , maxCount:7}]);

```

## 5. Testing Phase :-

### Home Page :-





## Want to Sell Property?

Let us create a tailored strategic marketing plan and keep track of the selling process.

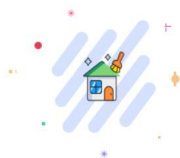
Post Property



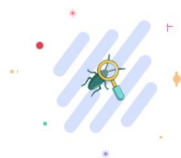
### Property Services Coming-soon



Home Loan



Home Cleaning



Pest Control



Sanitizing

### Download Our App



Property alert



Instant connect




Mark favorite


## Login And Register Page :-

### Login

LOGIN

or login using

 Facebook

 Google+


[Don't have an account?](#) **Sign Up**


### Register

I AM:

REGISTER

or register using

 Facebook

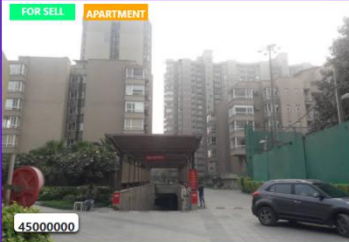
 Google+

[Already have an account?](#) **Login Now**

## Searched Property :-

FOR SELL

APARTMENT



45000000

M2K Victoria Gardens , Model Town , Delhi , Delhi -- 110009

Sq. Ft.


4

3

VIEW DETAIL

FOR SELL

VILLA



150000000

Vasant Kunj P.O , Vasant Kunj , Delhi , Delhi -- 110070

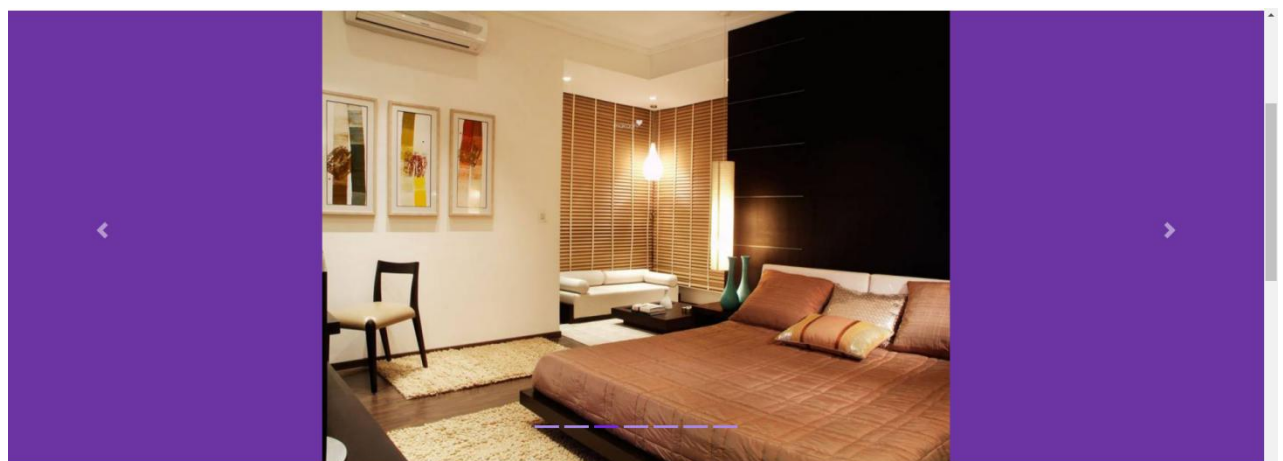
Sq. Ft.

5

4

VIEW DETAIL

## Property Details :-



### 4 BHK Apartment - 3,450 sq ft

M2K Victoria Gardens , Model Town , Delhi , Delhi -- 110009



45000000

Status  
Ready To Move

Bedroom  
4

Bathroom  
3



Rohit Mittal(broker)

Connect with the broker right now

Enter Your Name

Enter Your Email

Enter Your Number

CONNECT

## Description

It has an area of 3450 sqft . The property is available at a price of Rs. 4.50 cr . It is an unfurnished property. It has shared book library. Other amenities include car wash and laundromat facilities, staff quarter, spa/sauna/steam, tennis court, swimming pool, board games zone, yoga and meditation area, music and dance corner and concierge services. This residential property is ready-to-move-in. It is made in way to provide a comfortable living. It is situated in the proximity of all the important facilities.

## Amenities

  
Intercom  
Available

  
Power Backup  
Available

  
Lift  
Available

  
Swimming Pool  
Available

  
Parking  
Available

  
Gym  
Available

## Amenities

  
Intercom  
Available

  
Power Backup  
Available

  
Lift  
Available

  
Swimming Pool  
Available

  
Parking  
Available

  
Gym  
Available

  
Sports Facility  
Available

  
Jogging Track  
Available

  
24X7 Security  
Available

  
Garden  
Available

  
Laundry  
Not Available

  
Basement  
Not Available

## Furnishings

  
Gas Connection  
Not Available

  
Microwave  
Not Available

  
Sofa  
Not Available

  
AC  
Not Available

  
Wardrobe  
Not Available

  
TV  
Not Available

  
Refrigerator  
Not Available

  
Wifi  
Not Available

  
Dining Table  
Not Available

  
Bed  
Not Available

  
Barbeque  
Not Available

  
Water Heater  
Not Available

## Post Property Page :-

### General Property Info

Property Title

Type                      Catagory                      Price

Location

### Description

Description

### Details

Area Size Sq.Ft (Only Digits)                      Carpet Area (Only Digits)

Area Size

Carpet Area

Year Build                      Bedrooms                      Bathrooms

Year Build

### Property Anemities

☐ Gym

☐ Basement

☐ Lift

☐ Parking

☐ Swimming-Pool

☐ Intercom

☐ Laundry

☐ Garden

☐ Sports Facility

☐ Security

☐ Jogging Track

☐ Power Backup

### Property Furnishings

☐ Air Conditioning

☐ Gas Connection

☐ Refrigerator

☐ TV

☐ Microwave

☐ Sofa

☐ Wardrobe

☐ Bed

☐ Dinning Table

☐ Barbeque

☐ WiFi

☐ Water Heater

### Property Images


You can Choose Only 7 Images At Max

No file chosen


Choose Files

SUBMIT PROPERTY

## Dashboard :-







**You Have broker Profile**  
(You Can Upload Property With broker Profile)




Choose File No file chosen

Remove

Rohit Mittal  


7004885175  

### Your Uploaded Properties




Adarsh Palm Meadows

**House for Rent**


 5000 sq. ft.

[VIEW DETAILS](#)




M2K Victoria Gardens

**4 BHK Apartment**


 3,450 sq. ft.

[VIEW DETAILS](#)



Vasant Kury P-02

**5 BHK Villa**

 9000 sq. ft.

[VIEW DETAILS](#)

## **Finishing The Project :-**

### **Facing Problems During Development :**

- **Requirement Gathering Phase** : It was very important step. If the requirements were not good then the project may failed. I had faced lot of problems in gathering data.
- **During Design Phase** : At that time, I was confused about flow of data, but after some research I finally come to final design.
- **Development Phase** : It was a crucial part of the project. At that time I was confused about which technology to use but I managed to do that.
- **Testing Phase** : It was the necessary part of the project. This part helped to test the whole project,at that time I had come across some bugs but I had managed to fix them.

## **Conclusion :-**

The Property Dekho Project proves to be a strong system that has followed all the industry standards. Database is created with good design,the system can comply with any demand of future.