LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Flow of money is essence of economy. Purchasing and selling the assets is one of the common thing humans are doing and always look for gold in this field.

A real time website serves the need of the buyer and seller such that it could connect a potential buyer with a seller.

Property management is the operation, control, and oversight of real estate as used in its most broad terms. Management indicates a need to be cared for, monitored and accountability given for its useful life and condition. This is much akin to the role of management in any business.

Property management is also the management of personal property, equipment, tooling, and physical capital assets that are acquired and used to build, repair, and maintain end item deliverables. Property management involves the processes, systems, and manpower required to manage the life cycle of all acquired property as defined above including acquisition, control, accountability, responsibility, maintenance, utilization, and disposition.

But the major problem with the websites is that there is no guarantee that the property uploaded by the seller is true documents and it is not fraud. Our project mainly focuses on this area and have a document verification phase such that only the verified documents are get the view of the buyers.

1.1 Purpose:

Along with all the general purposes of the real estate management systems, this project focuses on the security of the money, that is tries to make sure that buyers are investing on the property that is not fraud.

1.2 Scope:

In 21st century with the advancement of technology, people are googling for anything and everything. Real Estate Website has got more scope across India and worldwide.

This is because of the changing trends It's a truth that people search for agents through online.

Also people are bored of the brokers who are making money out of nowhere. Everything is digital now. The old banner ads are gone now. Hence the people want online system that eliminates the need for broker and also provide a investing goal such that the hard earned money is not ruined.

Hence our website is a perfect match for the current trends and have lot of scope.

CHAPTER 2

Overall Description

We aim to connect the potential buyers and the sellers across the globe. Sellers upload property details with valid documents. We overcome the problem of fraud sale by having a verification process with the documents. Only verified documents which are according to wish list of the buyer are shown to the buyer. Buyer may choose the property he is interested in and contact the owner. Meanwhile once the buyer is interested on some property the seller of that property is sent a notification.

Hence for such implementation we are using various platforms namely HTML5,CSS,Atom, AngularJS and a few more with support from hardware below are the mentioned software as well as hardware requirement and where are they used.

2.1 Specific Requirements

2.1.1 Software Requirement

Front-end	:	Html5,Css3,Bootstrap,Angularjs,Jquery.
IDE used	:	Atom, Sublime.
Tools	:	Postman, Chrome dev tools.
Back-end	:	Nodejs.
Packages	:	mysql,express-fileupload, multer.
Web-server	:	Http server in Nodejs.
Database	:	Mysql.
Web-Browser	:	Google-Chrome,Mozilla- Firefox,Safari.

2.1.2 Hardware Requirement

Processor	:	2 CPU Cores or higher(2x1.6GHz or more).
RAM	:	2GB or more
Hard Disk	:	10GB.

2.2.3 Functionality

Both functional and nonfunctional aspects are important in order to have a successful project which performs according to what is planned considering user experience and business logic.

2.2.3.1 FUNCTIONAL REQUIREMENTS:

- Creating valid user-names for registration by admin
- Registration of the Users by using valid username and providing their details.
  - Updating of the wishlist.
  - Adding the properties for sale.
  - Verification of the properties by the employees. Namely khata certificate, tax receit and encumberation certificates has to be uploaded by the sellers and same will be verified by each of the expert employees.
  - View the relevant properties according to the wishlist.
  - Commit the interest in some of the properties.
  - Getting notifications about the verified property and people who are interested to buy the property of the seller.

2.2.3.2 NON-FUNCTIONAL REQUIREMENTS:

- User Friendly: A good user interface which needs no explanation and easy to use.
- Security.
- Prevention of unauthorized access(like users logging in as employee).
- Form validations and force the user to create a strong password.
- Limiting the number of ports to be accessed.
- Backup: Retrieve and recovery of data is possible in cases of damage ,corrupt or loss of data
- Performance: Optimizing the Storage data and providing higher hardware resources to the server.

CHAPTER 3

Detailed Design

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one.

Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

3.1 DFD Level 0

DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analysed or modelled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

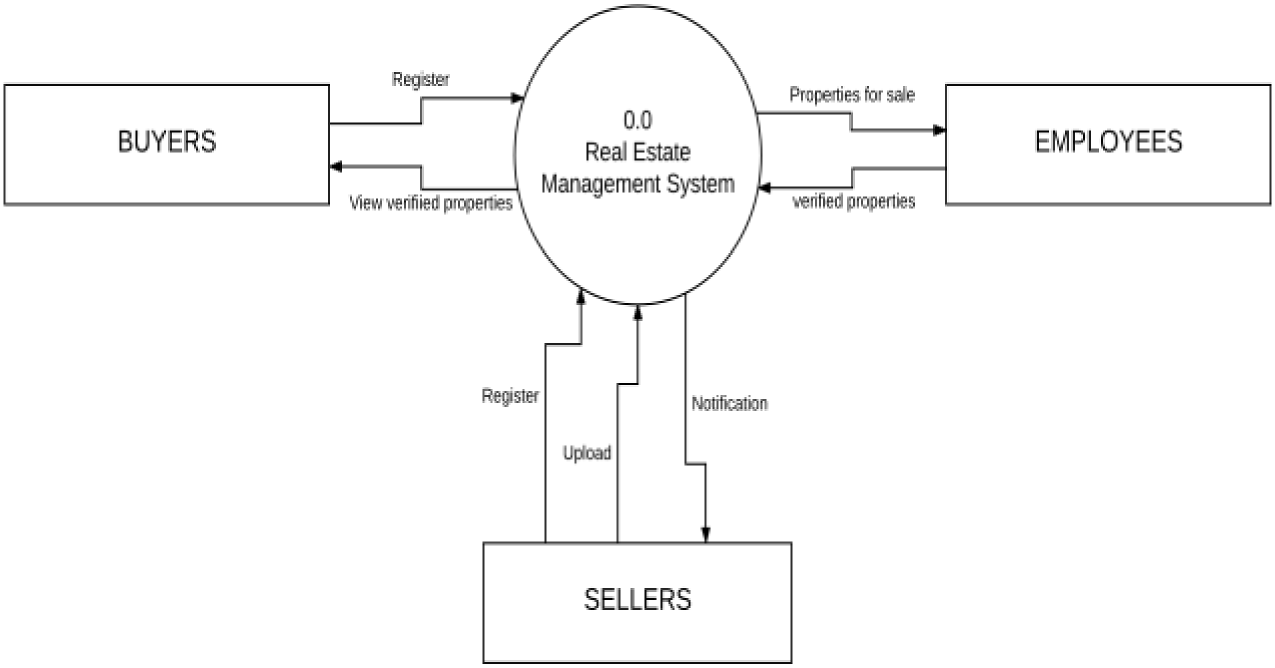


Figure 3.1 DFD level 0

Buyers and Sellers connect themselves to our system by registering with an email id and mobile number. Buyers will have a wish list of what type of property they are looking for, according to that they will get see the properties which are relevant and are verified. Once the seller uploads the property, it will be sent for verification, which will be done by employees. Notification of the verified properties and also the people who are interested in the property are notified to the user.

3.2 DFD Level 1

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

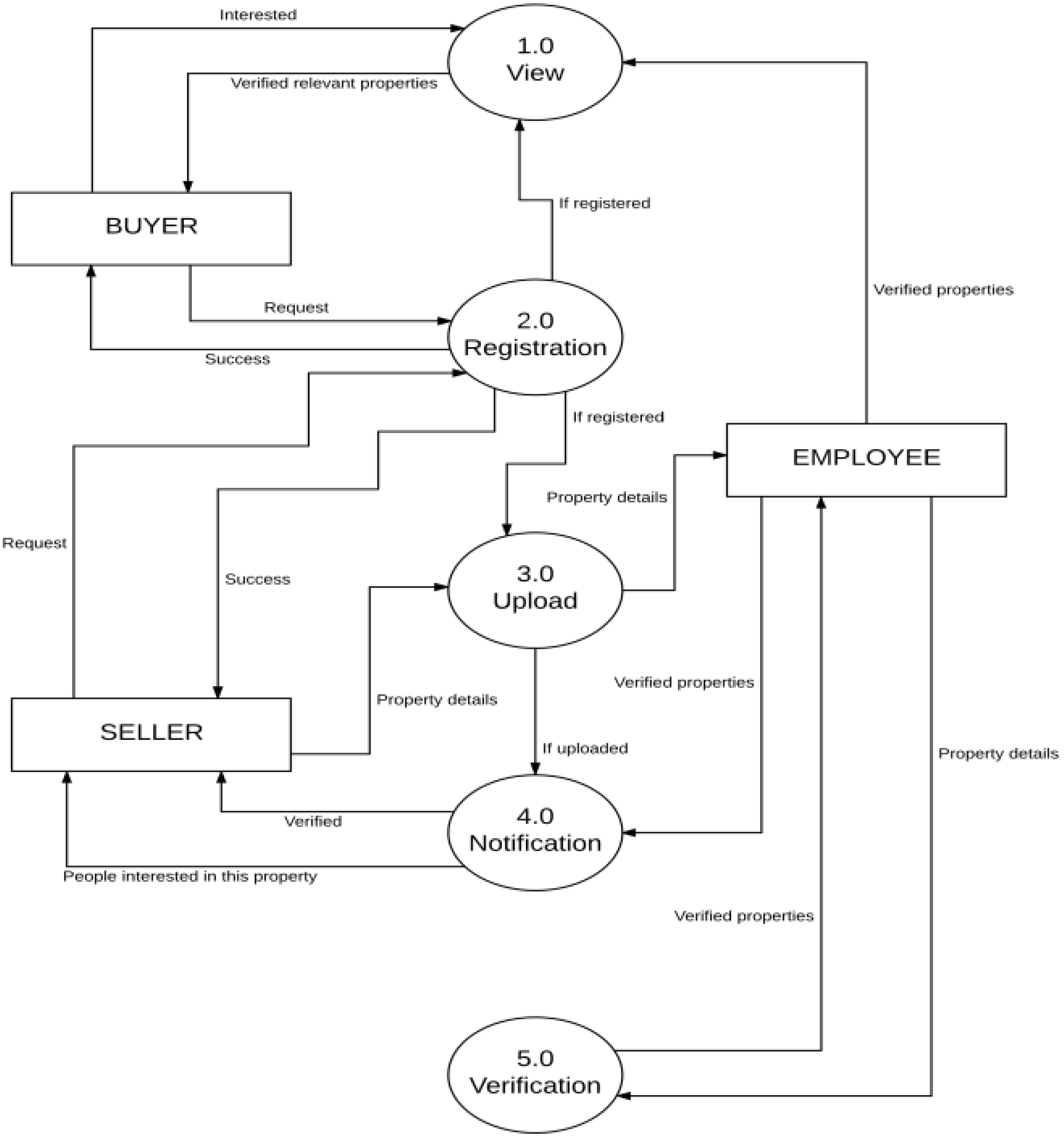


Figure 3.2 DFD level 1

Mainly there are five functionalities of the system namely registration, upload, verification, view and notification. Once a property is on for upload its documents should be verified by the employees. Only the verified properties can be viewed by the buyers.

3.3 DFD Level 2

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.

3.3.1 VIEW

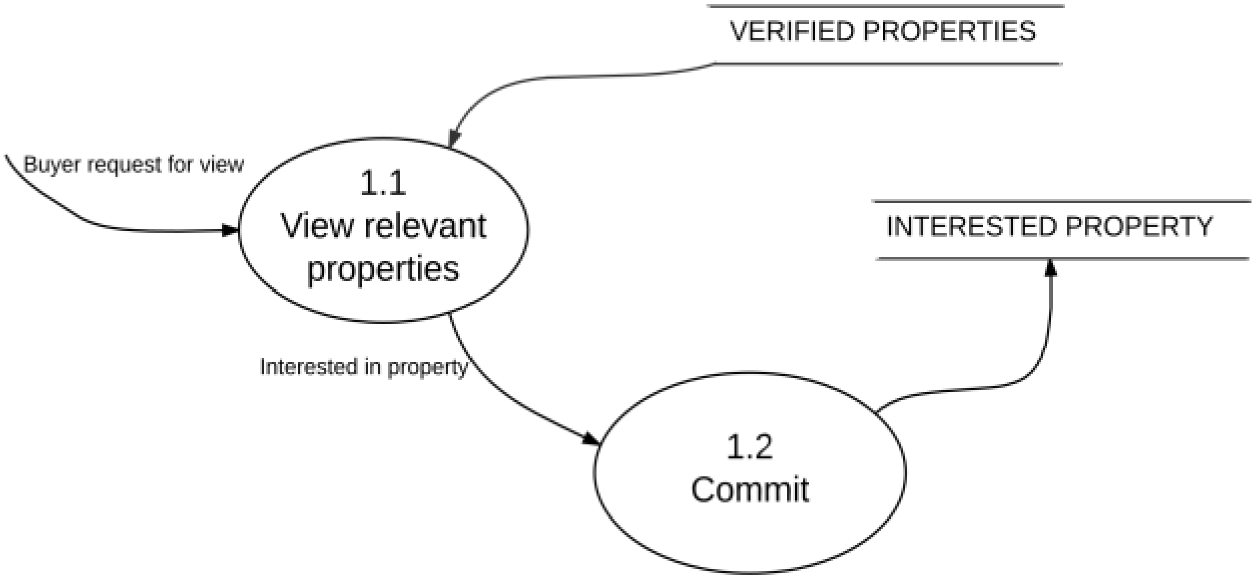


Figure 3.3 DFD level 2 view

When a user requests for a property view he will be shown relevant properties which are verified according to his wish list. On seeing the property he can commit for property that he is interested in that property. It will be recorded in the interested table and accordingly notification will be sent to seller of that property.

3.3.2 REGISTRATION FORM

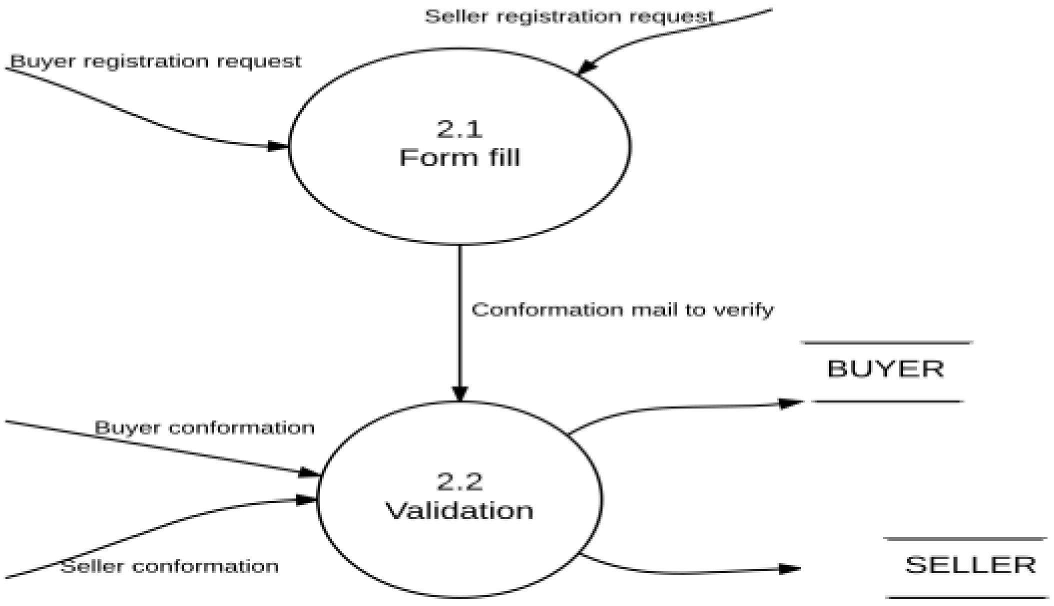

Seller registration request
Buyer registration request
2.1
Form fill
Conformation mail to verify
BUYER
Buyer conformation
2.2
Validation
Seller conformation
SELLER

Figure 3.4 DFD level 2 registration

Signup form is for the registration of the user with the system. He needs to fill out a form for registration, it will be validated and on validation they can access to the system by logging into the system.

3.3.3 UPLOAD

Seller will upload the property details and the required documents of the property for the verification. Also he will specify the amount for the sale. It will be updated in the property table.

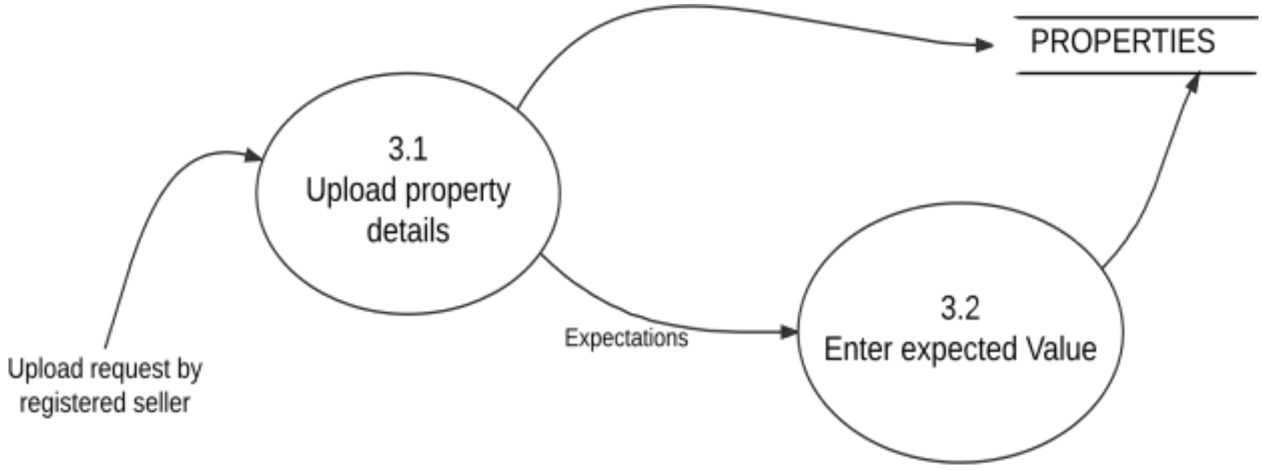


Figure 3.5 DFD level 2 upload

3.3.4 VERIFICATION

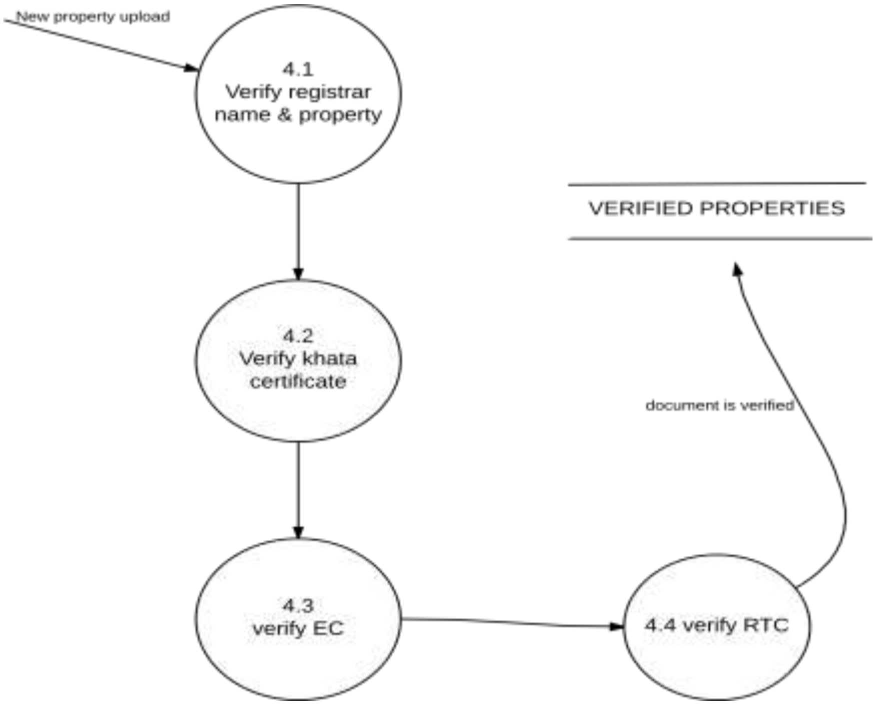


Figure 3.6 DFD level 2 verification

Verification is done in three stages by three of the employees, mainly khata certificate, tax certificate and encumberation certificate will be verified.

3.3.5 NOTIFICATION

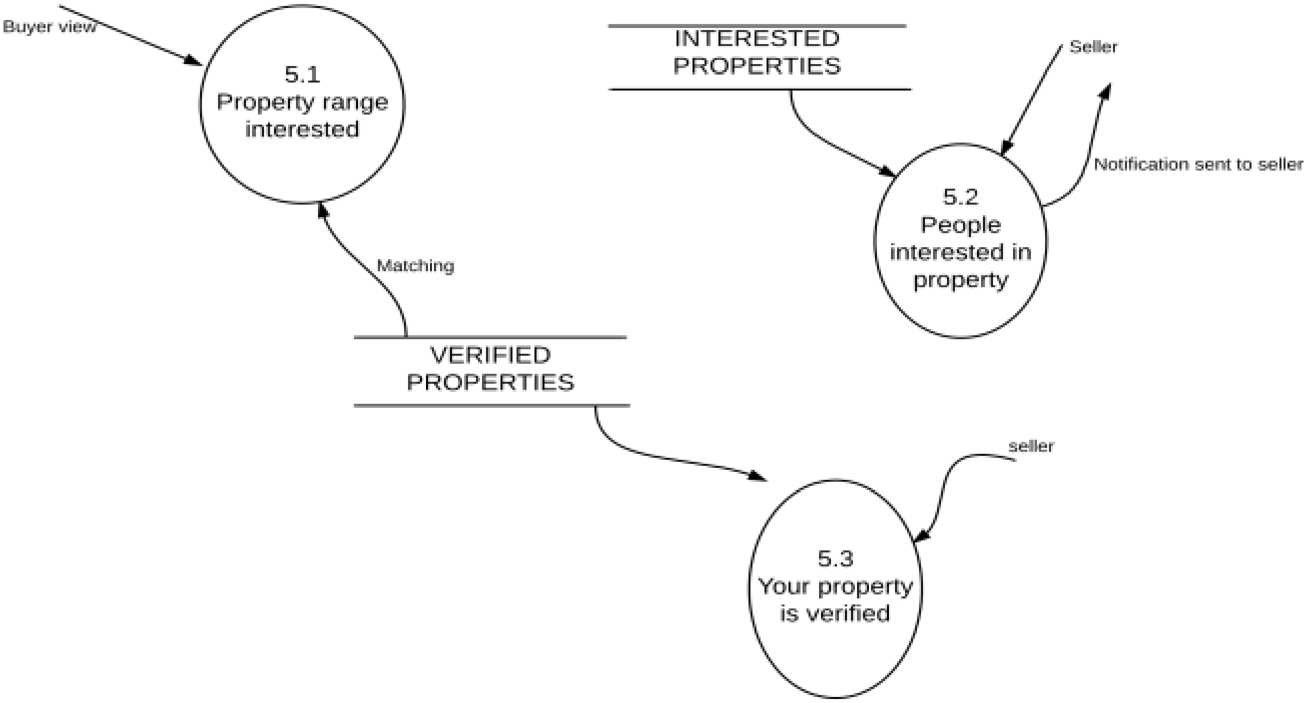

Buyer view
5.1
Property range interested
INTERESTED PROPERTIES
Seller
Notification sent to seller
5.2
People interested in property
Matching
VERIFIED PROPERTIES
seller
5.3
Your property is verified

Figure 3.7 DFD level 2 notification

Notification will be sent to the users if property uploaded by him is verified or if anybody shows interest in the property.

CHAPTER 4

ER Diagram

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

1. Entities
2. Relationships
3. Attributes

Steps involved in creating an ERD include:

1. Identifying and defining the entities
2. Determining all interactions between the entities
3. Analysing the nature of interactions/determining the cardinality of the relationships
4. Creating the ERD

An entity-relationship diagram (ERD) is crucial to creating a good database design. It is used as a high-level logical data model, which is useful in developing a conceptual design for databases.

An entity is a real-world item or concept that exists on its own. Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity.

An attribute of an entity is a particular property that describes the entity. A relationship is the association that describes the interaction between entities. Cardinality, in the context of ERD, is the number of instances of one entity that can, or must, be associated with each instance of another entity. In general, there may be one-to-one, one-to-many, or many-to-many relationships.

ER DIAGRAM: REAL ESTATE MANAGEMENT SYSTEM

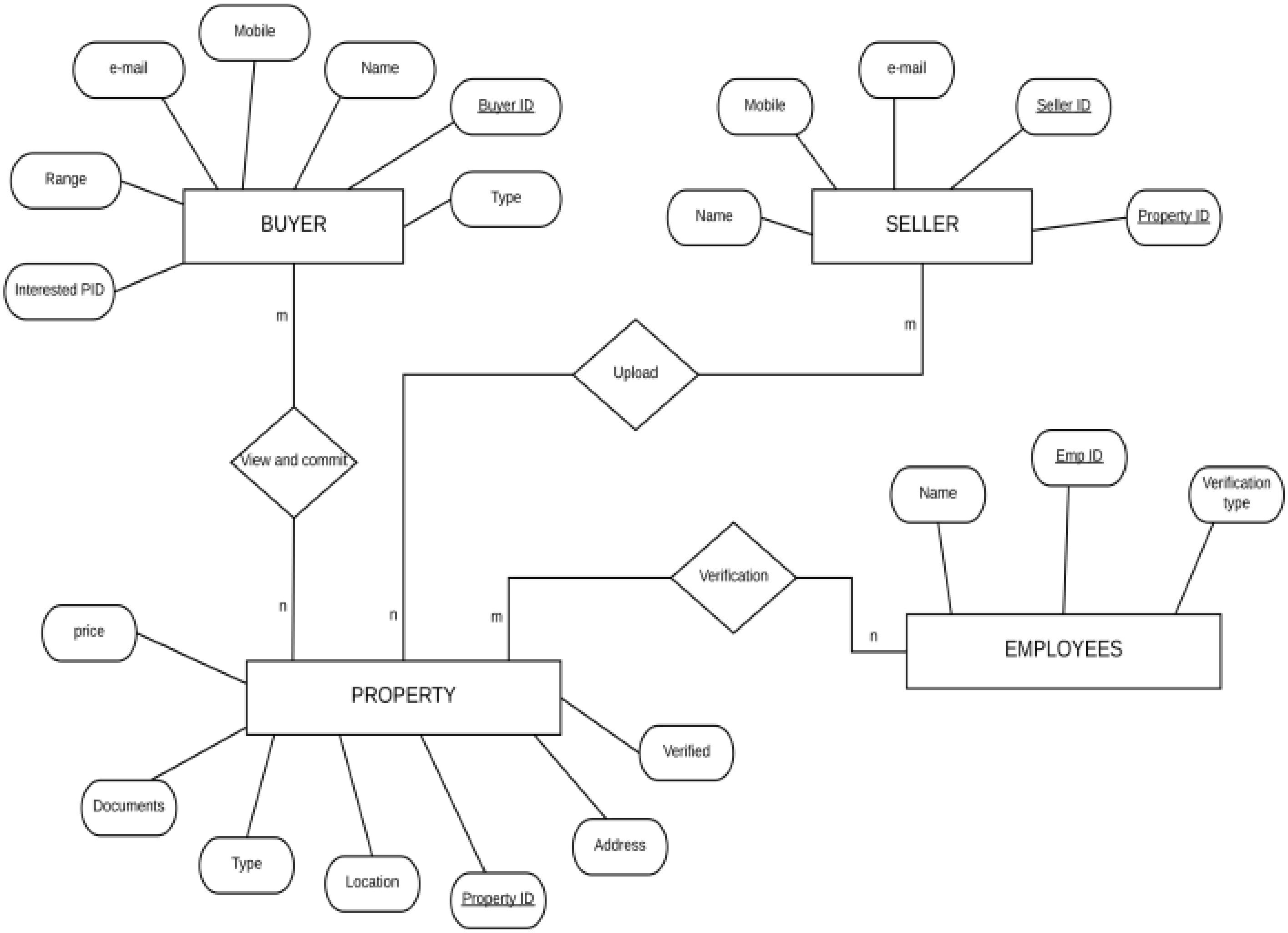


Figure 4.1 ER diagram of real estate management system

CHAPTER 5

Relational Schema and Normalization

5.1. 1st Normal Form (1NF)

First normal form (1NF) is a property of a relation in a relational database. A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain. The first definition of the term defined a relation to be in first normal form when none of its domains have any sets as elements.

First normal form is an essential property of a relation in a relational database. Database normalization is the process of representing a database in terms of relations in standard normal forms, where first normal is a minimal requirement.

First normal form enforces these criteria:

1. Eliminate repeating groups in individual tables.
2. Create a separate table for each set of related data.
3. Identify each set of related data with a primary key

5.2 2nd Normal Form (2NF)

Second normal form (2NF) is a normal form used in database normalization. A relation that is in first normal form (1NF) must meet additional criteria if it is to qualify for second normal form. Specifically: a relation is in 2NF if it is in 1NF and no non-prime attribute is dependent on any proper subset of any candidate key of the relation. A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation.

Put simply, a relation is in 2NF if it is in 1NF and every non-prime attribute of the relation is dependent on the whole of every candidate key.

5.3 3rd Normal Form (3NF)

The third normal form (3NF) is a normal form used in database normalization.Definition states that a table is in 3NF if and only if both of the following conditions hold:

The relation R (table) is in second normal form (2NF)

Every non-prime attribute of R is non-transitively dependent on every key of R.

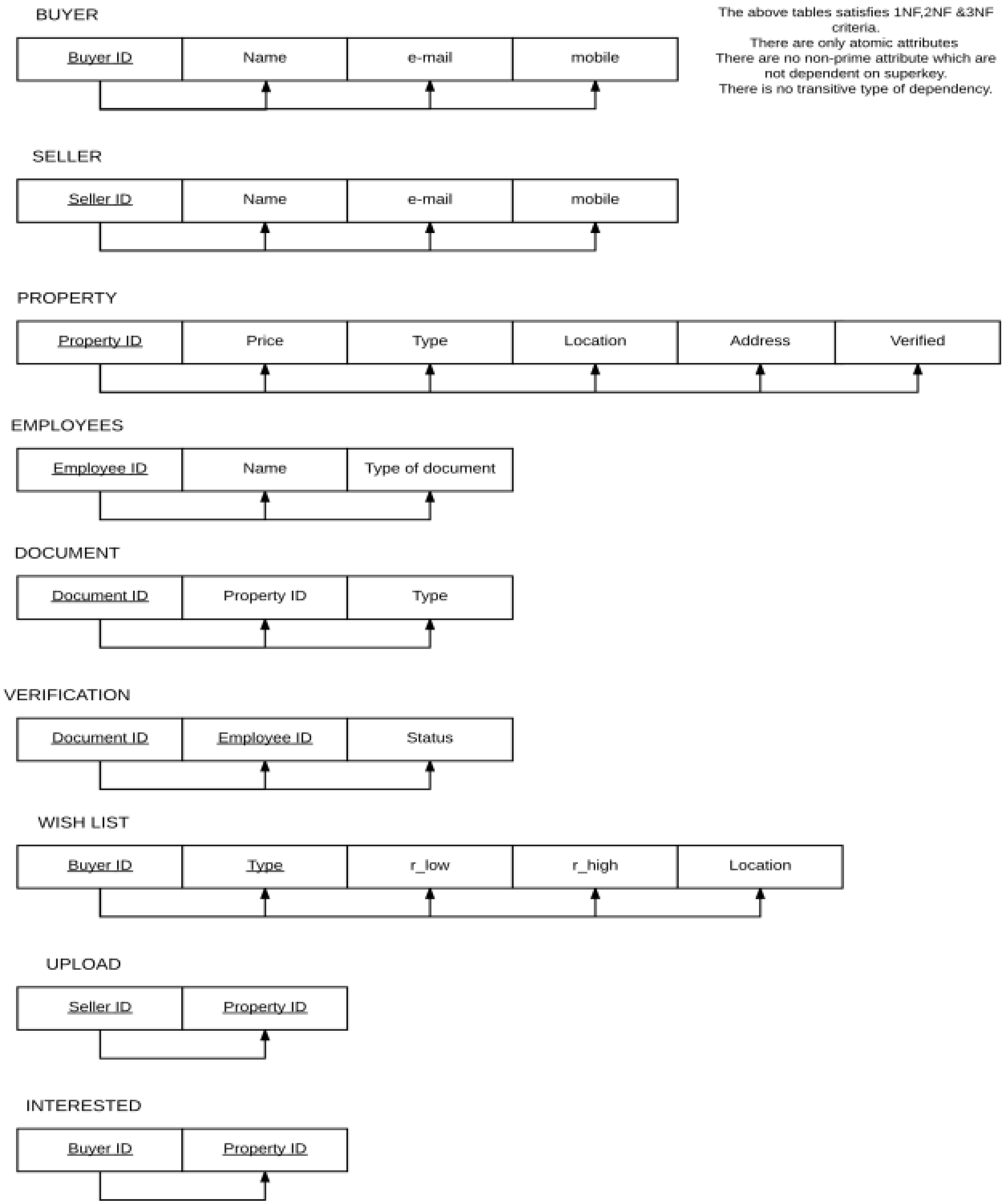
A non-prime attribute of R is an attribute that does not belong to any candidate key of R. A transitive dependency is a functional dependency in which X → Z (X determines Z) indirectly, by virtue of X → Y and Y → Z (where it is not the case that Y → X).

A 3NF definition that is equivalent to Codd's, but expressed differently, was given by Carlo Zaniolo in 1982. This definition states that a table is in 3NF if and only if, for each of its functional dependencies X → A, at least one of the following conditions holds:

X contains A (that is, X → A is trivial functional dependency), or

X is a superkey, or

Every element of A-X, the set difference between A and X, is a prime attribute (i.e., each attribute in A-X is contained in some candidate key).



The above tables satisfies 1NF,2NF &3NF criteria.
There are only atomic attributes
There are no non-prime attribute which are not dependent on superkey.
There is no transitive type of dependency.

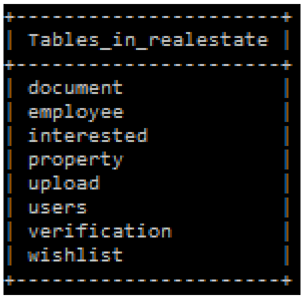
Figure5.1 3rd Normal Form

CHAPTER 6

REPORT

Name of the database**: realestate**

Tables in the realestate

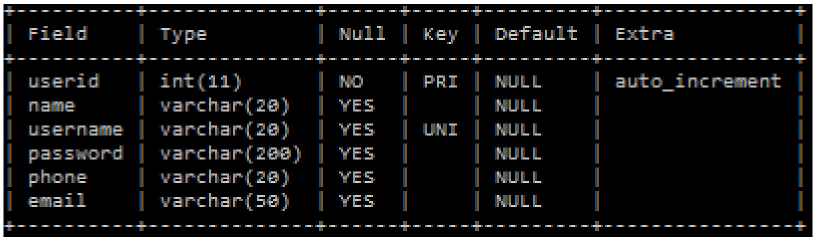


```
+----------------------+
| Tables_in_realestate |
+----------------------+
| document             |
| employee             |
| interested           |
| property             |
| upload               |
| users                |
| verification         |
| wishlist             |
+----------------------+
```

Table 5.1 list of tables in database realestate

There are 8 tables in the database realestate.

6.1 Users table

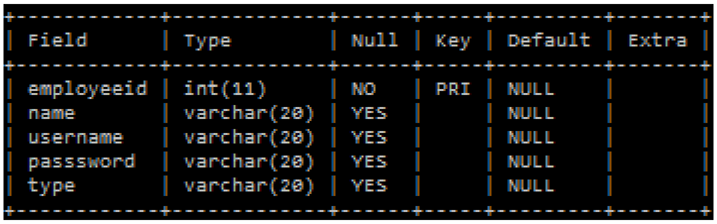


```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| userid   | int(11)      | NO   | PRI | NULL    | auto_increment |
| name     | varchar(20)  | YES  |     | NULL    |                |
| username | varchar(20)  | YES  | UNI | NULL    |                |
| password | varchar(200) | YES  |     | NULL    |                |
| phone    | varchar(20)  | YES  |     | NULL    |                |
| email    | varchar(50)  | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

Table 5.2 defining users

User table is to store the information about the users of the system, each user will have a userid which is a primary key attribute.

6.2 Employee table

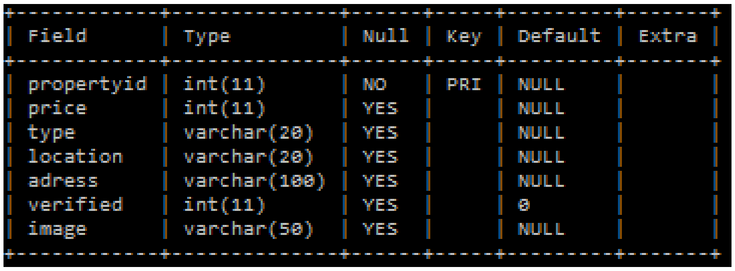


```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| employeeid | int(11)     | NO   | PRI | NULL    |       |
| name       | varchar(20) | YES  |     | NULL    |       |
| username   | varchar(20) | YES  |     | NULL    |       |
| passsword  | varchar(20) | YES  |     | NULL    |       |
| type       | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Table 5.5 defining employee

Employee table have the details about the employees, each employee will handle the verification of certain type of document. Each employee is uniquely identified by his employeeid which is primary key.

6.3 Property table

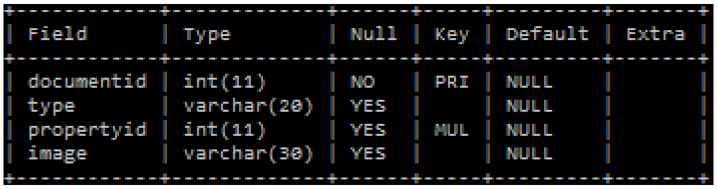


```
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| propertyid | int(11)      | NO   | PRI | NULL    |       |
| price      | int(11)      | YES  |     | NULL    |       |
| type       | varchar(20)  | YES  |     | NULL    |       |
| location   | varchar(20)  | YES  |     | NULL    |       |
| adress     | varchar(100) | YES  |     | NULL    |       |
| verified   | int(11)      | YES  |     | 0       |       |
| image      | varchar(50)  | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

Table 5.3 defining property

Each uploaded property will have a propertyid which is primary key. For each property, associated information is stored such as price of the property, type of the property and other details are specified.
If verified column for a property is 1 then it is verified otherwise its not. If all the documents related to a particular property is verified then verified column is set to 1.

6.4 Document table

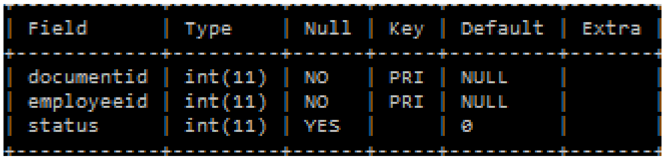


```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| documentid | int(11)     | NO   | PRI | NULL    |       |
| type       | varchar(20) | YES  |     | NULL    |       |
| propertyid | int(11)     | YES  | MUL | NULL    |       |
| image      | varchar(30) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Table 5.4 defining document

Each of the document will be identified by its documentid and will be associated with a propertyid. Each of the document has a type and hence will be verified accordingly.

6.5 Verification table

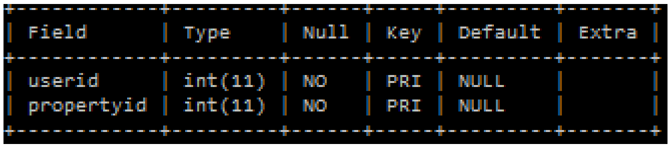


```
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| documentid | int(11) | NO   | PRI | NULL    |       |
| employeeid | int(11) | NO   | PRI | NULL    |       |
| status     | int(11) | YES  |     | 0       |       |
+------------+---------+------+-----+---------+-------+
```

Table 5.8 defining verification

Here each document is mapped into the employeeid, who is going to verify a particular property. Initially status is set to 0, implies that property is not verified, once the employee verifies the property status flag is set to 1.

6.6 Upload table

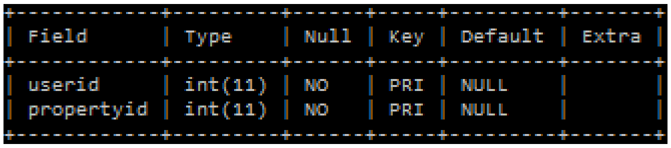


```
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| userid     | int(11) | NO   | PRI | NULL    |       |
| propertyid | int(11) | NO   | PRI | NULL    |       |
+------------+---------+------+-----+---------+-------+
```

Table 5.6 defining upload

If a seller uploads a property his userid and propertyid together will be stored in upload table.

6.7 Interested table

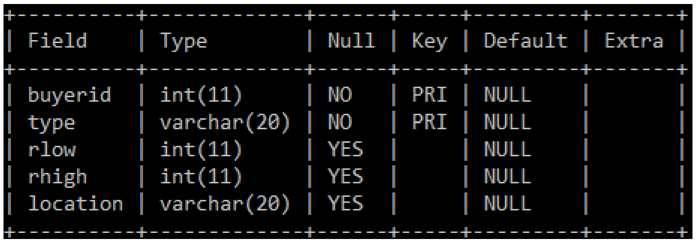


```
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| userid     | int(11) | NO   | PRI | NULL    |       |
| propertyid | int(11) | NO   | PRI | NULL    |       |
+------------+---------+------+-----+---------+-------+
```

Table 5.7 defining interested

If a buyer commits his interest on a property his userid and propertyid together will be stored in interested table.

6.8 Wish list table



```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| buyerid  | int(11)     | NO   | PRI | NULL    |       |
| type     | varchar(20) | NO   | PRI | NULL    |       |
| rlow     | int(11)     | YES  |     | NULL    |       |
| rhigh    | int(11)     | YES  |     | NULL    |       |
| location | varchar(20) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

Table 5.9 defining wishlist

When buyer makes a wish list for a property, that is type of property, range of price that he is looking for and location of the property, if will be recorded in wishlist table. According to the wishlist table verified properties are shown to the buyer.

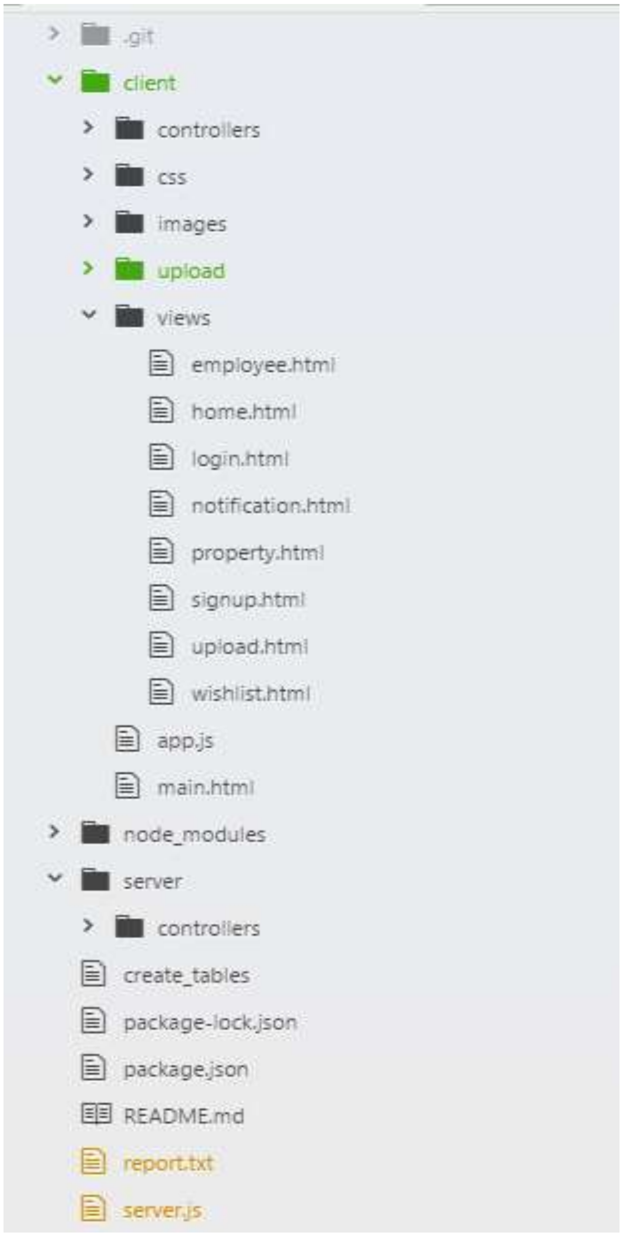
6.9 Organization of the project

Nodejs is used for the backend and front end is done using html, css and angular js.

As it is single page application, main.html is the main page of the application where all the controllers and other pages are loaded.

Server.js is the main backend file to run the server. It requested for node modules and other controllers and handles the backend apis.

All the uploaded documents will be stored in the upload directory.



```
> .git
v client
    > controllers
    > css
    > images
    > upload
    v views
        employee.html
        home.html
        login.html
        notification.html
        property.html
        signup.html
        upload.html
        wishlist.html
    app.js
    main.html
> node_modules
v server
    > controllers
  create_tables
  package-lock.json
  package.json
  README.md
  report.txt
  server.js
```

Figure 6.1 Organization of the project

6. 10 CODE SNIPPETS

1. server.js

```
//Loading packages for the server
var mysql                   =  require('mysql');
var url                     =  require('url');
var express                 = require('express');
var bodyParser              = require('body-parser');
var morgan                  = require('morgan');
var loginController     r          = require('./server/controllers/loginController');
var homeController     = require('./server/controllers/homeController');
var employeeController          = require('./server/controllers/employeeController');
var fs                                  = require('fs');
const fileUpload = require('express-fileupload');

//Connecting with the database
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : '@Shar1234',
  database : 'realestate'
});

var app = express();

var router = express.Router();


var server=app.listen(3000,function(){
  console.log("Real estate app is online at port no 3000");
})

//Module for the file upload
app.use(fileUpload());

app.use(bodyParser.urlencoded({
  extended: true
}));


app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(morgan('dev'));
app.use('/', express.static(__dirname + '/client/'));

//End  points of the system

app.post('/login', loginController.login);
app.post('/signup', loginController.signup);

app.post('/emplogin',employeeController.emplogin);
app.post('/verify',employeeController.verify);
app.get('/getverify',employeeController.getverify);

app.post('/addwishlist', homeController.addwishlist);
app.get('/getnotifications',homeController.getnotifications);
app.get('/getproperty',homeController.getproperty);
app.post('/interested',homeController.interested);

app.post('/uploadproperty', homeController.uploadproperty);


app.get('/', function(req, res) {
  res.sendFile(__dirname + '/client/main.html');
});

/*
Server js is the main backend file where all the backend processes are linked and controlled.
Home controller
Login controller
And Employee controllers are controlled by server.js where the actual database operations are performed

*/
```

2. app.js

```
//app.js is the main client side file it defines the angular module and also controls routes
var app = angular.module("realApp", ['ngResource', 'ngRoute']);


//In the configuration all the routes are handled to a controller
app.config(function($routeProvider) {
  $routeProvider
  .when('/login', {
    templateUrl: '/views/login.html',
    controller: 'loginController'
  })
  .when('/signup', {
    templateUrl: '/views/signup.html',
    controller: 'loginController'
  })
  .when('/home',{
    templateUrl: '/views/home.html',
    controller: 'homeController'

  })
  .when('/employee', {
    templateUrl: '/views/employee.html',
    controller: 'employeeController'
  })
  .when('/property', {
    templateUrl: '/views/property.html',
    controller: 'homeController'
  })
  .when('/upload', {
    templateUrl: '/views/upload.html',
    controller: 'homeController'
  })
  .when('/notification', {
    templateUrl: '/views/notification.html',
    controller: 'homeController'
  })
  .when('/wishlist', {
    templateUrl: '/views/wishlist.html',
    controller: 'homeController'
  })

  .otherwise({
    redirectTo: '/login'
  })
})

//Main controller is for the whole page , as it is a single page application
app.controller('mainController', function($scope) {
  $scope.main = "Real Estate Management";
})
```

CHAPTER 5

Conclusion and Future Enhancement

7.1 Summary

Based on the goal that we set at the beginning, we analyzed the existing work flow and issues in the current real estate management system and then proposed a new system in order to avoid the brokers and also limiting the fraud sale in the industry. The analysis to the new system is necessary in order to successfully implement it in the future. We analyzed the functional requirements, data requirements and non-functional requirements. The analysis was a foundation of the new system design. The conceptual design focused on analyzing and designing the basic elements of a database system according to the requirement analysis of the new system, we created an E-R diagram, relational schema, data dictionaries, DFDs, routine queries and report types. Following the design a set of tables with values were created in database. The success of the project is based on the users who participate actively.

7.2 Limitations

Main limitation of our project is it not a platform for sale, it is just a platform for connecting the potential buyer with the seller. Also employees at the backend must be highly qualified for the document verification , if they miss to find out false documents then it is the buyers who will suffer.

It is also important to send notifications through mobile message or an email which is more appropriate.

For the people who are not sure about the property they want to buy, need of brokers then this is not the solution for them. These are some of the limitations which we can overcome in the future.

7.3 Future Enhancement

7.3.1 Real time sale on the platform

Now in our system we are just connecting the buyers and sellers, but it future it may be not only connecting them but also they can sell or buy on our platform. Without the broker our platform will act as a middleman for their secured transactions and they can buy as and when they are interested in the product and negotiate with the sellers.

Also we will improve the user experience and robustness of our system.

REFERENCES

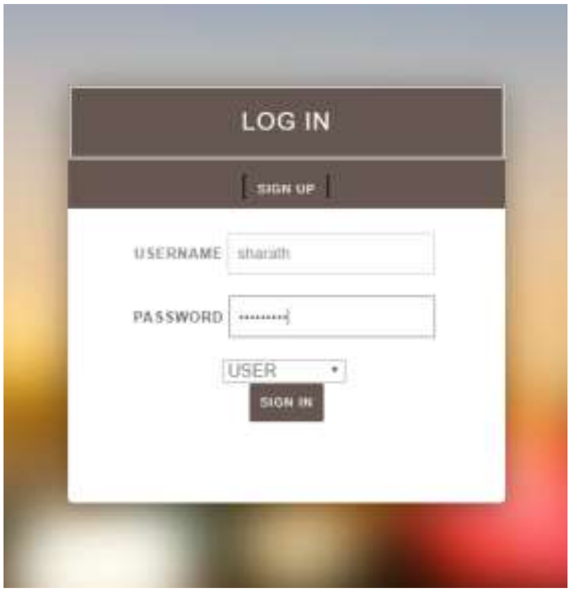
[1] Raghu Ramakrishnan, Database Management System, McGraw Hill Higher Education, 3rd Edition, 2002, ISBN-13 9780071231510

[2] Shamkant B. Navathe, Ramez Elmasri, Fundamentals of Database Systems, Pearson Education, 5th Editon, 2011, ISBN-13 9788131758984

[3] Complete Reference to Nodejs , “Node.js in Action”. Cantelon. Published: 2013.

SNAPSHOTS

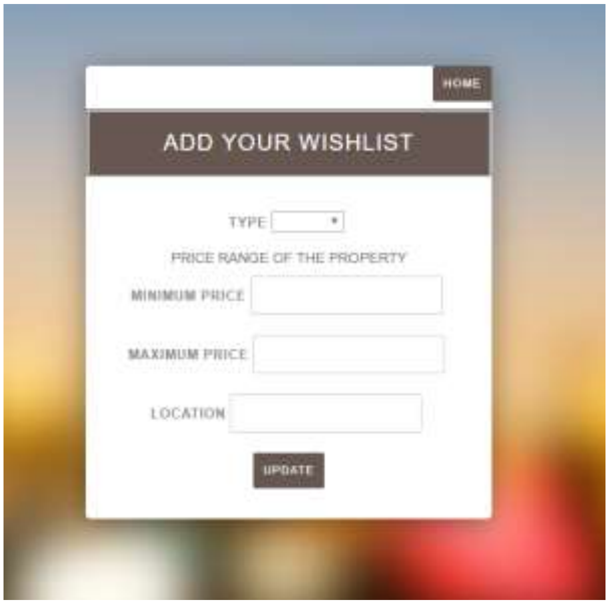
- **Form for login wish list**



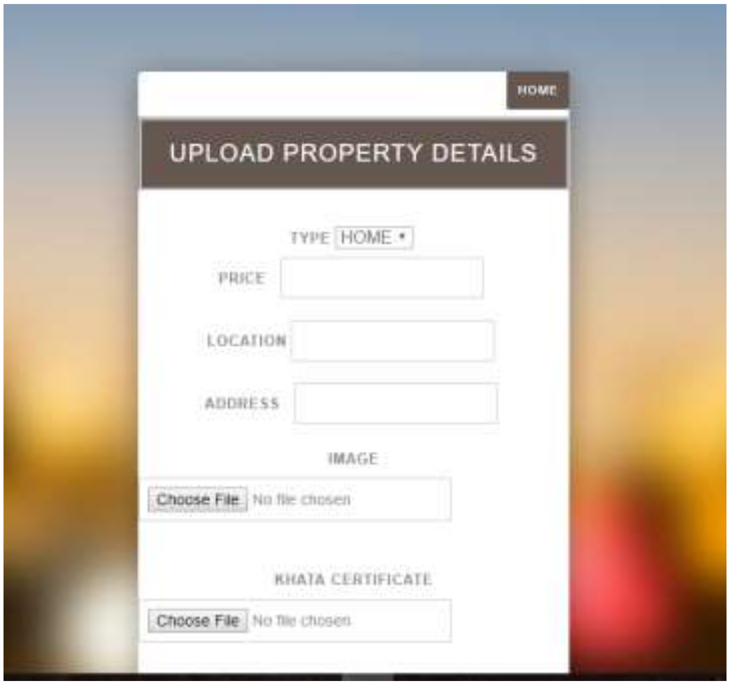
USERNAME sharath

USER

- **Form for adding wish list**



- **Form for uploading the property details**



UPLOAD PROPERTY DETAILS

TYPE HOME ▾

PRICE

LOCATION

ADDRESS

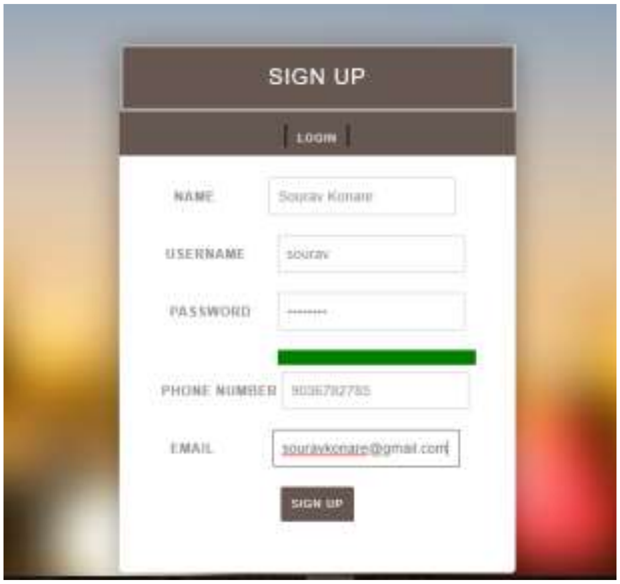
IMAGE

No file chosen

KHATA CERTIFICATE

No file chosen

- **Form for Signup with all the validations.**



SIGN UP

NAME Sourav Konare

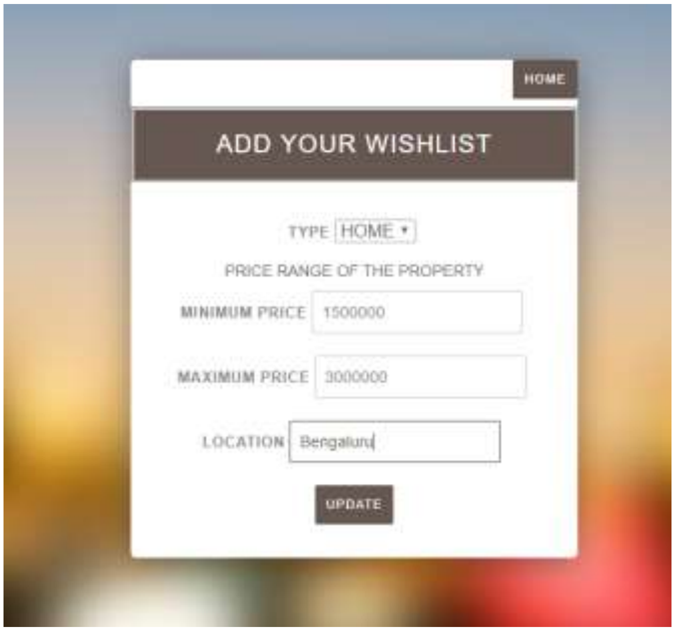
USERNAME sourav

PASSWORD *********

PHONE NUMBER 9036782785

EMAIL souravkonare@gmail.com

- **Adding details in the wishlist**


HOME

ADD YOUR WISHLIST

TYPE HOME ▾

PRICE RANGE OF THE PROPERTY

MINIMUM PRICE 1500000

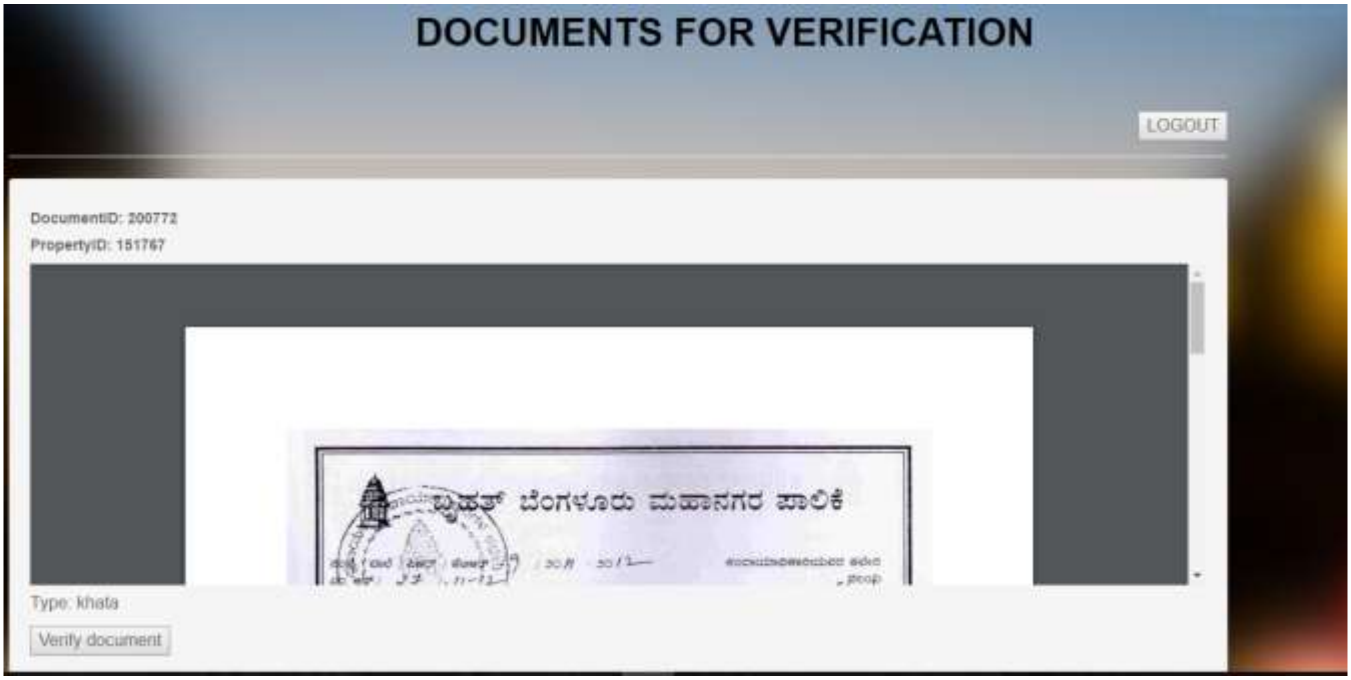
MAXIMUM PRICE 3000000

LOCATION Bengaluru

UPDATE

- **Document Verification by the employee**

-


DOCUMENTS FOR VERIFICATION

LOGOUT

DocumentID: 200772

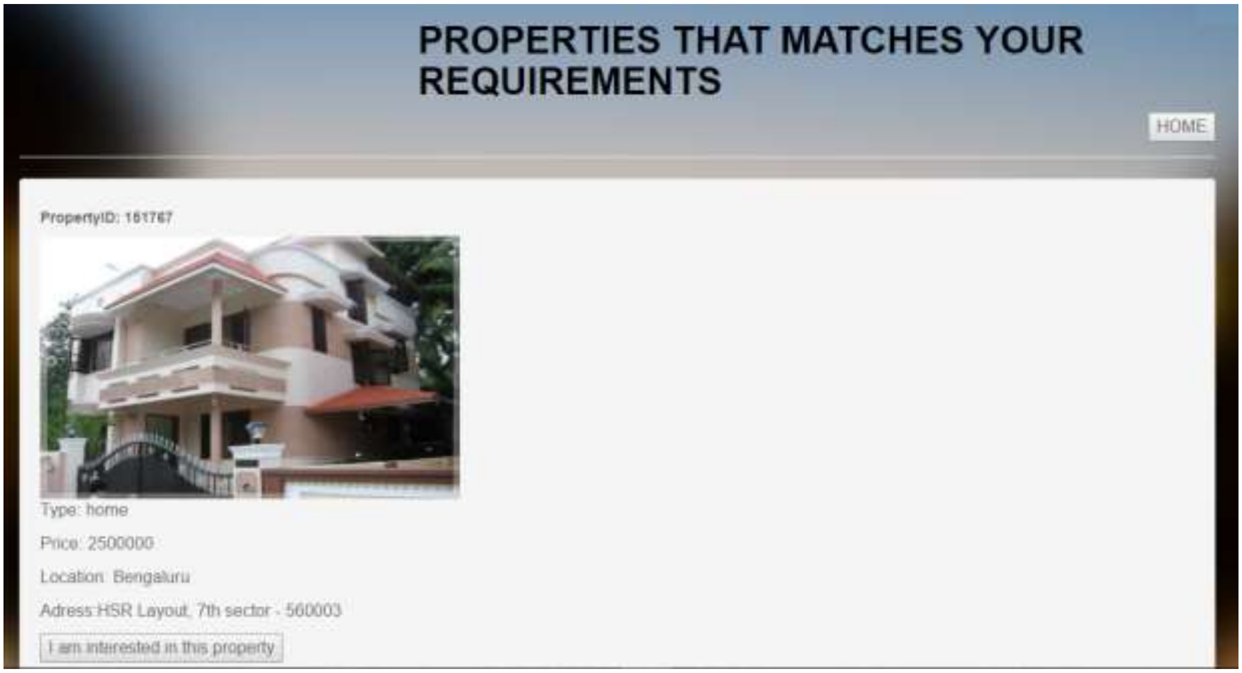
PropertyID: 151767

ಬೃಹತ್ ಬೆಂಗಳೂರು ಮಹಾನಗರ ಪಾಲಿಕೆ

Type: khata

Verify document

- **Displaying the properties that are relevant**



PROPERTIES THAT MATCHES YOUR REQUIREMENTS

PropertyID: 161767

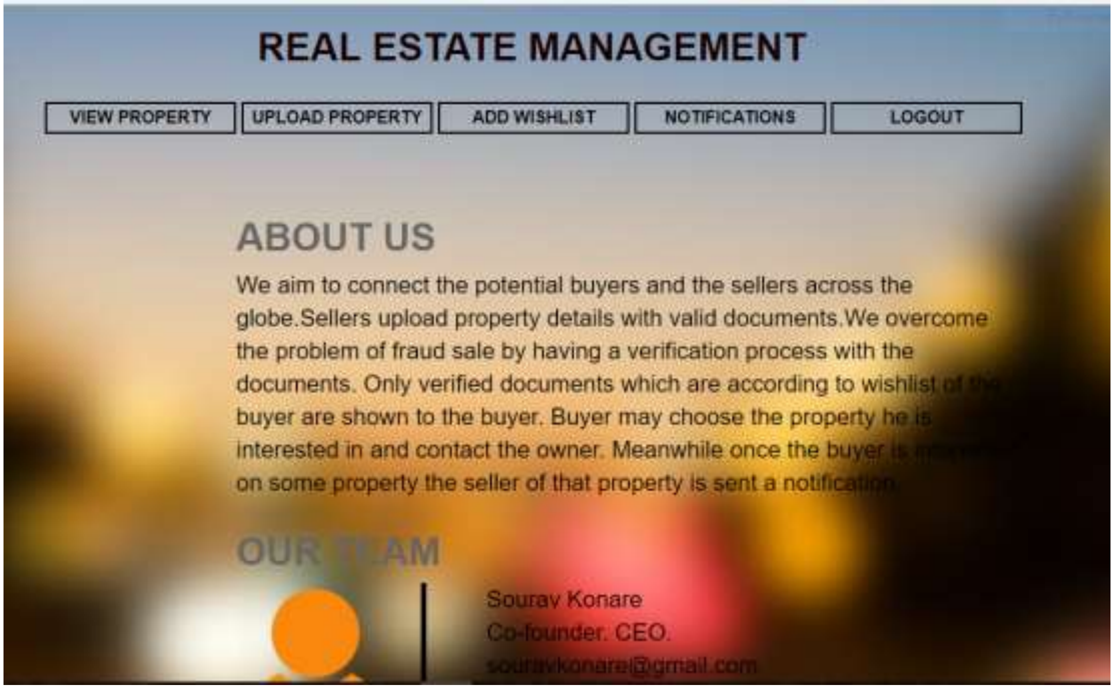
Type: home

Price: 2500000

Location: Bengaluru

Adress:HSR Layout, 7th sector - 560003

- **Home page after the login**



REAL ESTATE MANAGEMENT

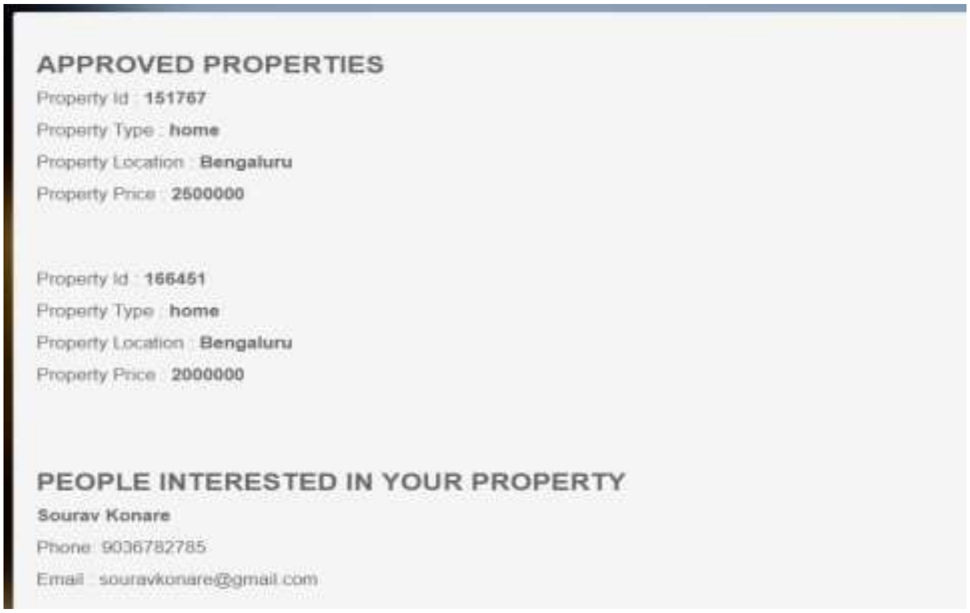
ABOUT US

We aim to connect the potential buyers and the sellers across the globe.Sellers upload property details with valid documents.We overcome the problem of fraud sale by having a verification process with the documents. Only verified documents which are according to wishlist of the buyer are shown to the buyer. Buyer may choose the property he is interested in and contact the owner. Meanwhile once the buyer is interested on some property the seller of that property is sent a notification.

OUR TEAM

Sourav Konare
Co-founder. CEO.
souravkonare@gmail.com

- **Notifications**



APPROVED PROPERTIES

Property Id : **151767**

Property Type : **home**

Property Location : **Bengaluru**

Property Price : **2500000**

Property Id : **166451**

Property Type : **home**

Property Location : **Bengaluru**

Property Price : **2000000**

PEOPLE INTERESTED IN YOUR PROPERTY

Sourav Konare

Phone: 9036782785

Email : souravkonare@gmail.com