****

**The Ghar Real Estate**

A Major Project Report Submitted In partial fulfillment For the award of the Degree of **BACHELOR OF TECHNOLOGY** In the Department of Computer science and Engineering

**Submitted by**:

Harshal Paliwal

Jainil Jain

Kailash Joshi

Varun Sharma

**Department of Computer Science and Engineering**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY**

RAJASTHAN TECHNICAL UNIVERSITY

**May 2025**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY**

****

# **CERTIFICATE**

This is to certify that this project report “The Ghar Real Estate ” is the confide work of “Harshal Paliwal”, “Jainil Jain”, “Kailash Joshi” and “Varun Sharma” who have carried out the project work under my supervision. I approve this project for submission of the Bachelor of Technology in the Department of Computer Science and Engineering, Techno India NJR Institute of Technology, affiliated to Rajasthan Technical University, Kota.

Mr. Aaditya Maheshwari

Project In-charge

Department of Computer Science

# **ABSTRACT**

**1. Purpose**

**1.1. Introduction**

The Ghar Real Estate project is a MERN-stack web application that allows users to search, view, and explore property listings. The backend uses Express.js and MongoDB to handle data storage and API routes, while the frontend is built with React for a clean, responsive interface. Users can search properties by keyword, view detailed listings, and navigate between sections like Home, About, and Contact. The app is deployed on Render, demonstrating a full-stack deployment pipeline. It provides a simple yet functional property search experience,.

**1.2. Scope**

The Ghar platform enables users to:  
● Search and view real estate property listings across various locations.  
● Access detailed property info including images, prices, and descriptions.  
● Use a keyword-based search bar for quick filtering of properties.  
● Navigate between Home, About, and Contact pages via a responsive UI.  
● Enjoy a clean, mobile-friendly layout built with React.  
● Fetch property data through a Node.js + Express backend with MongoDB.  
● Explore full-stack deployment via Render.  
● Interact with modular UI components like Property Cards.  
● View search results dynamically and seamlessly.

**2. Document overview**

The remainder of this document is 8 chapters, the first providing an introduction to the project. It lists all the functions performed by the system. The second chapter consists of software requirements specification. The third chapter provides details about system analysis and design. The fourth chapter gives data dictionary information. The fifth chapter consists of snapshots of the complete project. The sixth chapter gives testing for the project. The seventh chapter tells about the conclusion and future enhancements of the project. The final chapter concerns the bibliography. This document is meant for describing all the features and procedures that were followed while developing the system. This document specially mentions the details of the project, how it was developed, the primary requirement, as well as various features and functionalities of the project, and the procedures followed in achieving these objectives.

# **ACKNOWLEDGEMENT**

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed guide **Dr. Naresh Mali (Project In-charge), TINJRIT** for their valuable guidance, encouragement, and help in completing this work. Their useful suggestions for this whole work and cooperative behaviour are sincerely acknowledged. At the end, I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly during this project work.

Harshal Paliwal

Jainil Jain

Kailash Joshi

Varun Sharma

Place - Udaipur

Date - 09/05/2025

# 

# **TABLE OF CONTENTS**

**Table of Contents**

Abstract....................................................................................................................................... 3

Acknowledgement ...................................................................................................................... 5

Table of Contents ...................................................................................................................... 6-7

List of Figures.............................................................................................................................. 8

**CHAPTER – 1 Introduction**....................................................................................................9-10

Introduction .................................................................................................................................. 9

1.1.Purpose.................................................................................................................................. 10

**CHAPTER – 2 Software Requirement Specification** ......................................................... 11-18

2.1. Purpose................................................................................................................................. 11-13

2.2. Overall Description................................................................................................................ 14-15

2.3. Functional Requirements................................................................................. 15-16

2.4. Use Cases.......................................................................................................................... 17

2.5. Non functional requirements……………………………………………………………………17

2.6. Hardware and Software Requirements ................................................................... 18

**CHAPTER – 3 System Analysis and Design** ...................................................................... 19-27

3.1. Analysis of current system .............................................................................. 20

3.2 Benefits of using this over current system .............................................................................................. 21-22

3.3. Limitations of this system ................................................................................................................. 23

3.4. Data flow diagram …........................................................................................................... 23-26

3.5 Class Diagram ….............................................................................................................. 27

3.6 Context Diagram .............................................................................................................. 27

3.7. Sequence Diagram .......................................................................................................... 27

**CHAPTER – 4 Data Dictionary** ............................................................................................ 28-38

4.1. Table Details .......................................................................................................................28-38

4.2. E-R Diagram ................................................................................................................... 38

**CHAPTER - 5 Screenshots** ................................................................................................ 39-46

**CHAPTER – 6 Testing** .......................................................................................................... 47-52

**CHAPTER – 7 Conclusion and Future Enhancements** ..................................................... 53-54

7.1. Limitations .......................................................................................................................... 53

7.2. Future Enhancements ........................................................................................................ 53

7.3. Conclusion ..........................................................................................................................54

**CHAPTER – 8 Project Timeline…………………………**…...................................................... 55

**CHAPTER – 9 Bibliography and Dependencies** .................................................................. 56

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page Number** |
| Fig 1 | Level 0 DFD | 20 |
| Fig 2 | Level 1 DFD | 20 |
| Fig 3 | Level 2 DFD | 21 |
| Fig 4 | UML Class Diagram | 22 |
| Fig 5 | E-R Diagram | 38 |
| Fig 6 | Project Timeline | 46 |

# 

# **CHAPTER – I**

## **INTRODUCTION**

**1. Purpose**

**1.1. Introduction**

The Ghar Real Estate project is a MERN-stack web application that allows users to search, view, and explore property listings. The backend uses Express.js and MongoDB to handle data storage and API routes, while the frontend is built with React for a clean, responsive interface. Users can search properties by keyword, view detailed listings, and navigate between sections like Home, About, and Contact. The app is deployed on Render, demonstrating a full-stack deployment pipeline. It provides a simple yet functional property search experience,.

**1.2. Scope**

The Ghar platform enables users to:  
● Search and view real estate property listings across various locations.  
● Access detailed property info including images, prices, and descriptions.  
● Use a keyword-based search bar for quick filtering of properties.  
● Navigate between Home, About, and Contact pages via a responsive UI.  
● Enjoy a clean, mobile-friendly layout built with React.  
● Fetch property data through a Node.js + Express backend with MongoDB.  
● Explore full-stack deployment via Render.  
● Interact with modular UI components like Property Cards.  
● View search results dynamically and seamlessly.

**1.1.3. References**

[IEEE] The applicable IEEE standards are published in “IEEE Standards Collection,” 2001 edition.

[Bruade] The principal source of textbook material is “Software Engineering: An Object-Oriented Perspective” by Eric J. Bruade (Wiley 2001).

**1.1.4. Document Overview**

The remainder of this document is 8 chapters, the first providing an introduction to the project. It lists all the functions performed by the system. The second chapter consists of software requirements specification. The third chapter provides details about system analysis and design. The fourth chapter gives data dictionary information. The fifth chapter consists of snapshots of the complete project. The sixth chapter gives testing for the project. The seventh chapter tells about the conclusion and future enhancements of the project. The final chapter concerns with the bibliography. This document is meant for describing all the features and procedures that were followed while developing the system. This document specially mentions the details of the project, how it was developed, the primary requirement, as well as various features and functionalities of the project, and the procedures followed in achieving these objectives.

# 

# **CHAPTER – II**

## **SOFTWARE REQUIREMENT SPECIFICATION**

### **2.1. Purpose**

**2.1.1. Introduction**

This Software Requirements Specification provides a complete description of all the functions and specifications of the web application Ghar Real Estate. The main objective of this application is to simplify the property search experience by allowing users to browse, search, and view detailed listings of real estate properties. It enables seamless navigation between pages, displays relevant property data dynamically, and supports efficient data management through a MERN stack backend. The application is optimized for responsiveness and real-world deployment on Render.

**2.1.2. Scope**

The Ghar platform enables users to:  
● Search and view real estate property listings across various locations.  
● Access detailed property info including images, prices, and descriptions.  
● Use a keyword-based search bar for quick filtering of properties.  
● Navigate between Home, About, and Contact pages via a responsive UI.  
● Enjoy a clean, mobile-friendly layout built with React.  
● Fetch property data through a Node.js + Express backend with MongoDB.  
● Explore full-stack deployment via Render.  
● Interact with modular UI components like Property Cards.  
● View search results dynamically and seamlessly.

**2.1.3. Glossary**

|  |  |
| --- | --- |
| **TERM** | **Definition** |
| Admin | User with full access to manage listings and backend property data. |
| User | Any visitor or customer browsing or searching for properties. |
| Listing | An individual property entry with price, description, location, and images. |
| Search Bar | UI element allowing users to input keywords to filter property listings. |
| Property Card | Component displaying a property's summary including price, title, and image. |
| MongoDB | NoSQL database used to store property data. |
| Express.js | Node.js framework used for building the backend REST API. |
| React | JavaScript library used for building the dynamic frontend UI. |
| Route | Goods and Services Tax Identification Number in India. |
| Render | Hosting service used for deploying both the frontend and backend. |
| API | Backend endpoints allowing CRUD operations on property listings. |
| Deployment | Process of launching the full-stack application for public access. |

**2.1.4. References**

[IEEE] The applicable IEEE standards are published in “IEEE Standards Collection,” 2001 edition.

[Bruade] The principal source of textbook material is “Software Engineering: An Object-Oriented Perspective” by Eric J. Bruade (Wiley 2001).

**2.1.5. Document Overview**

The remainder of this document is 8 chapters, the first providing an introduction to the project. It lists all the functions performed by the system. The second chapter consists of software requirements specification. The third chapter provides details about system analysis and design. The fourth chapter gives data dictionary information. The fifth chapter consists of snapshots of the complete project. The sixth chapter gives testing for the project. The seventh chapter tells about the conclusion and future enhancements of the project. The final chapter concerns the bibliography.

### **2.2. Overall Description**

**2.2.1 Product Perspective**

Ghar Real Estate is a standalone full-stack web application comprising:

a. **Frontend**: React-based client (client/src) with modular components (e.g., PropertyCard, Navbar) and state handled via useState/useEffect.

b. **Backend**: Node.js/Express server (index.js, routes/) exposing RESTful APIs for property data.

c. **Database**: MongoDB schema storing property details like title, location,

image, and price.

d. **Infrastructure**: Deployed using Render; project structured for separation of client and server code, configured with package.json script

**2.2.2 User Classes and Characteristics**

a. **Visitor/User**: General public browsing available real estate listings. Requires a responsive UI and intuitive navigation.

b. **Admin**: Authorized personnel with privileges to add, update, or remove listings and manage backend data.

**2.2.3 Operating Environment**

a. **Supported browsers**: Chrome, Firefox, Edge (latest versions).

b. **Server**: Node.js v18+, MongoDB, Render-hosted environment.

c. **Deployment**: Full-stack deployment using Render’s free hosting tier.

**2.2.4 Design and Implementation Constraints**

a. Hosted frontend and backend must operate over HTTPS.

b. Environment variables are managed securely via Render’s dashboard.

c. All property operations are done via REST APIs.

**2.2.5 Assumptions and Dependencies**

a. Continuous internet access is assumed.

b. Cloud image hosting service Firebase integrated for

property images.

c. React and Node dependencies managed via npm.

### **2.3 Functional requirements**

a. **User Authentication & Authorization**

- Login/Logout for Users.

B**. User Management**

- Create, Read, Update, Delete (CRUD) user records.

- User gets a randomly generated username.

- User can update username, email, password, and their avatar.

c. **Property management**

Create Property Listing

* Input property name, description, and address.
* Upload up to 6 images (first image = cover photo, size < 2MB).

Listing Type Selection

* Choose between:
  + Sell
  + Rent
  + Optional tags: Parking Spot, Furnished, Offer.

Property Features

* Number of Beds and Baths.
* Toggle additional features (e.g., parking, furnished, offer).

Pricing Details

* Regular price (₹ or ₹/month based on rent/sale).
* Discounted price (if applicable).

Image Upload Handling

* Validates image count (max 6) and file size (< 2MB each).
* First image is shown as cover in listings.

Form Submission

* After filling the form, user clicks "Create Listing".
* Data is sent to backend via REST API for listing creation.

### **2.4 Use cases**

This system has three user modules: **Administrator**, **Property Manager**, and **Sales Agent**. Each module has dedicated permissions and responsibilities to streamline real estate listing and rental workflows.

**[1] Administrator Module**

The Administrator has full control and visibility across all system features:

* Manage all listings (view, filter, edit, delete)
* Approve or reject listings
* View property statistics
* Manage user roles (add/remove Property Managers and Sales Agents)
* Manage system-wide settings (listing rules, image limits, rent/sell options)
* Access audit logs of user actions
* Manage location categories and amenities

**[2] Property Manager Module**

The Property Manager is responsible for listing and maintaining properties:

* Create Listing  
  (Add name, description, address, images, select Rent/Sell, set prices, add amenities)
* Edit Listing (update fields or change listing type)
* Delete Listing
* Upload property images (limit: 6, first image = cover)
* Mark listing as Featured or Discounted
* Add or update metadata (beds, baths, furnished, parking)
* Set regular and discounted price
* View listing performance analytics (views, leads)

### **2.5 Hardware & Software Requirements**

**2.5.1 Hardware:**

As this is a web application it runs on a browser, preferably Google Chrome.

So the minimum hardware requirements for smooth functioning of this web application are:

1. **Windows -**

Windows 7, Windows 8, Windows 8.1, Windows 10 or later

* An Intel Pentium 4 processor or later that's SSE3 capable

Note: Servers require Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, or Windows Server 2016

1. **Mac -**

To use Chrome browser on Mac, you'll need:

* OS X El Capitan 10.11 or later

1. **Linux -**

64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, or Fedora Linux 32+

An Intel Pentium 4 processor or later that's SSE3 capable

1. **Android -**

Android Marshmallow 6.0 or later

**2.5.2 Software:**

It is recommended to run this application in google chrome browser version-102.0.5005.62

**CHAPTER – III**

# **SYSTEM ANALYSIS AND DESIGN**

### **3.1. Analysis of current system**

1. **Architecture & Components**

* **Client (React + Redux):** Single-Page Application handling UI flows for bookings, duty slips, invoices, vehicles, drivers, and admin functions. State is managed via Redux slices (session, user, locale, theme, common).
* **Server (Node.js + Express):** RESTful API defined in `routes/` (e.g. `dutySlips.js`, `createInvoice.js`), with business logic split into services.
* **Database (MongoDB):** Normalized schema with tables for users, customers, cars, drivers, duty\_slips, invoices, trip\_details, etc.
* **Image Storage**: Uses Firebase bucket for storing User avatars and property images
* **Deployment**: Deployed on a serverless service Render. Environment configured through `.env`.

1. **Key Workflows**

Create Property Listing

* Input property name, description, and address.
* Upload up to 6 images (first image = cover photo, size < 2MB).

Listing Type Selection

* Choose between:
  + Sell
  + Rent
  + Optional tags: Parking Spot, Furnished, Offer.

Property Features

* Number of Beds and Baths.
* Toggle additional features (e.g., parking, furnished, offer).

Pricing Details

* Regular price (₹ or ₹/month based on rent/sale).
* Discounted price (if applicable).

Image Upload Handling

* Validates image count (max 6) and file size (< 2MB each).
* First image is shown as cover in listings.

Form Submission

* After filling the form, user clicks "Create Listing".
* Data is sent to backend via REST API for listing creation.

### **3.2. Benefits of using this ERP over current practice:**

**1. Convenience and Accessibility**

Users can browse, list, and manage properties anytime, from anywhere, without needing to visit an office or call agents.

**2. Faster Transactions**

Instant listing uploads, search filters, and automated form submissions speed up the rental or sale process.

**3. Real-Time Updates**

Changes to price, availability, or property details reflect instantly—no delays or outdated information.

**4. Centralized Information**

All property data—photos, location, features, and prices—are stored and accessed from one place.

**5. Wider Reach**

Apps allow listings to be discovered by more users via search engines, social media, and sharing tools.

**6. Better Communication**

In-app chat, email notifications, and scheduled visit tools reduce missed inquiries and improve response times.

**7. Cost Efficiency**

Reduces need for physical marketing materials, field visits, or middlemen—saving both time and money.

**8. Data Analytics**

Owners and agents can track views, leads, and conversions to optimize pricing and listing strategy.

### 

### **3.3. Limitations of this web application:**

### **A. Web App Limitations**

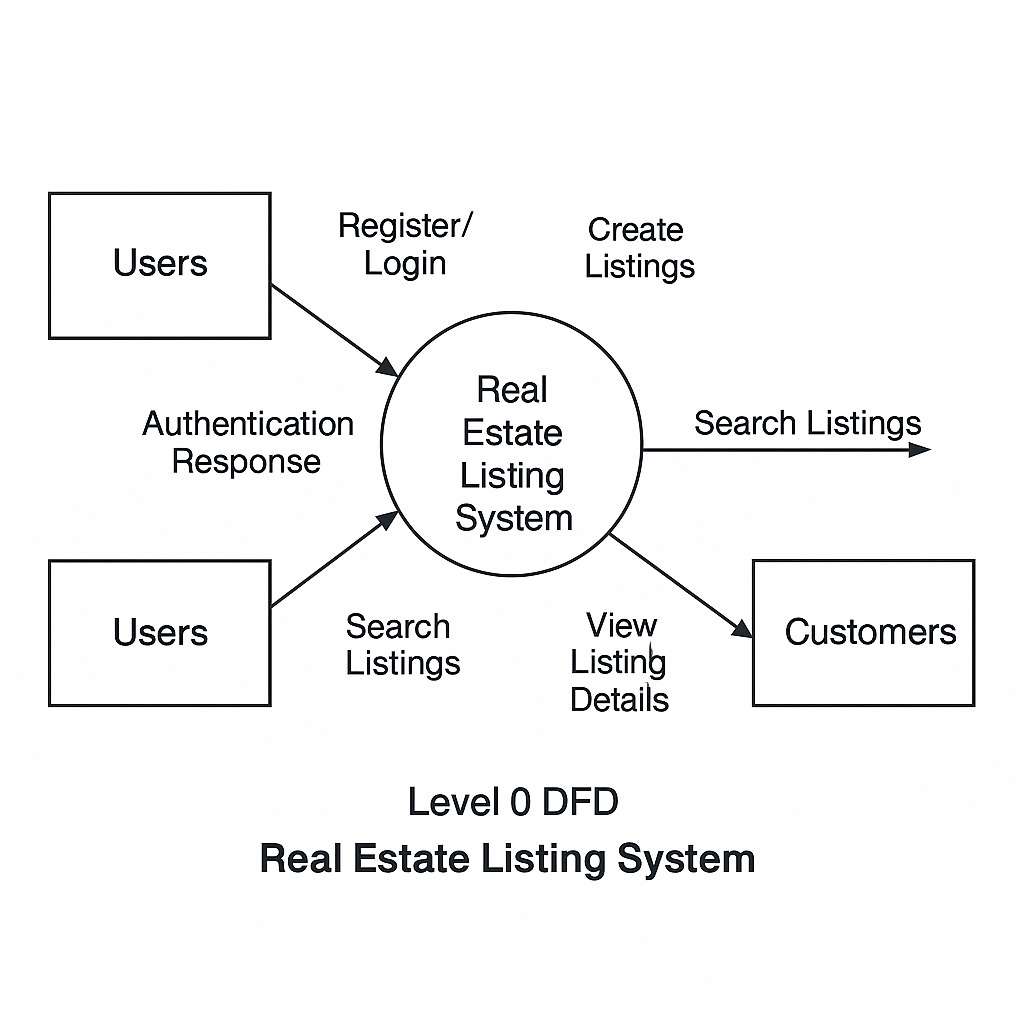
1. **Internet Dependency**  
   Users need a stable internet connection; no offline access to listings or saved data.
2. **Limited Mobile Experience**  
   If not optimized, web apps can feel clunky or slow on mobile devices compared to native apps.
3. **Security Concerns**  
   Sensitive user data (identity, financials) can be at risk without proper encryption and security practices.
4. **Scalability Issues**  
   High traffic without proper backend infrastructure can slow down or crash the application.
5. **Performance on Low-End Devices**  
   Web apps can lag on older devices, especially if they rely on heavy frontend frameworks.

### **B. General Digital Limitations**

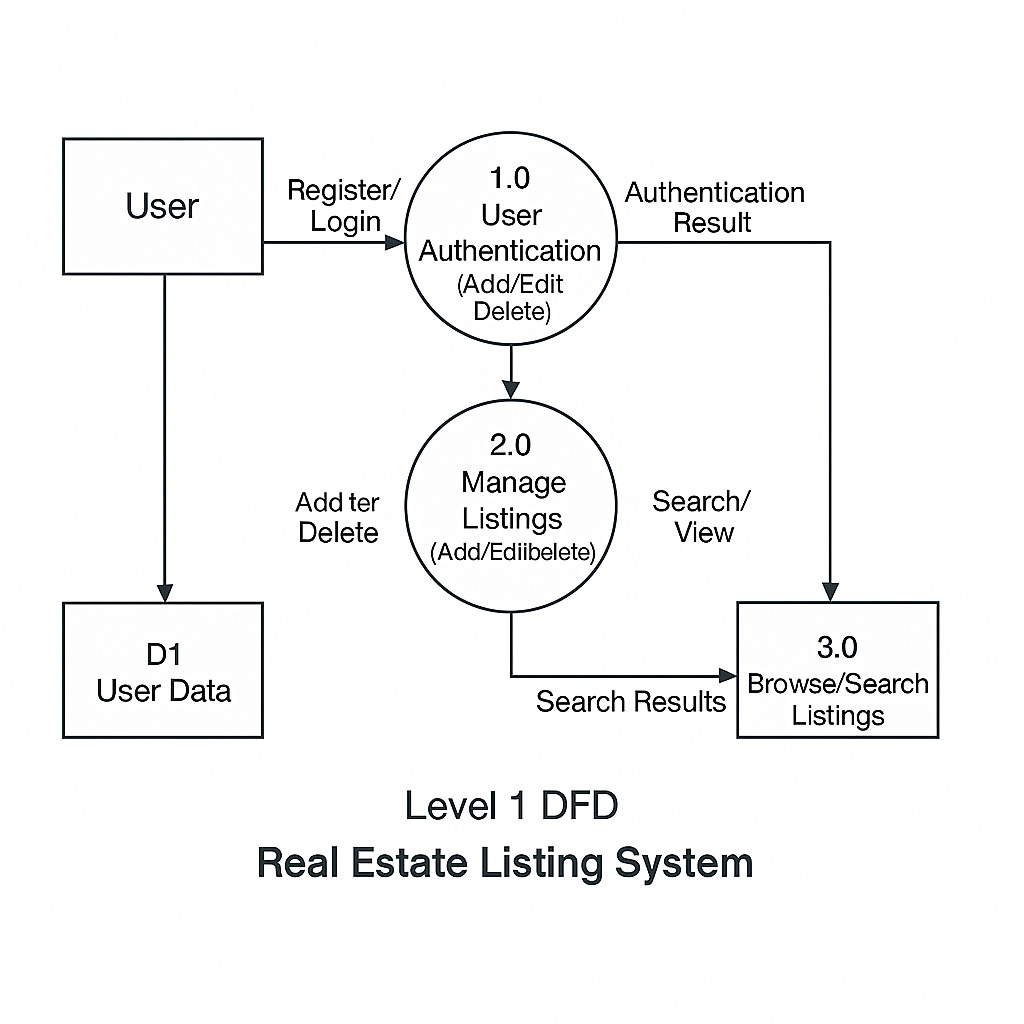
1. **Lack of Physical Interaction**  
   Users cannot physically inspect the property or neighborhood directly through the app.
2. **Fraud Risk**  
   Fake listings or misleading information can exist without strict verification systems.
3. **Digital Literacy Requirement**  
   Some users (especially elderly or rural populations) may find it difficult to navigate the app.
4. **Over-Reliance on Photos**  
   Visuals can be edited or not represent real conditions—users might form incorrect expectations.
5. **Trust Issues**  
   Without personal interaction, trust in agents or platforms may be lower compared to in-person dealings.

### **3.4. Data Flow Diagram (DFD)**

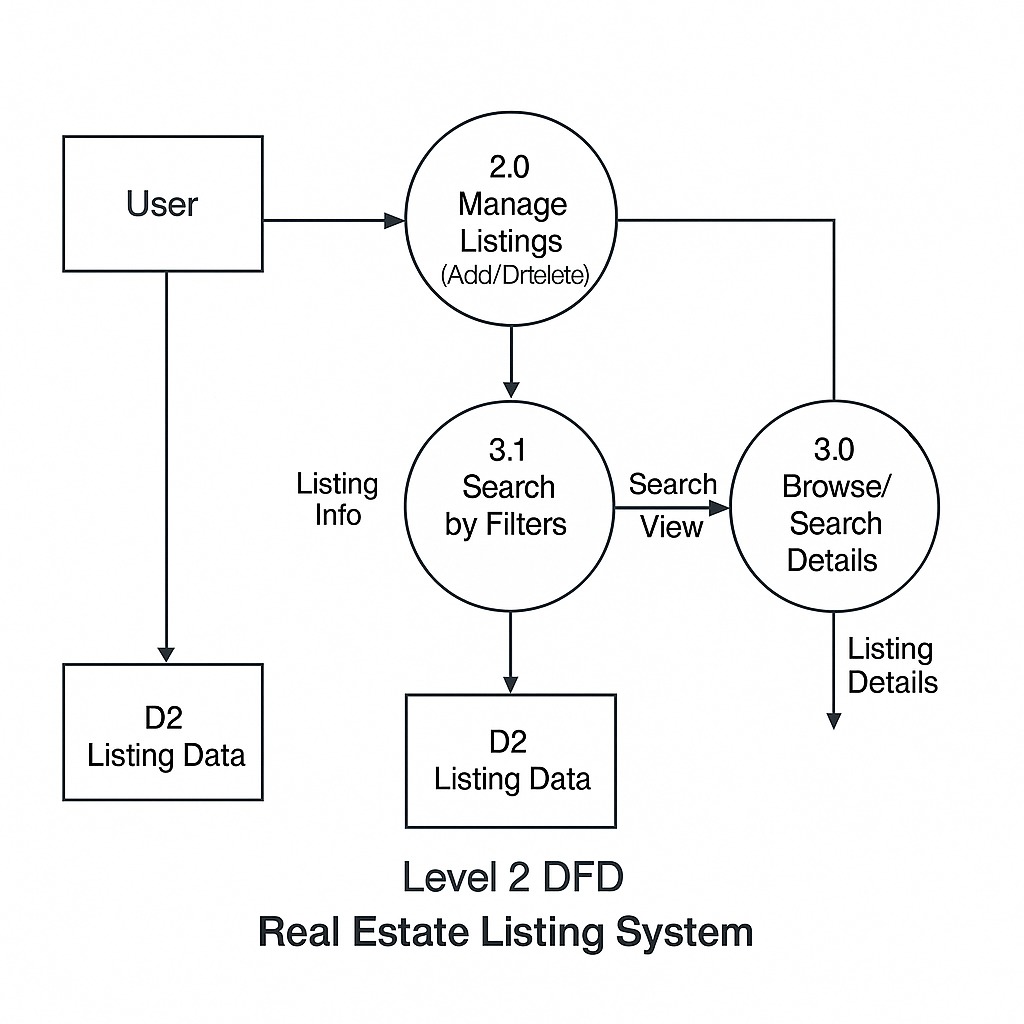
The DFD (also known as bubble chart) is a simple graphical formalism that can be used to represent a system in terms of the input data into the system, various processes carried on these data, and the output data generated by the system. The main reason why the DFD technique is so popular is because the DFD is a very simple formalism – it is simple to understand and use. A DFD model uses a very limited number of primitive symbols to represent the functions performed by a system and the data flow among the functions. Starting with a set of high-level functions that a system performs, a DFD model hierarchy represents various sub-functions.



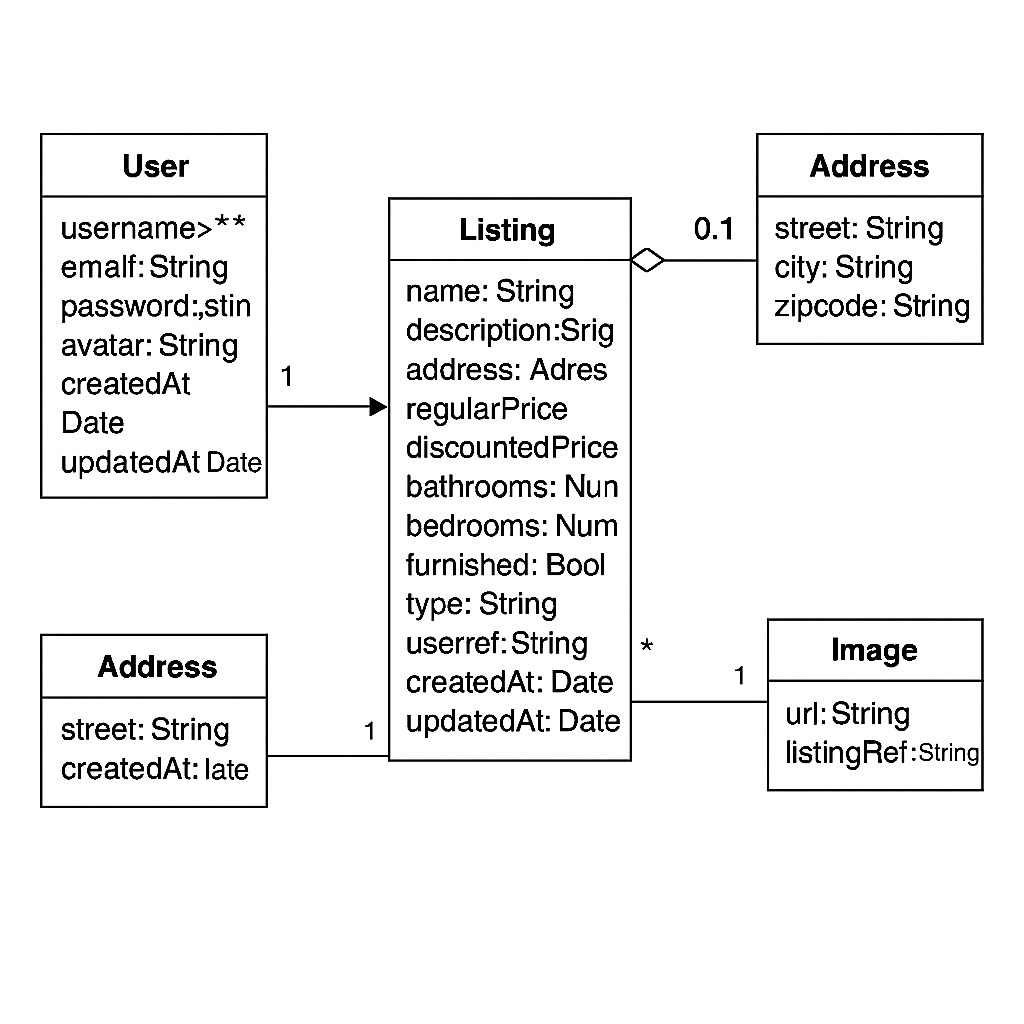
**Fig 1. Level 0 DFD diagram for The Ghar Real Estate**



**Fig 2. Level 1 DFD diagram for The Ghar Real Estate**



**Fig 3. Level 2 DFD for The Ghar Real Estate**



**Fig 4. UML class diagram**

# 

# **CHAPTER – IV**

## **DATA DICTIONARY**

### **4.1 Definition**

A data dictionary is a catalogue-a-repository of the elements in a system. As the name suggests, their elements centre on data and the way they are structured to meet user requirements and organisation needs. In a data dictionary, you will find a list of all the elements composing the data flowing through a system. The major elements are data flows, data stores, and processes. The data dictionary stores details and descriptions of these elements.

If analysis wants to know characters are in a data item by what other names it is referenced in the system, or where it is referenced in the system, or where it is issued in the system, they should be able to find the answers in issued in the system, they should be able to find the answer in the properly developed data dictionary.

The Dictionary contains two types of descriptions for the data following through the system.

**1. Data Elements**

The most fundamental data is the elements. They are building blocks for all other data in the system. Data elements are also alternatively known as fields, data items, or elementary items.

**2. Data Structure**

A data structure is a set of items that are related to one another and described

components in the system.

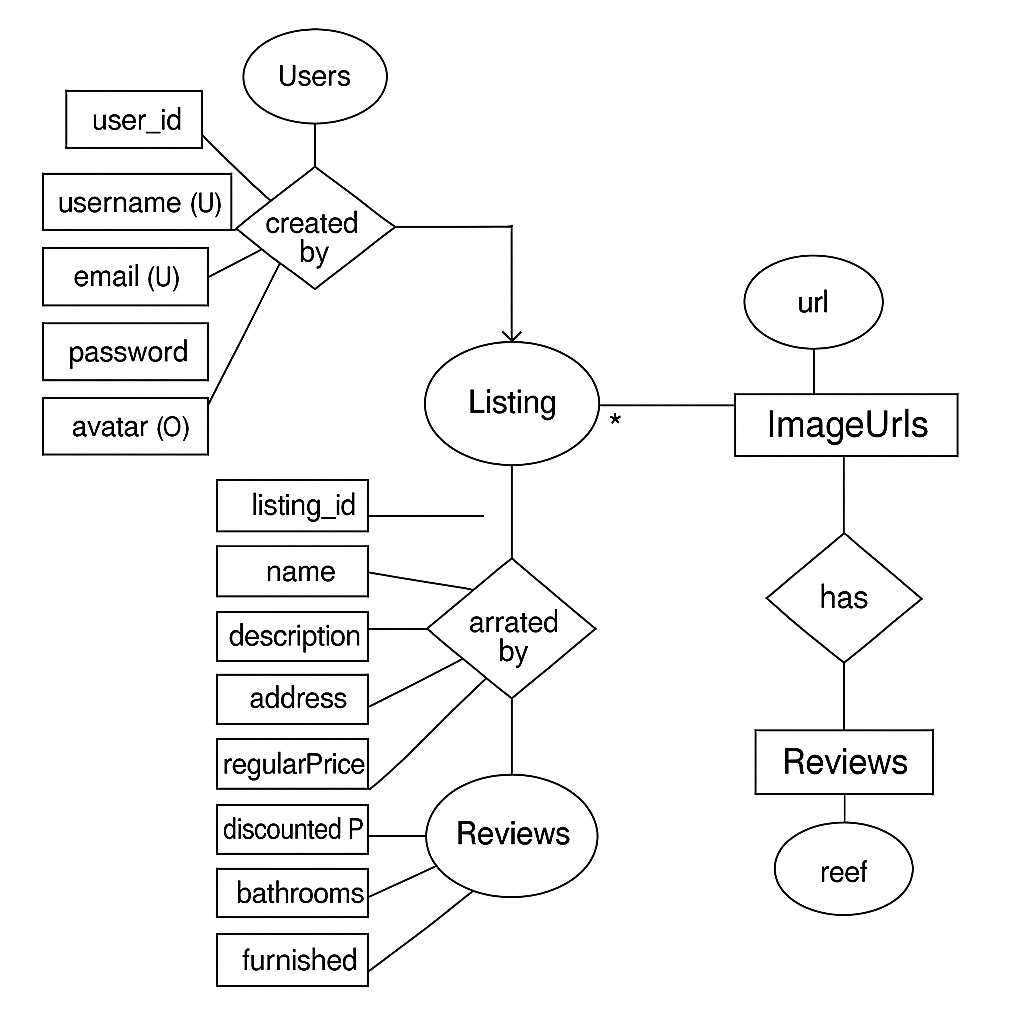
### **4.2 Table Details**

**Table 4.2.1 listing**

| **Field Name** | **Description** | **Constraint** | **Size** | **Data Type** |
| --- | --- | --- | --- | --- |
| name | Name of the listing | Required | Varies | String |
| description | Description of the listing | Required | Varies | String |
| address | Address of the property | Required | Varies | String |
| regularPrice | Regular price | Required | Number | Number |
| discountedPrice | Discounted price | Required | Number | Number |
| bathrooms | Number of bathrooms | Required | Number | Number |
| bedrooms | Number of bedrooms | Required | Number | Number |
| furnished | Whether furnished or not | Required | Boolean | Boolean |
| parking | Whether parking is available | Required | Boolean | Boolean |
| type | Type of listing (e.g. rent) | Required | Varies | String |
| offer | Whether offer is available | Required | Boolean | Boolean |
| imageUrls | List of image URLs | Required | Array | Array |
| userRef | Reference to user ID | Required | String | String |
| createdAt | Auto-generated timestamp | Auto-generated | Date | Date |
| updatedAt | Auto-updated on modification | Auto-generated | Date | Date |

**Table 4.2.2 user**

| **Field Name** | **Description** | **Constraint** | **Size** | **Data Type** |
| --- | --- | --- | --- | --- |
| username | User's unique name | Required, Unique | Varies | String |
| email | User's email address | Required, Unique | Varies | String |
| password | User's hashed password | Required | Varies | String |
| avatar | Profile picture URL | Optional, Default | Varies | String |
| createdAt | Creation timestamp | Auto-generated | Date | Date |
| updatedAt | Last update timestamp | Auto-generated | Date | Date |



**Fig 5. E-R Diagram**

# 

# **CHAPTER – V**

## **SCREENSHOTS**

**5.1 Login Page**

**A screenshot of a login page

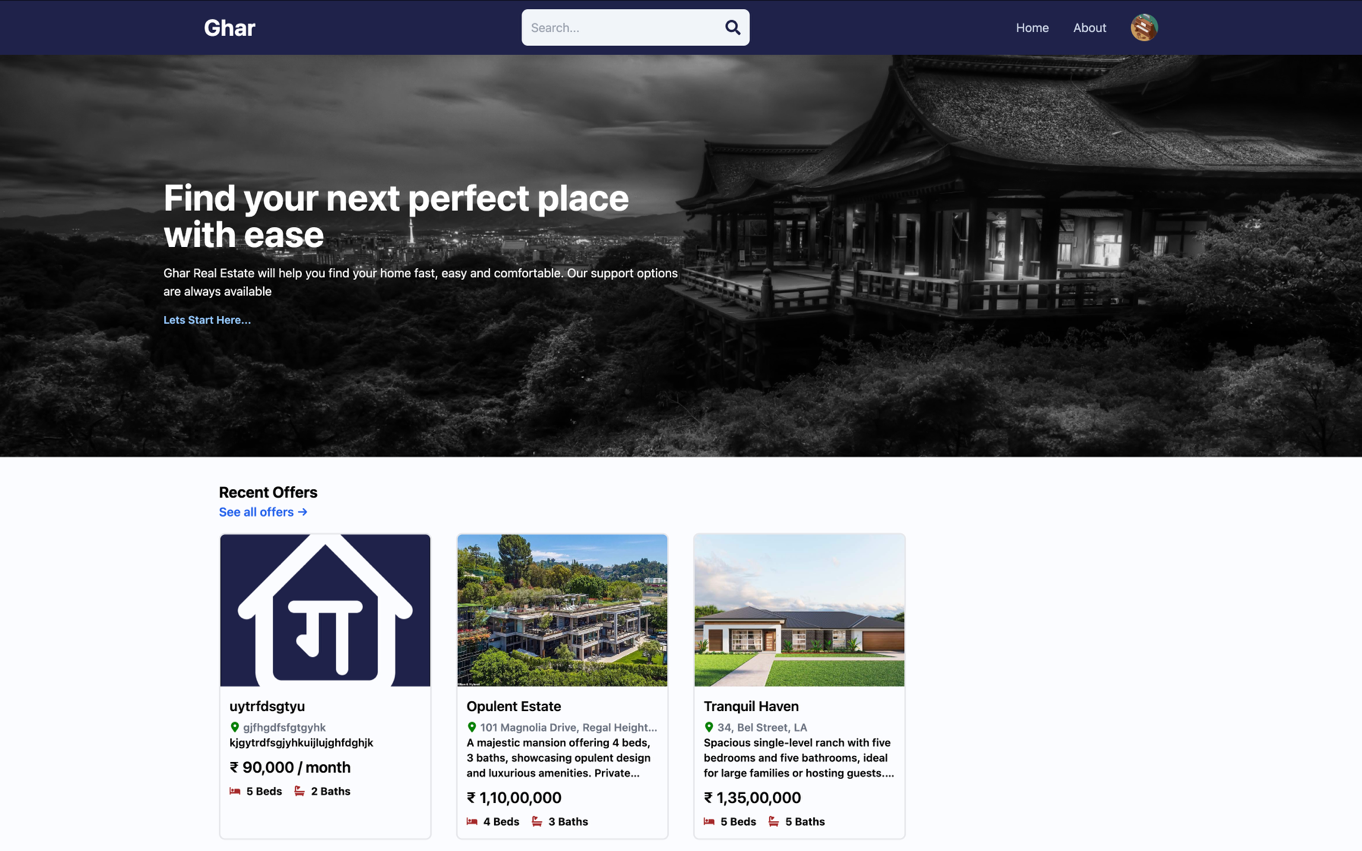
AI-generated content may be incorrect.**

**5.2 Signup Page**

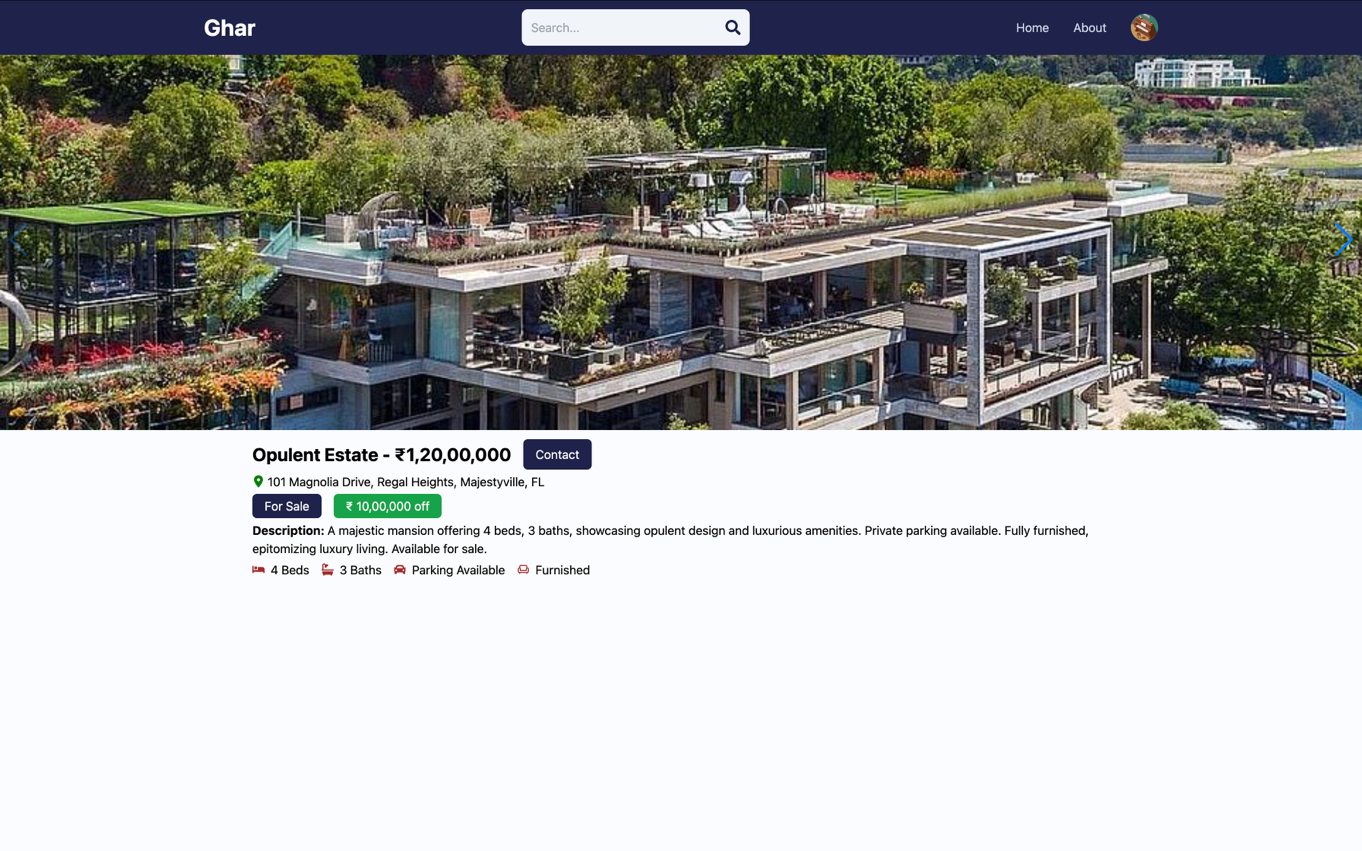
**A screenshot of a login page

AI-generated content may be incorrect.**

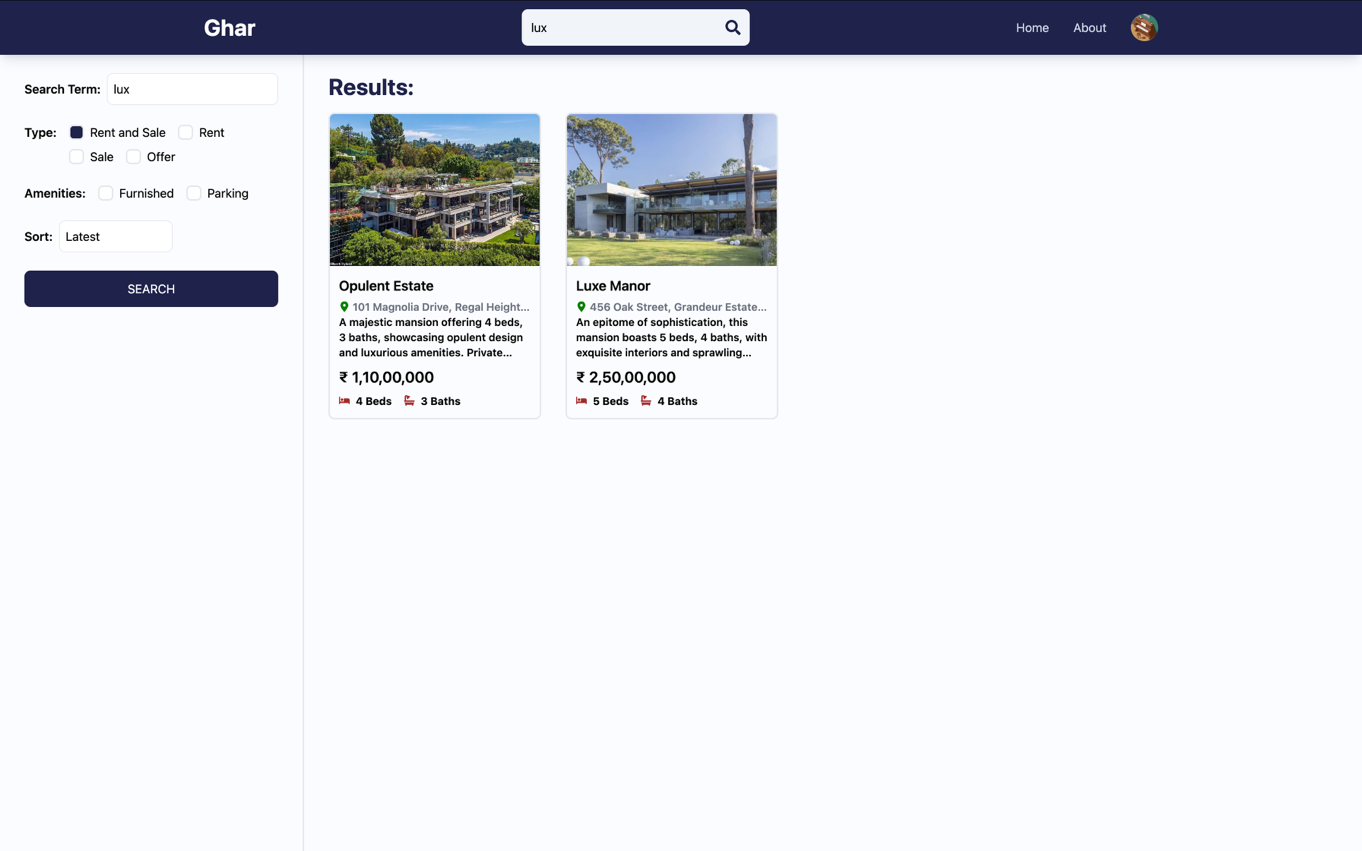
**5.3 Home Page**



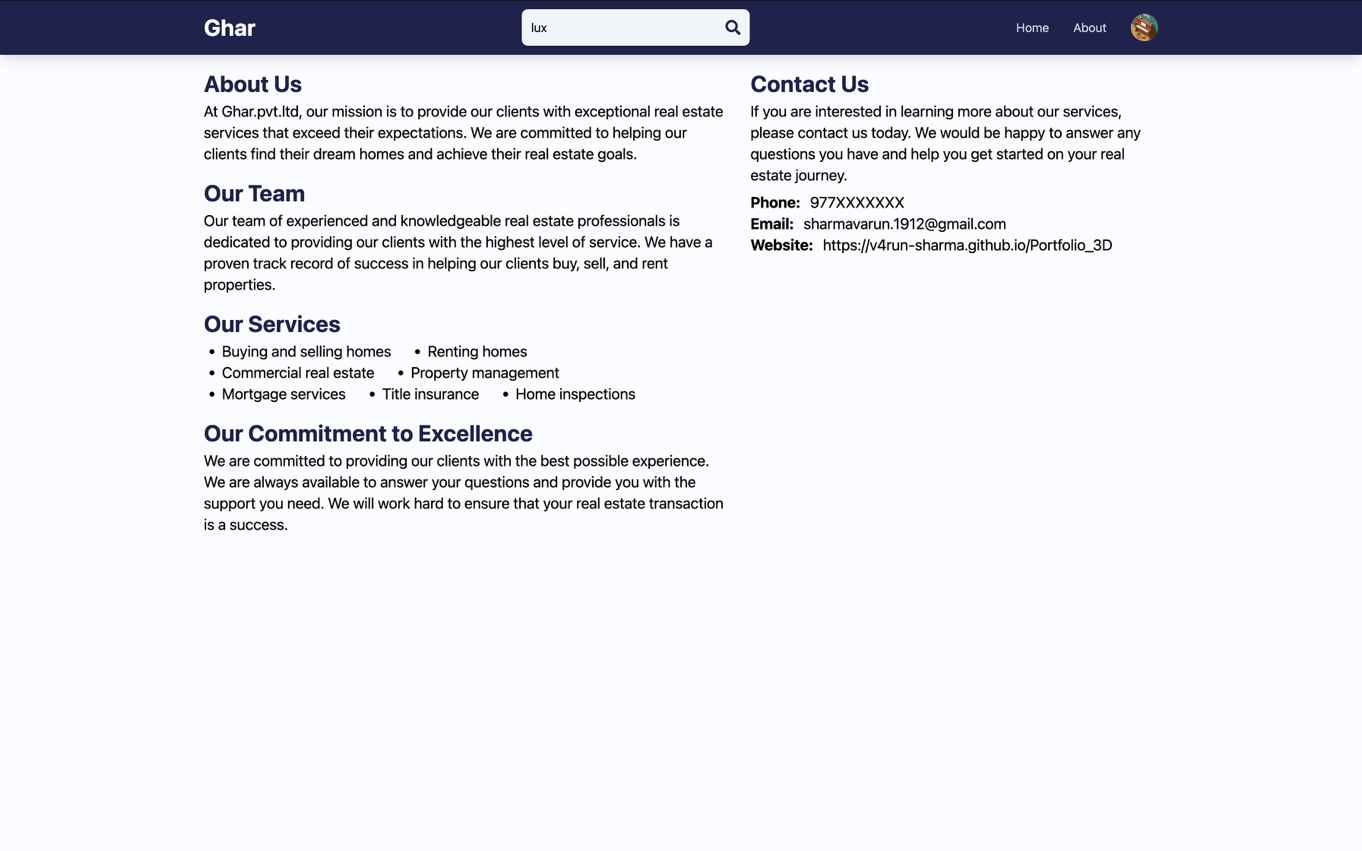
**5.4 Listing page**

****

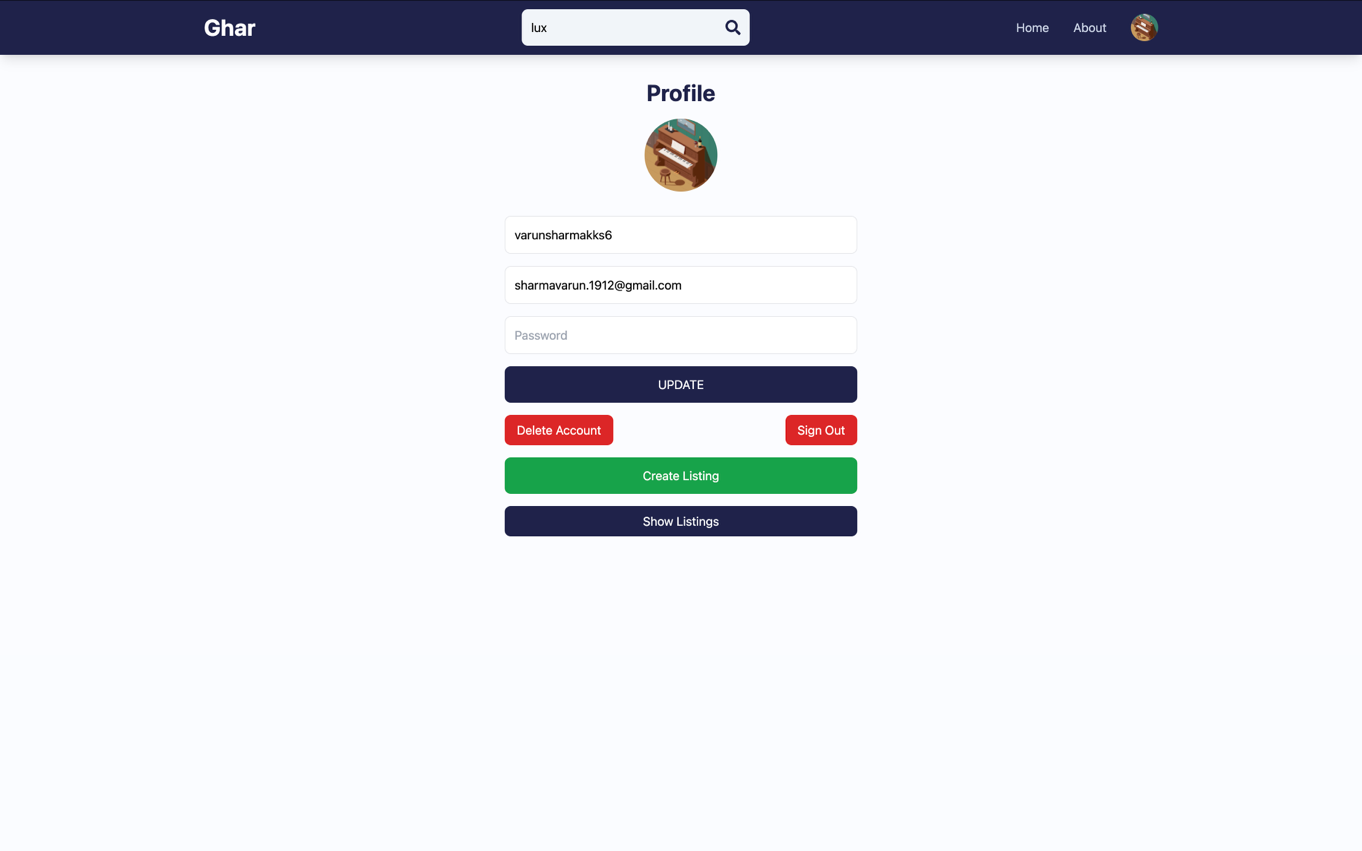
**5.5 Search Page**

****

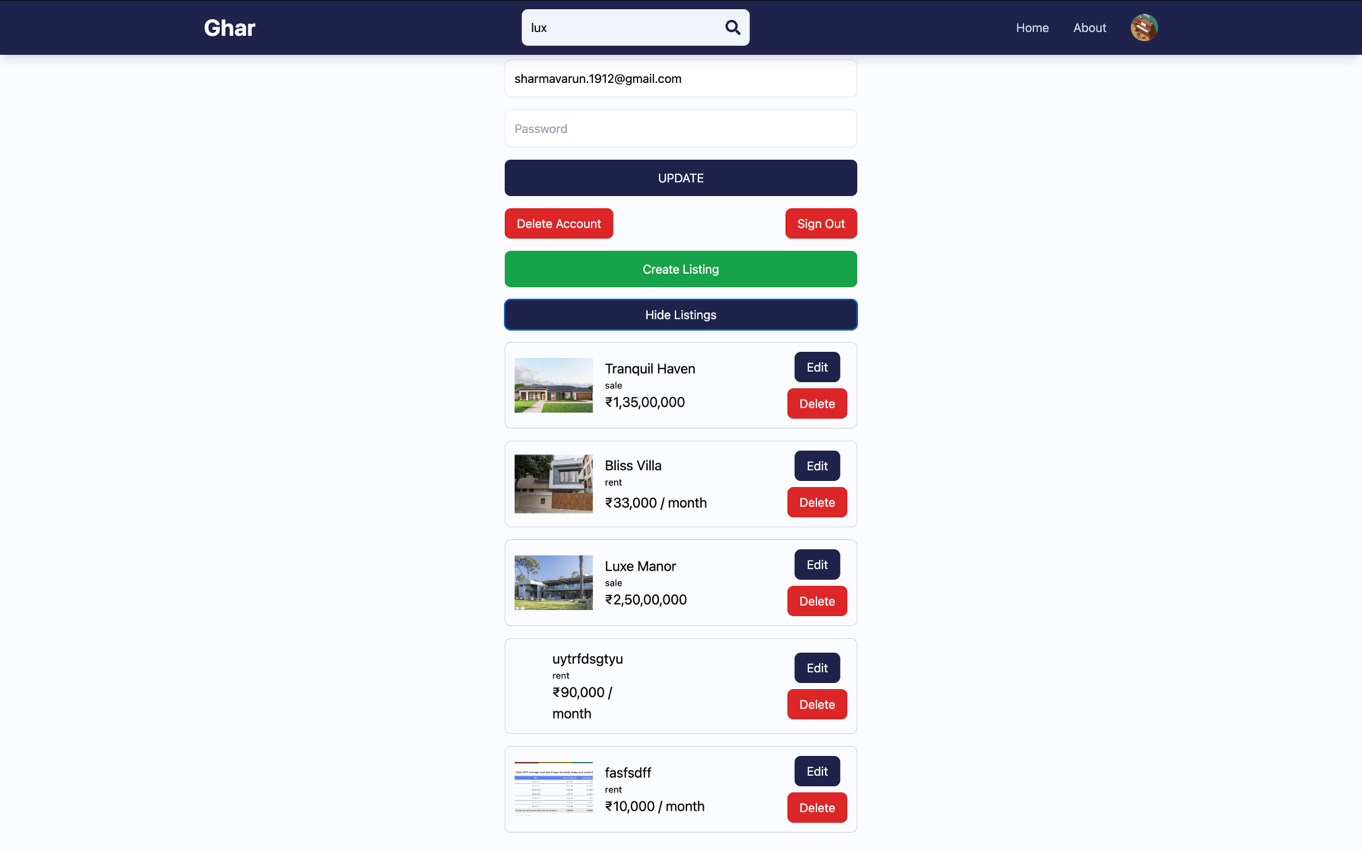
**5.6 About Page**

****

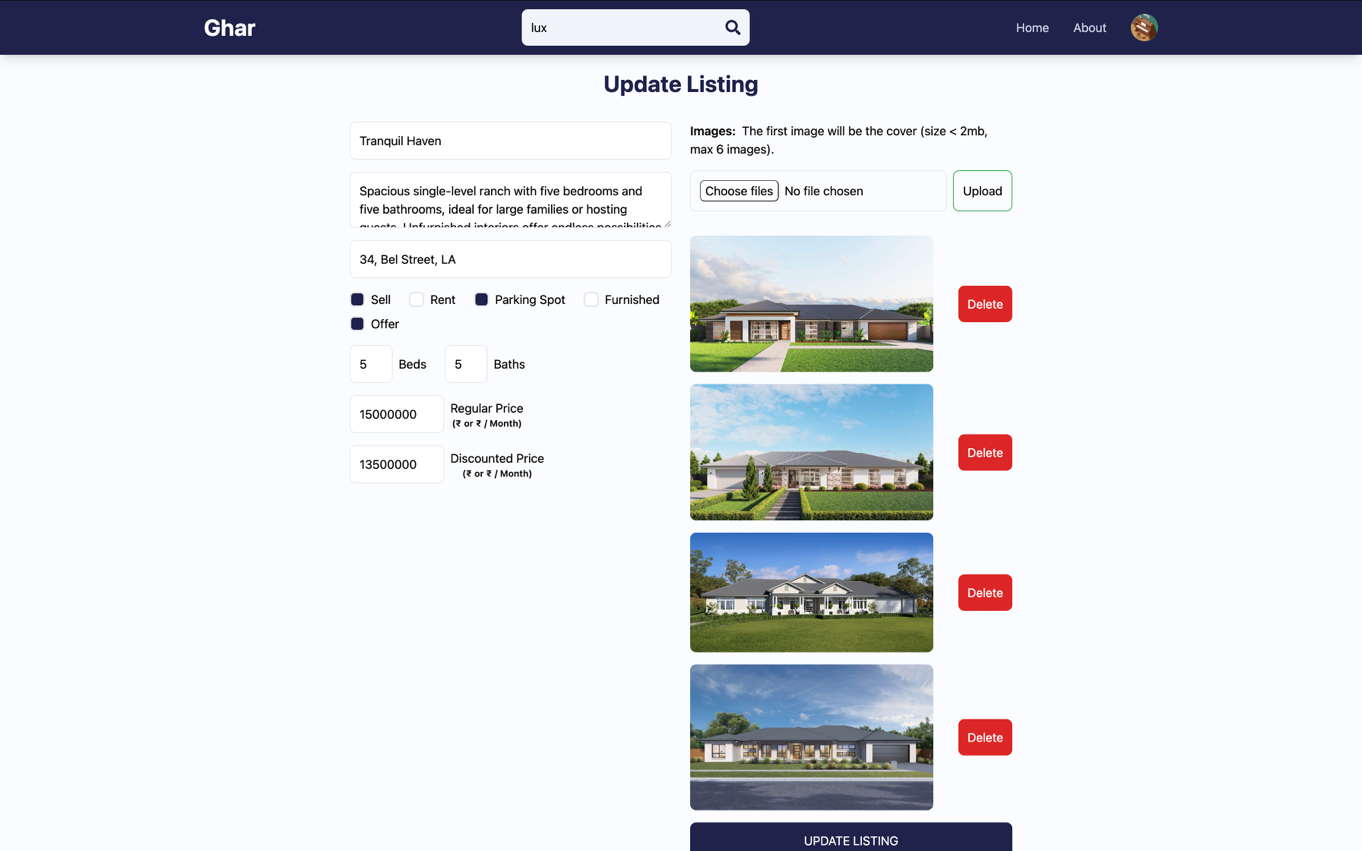
**5.7 Profile Page**

****

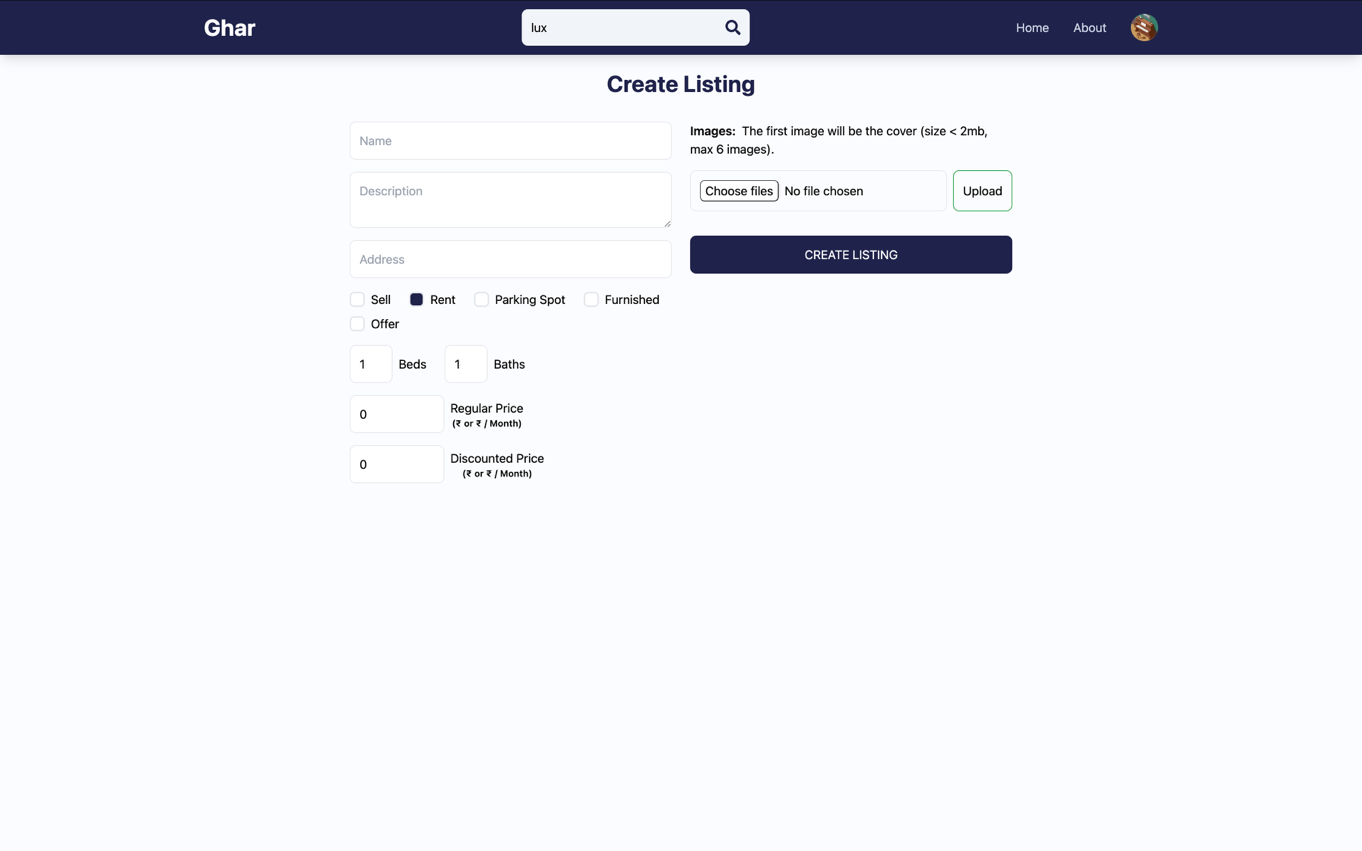
**5.8 User Listings**

****

**5.9 Update Listings Page**

****

**5.10 Create Listing Page**

****

# **CHAPTER – VI**

## **TESTING**

### **6.1 Testing Methodology**

Companies rely on software more than ever to provide and manage information with strategic and operational importance and to provide key decision support. Rising customer expectations for fault-free, requirements-exact software have increased awareness of the importance of software testing as a critical activity.

We begin the testing process by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process very

fies that the application meets the requirements specified in the system requirements document and is bug free. At the end of each testing day, we prepare a summary of completed and failed tests. Applications are not allowed to launch until all identified problems are fixed. A report is prepared at the end of testing to show exactly what was tested and to list the final outcomes.

Our software testing methodology is applied in three distinct phases: unit testing, system testing, and acceptance were testing.

**Unit Testing**: The programmers conduct unit testing during the development phase. Programmers can test their specific functionality individually or with other units. However, unit testing is designed to test small pieces of functionality rather than the system as a whole. This allows the programmers to conduct the first round of testing to eliminate bugs before they reach the testing staff. In unit testing the analyst tests the programs making up a system. For this reason, unit testing is sometimes called program testing. Unit testing gives stress on the

modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained within that module alone. The errors resulting from the interaction between modules are initially avoided.

For example, a hotel information system consists of modules to handle reservations; guest checking and checkout; restaurant, room service and miscellaneous charges; convention activities; and accounts receivable billing. For each, it provides the ability to enter, modify or retrieve data and respond to different types of inquiries or print reports. The test cases needed for unit-testing should exercise each condition and option. Unit testing can be performed from the bottom up, starting with smallest and lowest-level

modules and proceeding one at a time. For each module in bottom-up testing a short program is used to execute the module and provides the needed data, so that the module is asked to perform the way it will when embedded within the larger system.

**System Testing:** The objective of system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and to ensure that the computer system and the associated clerical and other procedures work together. The initial phase of system testing is the responsibility of the analyst who determines what

conditions are to be tested, generates test data, produces a schedule of expected results, runs the tests and compares the computer produced results with the expected results with the expected results. The analyst may also be involved in procedures testing. When the analyst is satisfied that the system is working properly, he hands it over to the users for testing. The importance of system testing by the user must be stressed. Ultimately it is the user must verify the system and give the go-ahead. During testing, the system is used experimentally to ensure that the software does not fail, i.e.,

that it will run according to its specifications and in the way users expect it to. Special test data is input for processing (test plan) and the results are examined to locate unexpected results. A limited number of users may also be allowed to use the system so analysts can see whether they try to use it in unexpected ways. It is preferable to find these surprises before the organisation implements the system and depends on it. In many organisations, testing is

performed by persons other than those who write the original programs. Using persons who do not know how certain parts were designed or programmed ensures more complete and unbiased testing and more reliable software.

The system is tested as a complete, integrated system. System testing first occurs in the development environment but eventually is conducted in the production environment. Functionality and performance testing are designed to catch bugs in the system, unexpected results, or other ways in which the system does not meet the stated requirements. The testers create detailed scenarios to test the strength and limits of the system, trying to break it

if possible. Editorial reviews not only correct typographical and grammatical errors but also improve the system’s overall usability by ensuring that on-screen language is clear and helpful to

users. Accessibility reviews ensure that the system is accessible to users with disabilities.

System testing consists of the following five steps:

i. Program testing

ii. String testing

iii. System testing

iv. System documentation

v. User acceptance testing

**Program Testing**

A program represents the logical elements of a system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly with other programs. It is the

responsibility of a programmer to have an error free program. At The time of testing the system, there exist two types of errors that should be checked. These errors are syntax and logic. A syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted key words are common syntax

errors. These errors are shown through error messages generated by the computer. A logic error, on the other hand, deals with incorrect data fields out of range items, and invalid combinations. Since the logical errors are not detected by the compiler, the programmer must examine the output carefully to detect them. When a program is tested, the actual output is compared with the

expected output. When there is a discrepancy, the sequence of the instructions must be traced to determine the problem. The process is facilitated by breaking the program down into self-contained portions, each of which can be checked at certain key points.

**String Testing**

Programs are invariably related to one another and interact in a total system. Each program is tested to see whether it conforms to related programs in the system. Each part of the system is tested against the entire module with both test and live data before the whole system is ready to be tested.

**System Testing**

System testing is designed to uncover weaknesses that were not found in earlier tests. This includes forced system failure and validation of the total system as it will be implemented by its user in the operational environment. Under this testing, generally we Take low volumes of transactions based on live data. This volume is increased until the maximum level for each transaction type is reached. The total system is also tested for recovery and fallback after various major failures to ensure that no data are lost during the emergency. All this is done with the old system still in operation. When we see that the proposed system is successful in the test, the old system is discontinued.

**System Documentation**

All design and test documentation should be well prepared and kept in the library for future

reference. The library is the central location for maintenance of the new system.

**User Acceptance Testing**

An acceptance test has the objective of selling the user on the validity and reliability of the

system. It verifies that the system's procedures operate to system specifications and that the integrity of important data is maintained. Performance of an acceptance test is actually the user's show. User motivation is very important for the successful performance of the system. After that, a comprehensive test report is prepared. This report shows the system's tolerance, performance range, error rate and accuracy.

### **6.2 Test Report with test data**

**TEST REPORT WITH TEST DATA**

(To be filled by System Analyst/Programmer)

**Project Name: The Ghar Real Estate**

|  |  |  |
| --- | --- | --- |
| A. | INTERFACE TESTING  1) User-friendliness  2) Consistent menus | OK  NA |
| B. | CONTROL FLOW TESTING  1) IF-THEN-ELSE  2) DO WHILE  3) CASE-SWITCH | OK  OK  OK |
| C. | VALIDATION TESTING  1) Check for improper or inconsistent typing  2) Check for erroneous initialization or default values  3) Check for incorrect variable names  4) Check for inconsistent Data Types  5) Check for relational/arithmetic operators | OK  OK  OK  OK  OK |
| D. | DATA INTEGRITY/SECURITY TESTING  1) Data Insertion/ Deletion/ Updating  2) Boundary condition (Underflow, Overflow Exception)  3) Check for unauthorised access of data  4) Check for data availability | OK  OK  OK  OK |
| E. | EFFICIENCY TESTING  1) Throughput of the system  2) Response time of the system  3) Online disk storage required by the system  4) Primary memory required by the system | OK  OK  OK  OK |
| F. | ERROR HANDLING ROUTINES  1) Error description are intelligent/ understandable  2) Error recovery is smooth  3) All error handling routines are tested and executed at least once | OK  OK  OK |

# 

# **CHAPTER – VII**

## **CONCLUSION AND FUTURE ENHANCEMENTS**

### **7.1 Limitations:**

### **A. Web App Limitations**

1. **Internet Dependency**  
   Users need a stable internet connection; no offline access to listings or saved data.
2. **Limited Mobile Experience**  
   If not optimized, web apps can feel clunky or slow on mobile devices compared to native apps.
3. **Security Concerns**  
   Sensitive user data (identity, financials) can be at risk without proper encryption and security practices.
4. **Scalability Issues**  
   High traffic without proper backend infrastructure can slow down or crash the application.
5. **Performance on Low-End Devices**  
   Web apps can lag on older devices, especially if they rely on heavy frontend frameworks.

### **B. General Digital Limitations**

1. **Lack of Physical Interaction**  
   Users cannot physically inspect the property or neighborhood directly through the app.
2. **Fraud Risk**  
   Fake listings or misleading information can exist without strict verification systems.
3. **Digital Literacy Requirement**  
   Some users (especially elderly or rural populations) may find it difficult to navigate the app.
4. **Over-Reliance on Photos**  
   Visuals can be edited or not represent real conditions—users might form incorrect expectations.
5. **Trust Issues**  
   Without personal interaction, trust in agents or platforms may be lower compared to in-person dealings.

### **7.2 Future enhancements:**

**A. Web App Limitations – Possible Enhancements**

1. **Offline Support via Service Workers**
   * Enable caching of listings and saved data so users can browse offline (e.g., using Progressive Web App (PWA) features).
2. **Mobile Optimization / Responsive Design**
   * Implement a mobile-first design, lazy loading, and adaptive components to ensure smooth UX across devices.
   * Consider developing a dedicated mobile app using React Native or Flutter for even better performance.
3. **Enhanced Security Measures**
   * Use HTTPS, JWT for secure authentication, and encrypt sensitive data at rest and in transit.
   * Implement two-factor authentication and periodic security audits.
4. **Backend Scalability**
   * Use scalable infrastructure like Kubernetes, AWS Lambda, or containerized microservices.
   * Integrate load balancing, auto-scaling, and database optimization strategies.
5. **Performance Optimization for Low-End Devices**
   * Minimize use of heavy JavaScript frameworks; optimize bundle size, use code splitting, and serve compressed assets.
   * Implement skeleton loaders and reduce DOM complexity.

**B. General Digital Limitations – Possible Enhancements**

1. **Virtual Tours & Augmented Reality**
   * Add 360-degree virtual tours or AR views of properties to simulate physical inspection.
2. **Listing Verification System**
   * Introduce a verified badge system, manual moderation, or AI-based fraud detection for listings.
   * Enable user reporting and reviews to flag fake content.
3. **Simplified UI & Accessibility Features**
   * Use icons, multilingual support, voice navigation, and tutorial modes to help less digitally literate users.
4. **Transparency in Photos**
   * Include photo timestamps, AI-verified images, or comparison sliders showing original vs. edited photos.
5. **Reputation & Review System**
   * Implement agent ratings, verified user reviews, and trust scores to build credibility in the platform.

### **7.3 Conclusion:**

* The current system successfully digitizes duty slip and invoicing workflows, centralizing operations in a modern web stack.
* It addresses manual bottlenecks but still faces challenges in scalability, security, and advanced reporting.

# 

# **CHAPTER - VIII**

## **Project Timeline**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tasks** | **Nov–Dec 2024** | **Jan–Feb 2025** | **Feb–Mar 2025** | **Mar–Apr 2025** | **April 2025** |
| Requirement Gathering | ✅ |  |  |  |  |
| Planning & Research | ✅ |  |  |  |  |
| Logical Design | ✅ |  |  |  |  |
| Prototyping | ✅ | ✅ |  |  |  |
| Development |  | ✅ |  |  |  |
| Testing |  | ✅ |  |  |  |
| Feedback |  | ✅ | ✅ | ✅ |  |
| Deployment |  |  |  |  | ✅ |

**Fig. 6**

**Project timeline**

This project is divided into 8 phases namely Requirement gathering, Planning and Research, Logical design, Prototyping, Development, Testing, Feedback, and Deployment.

After the month of April 2025, end-user testing was carried out.

It has been thoroughly tested and is currently deployed.

**CHAPTER – IX**

## **BIBLIOGRAPHY & DEPENDENCIES**

### **9.1 References and Bibliography:**

1. React Documentation *– A JavaScript library for building user interfaces*. Retrieved from<https://reactjs.org>
2. *Node.js Documentation* : Retrieved from <https://nodejs.org/en/docs>
3. *Bootstrap – The most popular HTML, CSS, and JS library*. Retrieved from<https://getbootstrap.com>
4. *GitHub - Code hosting platform for version control*. Retrieved from<https://github.com>
5. Stack Overflow and Medium - *for issue resolution*

### **9.2 Dependencies:**

* Mongoose - for database layer
* Jsonwebtoken - for token verification
* Bcryptjs - to hash passwords before storing
* Nodemailer - for password recovery