AKAR - A recommendation system for real estate investment

AKAR - A recommendation system for real estate investment

Project Members

1- Hazem Mohamed Abd El Naeem Mesbah | 20-00070

2- Shehab ElDin Ahmed Abd El Rauf Qutb | 20-00071

3- Menna Ramadan Mahmoud | 20-00075

4- Sohila Ibrahem Abd El Hamid Hassan | 20-00373

5- Nada Ali Hassan Mohamed | 20-00639

6- Shahd Maged Mohamed Morsy | 20-00525

7- Myriam Jan Gendy Hanna | 20-00257

Project Supervisors Prof

PROF / Amira Idrees

Teaching Assistant

Eng. Samah Essam

Abstract

An overview of the idea of the project, which is to make it easier for the user to buy the property without the need for an intermediary and to be exposed to the well-known cases of fraud that most users are exposed to. The site displays a group of properties, including which allows the user to communicate directly with the owner without the need for an intermediary, because in this case the intermediary will be between the seller and the buyer. It is the site.

The site also provides a unique feature for the user, where he can place any property he wants to know its price. The site requests a set of data, from which it analyzes that data in terms of area, city, and other things, and then shows him a suitable price for that property.

The site also allows the user to search for a property that suits the appropriate amount he has.

The result

result of the idea: We implemented seven pages

The Home page, which displays a set of data about the types of properties available to us and a list of those properties

About us page that displays an overview of the project

On the search page, in which the user add the price available to him and the area he wants at this price, and he is presented with a group of properties at this price and area.

The chat bot page is a page on which a set of data is requested that the user is required to enter, and it responds to him at an expected price.

Property list page that displays a group of properties that the admin has added

The contact page is where the user communicates with the admin so that he can connect him directly to the property owner

Admin page. The admin logs in and then logs in to the dashboard so that he can add, modify, and delete any property.

CH1 Introduction

CH2 RELATED EXISTING SYSTEMS

CH3: SYSTEM REQUIREMENTS ENGINEERING AND PLANNING

CHAPTER4: SYSTEM DESIGN

CHAPTER FIVE: SYSTEM IMPLEMENTATION

CHAPTER SIX: Machine Learning and models

CHAPTER 7: PROJECT CONCLUSION AND FUTURE WORK

List Of Figure

Chapter 1 Introduction

Introduction:

The website offers means for users to get information on vacant units, see their data, and look at the building's and the unit's associated images, saving them the time, money, and trouble of physically visiting the unit. It also offers a variety of payment options. In addition to saving you money on advertising campaigns and giving the company's representatives inflated commissions, the programme also enables you to construct advertising models for the properties you have available and send them to each client's email.

Problem Background:

We have seen that fraud or brokers taking advantage of people is a concern when purchasing or renting real estate. We also found the problem of fraud and fraud from many brokers.

Problem Statement:

As we explained in the previous paragraph Problem Background about the brokers fraud and wasting the investor's time through a lot of tricks.

There are some examples like:

Example1: Over-specification

You will find an advertisement in newspapers or on websites that the property has excellent specifications and a low price, and when you communicate with the real estate broker and want to inspect the property, you will find it telling you that the property has unfortunately been sold and there are other properties with other specifications, and therefore the broker succeeded in reaching a potential client, which is you.

Or he tells you to come to visit the property and find the actual specifications that are not in the advertisement and tells you that the conditions will improve soon and that the owner will improve the property or tells you that the area where the property is located will increase real estate prices for any reason of his invention. Samar's goal in this case is the commission, and it is possible to believe the false promises of Samar and the owner of the property and find yourself buying a property that you regret owning for the rest of your life.

Example2: Property by owner

It is possible to find an advertisement in newspapers or websites that the property is being sold by the owner and when you go and inspect the property and you like it, the person you are dealing with tells you that he is a broker, and you find yourself in front of two options: - Either sacrifice the property you liked or overlook the lie and complete the deal, and therefore you will pay a commission to this lying person.

Example3: Inspection costs

The real estate broker will tell you that you must pay an amount of money in order to prove your seriousness that you really want to buy or rent a property, and this amount in Egypt, for example, is about 100-200 pounds and varies according to the country in which you are, and of course you must provide the real estate broker with a means of transportation while you are watching real estate and thus an increase in expenses.

The amount of the inspection is taken by the illegal real estate broker, it may reach the case of some unscrupulous brokers to offer you properties that are not offered for sale in order to benefit from the inspection and when you ask them to buy the property, they tell you that unfortunately this property has been sold, although the property you are asking to buy is not offered for sale in the first place, but is used in the monument.

Example3: Property by owner

It is possible to find an advertisement in newspapers or websites that the property is being sold by the owner and when you go and inspect the property and you like it, the person you are dealing with tells you that he is a broker, and you find yourself in front of two options: - Either sacrifice the property you liked or overlook the lie and complete the deal, and therefore you will pay a commission to this lying person.

Example4: Unit Reservation Fees

When you like a property that you have visited and want to buy it already, the real estate broker tells you that he wants you to pay him an amount of money to reserve the real estate unit for you so that no one else buys it and in order to prove your seriousness in buying before negotiating with the owner, and tells you that this amount will be deducted from the commission that you will pay him, but the biggest problem is that the amount you will pay There is nothing to prove that you paid to this real estate broker, and if any difference occurs during the negotiation process, it will not You can get your money back.

Example5: Participation of more than one broker in the transaction

When you want to buy a property, you may be surprised by the presence of more than one broker in the process of selling or buying the property, and in good faith you assume that you will pay one commission according to the text of the law in the Arab countries, it is known that whatever the number of brokers, the commission is one divided between them, and you find in the end that each real estate broker asks you for a commission for his account and you have two options, either cancel the deal after all this effort or increase the amount of the commission and buy the property that you liked.

Example6: Agreement with real estate guards

When there is a property that has an announcement that it is for sale or rent and you go to ask the property guard about the owner, the guard will tell you that the owner asks you to contact a specific broker to negotiate with him, brokers make agreements with real estate guards to do so in exchange for money, so you find yourself forced to deal with a specific real estate broker and have to pay him a commission that you do not need.

Example7: Overpriced

Usually, the owner of the property asks the real estate broker to display the property at its fair price and you find the owner of the property does not exaggerate the price of the property, but the real estate broker when you go and talk to him, you will find him asking for an exaggerated price in the property and convinces you that the owner is the one who asked for it, and you have two options, either he reduces the imaginary price and convinces you that he has played a major role to reduce the price through negotiation and therefore needs a larger commission or puts pressure on you and makes you buy the property at a price greater than its real value Thus, the more money you pay as a price for the property, the greater the amount you get It is as a commission.

Example8: Difference for the broker

Some real estate brokers agree with the owner of the property that they will not take a commission from him in exchange for any amount that will come in excess of the fair price of the property is their right, and therefore they offer the property at more than its real value and the potential buyer comes and the real estate broker convinces him that the property is worth this high amount and because of the customer's lack of experience falls prey to these unethical transactions.

Problem solution

Example1: Detailed descriptions: Listings should include precise information about the property's location, size, price, amenities, features, and ownership history.

Example2: Clear ownership status: Transparency regarding the property's ownership status helps users avoid scams involving fake or duplicate listings.

Example3: Direct communication with the owner: Users should be able to contact the owner directly through the website, which reduces the need for third party involvement and reduces the risk of miscommunication.

Example4: Attach high-quality photos of properties for sale or rent: High-quality photos can help users get a better idea of the property before visiting it.

Significance of the project:

1.Increased visibility: A website allows real estate agents and agencies to showcase their properties to a wider audience, increasing their visibility and reach.

2. Price prediction: The website allows the user to add information about the property available to him in an area, and the site predicts an appropriate price for that property based on the data used by the site.

3. Searching for a property: The user can search for a suitable property based on price and area

4.availability: A website provides potential buyers with the ability to browse properties at any time, even outside of regular business hours.

5.Credibility and professionalism: A well-designed website can help establish credibility and professionalism for real estate professionals, which can be important in building trust with potential clients.

6.Marketing and advertising: A website serves as a platform for marketing and advertising properties, allowing agents to reach a larger audience and attract potential buyers.

7.Information hub: A website can serve as an information hub for potential buyers, providing details about properties, neighbourhoods, market trends, and other relevant information

8.Lead generation: A website can be used to capture leads through contact forms, inquiries about properties, or newsletter sign-ups.

Project Objective:

- Make the customer deal with the owner immediately and not through an intermediary.
- Building an electronic market that facilitates selling, searching, and predicting prices in a safe manner.
- Accelerating the wheel of investment in Egypt.
- Reducing the cost to the customer.

Scope and limits:

1. Individual pages for each site containing details, maps, and multimedia. Each of the best real estate destinations in Egypt must have a dedicated page for it on the site or application. These pages should provide detailed information about the destination, including its location, price, and things to do. It should also include maps, photos and videos to help visitors plan their purchase or rental.

2. The site allows the visitor to view more than one property, prefer it, and purchase many properties, but it does not allow the property to be owned by more than one person.

3. The site does not support the visitor adding a property and does not allow him to delete properties

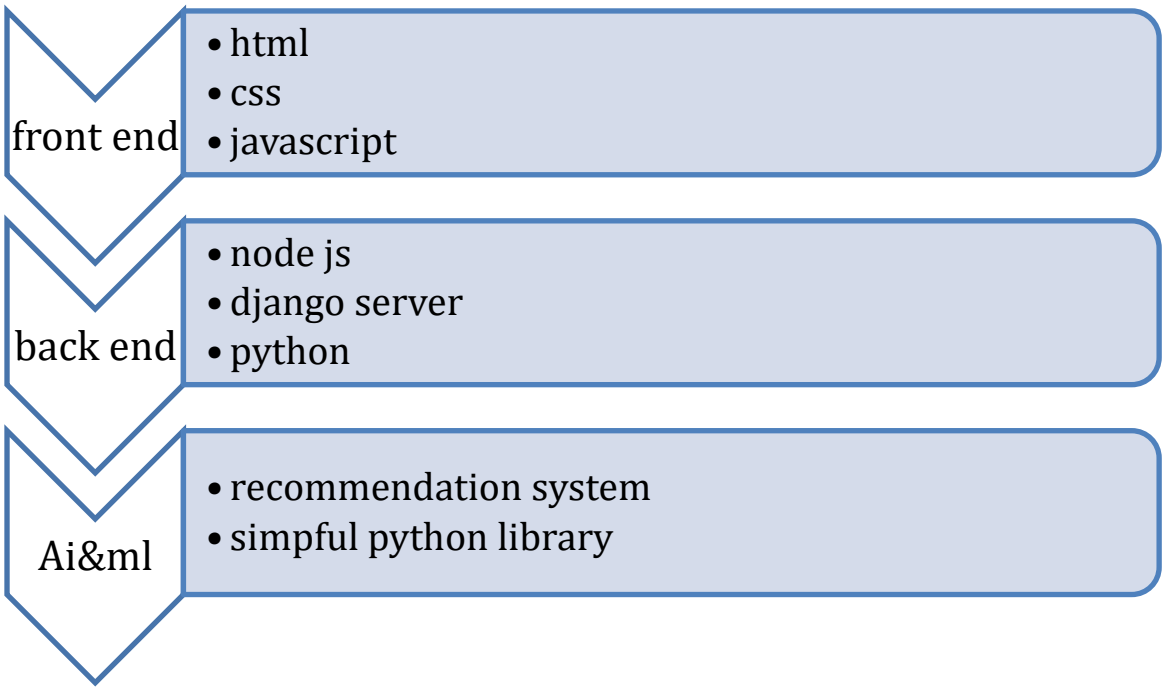
4. The site does not support the user's electronic registration

5. The website must also include a comprehensive guide to real estate in Egypt. Visitors should be able to search for properties by location, price, and amenities. They should also be able to book the property directly through the website

6. The project supports purchases from other countries outside Egypt or inside Egypt, and there are some reasons that may make the project support purchases from other countries outside Egypt and inside Egypt, and the reason is that the project aims primarily to provide all means of user comfort and requirements whether inside or outside Egypt. outside it. Another reason is that the project is designed to promote real estate available in Egypt or outside Egypt

7. Basic Recommendation Engine for Favorite Properties The basic recommendation engine can help visitors find favorite properties that interest them. The engine can use information about a visitor's interests and preferences to suggest preferred properties they are likely to enjoy.

Project Software Requirements:



Hardware Requirements:

Severe requirement

computers with 1GB RAM, 350MB hard disk space is required to install the software.

Client requirement

Computer with an internet connection and an internet browser is only required for the client to run the application.

Benefits of the Proposed System

- **No Property Dealer Required**

No property dealer is required because everything is well managed online.

- **Data Consistency**

The major benefit of our proposed system is Data consistency. Now, if we change data in one portion of the system, it can automatically change data in every related portion.

- **Data Security**

In our proposed system, the user has security because only an authorized person can enter in this system.

- **Data Accuracy**

Our proposed system is accurate because what you will post, will be checked by admin and it can also be editable.

- **Efficient System**

In our proposed system, the user can easily search, view, add and maintain their property in a quite efficient manner.

- **Removing Data Redundancy**

In our proposed system, we remove the data redundancy. Now, we get the information in a single place.

Project Expected Output:

The expected output of a real estate website would typically include:

- A clean and professional design with easy navigation

- High-quality images and detailed descriptions of properties

- Search functionality to filter properties by location, price, size, etc.

- Contact information for real estate agents or brokers

- Mortgage calculator or other financial tools

- Information about the local area, schools, amenities, etc.

- Testimonials or reviews from previous clients

- Blog or resources section with helpful articles and tips for buyers and sellers.

Report Outline:

The report layout for a real estate website typically includes the following elements:

1.Property details: This section provides information about the property, including its address, size, number of bedrooms and bathrooms, and any special features or amenities.

2.Photos: High-quality photos of the property are essential for showcasing its appearance and potential to potential buyers or renters.

3.Property description: A detailed description of the property, including its history, unique features, and any recent renovations or upgrades.

4.Pricing information: The report should include the asking price or rental rate for the property, as well as any additional fees or costs associated with the transaction.

5.Contact information: The report should provide contact details for the real estate agent or broker handling the property, as well as any other relevant parties involved in the transaction.

6.Virtual tour: Some real estate websites may include a virtual tour of the property to give potential buyers or renters a more immersive experience.

Chapter 2
RELATED EXISTING SYSTEMS

Introduction:

The real estate industry relies on a complex ecosystem of interrelated systems to function efficiently. These systems play a crucial role in facilitating transactions, managing properties, providing data and insights, and connecting stakeholders.

Existing Systems:

Here are some of the Existing systems used in real estate:

Example1: Multiple Listing Services (MLSs)

These online databases connect real estate agents and brokers, allowing them to share property listings and collaborate on transactions. Popular examples include Zillow, Trulia, and Realtor.com.

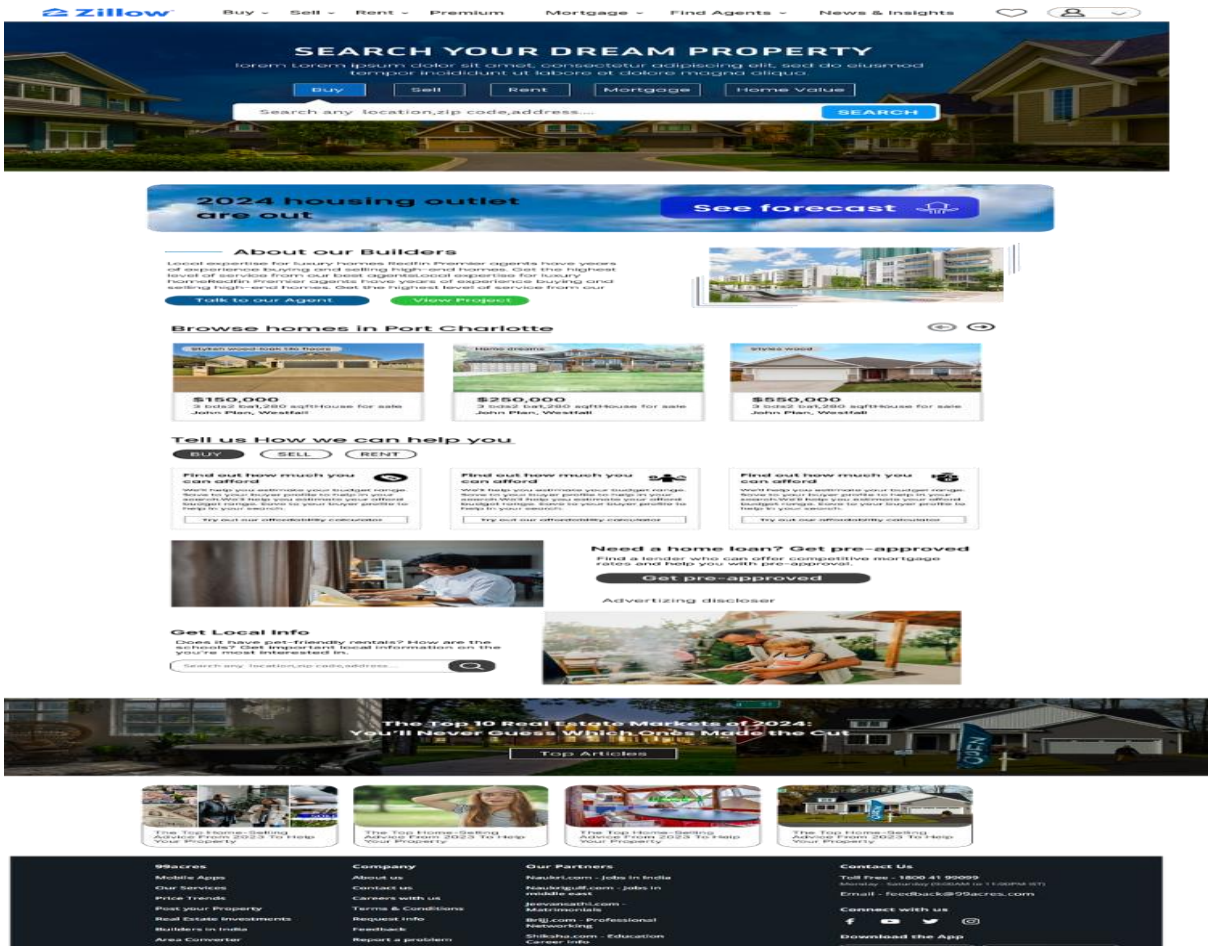
Benefits:

. Wider exposure for properties: Properties can be seen by a larger pool of potential buyers and sellers.

· Increased efficiency in finding buyers and sellers: Agents can easily see what other properties are available and match them with their clients' needs.

· Transparency in market data: Provides real-time data on market trends and prices.

Graphical User Interface (GUI) Design for zillow website



Figure(1)

advantages	disadvantages	solution
Massive real estate database	Data is not guaranteed	Improve data accuracy
Valuable tools and services	Anno ying ads	Offer ad-free options
User-friendly interface	Additional fees	Increase transparency about fees

Example2: Real Estate Management Systems (REMS)

These software applications manage the entire lifecycle of a property, from listing and tenant selection to rent collection and maintenance requests. Popular examples include AppFolio, Buildium, and Yardi Voyager.

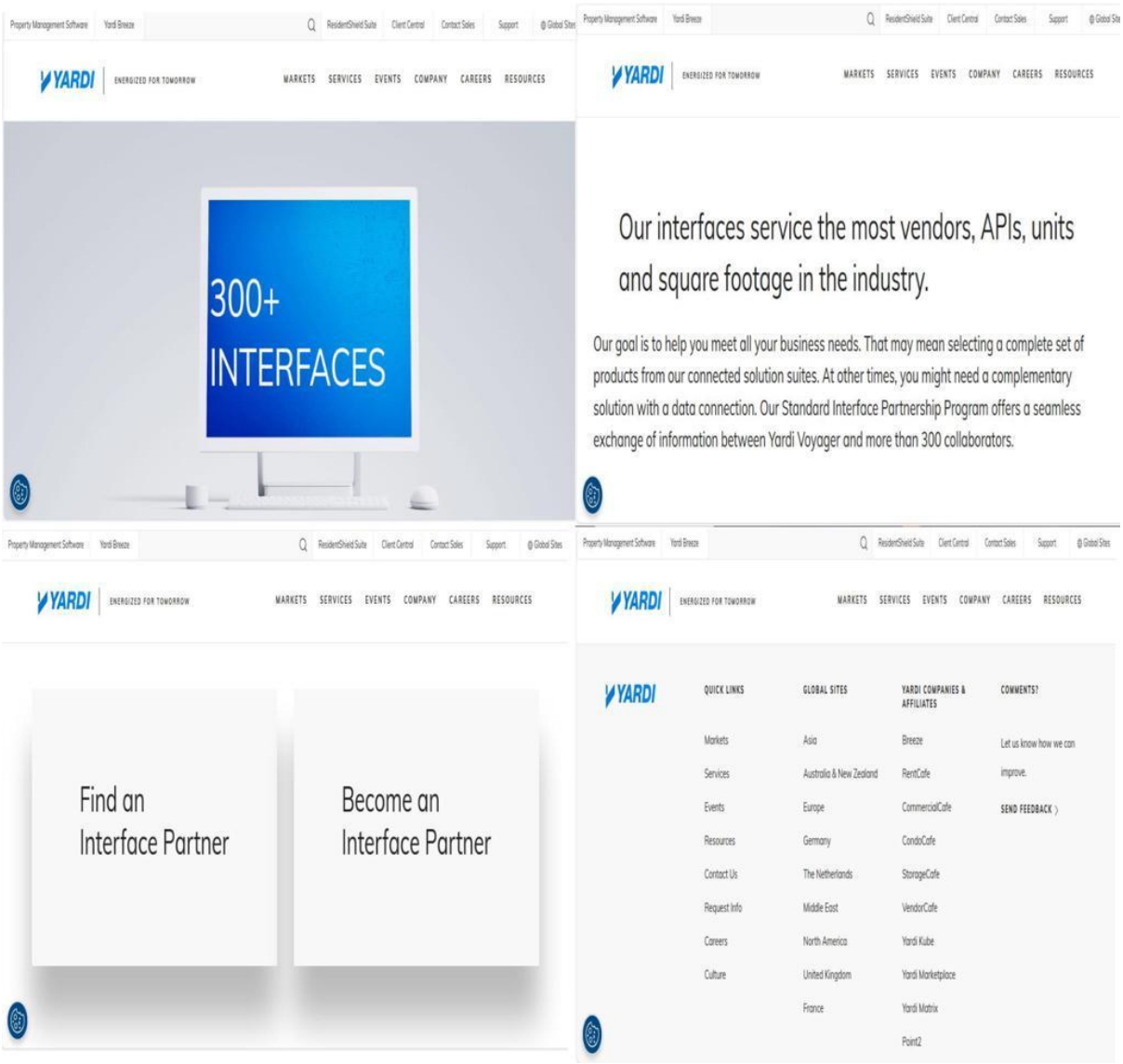
Benefits:

· Streamlining operations: Automates routine tasks and improves communication between landlords, tenants, and vendors.

· Improved data management: Provides access to real-time data on property performance and tenant activity.

· Reduced costs: Saves time and money by automating tasks and reducing administrative overhead.

Graphical User Interface (GUI) Design for Yardi website



Figure(2)

advantages	disadvantages	solution
Comprehensive property management software	Complexity	Increase platform flexibility
Scalability and customization	Limited flexibility	Improve data security
Data-driven insights	Cost	Develop more affordable tiers

Example3: Financial Technology (FinTech) Platforms

These platforms provide innovative solutions for financing real estate investments. Popular examples include Lending Home, Crowd Street, and Fundraise.

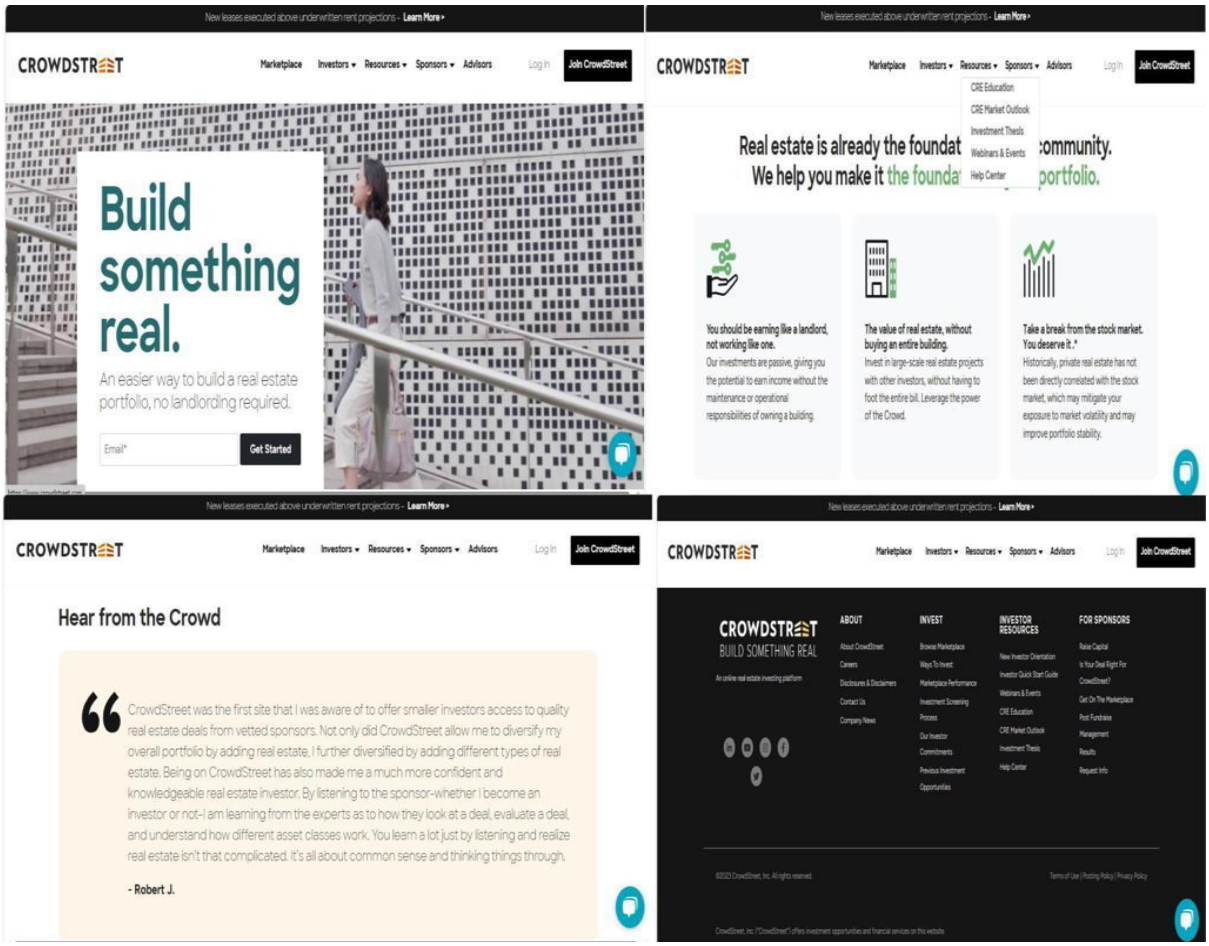
Benefits:

· Easier access to capital: Makes it easier for individuals to find funding for their real estate investments.

· New investment options: Offers alternative investment opportunities beyond traditional methods.

· Fractional ownership opportunities: Allows individuals to invest in a portion of a property, making real estate investing more accessible.

Graphical User Interface (GUI) Design for CrowdStreet website



Figure(3)

advantages	disadvantages	solution
Technology-driven platform	Minimum investment amounts	Offer lower investment options
Potential for high returns	Fees and expenses	Increase transparency around fees
Diversification options	Limited control and influence	Improve liquidity options

Example4: Property Data and Analytics Platforms

These platforms collect and analyze data on real estate markets, providing insights to investors. Popular examples include CoStar, Reonomy, and RealPage.

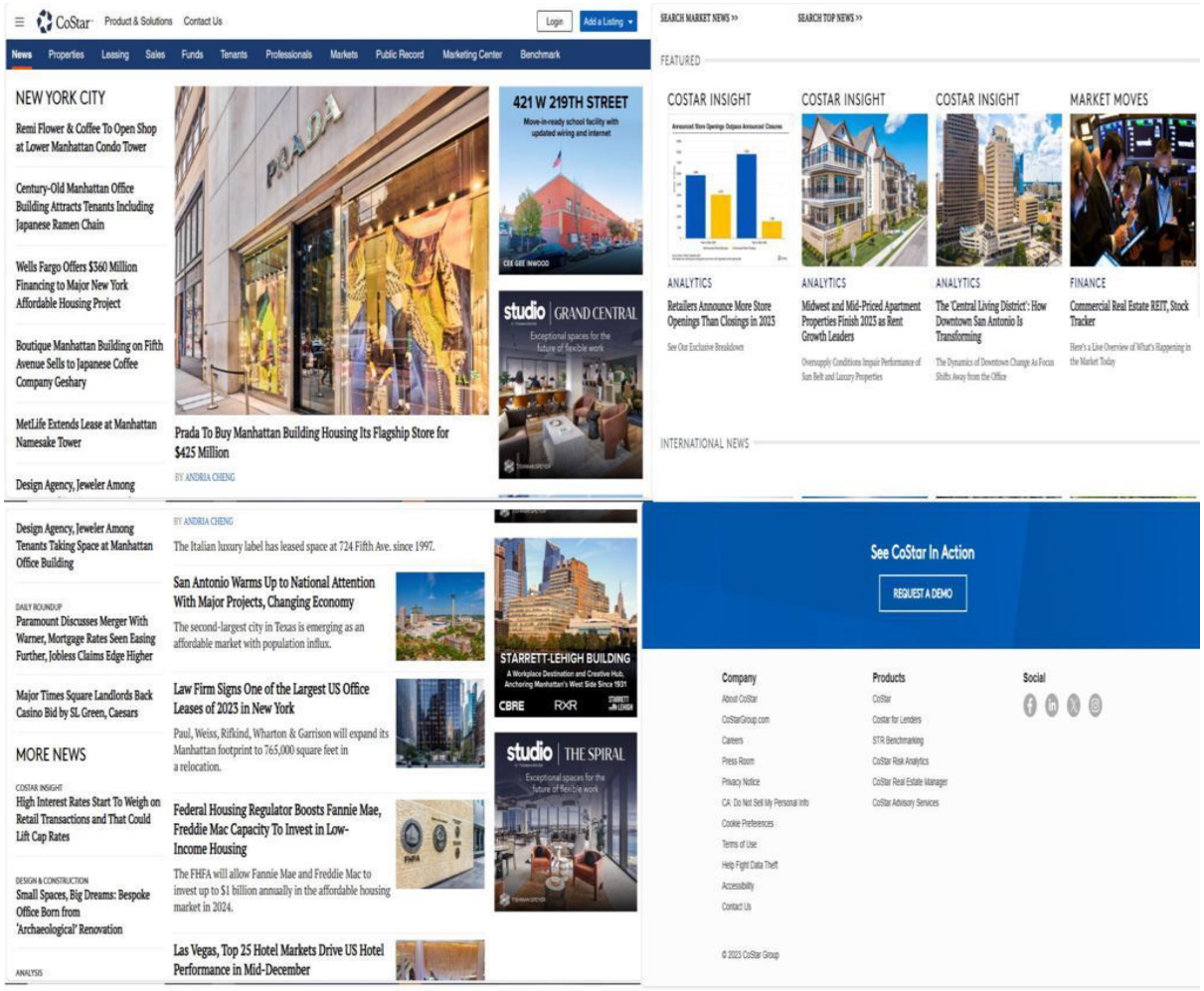
Benefits:

· Improved decision-making: Provides data-driven insights to help investors make informed decisions.

· Identification of investment opportunities: Helps investors identify potential investment opportunities based on market trends and data.

· Market trend analysis: Provides insight into market trends and potential risks.

Graphical User Interface (GUI) Design for CoStar website



Figure(4)

advantages	disadvantages	solution
Comprehensive commercial real estate data	Focus on large transactions	Expand focus to smaller markets
Powerful analytics and research tools	Cost	Improve data verification and validation
Market intelligence and news	Complexity	Develop tiered pricing models

Example5: Online Real Estate Platforms

These online platforms allow individuals to search for properties, connect with agents, and manage their real estate transactions. Popular examples include OpenDoor, Offerpad, and Redfin.

Benefits:

· Convenience and accessibility: Provides 24/7 access to property information and resources.

· Increased competition among sellers: Leads to potentially better prices for buyers.

· More information and resources for buyers: Provides buyers with more information and resources to make informed decisions.

Additional Systems:

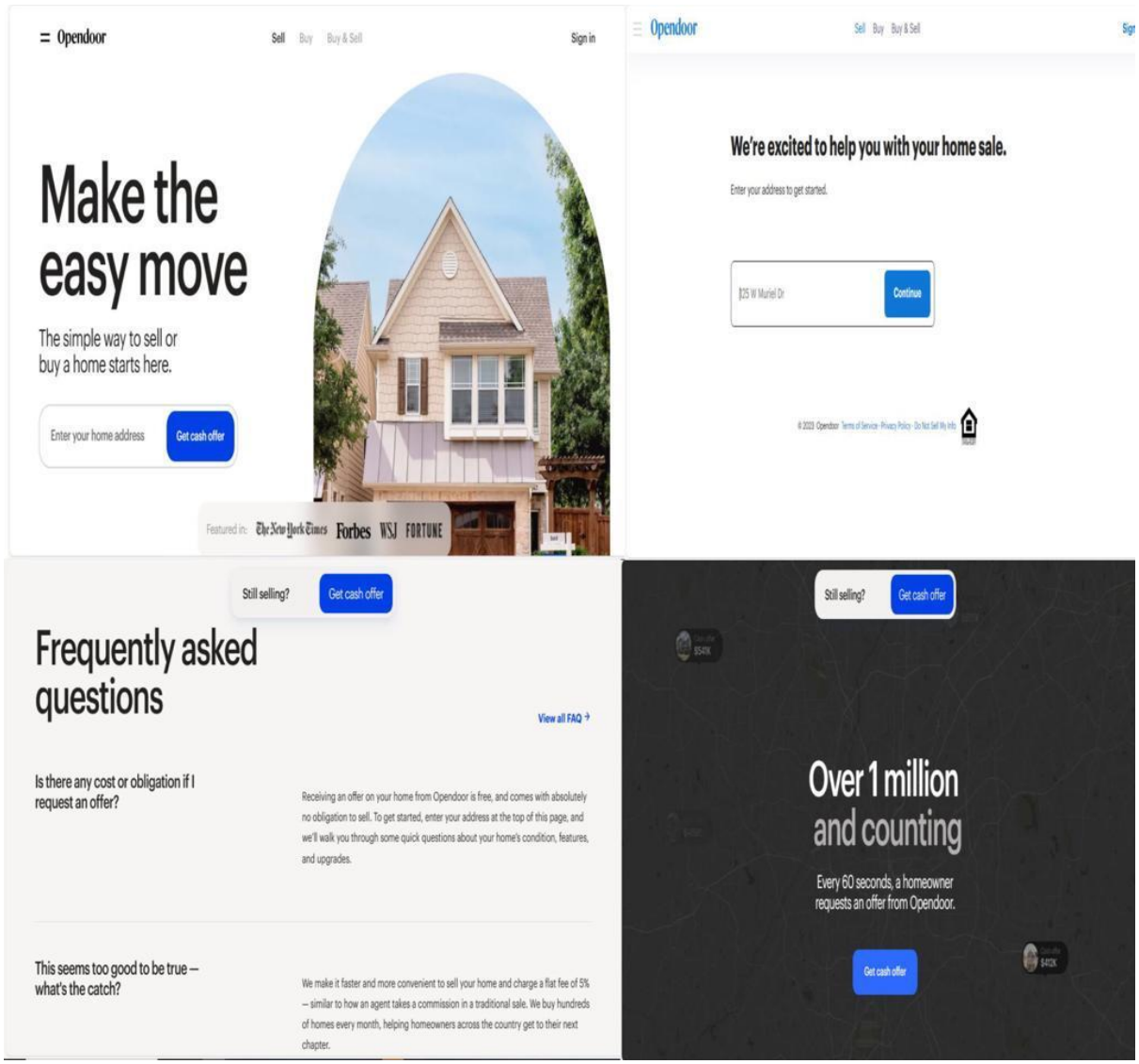
· Property Tax Management Systems: Streamline the process of paying property taxes and tracking assessments.

. Legal and Compliance Platforms: Help navigate complex real estate regulations and ensure legal compliance.

· Protech Solutions: Offer innovative technologies for various aspects of real estate, such as virtual tours and smart home integration.

The specific systems used in real estate can vary depending on the type of property, location, and individual needs. However, all of these systems play a critical role in facilitating transactions, managing properties, and providing information to stakeholders in the real estate industry.

Graphical User Interface (GUI) Design for OpenDoor website



Figure(5)

advantages	disadvantages	solution
Potential for higher net proceeds	Potential for hidden costs	Increase transparency and clarity around fees
Convenient and fast home selling process	Limited flexibility	Offer greater flexibility and customization
Competitive cash offers	Lower potential sale price	Address data privacy concerns

Overall Problems of Existing Systems:

While current real estate systems offer many benefits, they also face several general problems:

1. Lack of Transparency and Data Accessibility:

· Information about properties, market trends, and investment opportunities is often fragmented and difficult to find, especially for non-professionals.

· Data may be siloed within different systems, making it difficult to get a comprehensive picture of the market.

· Lack of transparency can lead to information asymmetry and potentially unfair practices.

2. Inefficiency and Manual Processes:

· Many tasks, such as paperwork, communication, and document management, still rely on manual processes.

· This can be time-consuming and prone to errors, leading to delays and inefficiencies.

· Manual processes also limit scalability and hinder the growth of businesses.

3. Limited Accessibility for Investors:

· Traditional real estate investment can be expensive and complex, often requiring significant capital and expertise.

. This excludes many individuals from participating in the real estate market, limiting their potential wealth creation opportunities.

· Lack of access to financing and fractional ownership options further restricts participation.

4. Fragmented and Siloed Systems:

· Existing systems often lack integration, creating a disjointed and inefficient experience for users.

· Information and data cannot easily flow between different systems, necessitating manual data entry and duplication of effort.

· This fragmentation makes it difficult to get a holistic view of the real estate landscape and hinders collaboration between different stakeholders.

5. Technological Limitations:

· Some existing systems rely on outdated technology, making them slow, inflexible, and difficult to use.

· This can limit the adoption of new technologies and hinder innovation in the real estate industry.

· Lack of user-friendly interfaces and mobile accessibility can also create barriers for some users.

6. Regulatory Challenges:

· Complying with complex and ever-changing regulations can be burdensome for real estate professionals and businesses.

· Lack of standardization and harmonization across different jurisdictions can further complicate matters.

· Regulatory challenges can stifle innovation and increase costs for businesses and consumers.

7. Cybersecurity Threats:

· Real estate transactions often involve sensitive personal and financial information, making them attractive targets for cyberattacks.

· Existing systems may not have adequate security measures in place to protect this data.

· Data breaches can have serious consequences, including financial losses, identity theft, and reputational damage.

8. Lack of Consumer Education:

· Many consumers lack the knowledge and understanding necessary to make informed decisions about their real estate transactions.

· This can lead to them being taken advantage of by unscrupulous actors or making costly mistakes.

. A lack of readily available educational resources and guidance can contribute to this problem.

Overall Solution Approach:

A one-stop solution for real estate aims to address the general problems of current systems by providing a comprehensive platform that integrates various aspects of the industry. It would ideally offer:

1. Centralized access to information:

· A single platform with property listings, market data, investment opportunities, regulations, and educational resources.

· Users can access all the information they need to make informed decisions in one place.

2. Streamlined processes:

· Automation and integration of tasks like paperwork, communication, and document management.

· Users can complete tasks efficiently and seamlessly, reducing time and effort.

3. Democratization of investment:

· Innovative solutions like fractional ownership and alternative financing make real estate investing more accessible and affordable.

· A wider range of individuals can participate in the market and benefit from potential wealth creation opportunities.

4. Enhanced transparency and data analysis:

· Real-time data and analytics tools providing insights into market trends, property performance, and investment risks.

· Users can make informed decisions based on data-driven insights.

5. Secure and compliant platform:

· Robust security measures and adherence to regulatory requirements.

· Protects user data and ensures fair and transparent transactions.

6. User-friendly interface and mobile accessibility:

· Intuitive and easy-to-use interface accessible on various devices.

· Users can access the platform and manage their real estate needs conveniently.

7. Collaborative environment:

· Features that facilitate communication and collaboration between different stakeholders, such as agents, investors, and property managers.

· Streamlines workflows, improves communication, and fosters a more efficient ecosystem.

8. Continuous innovation:

· Openness to adopting new technologies and integrating innovative solutions to adapt to changing market trends and user needs.

· Ensures the platform remains relevant and valuable to users in the long term.

·Benefits of a one-stop solution:

·Increased efficiency and productivity

·Reduced costs and time spent on administrative tasks

·Improved decision-making through data-driven insights

·Greater transparency and trust in the market

·Increased accessibility to real estate investment

·Enhanced security and compliance

·Improved user experience and satisfaction

·Challenges of implementing a one-stop solution:

·Integrating different systems and data sources

·Addressing regulatory requirements and compliance issues

·Gaining adoption from various stakeholders in the industry

·Ensuring user privacy and data security

·Continuously innovating and adapting to new technologies and market trends

Summary:

The real estate industry relies on various systems to facilitate transactions, manage properties, and provide information to stakeholders. Here are some of the major systems:

.Multiple Listing Services (MLSs)

·Real Estate Management Systems (REMS)

·Financial Technology (FinTech) Platforms

·Property Data and Analytics Platforms

·Online Real Estate Platforms

While current real estate systems offer many benefits, they also face several general problems:

·Lack of Transparency and Data Accessibility

.Inefficiency and Manual Processes

.Limited Accessibility for Investors

·Fragmented and Siloed Systems

.Technological Limitations

·Regulatory Challenges

.Cybersecurity Threats

·Lack of Consumer Education

While current real estate systems have many issues, there are many solutions

Centralized access to information

·Streamlined processes

.Democratization of investment

·Enhanced transparency and data analysis

.Secure and compliant platform

·User-friendly interface and mobile accessibility

.Collaborative environment

.Continuous innovation

Chapter 3
SYSTEM REQUIREMENTS ENGINEERING AND PLANNING

Introduction :

System Requirements Engineering (SRE) and Planning are crucial stages in building a successful real estate website. They lay the foundation for a user-friendly, efficient, and scalable platform that meets the needs of both buyers and sellers. Here's a breakdown of their roles and how they work together:

System Requirements Engineering (SRE):

· Identifying stakeholders: This includes buyers, sellers, real estate agents, administrators, and potential investors. Understanding their needs and expectations is crucial for defining the website's functionalities.

· Defining functional and non-functional requirements: Functional describe what the website should do (e.g., search for properties, manage listings, schedule viewings). Non-functional specify how it should perform (e.g., speed, security, scalability).

· Prioritizing requirements: Prioritize based on importance, urgency, and feasibility. This helps you focus on the essential features first.

Planning:

·Technology stack: Choose suitable technologies for backend and frontend development, considering performance, security, and scalability needs.

·Testing and quality assurance: Define testing procedures to ensure the website functions as intended. Early and frequent testing helps catch and fix bugs before launch.

·Deployment and maintenance: Plan for website deployment, including server configuration, data migration, and user onboarding. Define a maintenance strategy for ongoing updates and bug fixes.

Collaboration between SRE and Planning:

· SRE informs planning by providing detailed functional and non-functional requirements.

· Planning uses these requirements to define the project scope, development roadmap, and technology stack.

Feasibility Study:

Benefits of conducting a feasibility study:

Investing time and effort in a feasibility study can provide many benefits:

Reduces the risk of failure: By identifying potential challenges and barriers early, you can make informed decisions and avoid costly mistakes.

Improve decision making: The study provides valuable data and insights to help you make strategic decisions about the development and operation of our website.

Attracts investors and partners: A well-prepared feasibility study can be a valuable tool to attract investors or partners who believe in the potential of our project.

Provides a roadmap to success: The study serves as a roadmap for the development and growth of our website, helping us stay focused and on track.

A real estate feasibility study is an evaluation of the potential profitability of a real estate development project. The study takes into account various factors such as the cost of land, construction costs, market demand, zoning regulations, and financing options to determine whether the project is financially viable.

The feasibility study typically includes the following components:

1. Market analysis: This involves analyzing the local real estate market to determine demand for the type of property being developed. This can include demographic data, economic indicators, and information about competing properties.

2. Financial analysis: This involves determining the costs associated with the development, including land acquisition, construction, marketing, and ongoing maintenance. It also involves projecting potential revenue streams, including sales or leasing income, and analysing the return on investment.

3. Technical analysis: This involves evaluating the technical feasibility of the project, including reviewing zoning regulations, environmental impact, and other regulatory requirements.

4. Risk analysis: This involves identifying potential risks associated with the project, such as changes in market demand, construction delays, or unforeseen costs.

5. Site Analysis: A site analysis involves an evaluation of the physical attributes of the proposed property. This includes an assessment of the site's topography, accessibility, infrastructure, and environmental factors. The analysis helps to determine the suitability of the site for the proposed project and identify any potential issues that may need to be addressed.

The results of the feasibility study can help developers make informed decisions about whether to move forward with a real estate development project or not. It can also help investors assess the potential risks and rewards of investing in a particular project.

A real estate feasibility study is a critical tool for any real estate development project. It helps to determine the viability of the proposed project, identify potential risks and challenges, and provide recommendations for moving forward. Conducting a comprehensive feasibility study involves an in-depth analysis of various factors that could impact the project's success, including market analysis, site analysis, financial analysis, and legal analysis. With a well-conducted feasibility study, real estate investors, developers, and businesses can make informed decisions and maximise their returns on investment

Requirements Elicitation Techniques:

Choosing the right requirements elicitation techniques for your real estate location is crucial, and here are some effective techniques to consider:

1. User interviews and focus groups:

Individual Interviews: Conduct one-on-one interviews with potential users (buyers, sellers, agents) to understand their pain points, preferences, and expectations from a real estate website.

Focus groups: Gather a small group of users with similar characteristics to discuss their needs and brainstorm ideas about website features.

2. Use cases and scenarios:

Develop detailed use cases: Define specific scenarios that represent typical user interactions with the website. This helps define the necessary functions and data flows.

Storyboard: Create visual representations of user journeys through the website, highlighting key interactions and decision points.

3. Observation and competitor analysis:

Monitor user behaviour on existing real estate websites: Watch how users interact with popular platforms to identify best practices and potential improvements.

Competitor website analysis: Identify the features and functionality your competitors offer and evaluate their strengths and weaknesses.

Additional Resources:

Top 10 requirements elicitation techniques: **https://uxbooth.com/**

The complete guide to real estate website design and development: **https://www.examples.com/business/one-page-real-estate-website.html**

Targeted Users:

Identifying your target users is crucial to building a successful real estate website. Here are some key groups to consider:

1. Buyers:

First Time Home Buyers: Young professionals, couples or families looking for their first property.

Upgraded Buyers: Existing homeowners seeking to upgrade to a larger property or a different location. We will highlight features such as additional bedrooms, upgraded amenities, and educational areas.

Luxury Buyers: High net worth individuals or families looking for upscale properties.

Investors: Individuals or institutions looking for investment properties.

2. Sellers:

Individuals selling their own homes: Providing tools to estimate property values, create listings, and manage showings.

Real Estate Agents: Provide tools to manage listings, attract buyers, and communicate with clients.

Developers & Builders: New developments, virtual tours and financing options will be showcased.

3.Other:

Tenants: View rental listings and search functions**.**

Investors looking for commercial real estate: We will provide listings for office buildings, retail space, and industrial properties.

International Buyers: We will customise our website to fit specific languages and meet their needs.

Additional factors to determine target users:

Demographics: age, income, family size, location.

Lifestyle: needs, preferences and interests.

Technology provision: online behaviour, preferred communication channels.

Decision-making process: How they research and select properties.

There are several ways to target users, including:

Develop buyer personas: We will create detailed profiles of your ideal buyers and sellers to understand their motivations and pain points.

Using targeted advertising: Using online platforms to reach specific demographics and interests.

Build relationships with real estate agents and other relevant partners: We will expand user reach and gain valuable insights into specific market segments.

Functional Requirements Definition:

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its components. A function is nothing but inputs to the software system, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements in Software Engineering are also called Functional Specification.

Functional Requirements Specification:

1. User Registration and Login: - The website should allow users to register and create an account.

- Users should be able to log in using their email address and password.

2. Property Listings: - The website should allow real estate agents or property owners to list their properties for sale or rent.

- Each property listing should include details such as location, price, size, number of bedrooms and bathrooms, and photos.

3. Search Functionality:-Users should be able to search for properties based on criteria such as location, price range, property type, and amenities.

- The search results should be displayed in a user-friendly format with the option to filter and sort the listings.

4. Property Details: Users should be able to view detailed information about each property listing, including photos, floor plans, virtual tours, and contact information for the agent or owner.

5. Contact Form: The website should include a contact form that allows users to inquire about a specific property or request more information from the agent or owner.

6. **Agent/Owner Profiles:** Real estate agents and property owners should have the ability to create profiles that showcase their listings and contact information.

7.**Mobile Compatibility:** The website should be fully responsive and compatible with mobile devices for easy access on the go.

8.**Admin Panel:** An admin panel should be available for managing user accounts, property listings, inquiries, and other website content.

9.**Social Media Integration:** Users should have the option to share property listings on social media platforms such as Facebook, Twitter, and Instagram.

10.Email Notifications: The website may send email notifications to users regarding new listings matching their saved searches or updates on their inquiries.

11.Language Support: The website may support multiple languages for catering to an international audience.

Non-functional Requirements:

1. Performance: The website should load quickly and be responsive to user interactions.

2. Security: The website should have secure user authentication and data encryption to protect sensitive user information.

3. Usability: The website should have an intuitive and user-friendly interface that is easy to navigate and understand.

4. Compatibility: The website should be compatible with various web browsers and devices, including mobile phones and tablets.

5. Reliability: The website should be reliable and available for use at all times with minimal downtime for maintenance or updates.

Summary:

System Requirements Engineering and Planning (SREP) is a crucial phase in the development lifecycle of any system. It involves identifying, analyzing, and documenting the functional and non-functional requirements that the system must possess to meet the needs of its stakeholders. This process ensures the built system is efficient, effective, and meets the intended purpose.

Key aspects of SREP:

Feasibility study: Assessing the technical and economic feasibility of implementing the requirements within the given constraints.

Targeted Users: refers to a specific group of individuals or organizations who are most likely to be interested in or affected by a particular product, service, message, or initiative.

Requirements elicitation: Gathering and understanding the needs and expectations of stakeholders through various techniques like interviews, surveys, and workshops.

Requirements analysis: Analyzing the gathered requirements for completeness, consistency, feasibility, and prioritization.

Requirements documentation: Clearly documenting the requirements in a structured and unambiguous way, often using a Functional Requirements Specification (FRS) and other relevant documents.

Functional Requirements: A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its components. A function is nothing but inputs to the software system, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements in Software Engineering are also called Functional Specification.

Non-Functional Requirements: qualities or constraints that a system must possess, beyond its specific functionalities. NFRs describe how the system should work, rather than what it should do. They are as important as functional requirements for ensuring a successful and user-friendly system.

Chapter 4
SYSTEM DESIGN

Introduction:

System design is the process of defining and planning the architecture, components, modules, interfaces, and interactions of a complex software or hardware system. It involves making decisions about how different parts of a system will work together to achieve the desired functionality, performance, scalability, reliability, and maintainability. The goal of system design is to create a blueprint or roadmap for building a system that meets the requirements and objectives of a project. This includes breaking down the system into smaller subsystems, modules, or components, and determining how they will communicate and collaborate to accomplish the overall goals. System design takes into consideration various technical and non-technical aspects.

Context diagram:

Context Diagram is the highest level of data flow diagrams (DFD). A context diagram shows an overview of the system as a whole, and shows the external entities that interact with it.

Basic elements of a context diagram:

System:

The entire system is represented as a circuit.

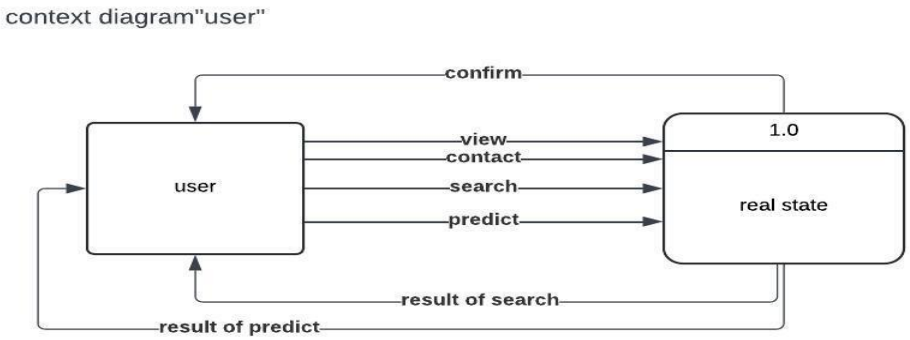
External Data Flows:

Represents data that enters or exits the system.

External Entities:

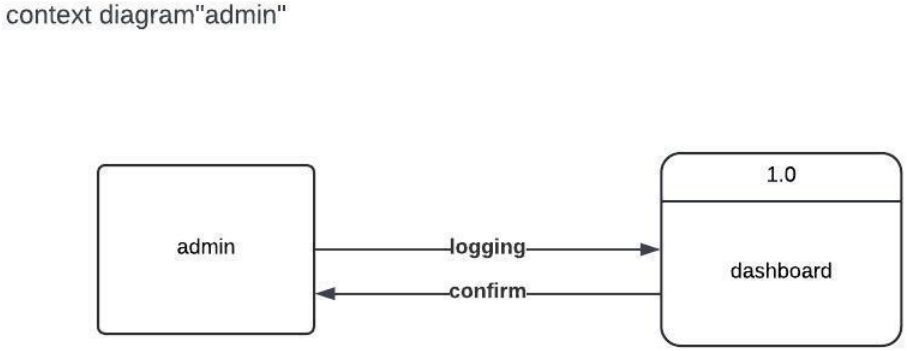
Represents the people, systems, or devices that interact with the system.

Context diagram “user”



Figure(6)

Context diagram “admin”



Figure(7)

Data flow diagram:

It is a graphical representation of the flow of data within a system or process. Explains how data moves between different processes, data sources, and storage destinations.

It is used to better understand, analyze and design systems.

The basic elements of a data flow diagram:

1-Operations:

Represents procedures that transform or manipulate data.

2-Data Flows:

Represents the movement of data between processes or entities.

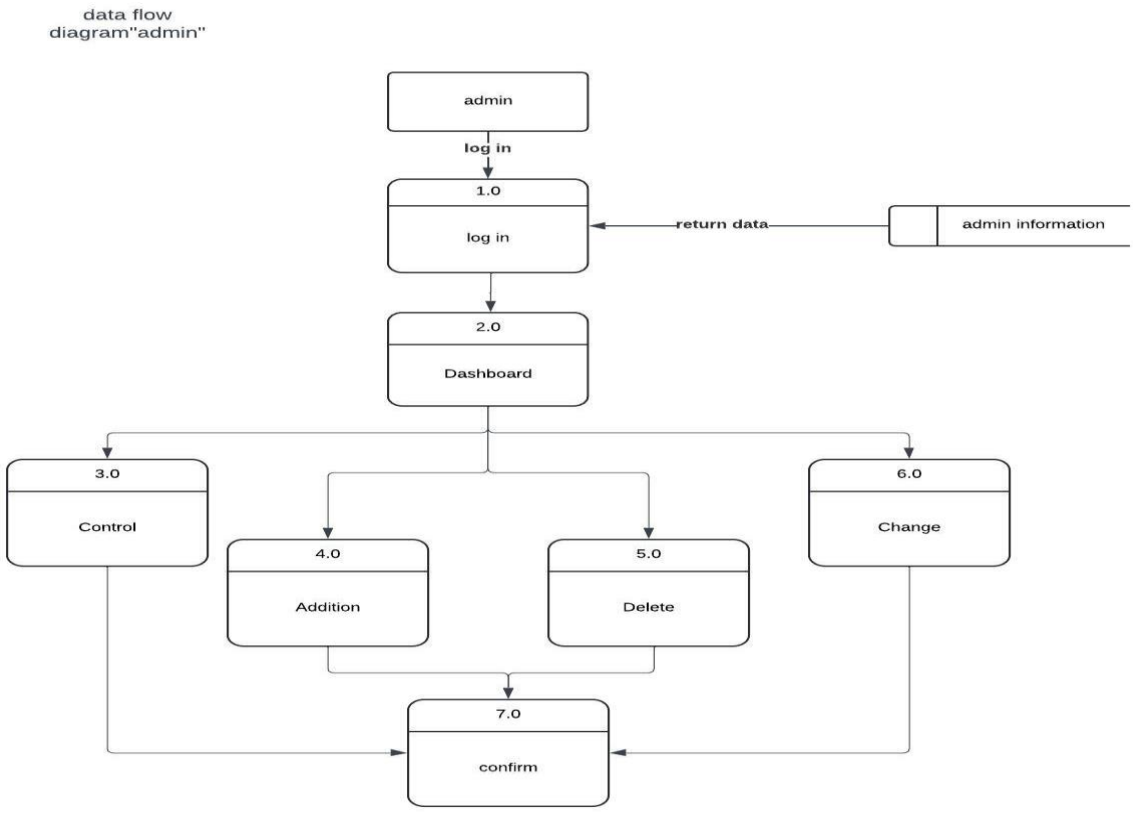
3-Data Sources and Sinks:

Represents external entities that generate or receive data.

4-Data Stores:

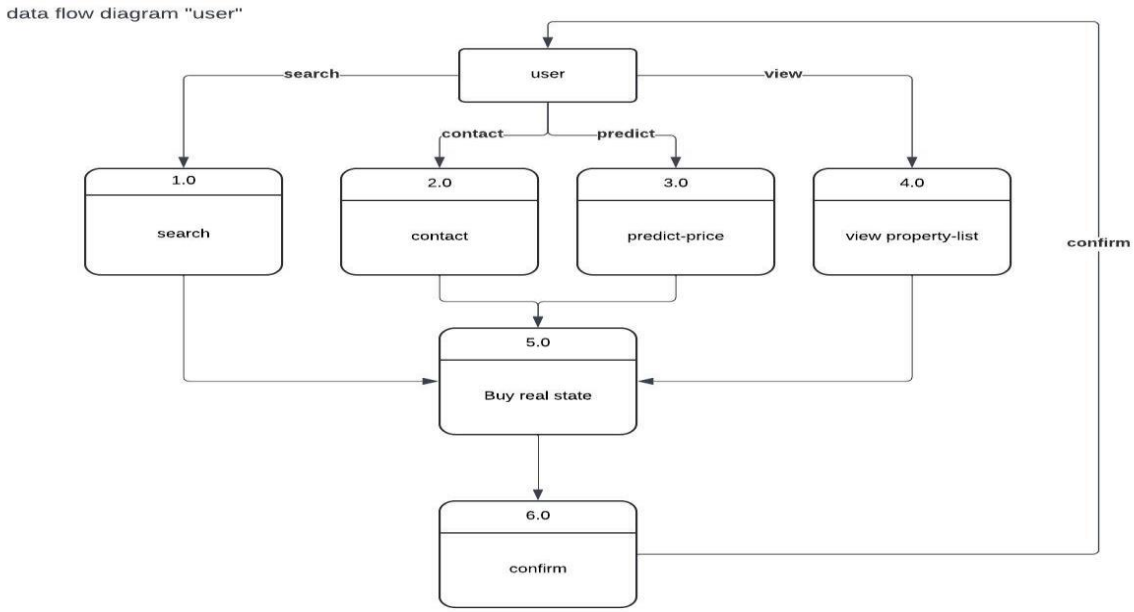
Represents where data is stored.

Data flow diagram “admin”



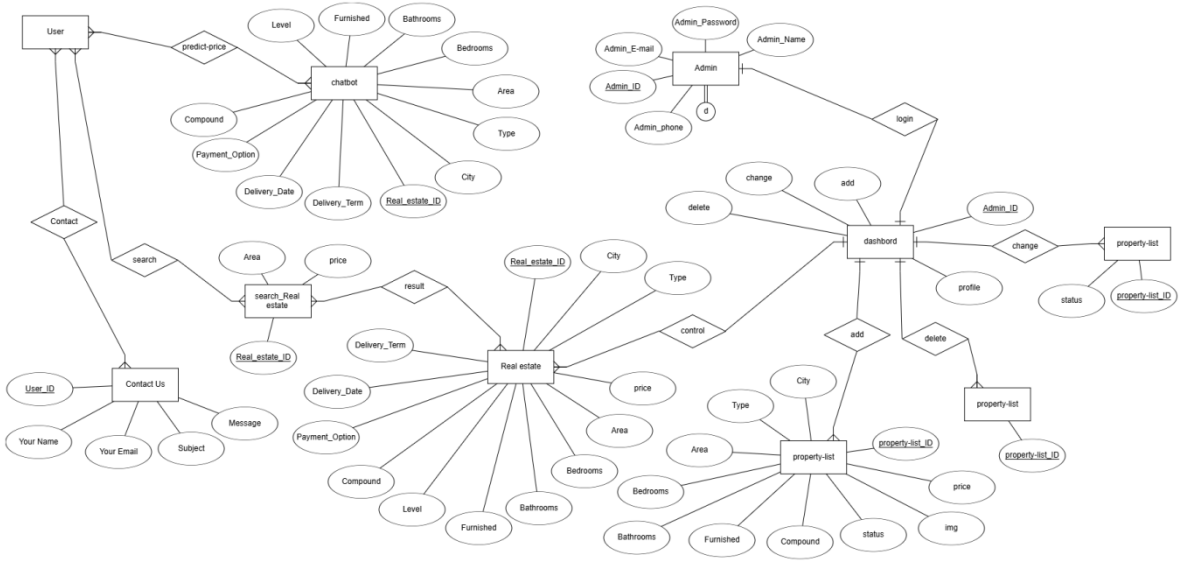
Figure(8)

Data flow diagram “user”



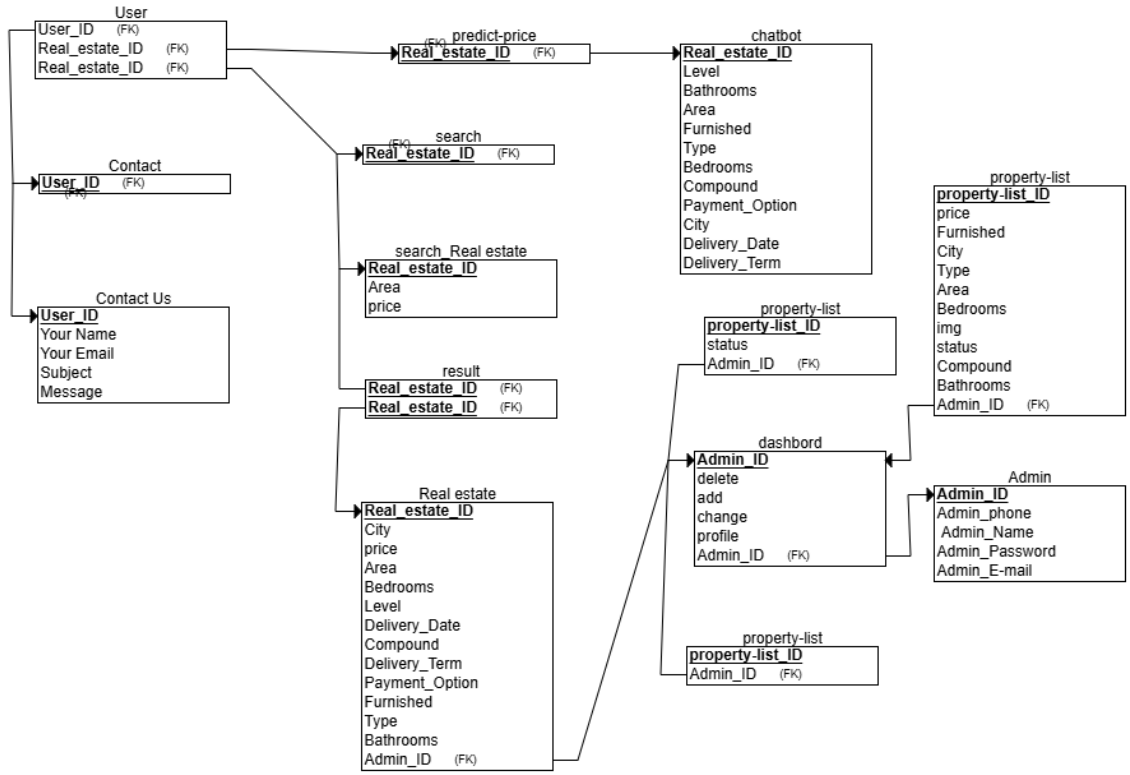
Figure(9)

Entity Relationship Diagram (ERD)



Figure(10)

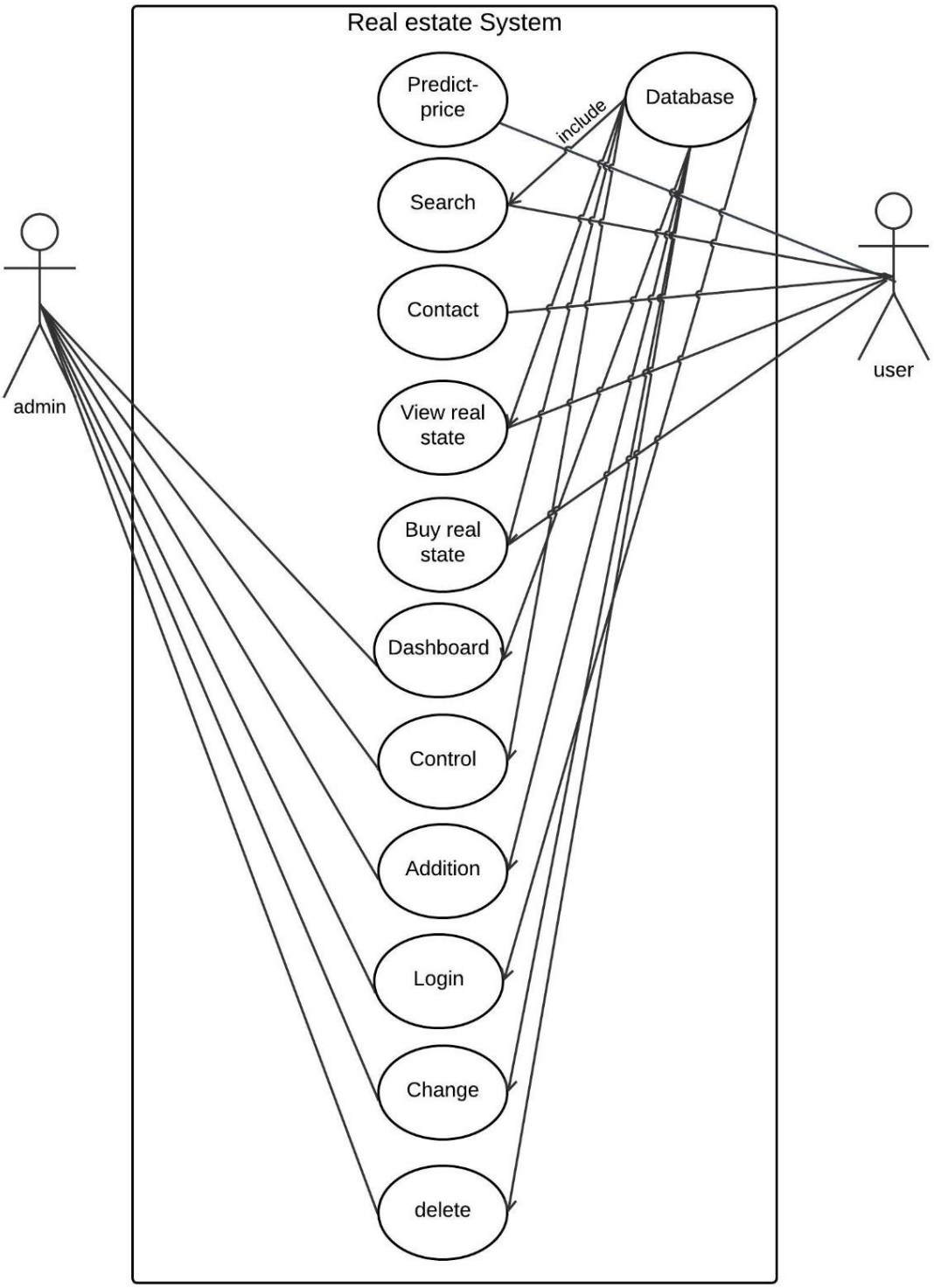
schema



Figure(11)

UML Use Case Diagram

UML Use Case Diagram is a type of diagram used in systems analysis and design. It aims to describe how users interact with the system and the activities they perform. The UML Use Case Diagram consists of the main elements: Actor, Use Case, and the relationships between them.



Figure(12)

UML Activity Diagram

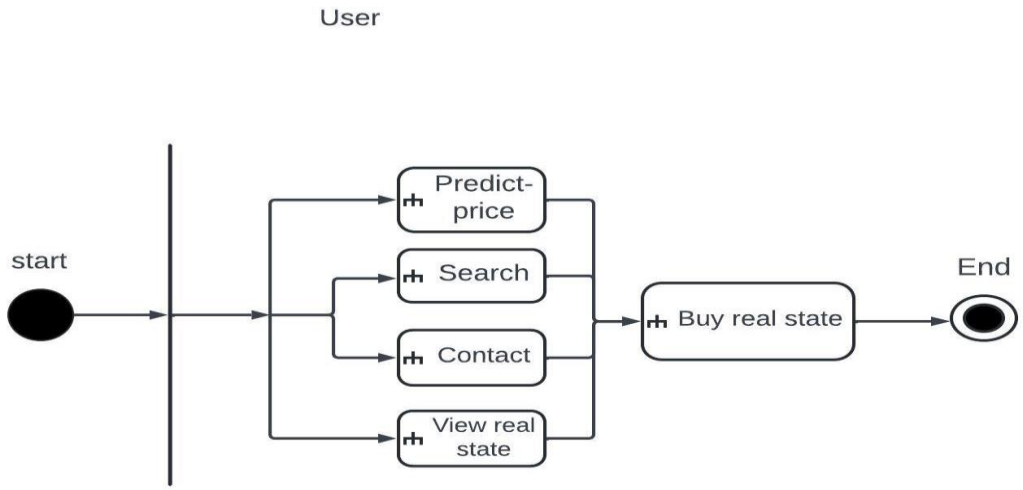
UML Activity Diagram is a type of diagram that is used to visualize a series of activities or processes in a particular system. This type of diagram aims to simplify and explain the workflow in a system.

The UML Activity Diagram consists of several main components, including:

1. Activities. 2. Transitions.
3. Conditions. 4. Start and End Nodes.

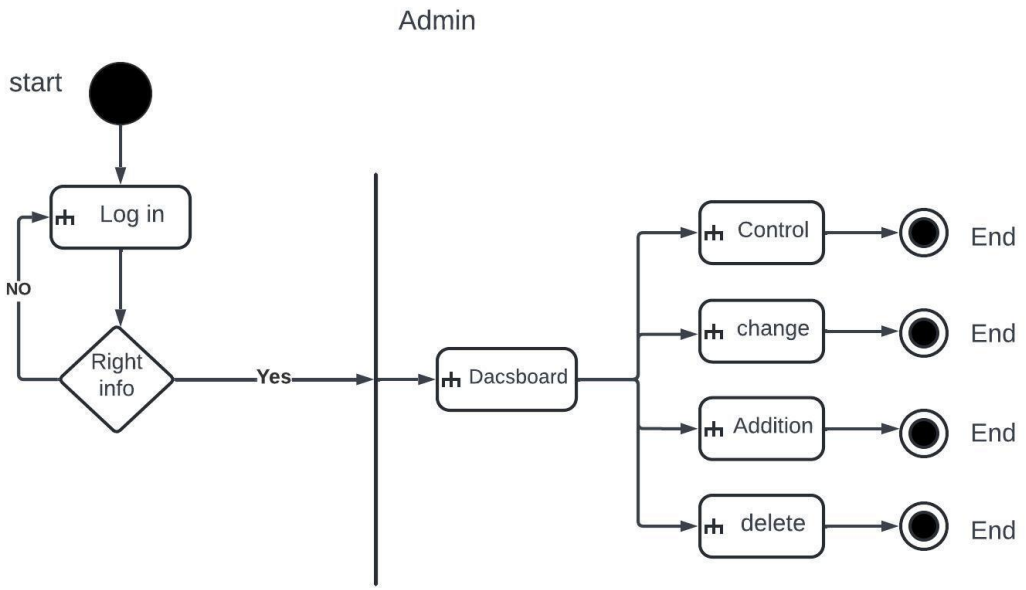
If you have a complex system and need to understand how the system works, using UML Activity Diagrams may be a good way to simplify this information and show it in an easy-to-understand format.

UML Activity Diagram "user"



Figure(13)

UML Activity Diagram "admin"

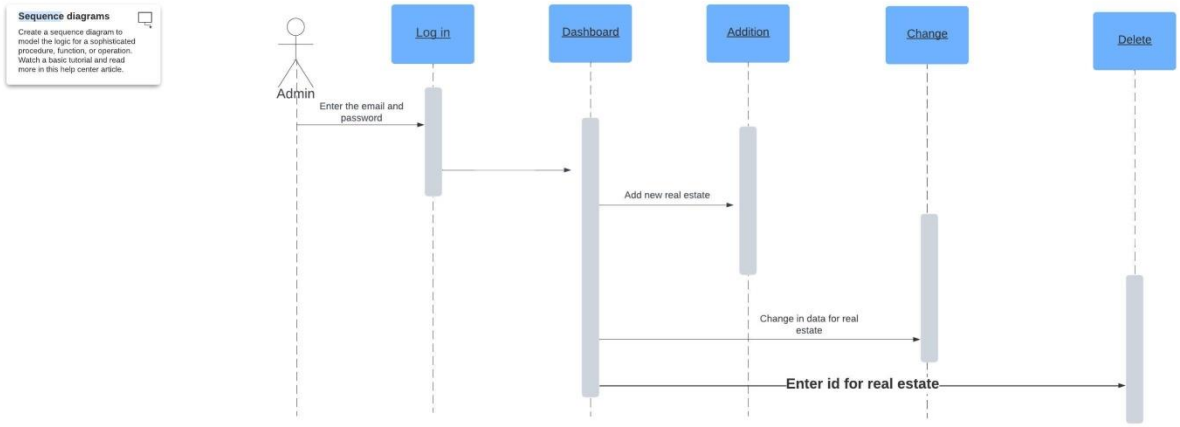


Figure(14)

UML Sequence Diagram

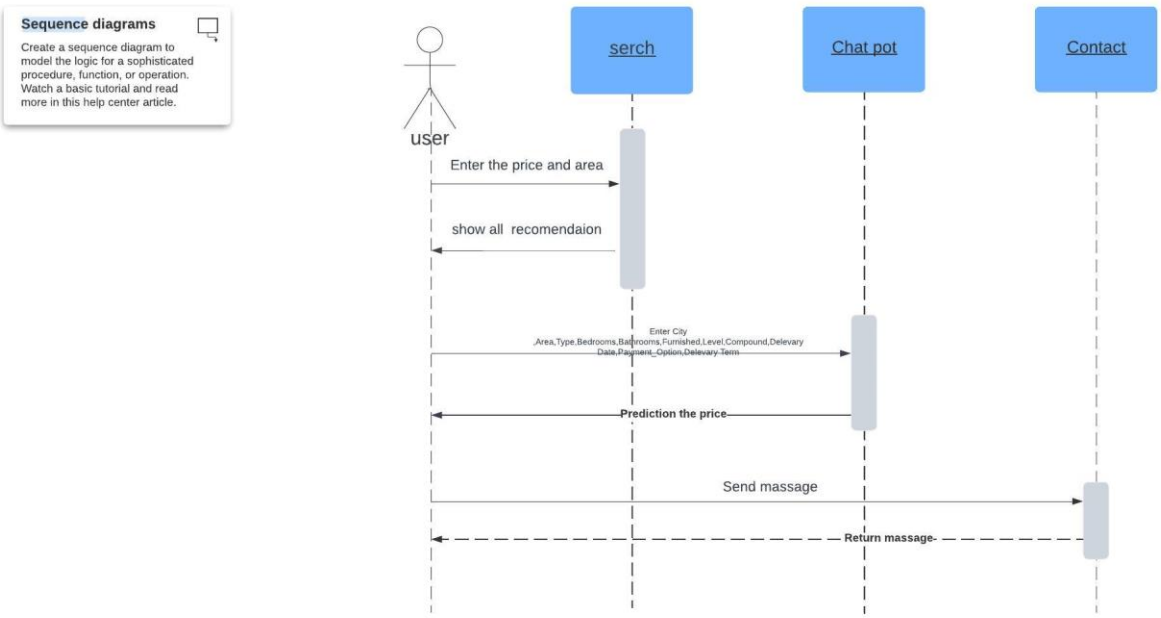
System analysis and design to clarify how the client and the rest of the various systems interact with each other during the implementation of operations in the system.

UML Sequence Diagram"admin"



Figure(15)

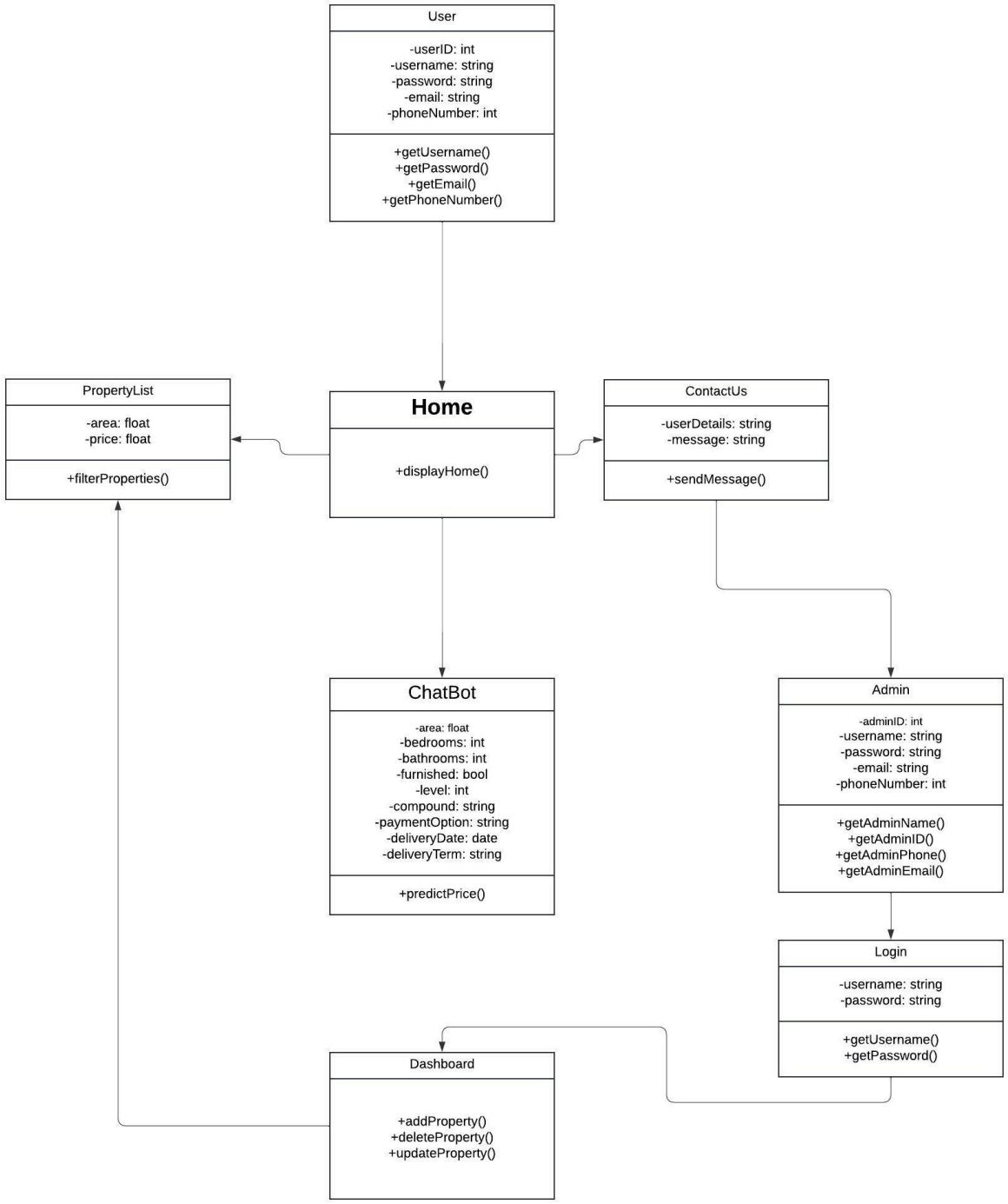
UML Sequence Diagram "user"



Figure(16)

UML Class Diagram

Here we see the relationship between all the classes and how the project proceeds for the admin and the customer.



Figure(17)

Summary

Context diagram:

is a graphic design that clarifies the interfaces and boundaries of the project or process at hand. It not only shows the process or project in its context, it also shows the project's interactions with other systems and users, a context diagram is "is the highest level view of a system . . . system as a whole and its inputs and outputs , a context diagram "shows the interactions between a system and other actors with which the system is designed to interface. System context diagrams can be helpful in understanding the context which the system will be part of."

Data flow diagram:

is a graphical or visual representation using a standardised set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM).

Entity Relationship Diagram (ERD):

An ERD visualizes the relationships between entities like people, things, or concepts in a database. An ERD will also often visualize the attributes of these entities.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram can illustrate the logical structure of databases.

UML Use Case Diagram:

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system.
Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

UML Activity Diagram:

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modeling how a collection of use cases coordinate to represent business workflows

UML Sequence Diagram:

are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focused and they show the order of the interaction visually by using the vertical axis of the diagram to represent time, what messages are sent and when.

UML Class Diagram:

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints.

Chapter 5
SYSTEM IMPLEMENTATION

Introduction

In this chapter, we will talk about the data base used on the site and the way we choose, classify and arrange it so that it is appropriate and serves the purpose required to implement the idea of the site. We will also talk about the user interface and the dynamics of the site.

Database Implementation

We initially encountered several challenges in gathering data that aligned with our project idea. Therefore, we began by collecting real estate information from the physical market, conducting research in the real estate sector. However, we faced the issue of not being able to gather sufficient data to use with our algorithms. We managed to collect data for fewer than 1,000 properties, which was inadequate for our needs. Consequently, we turned to web-based data sources, specifically Kaggel. Despite the difficulty in obtaining real estate data for all governorates of Egypt, we eventually found a database that met our project requirements. In the following, we will explain what we did after finding the suitable data.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Real_estate_ID	City	Type	Area	Bedrooms	Bathrooms	Furnished	Level	Compound	Payment_Option	Delivery_Date	Delivery_Term	Price
2	1	Nasr City	Duplex	400	3	3	No	7	Unknown	Cash	Ready to move	Finished	4000000
3	2	Camp Caesar	Apartment	160	3	3	No	10	Unknown	Cash	Ready to move	Finished	4000000
4	3	Smoha	Apartment	165	3	2	No	1	Unknown	Cash	Ready to move	Finished	2250000
5	4	Nasr City	Apartment	230	3	2	No	10	Unknown	Cash	Ready to move	Finished	1900000
6	5	New Cairo - El Tagamoa	Apartment	160	2	3	No	Ground	Eastown	Cash	Ready to move	Semi Finished	5800000
7	6	New Cairo - El Tagamoa	Apartment	222	4	3	No	1	Beit Al Watan	Cash	2024	Semi Finished	1844900
8	7	New Cairo - El Tagamoa	Duplex	290	5	5	No	Highest	Jayd	Cash	Ready to move	Finished	3900000
9	8	Sheikh Zayed	Apartment	144	2	2	No	1	Zayed 2000	Cash	Ready to move	Finished	1650000
10	9	New Cairo - El Tagamoa	Apartment	200	3	3	Unknown	Ground	Unknown	Cash	Unknown	Semi Finished	1560000
11	10	New Cairo - El Tagamoa	Apartment	146	3	2	Unknown	3	Unknown	Cash	2024	Semi Finished	992800
12	11	New Cairo - El Tagamoa	Apartment	153	4	3	No	1	Beit Al Watan	Cash	2024	Semi Finished	309825
13	12	Shorouk City	Apartment	165	3	2	No	2	Unknown	Cash	Ready to move	Finished	1150000
14	13	Smoha	Apartment	94	2	1	No	2	Unknown	Cash	Ready to move	Finished	550000
15	14	Sidi Beshr	Apartment	140	3	2	Yes	10	Unknown	Cash	Ready to move	Finished	2600000
16	15	Gesr Al Suez	Apartment	130	2	1	No	3	Unknown	Cash	Unknown	Finished	800000
17	16	New Cairo - El Tagamoa	Apartment	167	3	3	Unknown	3	90 Avenue	Cash	Unknown	Finished	480000
18	17	Mokattam	Penthouse	229	4	2	No	4	Unknown	Cash	Ready to move	Semi Finished	975000
19	18	New Capital City	Apartment	71	1	1	No	1	Armonia	Cash	Unknown	Semi Finished	143000
20	19	Nasr City	Apartment	220	3	2	No	9	Unknown	Cash	Ready to move	Core & Shell	1600000
21	20	New Damietta	Duplex	165	3	2	Unknown	Ground	Unknown	Cash	Unknown	Semi Finished	1100000
22	21	New Cairo - El Tagamoa	Apartment	184	3	3	Unknown	1	Palm Hills New Cairo	Cash	Unknown	Unknown	820000
23	22	New Cairo - El Tagamoa	Apartment	178	3	3	No	2	La Mirada	Cash	Ready to move	Finished	2350000

Figure(18)

Our primary dataset comprises 12 columns and 26,693 rows. We augmented this dataset by adding an additional column, the Real-estate-ID, to serve as the unique identifier to be utilized in the Entity-Relationship Diagram (ERD). However, this column is not included in the training of our model.

As we mentioned, the data consists of 12 columns, which we will now classify:

As the first column includes the city, we have made a comprehensive inventory of different cities within Egypt to make it easier for the user to find what he wants.

The second column includes the type of property, and we have collected some types, as these types are:

- Duplex
- Apartment
- Penthouse
- Studio
- Chalet
- Standalone Villa
- Twin house
- Town House

which makes it easier for the user to find his request.

In the third column:

This column includes the area, as we have made a comprehensive inventory of all the areas of the different properties and different types so that we have a large group of different properties and types of different sizes.

In the fourth and fifth column:

They include the number of rooms and bathrooms, and they vary according to the area and type of property. This data is considered complete of all requirements.

In the sixth column:

It includes the completion level, as it informs the user whether this property is complete and ready for delivery or not

In the seventh column:

It includes the level, urging us to know this property, whether it is the ground level or the upper level

In the eighth column:

It includes a compound so that the user knows where he will live

In the ninth column:

Includes the payment method to inform the user of the available methods of payment

In the tenth column:

It includes the delivery date, as it informs the user whether it is ready for delivery or not

In column eleven:

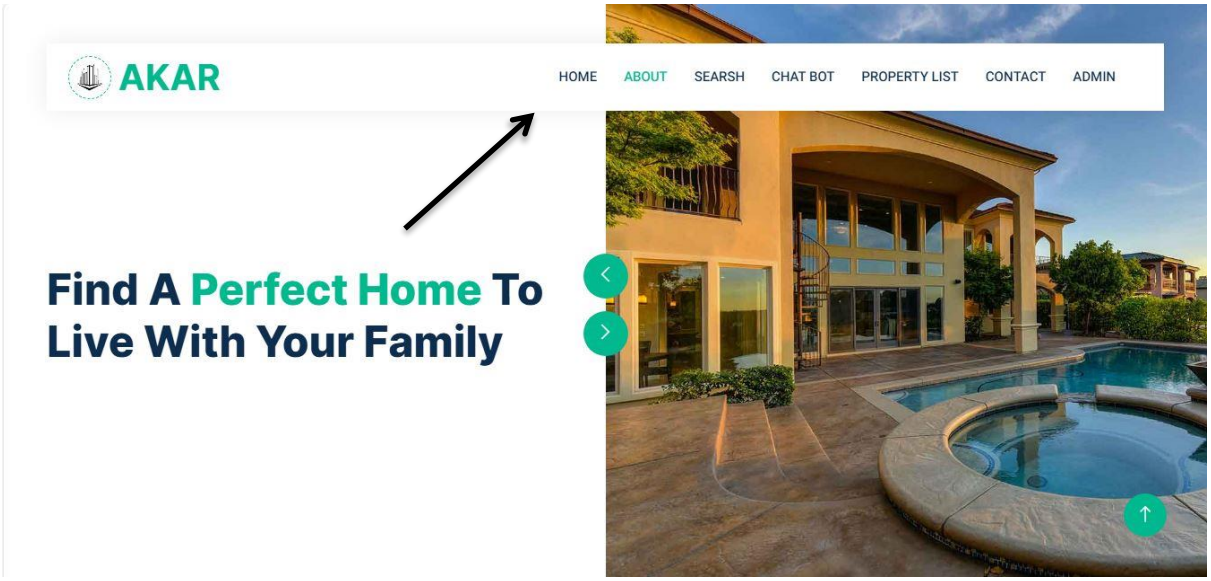
The type of finish includes whether it is fully finished or moderately finished

And in the last column:

It includes the price of the property, as each area differs in price and the user must find what suits him

Graphical User Interface Implementation

Web application



Figure(19)

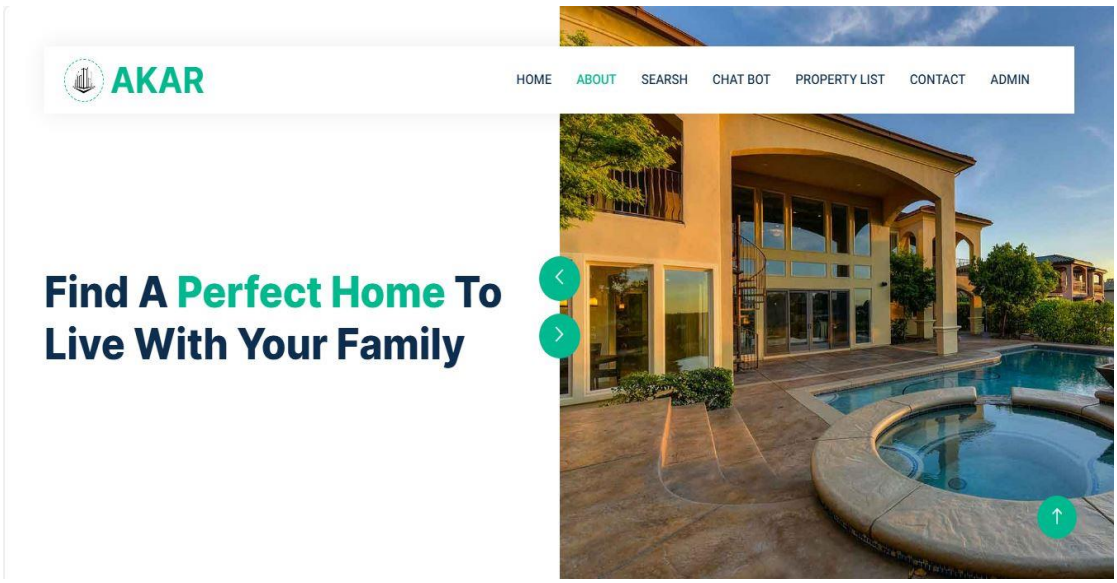
NAV BAR

At the beginning of the page, there is a NAV BAR that I use to go to all pages, and it is fixed on all pages for easy navigation between pages In this navigation bar there are many pages that, when you click on them, take you directly to their respective page. These pages are:

- Home
- About
- Search
- Chat Bot
- Property List
- Contact
- Admin

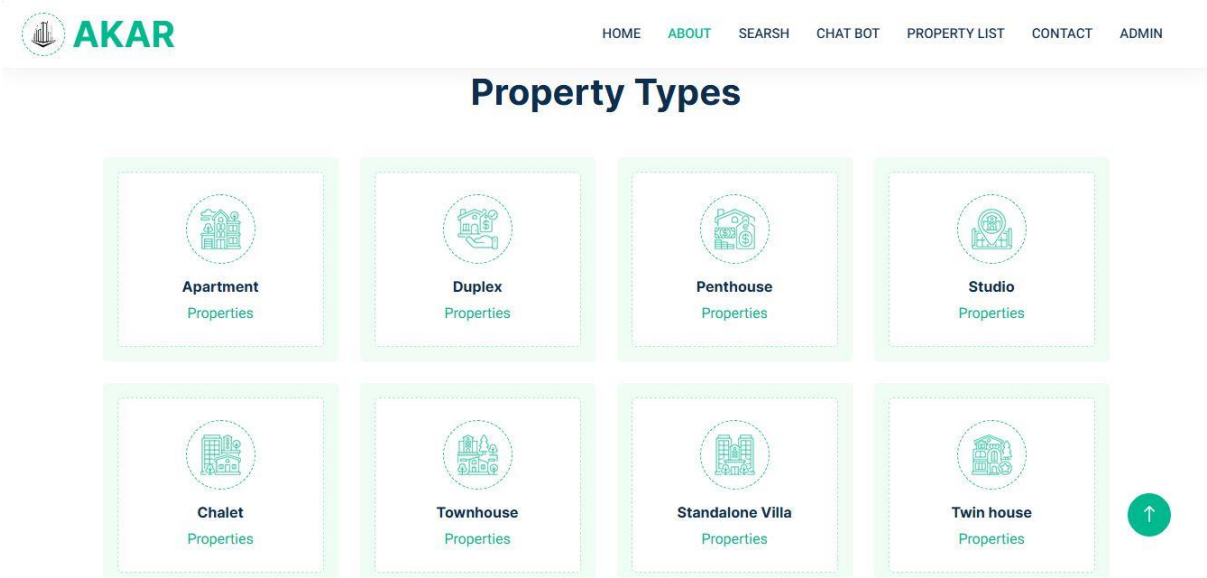
In Home page

The home page displays a set of features to let the user know what he will encounter on the site.



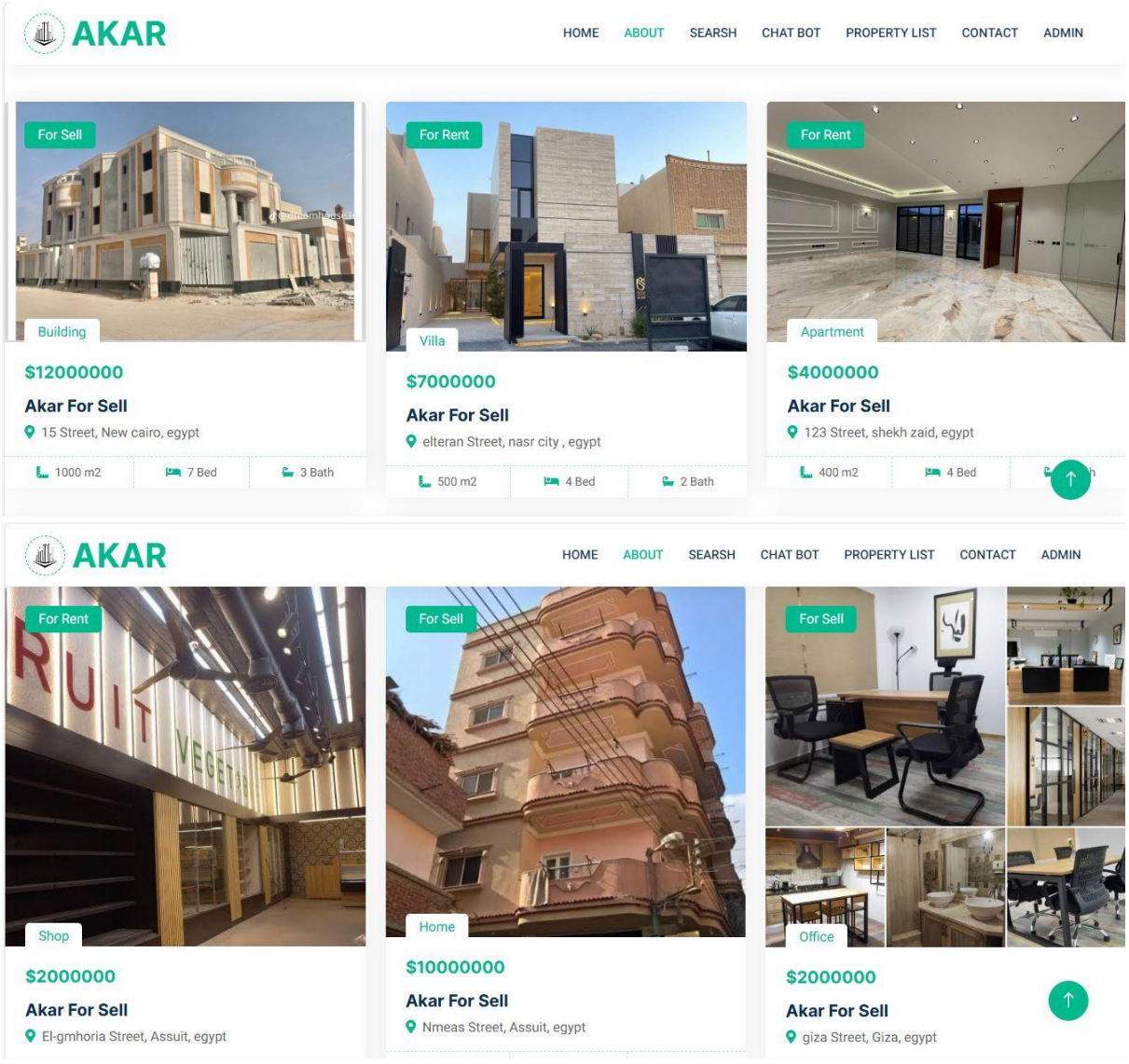
Figure(20)

After that, the interface that he will encounter at the moment of entry will be presented with a group of the types available with us.



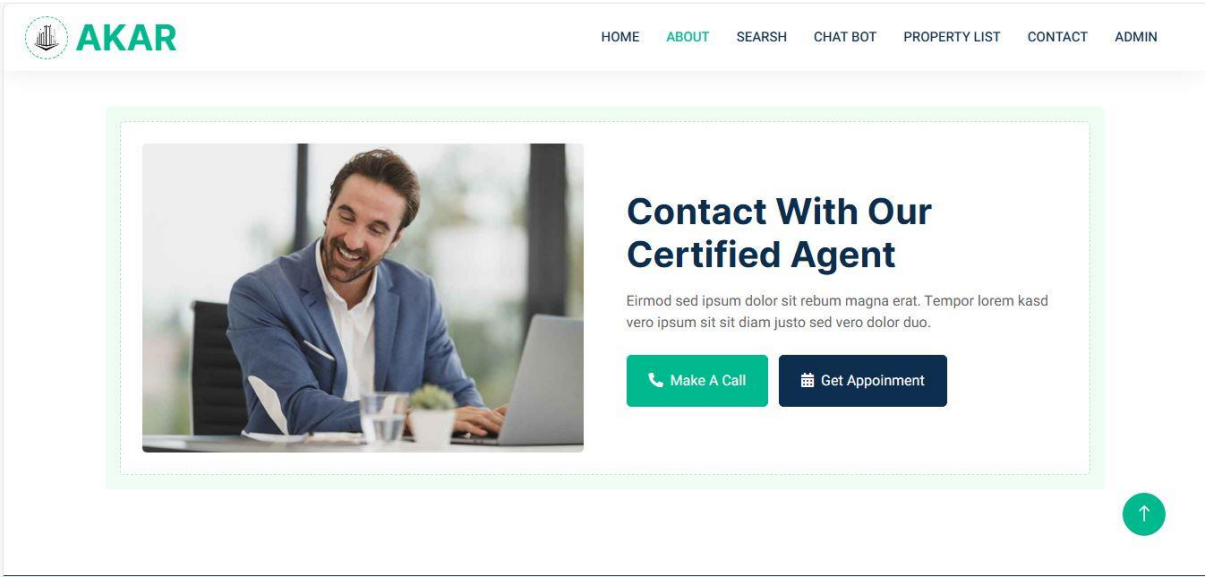
Figure(21)

When you scroll down, you will find a group of properties of various types displayed and in different conditions, whether they are for sale or for rent.



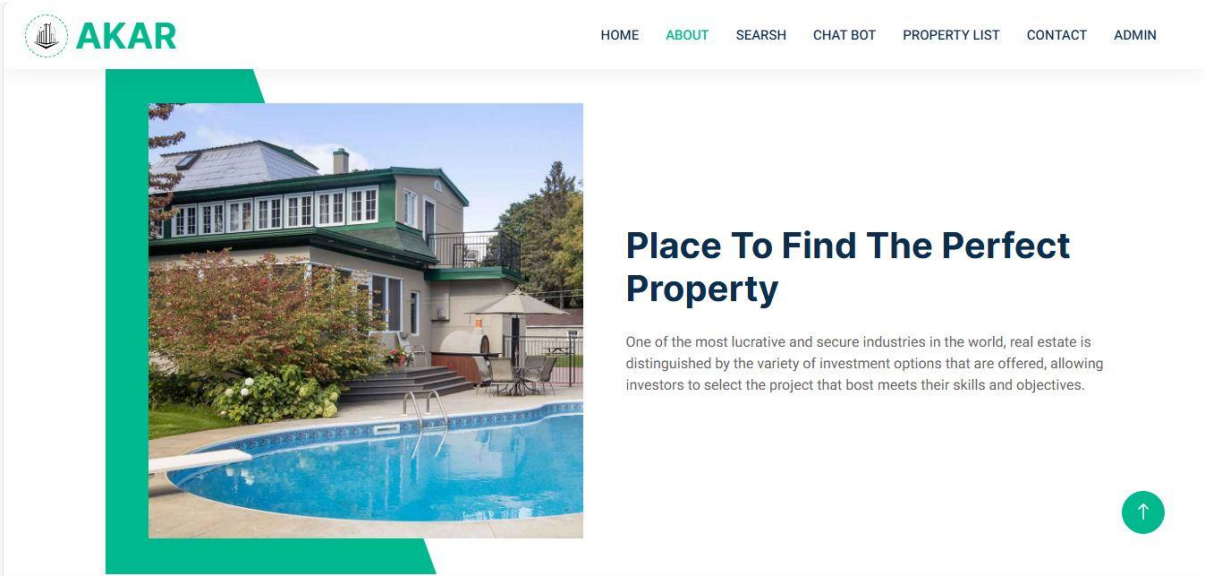
Figure(22)

There is also a user interface for direct connection to the contact page if the user makes calls and sends complaints to us.



Figure(23)

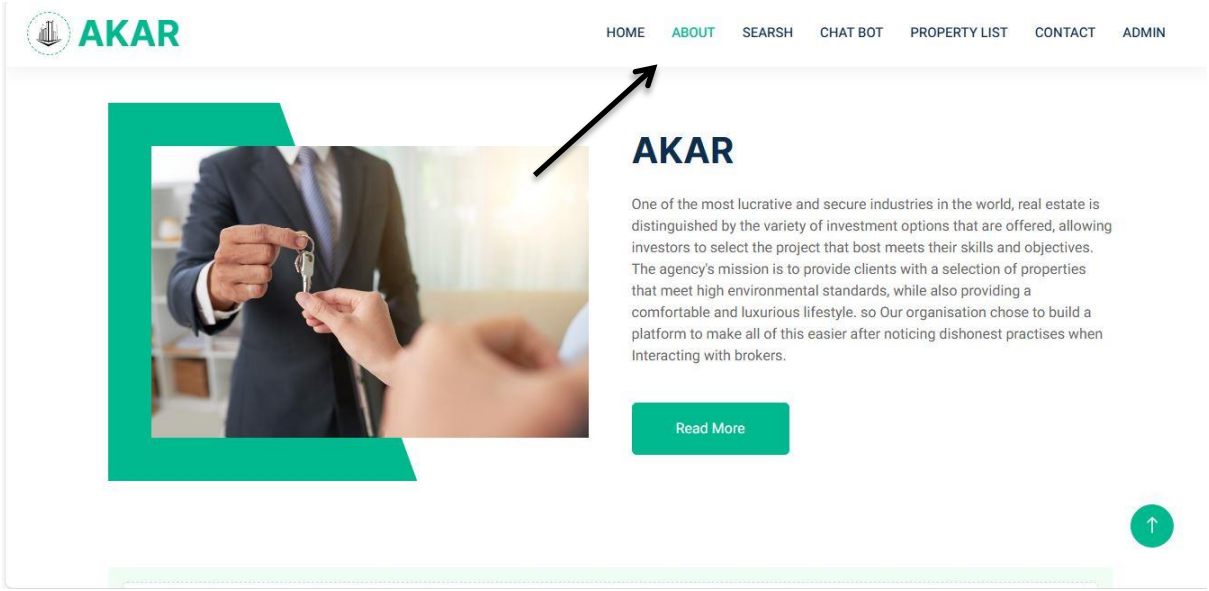
In this image, we present our website in a better way to the user, so that it gives him the best options.



Figure(24)

About us

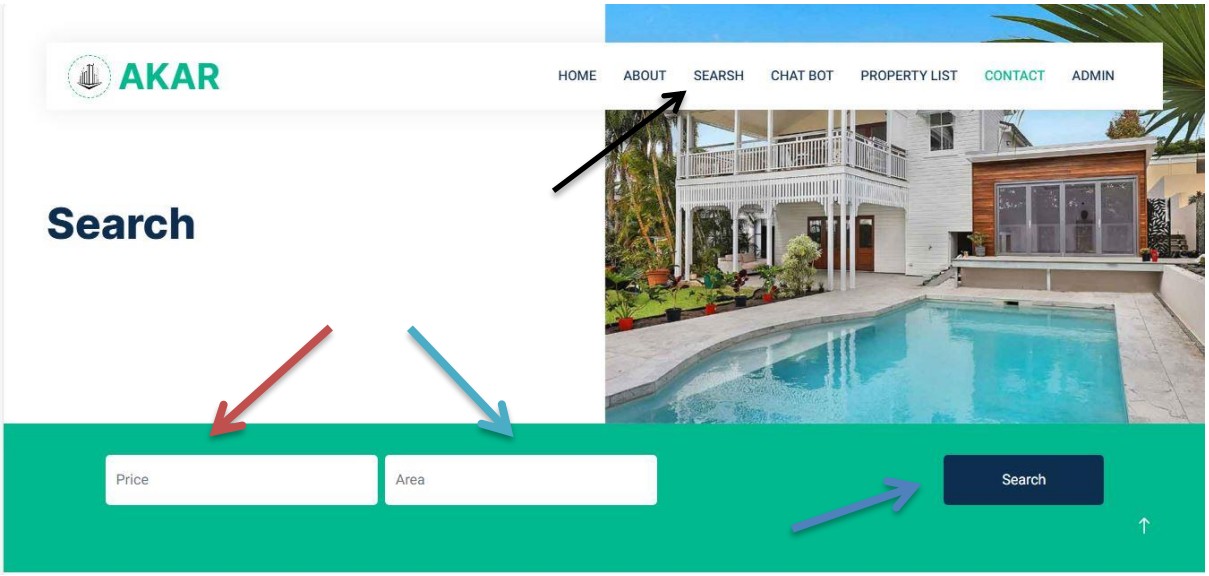
On this page we talk about information about us We also learned about real estate and its importance.



Figure(25)

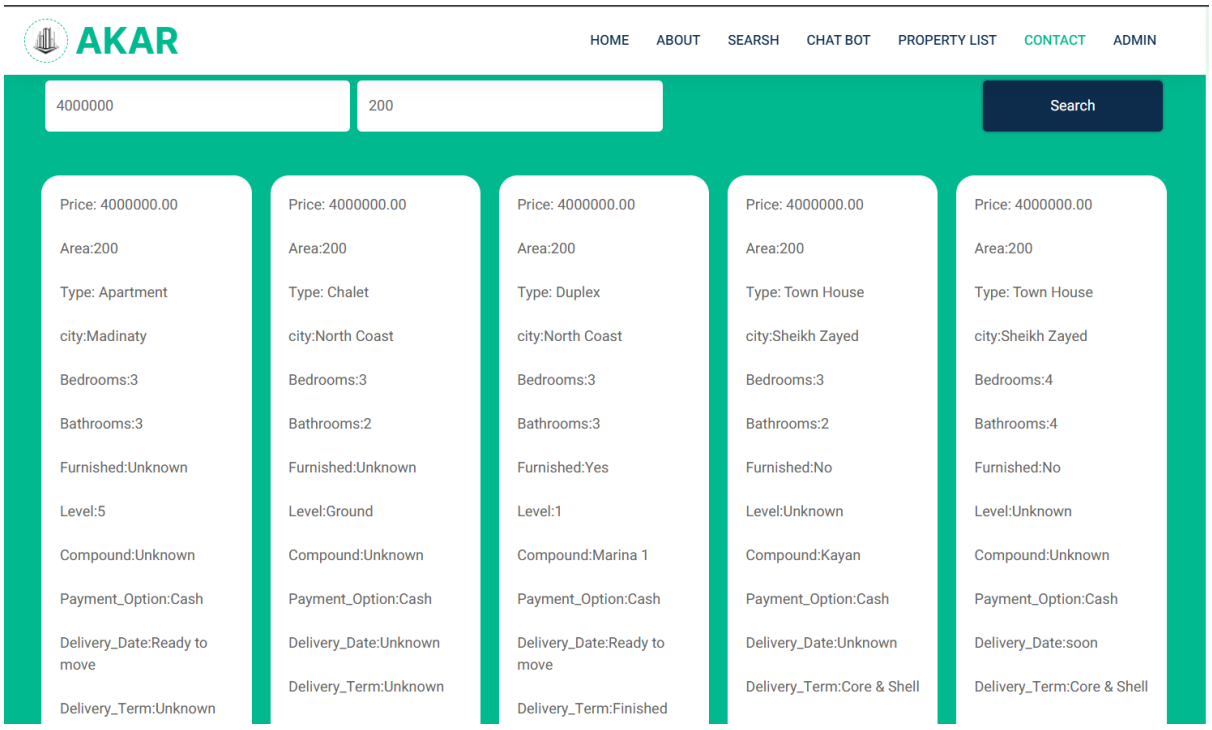
search

The search page consists of two fields, one for the price and the other for the area, and a button to execute. When those cells are full, the data API will be delivered to the data set and filtered and analyzed. The results related to that request will appear below. There is another advantage: if the user wants to search by price, this has no effect, and if he wants to search by area, filling it in has no effect.



Figure(26)

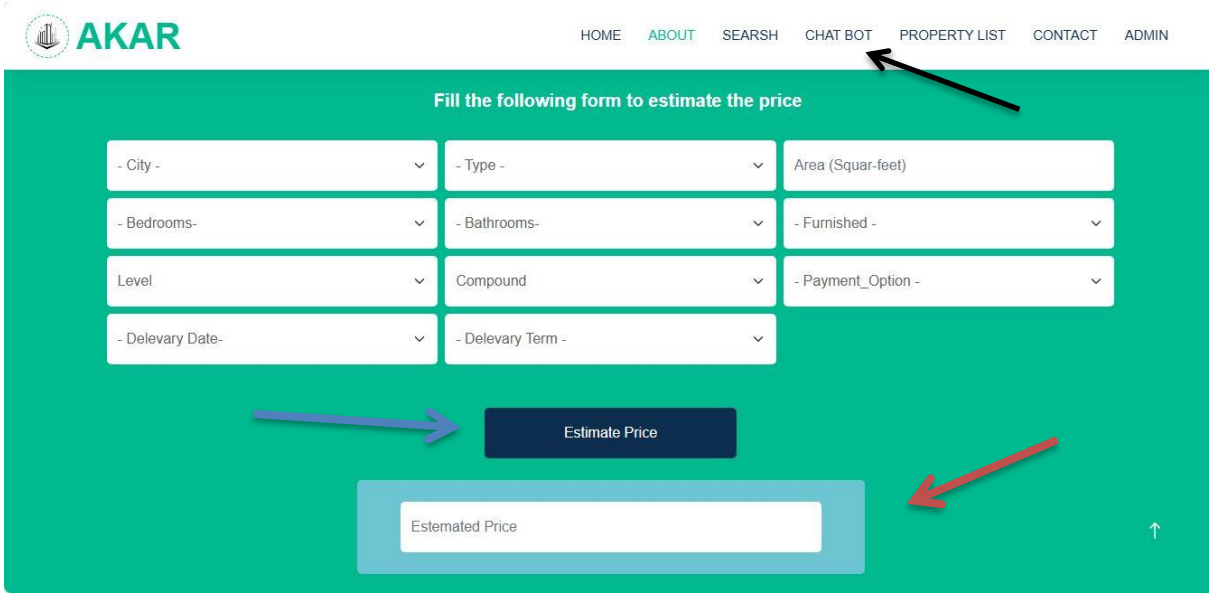
The result



Figure(27)

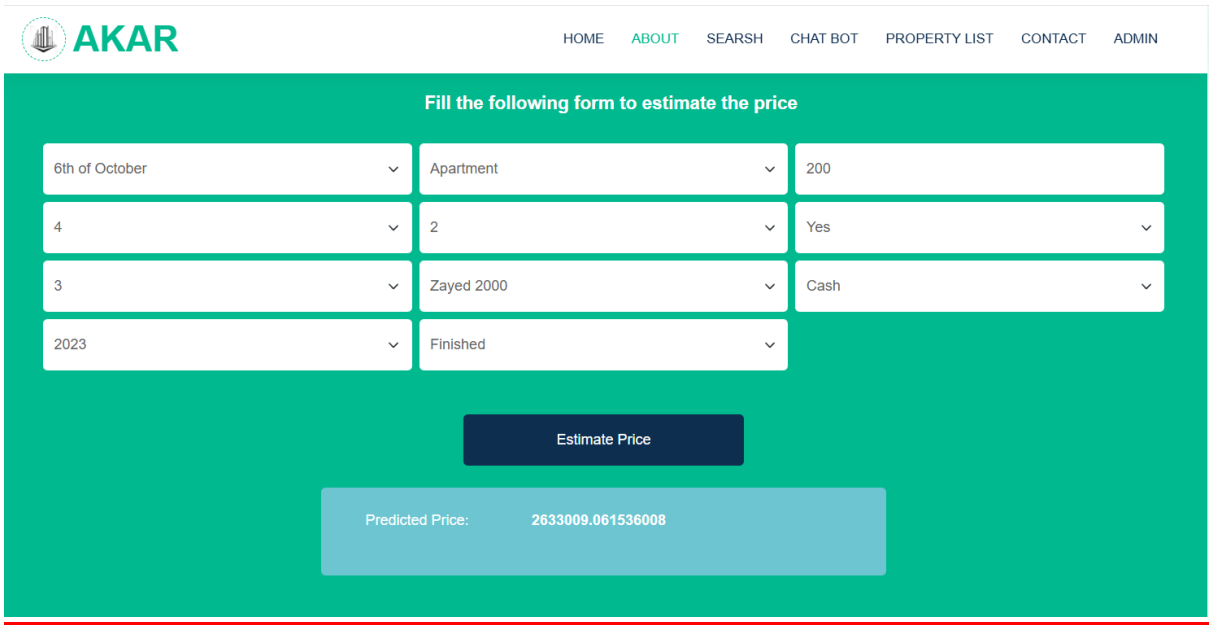
Chat bot

The chat bot page contains eleven fields that require filling out, and when you press a button Estimate price , the algorithm performs its basic cycle, and based on this data, it predicts a price and displays it in the field Estemated price.



Figure(28)

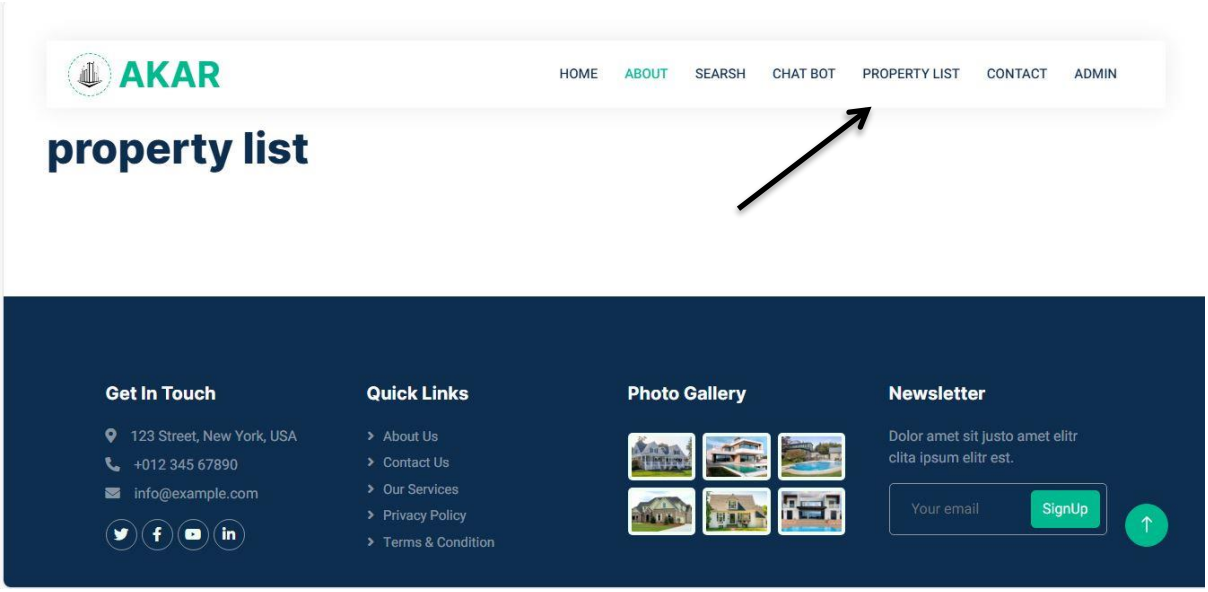
The result



Figure(29)

Property list

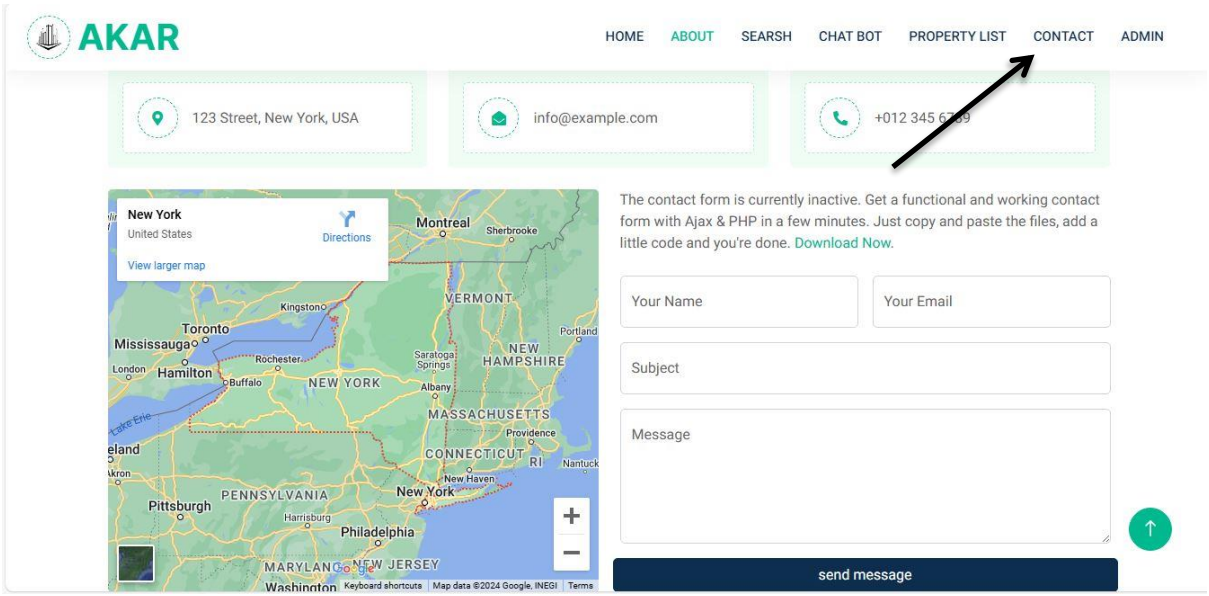
This page Property list displays the data that the admin adds and edits.



Figure(30)

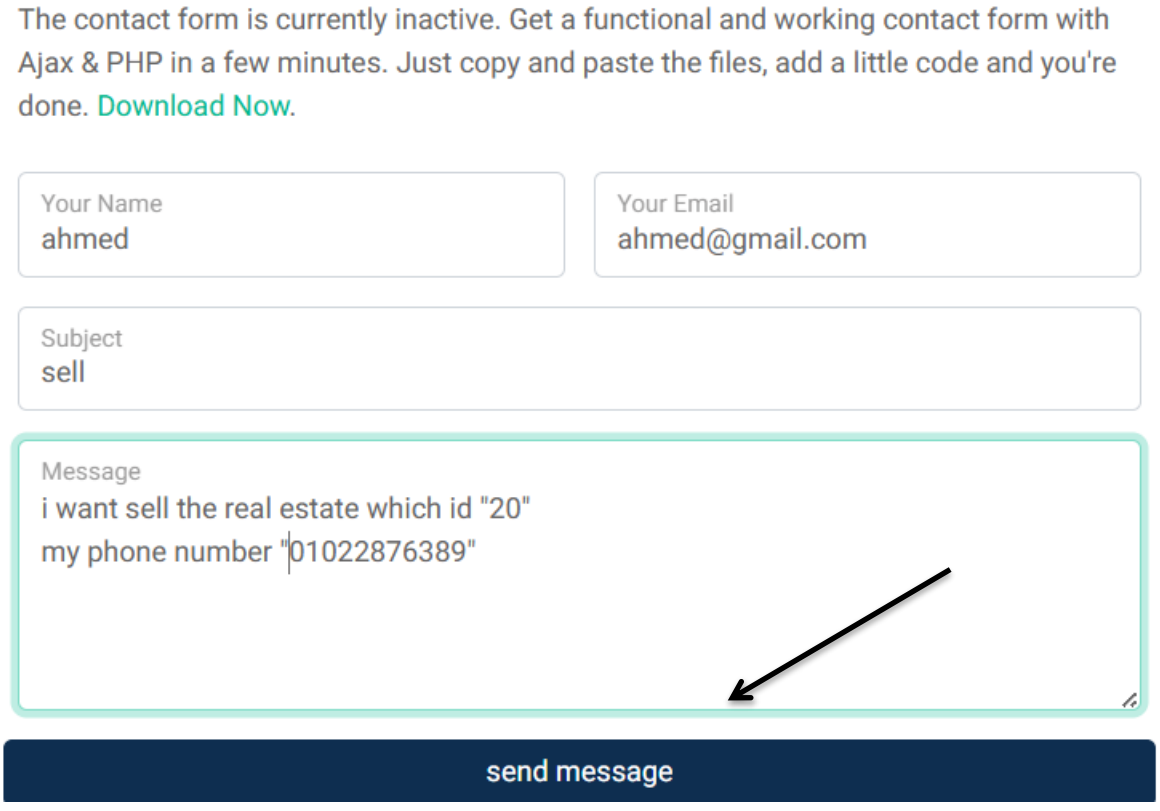
contact

On this page, it is the contact page, where the user writes his name, email, the subject of his request, and its message, and then presses the send button, from which that message will be saved in SQL data base.



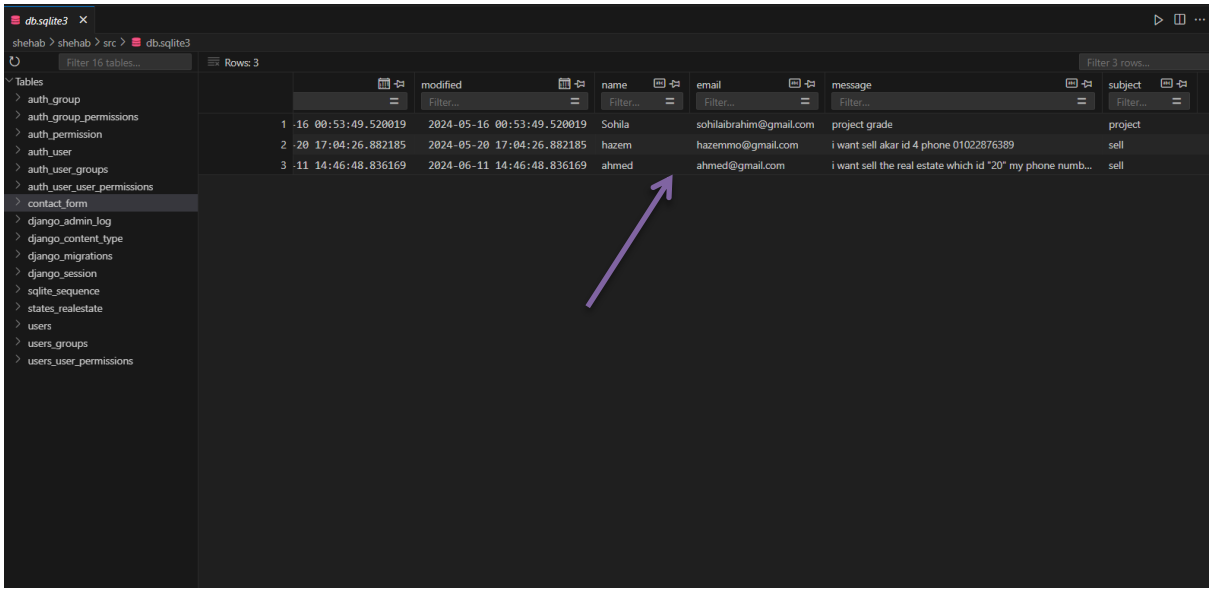
Figure(31)

If we fill this field and press send message



Figure(32)

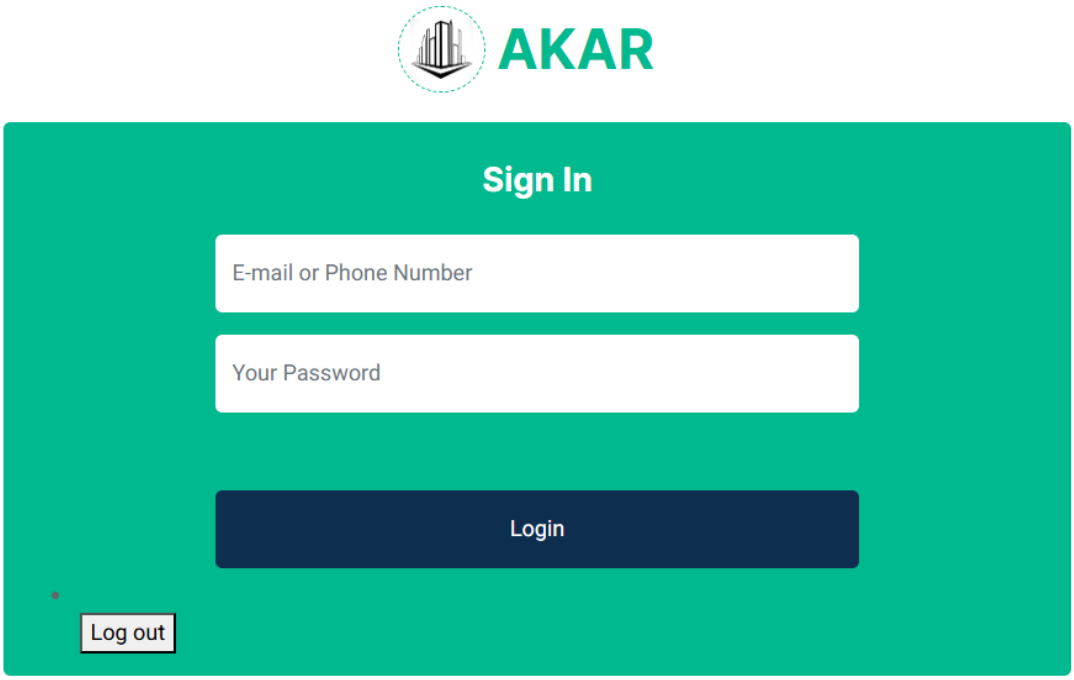
After press this button the data will send in SQL database admin that's result .



Figure(33)

Admin page

On this page two filled email admin and password after fill this filled and press login button the admin will go to dashboard page
And if press logout will go to home page.

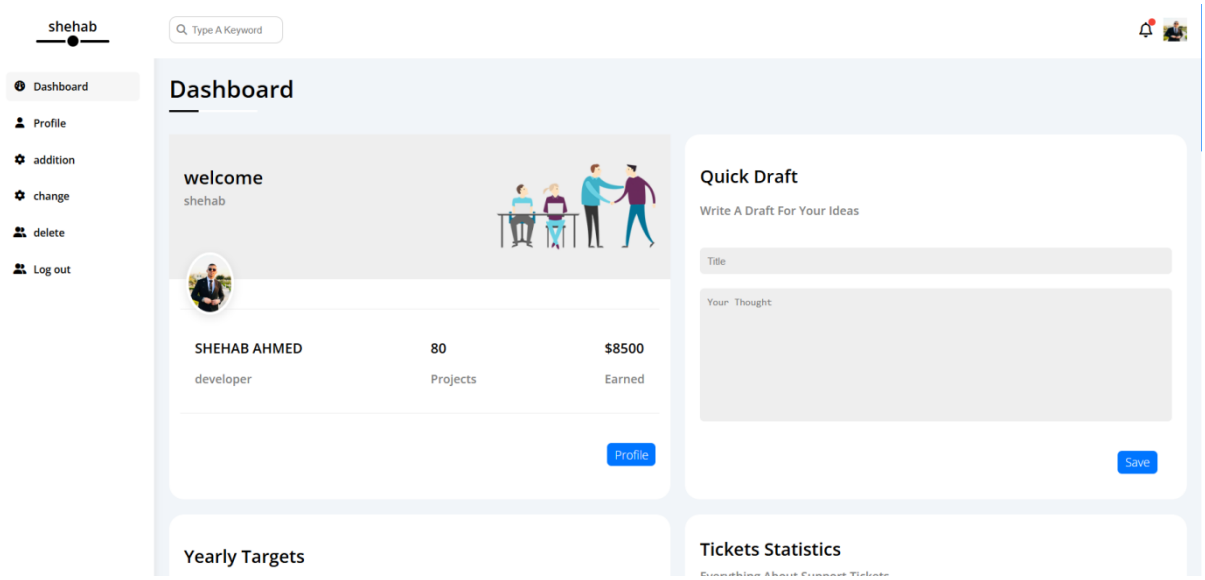


Figure(34)

After press login button will go to dashboard page .

Dashboard
On this page five page
Profile
Addition
Change
Delete
Logout

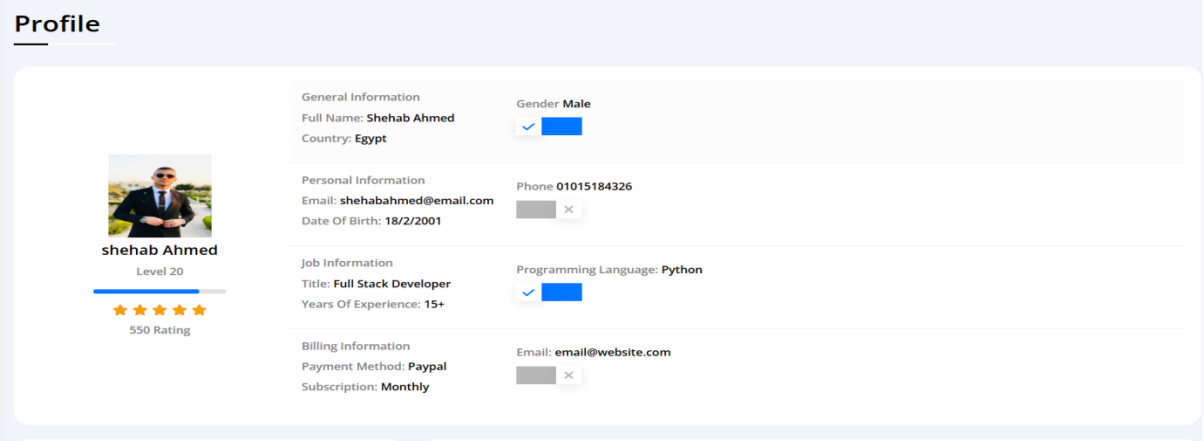
That’s dashboard tell us who is admin and how much earned .



Figure(35)

Profile page

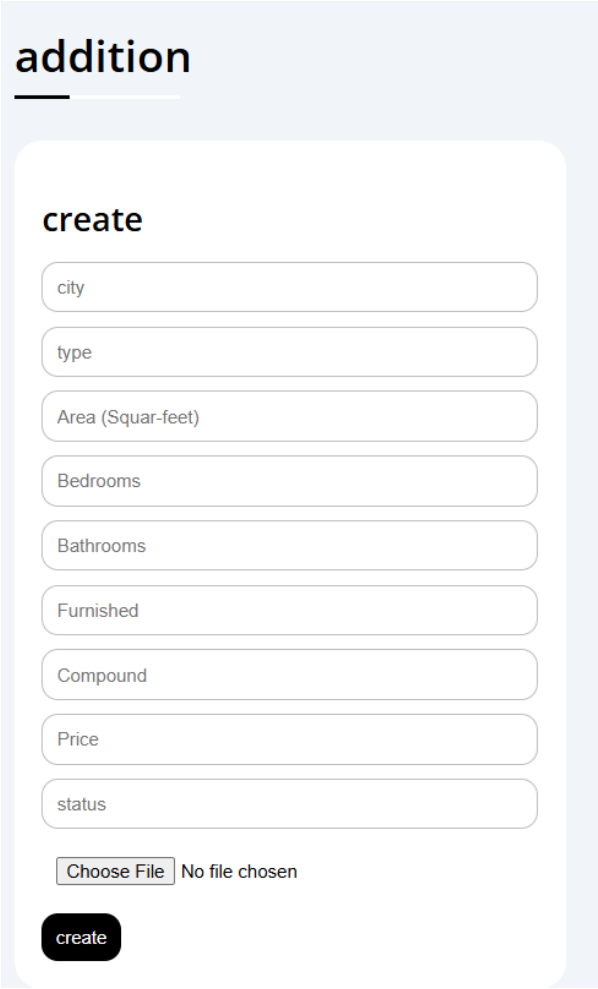
This page tell us some information about admin his name, phone , Country , Email and date of birth .



Figure(36)

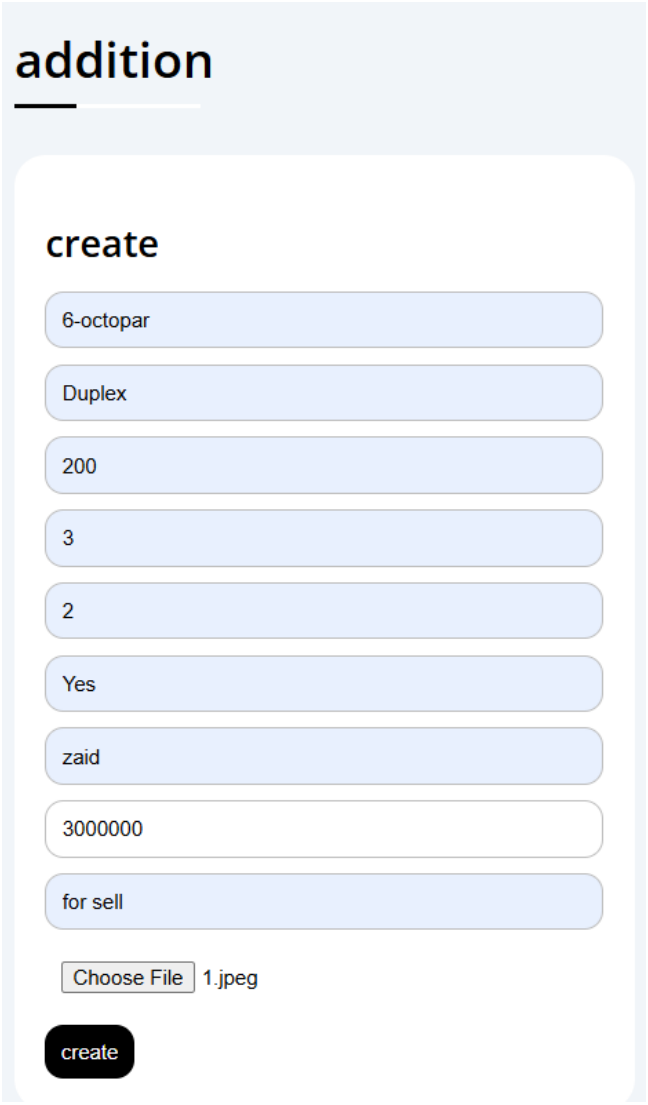
Addition page

This page to add new Real Estate in property list page.



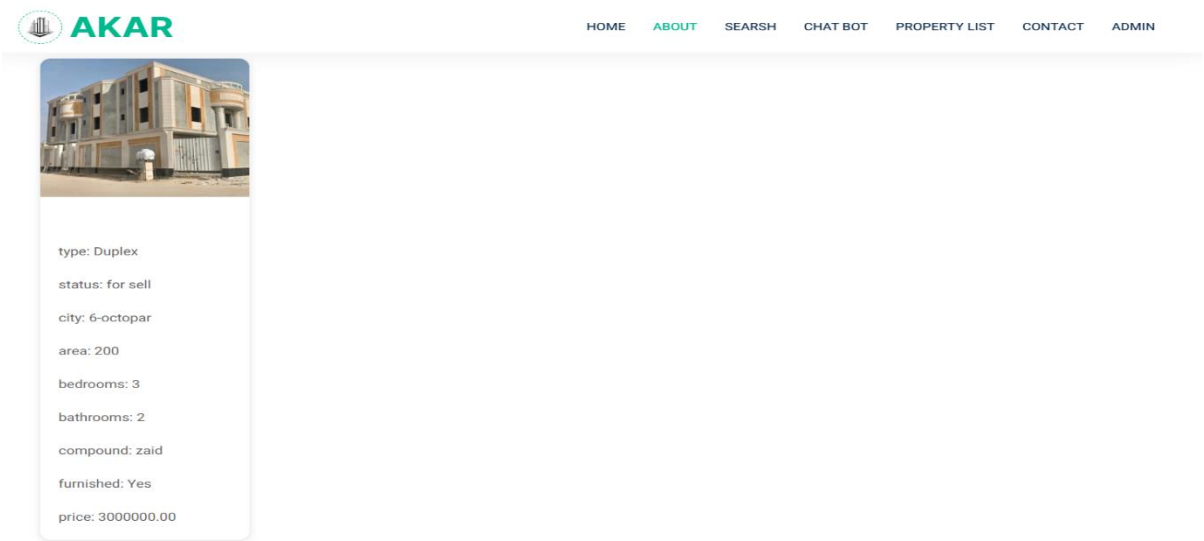
Figure(37)

After fill this field the result will show in property list page.



Figure(38)

Result



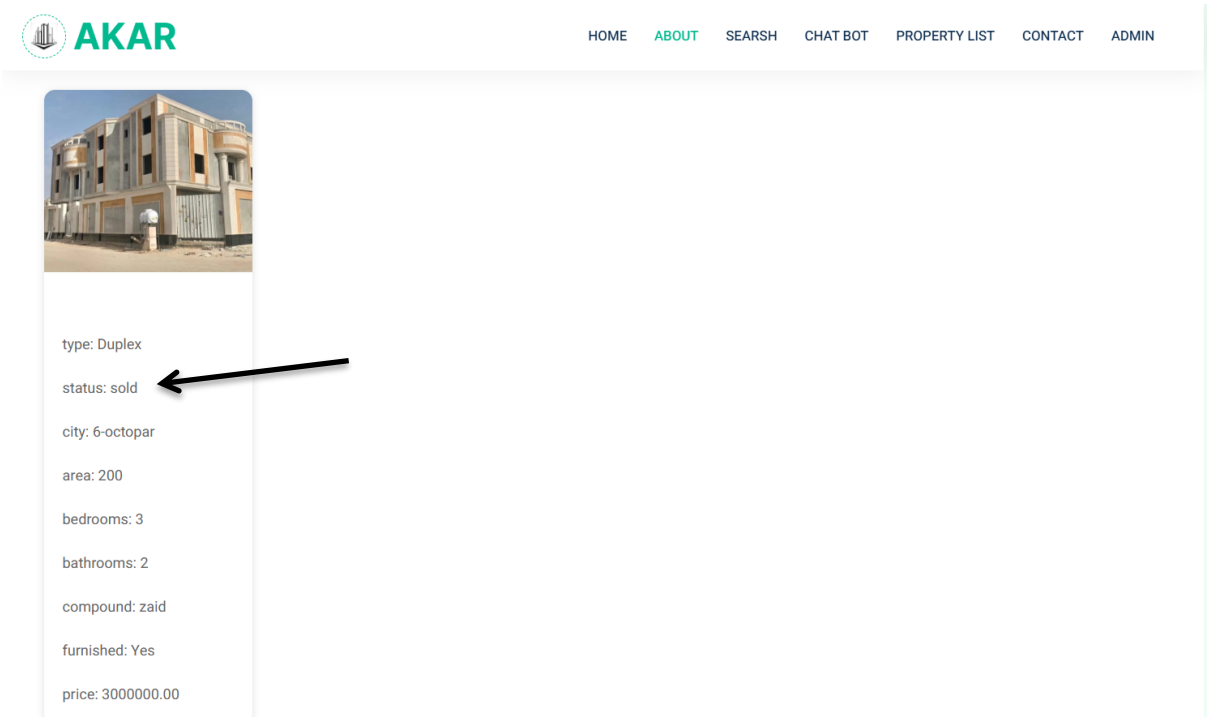
Figure(39)

Change page

This page use to make change status in any real estate by id .

Figure(40)

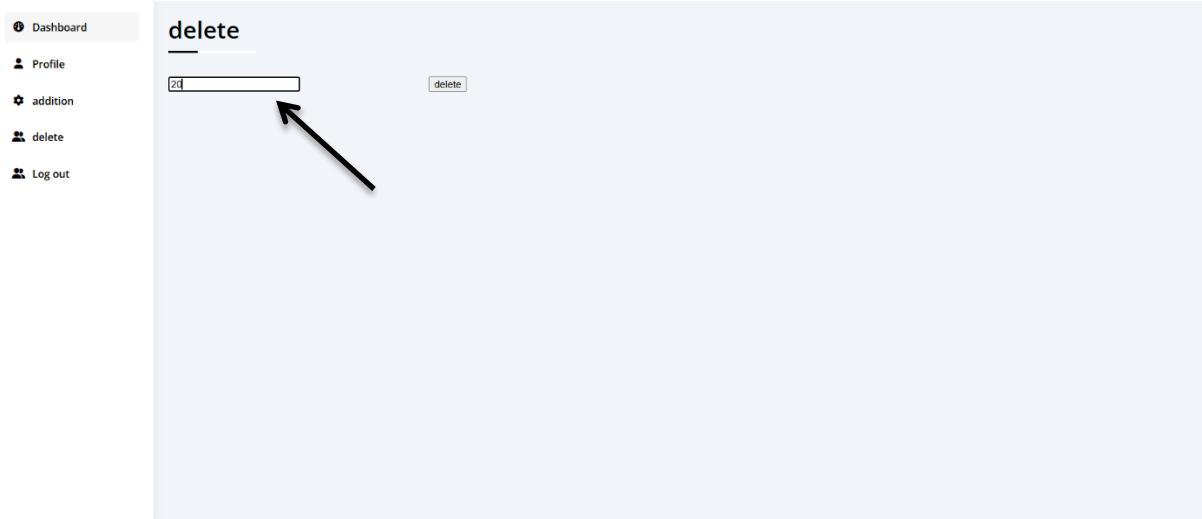
Result



Figure(41)

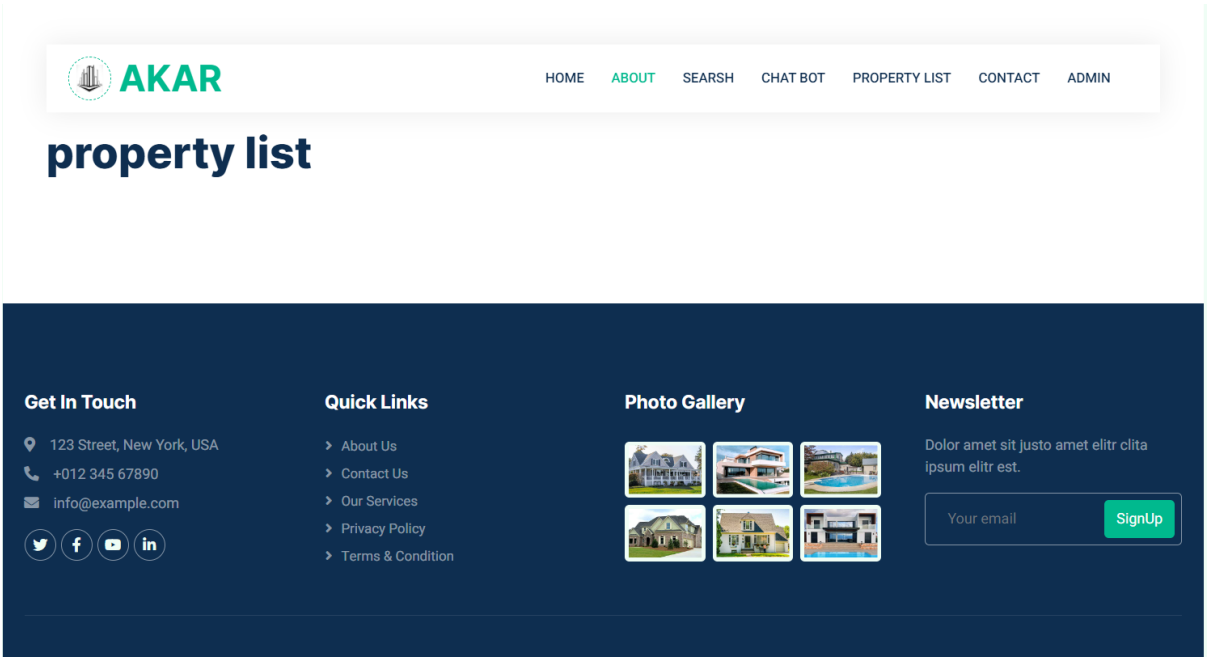
Delete page

If admin want to delete any real estate by id .



Figure(42)

Result



Figure(43)

Other Components Implementation

In search page

the code which use to make filtration to data in date set to view data which user search it.

```
from django.db.models import Value
from django.db.models.functions import Cast

class StateListView(generics.ListAPIView):
    permission_classes = [AllowAny, ]
    serializer_class = HouseSerializer

    search_fields = ['city', 'type', 'area', 'price']

    def get_queryset(self):
        Egypt_Houses_CSV_File_path = os.path.join(BASE_DIR, 'Egypt_Houses.csv')
        data = pd.read_csv(Egypt_Houses_CSV_File_path)

        # Filter based on query parameters
        queryset = data

        city = self.request.query_params.get('city', None)
        print("City:", city)
        if city:
            queryset = queryset[queryset['City'] == city]

        house_type = self.request.query_params.get('type', None)
        print("House Type:", house_type)
        if house_type:
            queryset = queryset[queryset['Type'] == house_type]

        area = self.request.query_params.get('area', None)
        print("Area:", area)
        if area:
            queryset = queryset[queryset['Area'] == int(area)]

        price = self.request.query_params.get('price', None)
        print("Price:", price)
        if price:
            queryset = queryset[queryset['Price'] == int(price)]

        return queryset.to_dict(orient='records')
```

Figure(44)

shehab > shehab > src > apps > states > serializers.py > RealStateSerializer > Meta

```python
from rest_framework import serializers

from .models import RealEstate

class HouseSerializer(serializers.Serializer):
    Real_estate_ID = serializers.IntegerField()
    City = serializers.CharField()
    Type = serializers.CharField()
    status = serializers.CharField()
    Area = serializers.FloatField()
    Bedrooms = serializers.IntegerField()
    Bathrooms = serializers.IntegerField()
    Furnished = serializers.CharField()
    Level = serializers.CharField()
    Compound = serializers.CharField()
    Payment_Option = serializers.CharField()
    Delivery_Date = serializers.CharField()
    Delivery_Term = serializers.CharField()
    Price = serializers.DecimalField(max_digits=12, decimal_places=2, )



class RealStateSerializer(serializers.ModelSerializer):
    class Meta:
        model = RealEstate
        fields = '__all__'
```

Figure(45)

Script which make show data from API in front end.

```javascript
<script>
    let btn =document.getElementById("searchButton");
    btn.addEventListener("click", function() {
        let contentView = document.querySelector(".content-veiw");
        document.getElementById("searchButton").addEventListener("click", function() {
            // Get filter values
            let Price = document.getElementById("priceInput").value;
            let area = document.getElementById("areaInput").value;



            // Construct the URL with filter parameters
            let url = `http://127.0.0.1:8000/states/?price=${Price}&area=${area}`;



            // Fetch data from the API
            fetch(url)
                .then(response => response.json())
                .then(data => {
                    console.log("Data from API:", data);
                    console.log( `http://127.0.0.1:8000/states/?price=${Price}&area=${area}}`);

                    contentView.innerHTML=""
                  for (let i = 0; i < data.results.length; i++) {
                   let box=  document.createElement("div");
                  box.className="box-li"
                  box.innerHTML=` Price: ${data.results[i].Price} </br> </br>Area:${data.results[i].Area} </br> </br> Type: ${data.results[i].Type} </br> </br> city:${data.re



                  contentView.appendChild(box)
                   }
                   console.log( data.results.length)
                })
                .catch(error => console.error("Error fetching data:", error));
        });
    });
</script>
```

Figure(46)

In chat bot page

The code of predict in server to make contact between front and Algorithm by API .

```python
# Predict house price
def predict_house_price(model, label_encoders, input_data):
    # Define the features based on the input requirement
    features = [
        'City', 'Type', 'Area', 'Bedrooms',
        'Bathrooms', 'Furnished', 'Level',
        'Compound', 'Payment_Option', 'Delivery_Date', 'Delivery_Term'
    ]

    input_df = pd.DataFrame([input_data], columns=features)

    # Encode categorical features
    for feature in input_df.select_dtypes(include=['object']).columns:
        if feature in label_encoders:
            input_df[feature] = label_encoders[feature].transform(input_df[feature])
        else:
            raise ValueError(f"Label encoder for {feature} is missing in the provided 'label_encoders' dictionary.")

    # Predict house price
    try:
        predicted_price = model.predict(input_df)
        return predicted_price[0]
    except ValueError as ve:
        # Handle unseen labels error
        if "y contains previously unseen labels" in str(ve):
            return {"error": str(ve)}
        else:
            raise ve

# Custom label encoder
class CustomLabelEncoder(LabelEncoder):
    def fit(self, y):
        super().fit(y)
        self.classes_ = np.append(self.classes_, '<unknown>')  # Add '<unknown>' for unseen labels
        return self

    def transform(self, y):
        unseen_labels_mask = ~np.isin(y, self.classes_)
        y_transformed = np.where(unseen_labels_mask, '<unknown>', y)  # Replace unseen labels with '<unknown>'
        return super().transform(y_transformed)
```

Figure(47)

```python
# Django view
class PredictedStateList(generics.GenericAPIView):
    permission_classes = [AllowAny, ]

    def get_serializer_class(self):
        return HouseSerializer

    def post(self, request, *args, **kwargs):
        # Get input features from request data
        input_data = request.data

        try:
            # Load label encoders and model
            label_encoders = load_label_encoders(filename='features_encoders.pkl')
            model = load_prediction_model(filename='final_model.txt')

            # Predict the price using the model and input features
            predicted_price = predict_house_price(model, label_encoders, input_data)

            # Return the predicted price as a response
            return Response({"predicted_price": predicted_price})

        except ValueError as ve:
            return Response({"error": str(ve)}, status=400)
        except Exception as e:
            return Response({"error": str(e)}, status=500)
```

Figure(48)

shehab > shehab > src > apps > states > urls.py > ...

```python
from django.urls import path, include

from . import views

from rest_framework import routers

router = routers.DefaultRouter()

router.register('property-list', views.StateViewSet, basename='property-list')


urlpatterns = [
    path('', views.StateListView.as_view(), name='state-list'),

    path('', include(router.urls)),


    path('predicted-states/', views.PredictedStateList.as_view(), name='predicted-state-list'),
]

```

Figure(49)

The script which use to show the result in frontend.

```
    <script src="js/main.js"></script>
    <script>
function estimate() {
    var inputs = $('.esInput');
    var data = {
        City: inputs.eq(0).val(),
        Type: inputs.eq(1).val(),
        Area: Number(inputs.eq(2).val()),
        Bedrooms: Number(inputs.eq(3).val()),
        Bathrooms: Number(inputs.eq(4).val()),
        Furnished: inputs.eq(5).val(),
        Level: Number(inputs.eq(6).val()),
        Compound: inputs.eq(7).val(),
        Payment_Option: inputs.eq(8).val(),
        Delivery_Date: inputs.eq(9).val(),
        Delivery_Term: inputs.eq(10).val()
    };

    fetch("http://127.0.0.1:8000/states/predicted-states/", {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(data => {
        console.log("Data from API:", data); // Log the entire data object
        // Check if predictedPrice is present in data object and display it
        if (data && data.predicted_price !== undefined) {
            $("#putResult").html(`<div class="row mb-4">
                <div class="col-4 text-white">Predicted Price:</div>
                <div class="col-8 text-white fw-bold">${data.predicted_price}</div>
            </div>`);
        } else {
            console.error('Predicted price is undefined or not present in the response data');
        }
    })
    .catch(error => {
        console.error('Error:', error);
    });
}
```

Figure(50)

In connect page

The code which use to send this message to SQL database.

shehab > shehab > src > apps > contacts > models.py > ...

```python
from django.db import models

from django_extensions.db.models import TimeStampedModel

# from apps.users.validators import valid_phone_number


class ContactForm(TimeStampedModel):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    message = models.TextField()
    subject = models.TextField()

    def __str__(self):
        return self.name


    class Meta:
        db_table = "contact_form"
        verbose_name = "Contact Form"
        verbose_name_plural = "Contact Forms"
        ordering = ["-created"]
```

Figure(51)

shehab > shehab > src > apps > contacts > serializers.py > ...

```python
from rest_framework.serializers import ModelSerializer

from .models import ContactForm



class ConactFormSerializer(ModelSerializer):
    class Meta:
        model = ContactForm
        fields = ("name", "email", "subject", "message", 'created',)
        read_only_fields = ('created',)
```

Figure(52)

In property list page

The code which connect between backend and front end.

shehab > shehab > src > apps > states > models.py > ...

```python
from django.db import models

class RealEstate(models.Model):

    REAL_ESTATE_TYPES = [
        ('Apartment', 'Apartment'),
        ('Duplex', 'Duplex'),
        ('Villa', 'Villa'),
        ('Townhouse', 'Townhouse'),
        ('Penthouse', 'Penthouse'),
        ('Studio', 'Studio'),
        ('Chalet', 'Chalet'),
        ('Other', 'Other'),
    ]



    type = models.CharField(max_length=50, choices=REAL_ESTATE_TYPES, null=True, blank=True)

    city = models.CharField(max_length=100, null=True, blank=True)
    area = models.PositiveIntegerField(null=True, blank=True)  # in square meters
    bedrooms = models.PositiveSmallIntegerField(null=True, blank=True)
    bathrooms = models.PositiveSmallIntegerField(null=True, blank=True)
    compound = models.CharField(max_length=100, null=True, blank=True)
    status = models.CharField(max_length=100, null=True, blank=True)
    furnished = models.CharField(max_length=50, choices=[('Yes', 'Yes'), ('No', 'No')], null=True, blank=True)

    price = models.DecimalField(max_digits=12, decimal_places=2, )
    image = models.ImageField(upload_to='real_estates/', null=True, blank=True)

    def __str__(self):
        return f"Real Estate ID: {self.real_estate_id} - {self.type} in {self.city}"

```

Figure(53)

The script which use to show the result in frontend.

```
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
    <script>

                document.addEventListener('DOMContentLoaded', () => {
                  let content =  document.querySelector(".contennt-veiw");
axios.get('http://127.0.0.1:8000/states/property-list/', {
})
.then(function (response) {
    myarr= response.data.results;
    for (let i = 0; i < myarr.length; i++) {
       let box= document.createElement("div");
       box.className="box-content";
      box.innerHTML=` <div class="box-img"> <img src="${myarr[i].image}" alt=""> </div>  </br> </br >  <div class="text-box"> type: ${myarr[i].type}</br> </br> status:
      content.appendChild(box)

    }

    console.log(content);
})
.catch(function (error) {
    console.error('Error fetching data:', error);
});
});

    </script>
```

Figure(54)

Summary

In summary, This data has been classified and arranged to be suitable for performing the purposes of the site and fulfilling the requirements of predicting prices and also searching for a suitable property for the user. We have explained its components to make it easier for you to understand what we will do.

We also mentioned, explained and simplified the user interface so that you have a background on our site and how to use it.

Chapter 6
Machine Learning and models

Introduction

In this chapter, we will talk about the Machine Learning of the site, its characteristics, the model used and the outputs, and also some things about another model that we tested, extracted its outputs, compared the two models, and chose the best one.

Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) focused on developing algorithms and techniques that enable computers to "learn" from data. Instead of programming computers to perform specific tasks explicitly, machine learning allows systems to improve their performance on a given task based on past experiences or available data.

Supervised Learning

In supervised learning, the model is trained on a labeled dataset, which means the input data is paired with the correct output. The goal is to learn the mapping between inputs and outputs so that the model can predict the correct output for new, unseen inputs. Common applications include classification (e.g., identifying spam emails) and regression (e.g., predicting house prices).

Unsupervised Learning

In unsupervised learning, the model is provided with input data without any corresponding output labels. The goal is to uncover hidden patterns or structures in the data. Common applications include clustering (e.g., customer segmentation) and dimensionality reduction (e.g., simplifying data for visualization).

Reinforcement Learning

In reinforcement learning, the model learns by interacting with an environment and receiving rewards or penalties based on its actions. The goal is to maximize the total reward over time. This type of learning is widely used in developing game strategies and controlling robots.

Applications of Machine Learning

Machine learning has a wide range of applications across various fields, including:

• Image Recognition: Such as facial recognition in smartphones.
• Natural Language Processing (NLP): Such as machine translation and text analysis.
• Medical Data Analysis: Improving disease diagnosis and treatment development.
• Financial Prediction: Analyzing stock markets and predicting price movements.
• E-commerce: Recommending products to customers based on their interests and behaviors.

Conclusion

Machine learning is a leading field in artificial intelligence, offering significant capabilities for data analysis and decision-making with greater accuracy and efficiency. As technology advances and data availability increases, machine learning becomes more essential and influential in various aspects of our daily lives.

Models and Results

Light GBM

Introduction to Light GBM

Light GBM, which stands for "Light Gradient Boosting Machine," is an open-source framework developed by Microsoft for creating machine learning models. Light GBM is based on the gradient boosting technique and is designed to be fast and efficient in handling large amounts of data while maintaining high accuracy in models.

What is Light GBM?

Light GBM is a library for developing machine learning models based on tree-based algorithms, using gradient boosting to enhance model performance. It is known for several features that make it preferred in practical applications, including:

- **Speed and Efficiency**: Light GBM is much faster in training and prediction compared to other frameworks like XG Boost. It uses various optimization techniques to efficiently utilize available hardware and handle data in parallel.
- **High Accuracy**: Light GBM produces highly accurate models due to its numerous improvements in tree-building and data handling processes.
- **Support for Large Data**: Light GBM can easily handle millions of data points, making it suitable for big data applications.

How Does Light GBM Work?

Light GBM works by constructing an ensemble of decision trees in a sequential manner, where each new tree attempts to correct the errors of the previous trees. This process involves the following steps:

1. **Start with a Simple Model**: Begin with a simple predictive model (usually the mean value of the target variable).
2. **Calculate Errors**: Compute the errors or differences between the current predictions and the actual values.

3. **Build a New Tree**: Construct a new tree that aims to minimize these errors as much as possible.
4. **Update the Model**: Update the model by adding the new tree to the current model.
5. **Repeat**: Repeat the above steps until a specified number of trees are built or the performance is sufficiently improved.

Features of Light GBM

- **Handling Large Data**: Light GBM can easily manage millions of data points thanks to techniques like histogram-based algorithms.
- **Speed and Efficiency**: Light GBM is known for its fast training and prediction capabilities.
- **Support for Various Data Types**: Light GBM can handle both categorical and numerical data seamlessly.

- **Handling Imbalanced Data**: Light GBM includes techniques to effectively deal with imbalanced data issues.

Applications of Light GBM

Light GBM is used in various applications, such as:

- **Financial Prediction**: Analyzing financial markets and predicting stock movements.
- **Medical Data Analysis**: Improving disease diagnosis and treatment development.
- **E-commerce**: Analyzing customer behavior and recommending products.
- **Text Analysis**: Document classification and sentiment analysis.

Conclusion

Light GBM is a powerful and efficient tool for developing machine learning models, especially in handling large amounts of data. Its high speed and accuracy make it a popular choice among developers and researchers across various fields.

```
[12] model = lgb.train(study.best_trial.params, train_set)
     preds = model.predict(X_test, num_iteration=model.best_iteration)

     # Calculate RMSE
     r2 = r2_score(y_test, preds)

     print('R2 Score of Final Test Data: ', r2)
```

```
[LightGBM] [Warning] Categorical features with more bins than the configured maximum bin number found.
[LightGBM] [Warning] For categorical features, max_bin and max_bin_by_feature may be ignored with a large number of categories.
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.002241 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 802
[LightGBM] [Info] Number of data points in the train set: 21354, number of used features: 10
[LightGBM] [Info] Start training from score 4393631.647326
R2 Score of Final Test Data:  0.668625715813098
```

```
[17] X.shape
```

```
(26693, 11)
```

```
[18] y.shape
```

```
(26693,)
```

Figure(55)

As we can see, this is the Score value produced by the lightGBM model, which is 66%.

X

They are the values of the variables that we enter as inputs to the model so that it outputs to us the value of the price that it predicts, which is the value of y.

Random Forest Regression

Introduction to Random Forest Regression

Random Forest Regression is a machine learning algorithm that belongs to the ensemble learning family. It is specifically used for regression tasks, where the goal is to predict a continuous output variable based on input features. The algorithm is an implementation of a random forest, which is essentially a collection of decision trees working together to produce a more accurate and stable prediction.

What is Random Forest Regression?

Random Forest Regression is part of the random forest algorithm family, which operates by constructing multiple decision trees during training and outputting the average prediction of the individual trees. This method reduces the risk of over fitting and improves generalization.

How Does Random Forest Regression Work?

Random Forest Regression works through the following steps:

1. **Bootstrap Sampling**: Random subsets of the training data are created with replacement, meaning some samples may be repeated, and others may be omitted. Each subset is used to train a separate decision tree.
2. **Training Individual Trees**: Each decision tree is trained independently on its bootstrap sample. During the training process, a random subset of features is selected at each split in the tree to introduce variability among the trees.
3. **Aggregation of Predictions**: For regression tasks, the predictions from all individual trees are averaged to produce the final output. This aggregation helps to reduce variance and improve the robustness of the model.

Features of Random Forest Regression

- **Robustness to Over fitting**: By averaging the predictions of multiple trees, Random Forest Regression reduces the risk of over fitting compared to individual decision trees.

- **Feature Importance**: The algorithm provides a way to measure the importance of each feature in making predictions, which can be useful for understanding the data and feature selection.
- **Handling Large Datasets**: Random Forest Regression can handle large datasets with high-dimensional feature spaces effectively.
- **Resilience to Noise**: The randomization and aggregation processes make the model more resilient to noise in the data.

Applications of Random Forest Regression

Random Forest Regression is widely used in various fields due to its versatility and effectiveness. Some common applications include:

- **Predicting Housing Prices**: Estimating the value of real estate properties based on features like location, size, and amenities.
- **Financial Forecasting**: Predicting stock prices, market trends, and economic indicators.
- **Environmental Modeling**: Forecasting weather patterns, air quality, and pollution levels.
- **Healthcare**: Predicting patient outcomes, treatment effectiveness, and disease progression.

Conclusion

Random Forest Regression is a powerful and flexible algorithm for regression tasks, combining the strengths of multiple decision trees to produce accurate and stable predictions. Its robustness to over fitting, ability to handle large and complex datasets, and feature importance insights make it a valuable tool in the machine learning toolkit.

```
[8] from sklearn.ensemble import RandomForestRegressor
    rf_model = RandomForestRegressor()
    rf_model.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
[9] from sklearn.metrics import r2_score
    y_pred = rf_model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    print(f'R2 Score: {r2}')
```

```
R2 Score: 0.6390484647280561
```

```
[11] X.shape
```

```
(26693, 11)
```

```
y.shape
```

```
(26693,)
```

Figure(56)

As we can see,
this is the Score value produced by the RandomForestRegressor model, which is 63%.

X

They are the values of the variables that we enter as inputs to the model so that it outputs to us the value of the price that it predicts, which is the value of y.

Linear Regression

Introduction to Linear Regression

Linear Regression is one of the simplest and most well-known machine learning algorithms in the fields of statistics and predictive analysis. It is used to determine the relationship between one or more dependent variables (outputs) and one or more independent variables (inputs) by fitting a linear model to the data.

What is Linear Regression?

Linear Regression is a statistical method used to model the relationship between a dependent variable (Y) and one or more independent variables (X). The goal is to find a linear equation (a straight line) that can be used to predict the values of the dependent variable based on the values of the independent variables.

The basic equation for simple linear regression (with one independent variable) is: Y=β0+β1X+ϵY = \beta_0 + \beta_1 X + \epsilonY=β0+β1X+ϵ

where:

- YYY is the dependent variable.
- XXX is the independent variable.
- β0\beta_0β0 is the intercept; the value of YYY when X=0X = 0X=0.
- β1\beta_1β1 is the regression coefficient (slope); the amount by which YYY changes when XXX changes by one unit.
- ϵ\epsilonϵ is the random error.

How Does Linear Regression Work?

1. **Data Collection**: Linear regression requires a historical dataset that contains values for the dependent variable and the independent variables.
2. **Model Fitting**: The least squares method is used to estimate the model parameters β0\beta_0β0 and β1\beta_1β1 in such a way that the sum of the squared differences between the actual values and the predicted values of YYY is minimized.
3. **Prediction**: Once the model parameters are estimated, the linear equation can be used to predict the values of the dependent variable YYY based on the values of the independent variables XXX.

4. **Model Evaluation**: The model's performance is evaluated using metrics such as the coefficient of determination (R2R^2R2), mean absolute error (MAE), and root mean squared error (RMSE) to understand the accuracy of the predictions.

Features of Linear Regression

- **Simplicity**: Linear regression is straightforward and easy to understand and implement.
- **Interpretability**: The model coefficients are easy to interpret, providing insight into the relationship between variables.
- **Speed and Efficiency**: Linear regression models can be trained quickly, even with large datasets.

Applications of Linear Regression

Linear regression is used in a wide range of fields, including:

- **Financial Forecasting**: Predicting stock prices and future returns.
- **Marketing Analysis**: Understanding the impact of advertising on sales.
- **Economics**: Studying the relationship between economic variables such as GDP and unemployment rates.
- **Healthcare**: Predicting health outcomes based on medical data.

Conclusion

Linear Regression is a powerful and simple tool in the fields of machine learning and statistics, helping to understand and interpret relationships between variables and predict future values. Its simplicity and effectiveness make it a popular choice among researchers and analysts in various domains.

```
[8] from sklearn.linear_model import LinearRegression
    model = LinearRegression()
    model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
[9] from sklearn.metrics import r2_score
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    print(f'R2 Score: {r2}')
```

```
R2 Score: 0.4702148854047089
```

```
[10] X.shape
```

```
(26693, 11)
```

```
y.shape
```

```
(26693,)
```

Figure(57)

As we can see,

this is the Score value produced by the Linear Regression model, which is 47%.

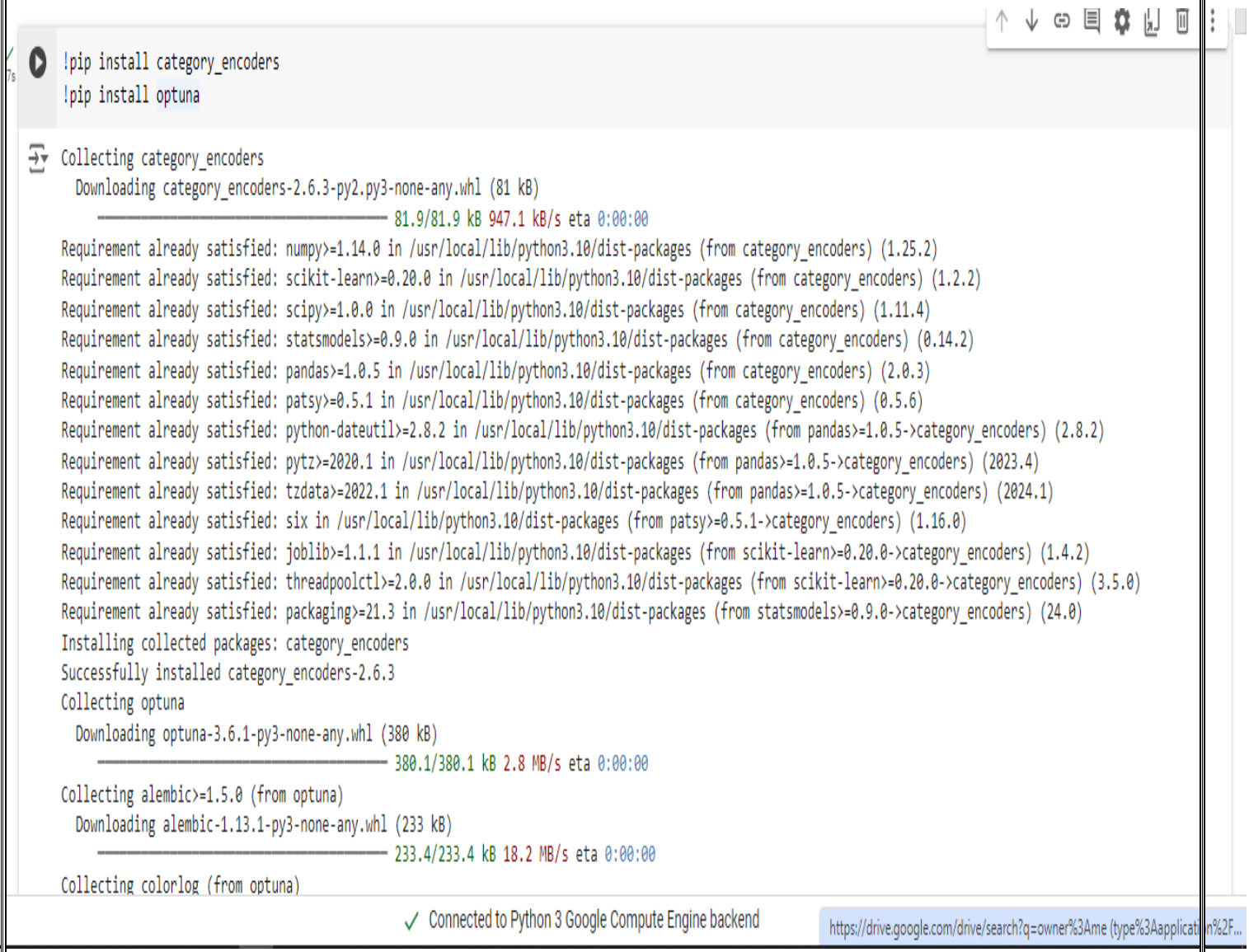
X

They are the values of the variables that we enter as inputs to the model so that it outputs to us the value of the price that it predicts, which is the value of y.

Now, as we see, the light GBM model was trained, which produced a Score value of 66%, the Random forest Regression model was trained, which produced a Score value of 63%, and the linear Regression model was trained, which produced a Score value of 47%. We find that the best results came from the light GBM model. This is the model that we used.

In our project.

Now we will provide a simplified explanation of each code that was used to train the light GBM model to make it predict well.



Figure(58)

This code installs two important libraries using the `pip` package manager:

1. **`Category encoders`**: A library for encoding categorical variables in data, making it easier to use them in computational models.
2. **`Op tuna`**: A library for automatic hyper parameter optimization in machine learning, aimed at improving model performance.

The `!` at the beginning of the line allows these commands to be executed in a Jupyter Notebook environment.

```
[2]  import optuna
     import pickle
     import pandas as pd
     import numpy as np
     import lightgbm as lgb
     from sklearn.metrics import mean_squared_error
     from sklearn.metrics import r2_score
     from category_encoders import BinaryEncoder
     from sklearn.compose import ColumnTransformer
     from sklearn.preprocessing import LabelEncoder
     from sklearn.model_selection import train_test_split
```

Figure(59)

1. **Importing Libraries**:
   - `import op tuna`: Imports the `op tuna` library, which is used for hyper parameter optimization.
   - `import pickle`: Imports the `pickle` library, which is used for saving and loading Python objects.
   - `import pandas as pd`: Imports the `pandas` library with the alias `pd`, used for data manipulation and analysis.
   - `import numpy as np`: Imports the `numpy` library with the alias `np`, used for numerical computing and array processing.
   - `import light GBM as lgb`: Imports the `light GBM` library with the alias `lgb`, used for implementing gradient boosting models.
   - `From sklearn.metrics import mean_squared_error`: Imports the `mean_squared_error` function from `sklearn`, used to calculate the mean squared error.
   - `From sklearn.metrics import r2_score`: Imports the `r2_score` function from `sklearn`, used to calculate the R^2 score (coefficient of determination).
   - `from category encoders import Binary Encoder`: Imports `Binary Encoder` from `category encoders`, used for encoding categorical variables into binary format.
   - `from sklearn.compose import Column Transformer`: Imports `Column Transformer` from

`sklearn`, used to apply different transformations to specific columns in the data.
    - `from sklearn preprocessing import Label Encoder`: Imports `Label Encoder` from `sklearn`, used to encode categorical labels into numeric form.
    - `from sklearn model selection import train test split`: Imports `train test split` from `sklearn`, used to split the data into training and testing sets.

Usage:

- **Op tuna**: For optimizing hyper parameters of machine learning models.
- **Pickle**: For saving and loading models or data.
- **Pandas**: For data manipulation and analysis.
- **NumPy**: For numerical operations and mathematical computations.
- **Light GBM**: For training high-performance gradient boosting models.
- **Mean Squared Error and R^2 Score**: For evaluating model performance.
- **Binary Encoder and Label Encoder**: For encoding categorical variables.
- **Column Transformer**: For applying specific transformations to certain columns in the data.
- **Train Test Split**: For splitting the dataset into training and testing subsets.

This code sets up the necessary tools and libraries for building and evaluating a machine learning model, including data preprocessing and encoding.

- This code reads data from a CSV file named `"Egypt_Houses_Price.csv"` using the `pandas` library.

```
[3] data = pd.read_csv("Egypt_Houses_Price.csv")
```

```
[4] data
```

	Real_estate_ID	City	Type	Area	Bedrooms	Bathrooms	Furnished	Level	Compound	Payment_Option	Delivery_Date	Delivery_Term	Price
0	1	Nasr City	Duplex	400.0	3	3	No	7	Unknown	Cash	Ready to move	Finished	4000000
1	2	Camp Caesar	Apartment	160.0	3	3	No	10	Unknown	Cash	Ready to move	Finished	4000000
2	3	Smoha	Apartment	165.0	3	2	No	1	Unknown	Cash	Ready to move	Finished	2250000
3	4	Nasr City	Apartment	230.0	3	2	No	10	Unknown	Cash	Ready to move	Finished	1900000
4	5	New Cairo - El Tagamoa	Apartment	160.0	2	3	No	Ground	Eastown	Cash	Ready to move	Semi Finished	5800000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
26688	26689	New Cairo - El Tagamoa	Stand Alone Villa	165.0	4	3	Unknown	Unknown	Unknown	Cash	Unknown	Semi Finished	4800000

North Town

Connected to Python 3 Google Compute Engine backend

Figure(60)

- The data read from the file is stored in a variable called `data` as a Data Frame, which is a data structure in `pandas` similar to a table (like a spreadsheet in Excel).In summary, this code reads house price data in Egypt from a CSV file and loads it into a Data Frame called `data` for further analysis or processing.

```
[5] data = data.drop(columns=['Real_estate_ID'])
```

```
[6] data
```

	City	Type	Area	Bedrooms	Bathrooms	Furnished	Level	Compound	Payment_Option	Delivery_Date	Delivery_Term	Price
0	Nasr City	Duplex	400.0	3	3	No	7	Unknown	Cash	Ready to move	Finished	4000000
1	Camp Caesar	Apartment	160.0	3	3	No	10	Unknown	Cash	Ready to move	Finished	4000000
2	Smoha	Apartment	165.0	3	2	No	1	Unknown	Cash	Ready to move	Finished	2250000
3	Nasr City	Apartment	230.0	3	2	No	10	Unknown	Cash	Ready to move	Finished	1900000
4	New Cairo - El Tagamoa	Apartment	160.0	2	3	No	Ground	Eastown	Cash	Ready to move	Semi Finished	5800000
...	...	...	...	...	...	...	...	...	...	...	...	...
26688	New Cairo - El Tagamoa	Stand Alone Villa	165.0	4	3	Unknown	Unknown	Unknown	Cash	Unknown	Semi Finished	4800000
26689	North Coast	Town House	240.0	3	2	Unknown	Unknown	Unknown	Cash	Unknown	Unknown	890000
26690	New Cairo - El Tagamoa	Town House	218.0	4	3	Unknown	Unknown	Unknown	Cash	Unknown	Finished	4000000
26691	New Cairo - El Tagamoa	Twin House	308.0	3	4	No	Unknown	Cairo Festival City	Cash	Ready to move	Semi Finished	13800000

Connected to Python 3 Google Compute Engine backend

Figure(61)

What the Code Does:

- This code removes the column named 'Real estate ID' from the Data Frame 'data'. After executing this line, the Data Frame 'data' will no longer contain the column 'Real estate ID'.

In summary, this code drops the specified column from the Data Frame, effectively removing it from the dataset.

```
[7]
label_encoders = {}
cat_feats = []
for col in data.columns:
    if data[col].dtype == 'object':
        cat_feats.append(col)
        label_encoders[col] = LabelEncoder()
        data[col] = label_encoders[col].fit_transform(data[col])

def save_label_encoders(label_encoders, filename):
    with open(filename, 'wb') as file:
        pickle.dump(label_encoders, file)

save_label_encoders(label_encoders, 'features_encoders.pkl')
```

```
[8]
data
```

	City	Type	Area	Bedrooms	Bathrooms	Furnished	Level	Compound	Payment_Option	Delivery_Date	Delivery_Term	Price
0	115	2	400.0	3	3	0	7	525	0	6	1	4000000
1	41	0	160.0	3	3	0	1	525	0	6	1	4000000
2	161	0	165.0	3	2	0	0	525	0	6	1	2250000
3	115	0	230.0	3	2	0	1	525	0	6	1	1900000

Connected to Python 3 Google Compute Engine backend

Figure(62)

- **`Label encoders`**: This is a dictionary where each key is the name of a categorical column and the corresponding value is a `Label Encoder` object.
- **`Cat feats`**: This is an empty list that will store the names of categorical columns.
- **Loop**: The code loops through each column in the Data Frame `data`.
- **Check Data Type**: It checks if the data type of the column is `'object'`, indicating that it contains categorical data.
- **Label Encoding**: If the column is categorical, it adds the column name to `cat feats`, creates a `Label Encoder` object for that column, and fits it to the column data to convert categorical labels into numeric form.
- **`Save label encoders` Function**: This function saves the `label encoders` dictionary to a file using pickle serialization.
- **Saving Label Encoders**: It calls the `save label encoders` function to save the label encoders to a file named `'features encoders PKL'`.

What the Code Does:

- This code loops through each column in the Data Frame and encodes categorical columns using `Label Encoder`. It saves the `Label Encoder` objects in a dictionary `label encoders`, where keys are column names, and values are the `Label Encoder` objects.
- It also saves these label encoders to a file named `'features encoders PKL'` using pickle serialization.

In summary, this code preprocesses categorical features by label encoding them and saves the label encoders for later use.

```
[9]  X = data.drop('Price', axis=1)
     y = data['Price']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
```

Figure(63)

- **`X` and `y`**: These variables represent the features (independent variables) and the target variable (dependent variable) respectively. Here, `X` contains all columns except for the column 'Price', and `y` contains only the column 'Price'.
- **Splitting the Dataset**:
  - The dataset is split into training and testing sets using the `train test split` function. `Test size=0.2` specifies that 20% of the data will be used for testing, while the remaining 80% will be used for training.
  - `Random state=42` ensures reproducibility of the results by fixing the random seed.
- **Further Splitting the Training Set**:
  - The training set is further split into training and validation sets using the `train test split` function again. This is often done to have a separate set for validation to tune hyper parameters and prevent over fitting.

- `Test size=0.2` specifies that 20% of the training data will be used for validation, while the remaining 80% will be used for training.

- `Random state=42` ensures consistency in the random splitting process.

What the Code Does:

- The code prepares the data for training a machine learning model by splitting it into features (`X`) and target (`y`) variables.
- It splits the dataset into three sets: training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune hyper parameters and evaluate model performance during training, and the testing set is used to evaluate the final model performance after training.

In summary, this code sets up the data for training, validation, and testing of a machine learning model.

```
[10] # Define the objective function to minimize
     def objective(trial):
         # Suggest values for the hyperparameters
         param = {
                 'objective': 'regression',
                 'metric': 'rmse',
                 'verbosity': -1,
                 'boosting_type': 'gbdt',
                 'lambda_l1': trial.suggest_float('lambda_l1', 1e-8, 10.0),
                 'lambda_l2': trial.suggest_float('lambda_l2', 1e-8, 10.0),
                 'num_leaves': trial.suggest_int('num_leaves', 20, 100),
                 'feature_fraction': trial.suggest_float('feature_fraction', 0.4, 1.0),
                 'bagging_fraction': trial.suggest_float('bagging_fraction', 0.4, 1.0),
                 'bagging_freq': trial.suggest_int('bagging_freq', 1, 7),
                 'min_child_samples': trial.suggest_int('min_child_samples', 20, 100),
             }

         # Create and train LightGBM model
         train_set = lgb.Dataset(X_train, y_train, categorical_feature=cat_feats)
         valid_set = lgb.Dataset(X_valid, y_valid, reference=train_set, categorical_feature=cat_feats)
         model = lgb.train(param, train_set, valid_sets=[valid_set])

         # Predict on validation set
         preds = model.predict(X_valid, num_iteration=model.best_iteration)

         # Calculate RMSE
         r2 = r2_score(y_valid, preds)
[10]     return r2


# Create a study object and specify the optimization direction to 'minimize'
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=50)  # You can adjust the number of trials

# Print the result of hyperparameters
print("Best hyperparameters: ", study.best_trial.params)
print("Best RMSE: ", study.best_trial.value)
```

```
[I 2024-06-09 10:40:04,046] A new study created in memory with name: no-name-67aeb7a7-f237-4a5c-8980-18b1ad3dc232
[I 2024-06-09 10:40:04,566] Trial 0 finished with value: 0.6774473112505535 and parameters: {'lambda_l1': 7.736898169800086, 'lambda_l2': 7.928689553168826, 'num
[I 2024-06-09 10:40:05,646] Trial 1 finished with value: 0.6793015771624905 and parameters: {'lambda_l1': 0.7211519788307089, 'lambda_l2': 9.246841944014243, 'nu
[I 2024-06-09 10:40:06,210] Trial 2 finished with value: 0.6900310252871666 and parameters: {'lambda_l1': 0.2578395512129361, 'lambda_l2': 1.506870638013448, 'nu
[I 2024-06-09 10:40:06,697] Trial 3 finished with value: 0.6775907427091734 and parameters: {'lambda_l1': 8.433410399328851, 'lambda_l2': 2.0042850329454915, 'nu
[I 2024-06-09 10:40:07,314] Trial 4 finished with value: 0.6826800248363184 and parameters: {'lambda_l1': 9.421453288367227, 'lambda_l2': 1.8975794476363743, 'nu
[I 2024-06-09 10:40:08,111] Trial 5 finished with value: 0.6860551672831141 and parameters: {'lambda_l1': 2.420865189747792, 'lambda_l2': 6.737852411897996, 'num
[I 2024-06-09 10:40:08,790] Trial 6 finished with value: 0.6829577988953692 and parameters: {'lambda_l1': 5.383557959418894, 'lambda_l2': 2.5941475250427346, 'nu
[I 2024-06-09 10:40:09,267] Trial 7 finished with value: 0.6714083922899232 and parameters: {'lambda_l1': 0.6101148487634974, 'lambda_l2': 1.1244709762185133, 'nu
[I 2024-06-09 10:40:09,722] Trial 8 finished with value: 0.6881969835277972 and parameters: {'lambda_l1': 2.454701142124997, 'lambda_l2': 6.138438495156284, 'num
[I 2024-06-09 10:40:10,156] Trial 9 finished with value: 0.676889700620779 and parameters: {'lambda_l1': 3.3638391197406405, 'lambda_l2': 2.244536732694904, 'num
[I 2024-06-09 10:40:10,550] Trial 10 finished with value: 0.6817135002640282 and parameters: {'lambda_l1': 5.571965000314072, 'lambda_l2': 4.40904699822698, 'num
[I 2024-06-09 10:40:12,359] Trial 11 finished with value: 0.6881120705101772 and parameters: {'lambda_l1': 2.5644285074440107, 'lambda_l2': 4.929912018911685, 'nu
[I 2024-06-09 10:40:14,017] Trial 12 finished with value: 0.6840058275452952 and parameters: {'lambda_l1': 0.00851163144096842, 'lambda_l2': 6.568612645540639, 'n
[I 2024-06-09 10:40:14,537] Trial 13 finished with value: 0.6858560255630439 and parameters: {'lambda_l1': 1.8351179855792534, 'lambda_l2': 0.24084521348203847,
[I 2024-06-09 10:40:15,020] Trial 14 finished with value: 0.6929435814174827 and parameters: {'lambda_l1': 4.039785310746755, 'lambda_l2': 3.712840801494695, 'nu
```

Connected to Python 3 Google Compute Engine backend

```
[10] [I 2024-06-09 10:40:20,635] Trial 24 finished with value: 0.6839461529451598 and parameters: {'lambda_l1': 4.599523978230781, 'lambda_l2': 4.962347900458303, 'nu
     [I 2024-06-09 10:40:21,181] Trial 25 finished with value: 0.6897307563245826 and parameters: {'lambda_l1': 3.345107640273933, 'lambda_l2': 4.472607578819511, 'nu
     [I 2024-06-09 10:40:22,723] Trial 26 finished with value: 0.6941709344273093 and parameters: {'lambda_l1': 5.112633898839134, 'lambda_l2': 3.872955395578221, 'nu
     [I 2024-06-09 10:40:25,283] Trial 27 finished with value: 0.6960393381016239 and parameters: {'lambda_l1': 6.171418504574549, 'lambda_l2': 4.248114395566523, 'nu
     [I 2024-06-09 10:40:27,189] Trial 28 finished with value: 0.6930675676860927 and parameters: {'lambda_l1': 6.365641259123062, 'lambda_l2': 5.428176377870256, 'nu
     [I 2024-06-09 10:40:27,792] Trial 29 finished with value: 0.6884470193083443 and parameters: {'lambda_l1': 6.279165317098199, 'lambda_l2': 7.641981582121156, 'nu
     [I 2024-06-09 10:40:28,433] Trial 30 finished with value: 0.6881654343528087 and parameters: {'lambda_l1': 7.5183091284311105, 'lambda_l2': 5.601407722856834, 'nu
     [I 2024-06-09 10:40:29,106] Trial 31 finished with value: 0.6868052957677098 and parameters: {'lambda_l1': 7.302449695060881, 'lambda_l2': 4.231753377582065, 'nu
     [I 2024-06-09 10:40:29,788] Trial 32 finished with value: 0.690016958988642 and parameters: {'lambda_l1': 5.413613514787365, 'lambda_l2': 9.538164108193303, 'num
     [I 2024-06-09 10:40:30,475] Trial 33 finished with value: 0.6944801219809489 and parameters: {'lambda_l1': 4.923863397065984, 'lambda_l2': 7.234496685609798, 'nu
     [I 2024-06-09 10:40:31,133] Trial 34 finished with value: 0.6949692833396575 and parameters: {'lambda_l1': 8.277283368136475, 'lambda_l2': 8.592203000940305, 'nu
     [I 2024-06-09 10:40:31,813] Trial 35 finished with value: 0.6926998286688768 and parameters: {'lambda_l1': 8.385421892050733, 'lambda_l2': 8.772869963393566, 'nu
     [I 2024-06-09 10:40:32,510] Trial 36 finished with value: 0.6937736052813147 and parameters: {'lambda_l1': 9.82887629959859, 'lambda_l2': 7.994559485695364, 'num
     [I 2024-06-09 10:40:33,231] Trial 37 finished with value: 0.6931100183124719 and parameters: {'lambda_l1': 8.889918153170889, 'lambda_l2': 8.441340364718746, 'nu
     [I 2024-06-09 10:40:33,923] Trial 38 finished with value: 0.6907897685270782 and parameters: {'lambda_l1': 4.947429578403166, 'lambda_l2': 7.138279586376923, 'nu
     [I 2024-06-09 10:40:34,611] Trial 39 finished with value: 0.6925541690983748 and parameters: {'lambda_l1': 7.1450683450012225, 'lambda_l2': 8.742759020833175, 'nu
     [I 2024-06-09 10:40:35,235] Trial 40 finished with value: 0.6867807609517651 and parameters: {'lambda_l1': 8.038337600973943, 'lambda_l2': 9.764136624256556, 'nu
     [I 2024-06-09 10:40:36,003] Trial 41 finished with value: 0.6933818050935993 and parameters: {'lambda_l1': 9.879258706150841, 'lambda_l2': 7.722015907548424, 'nu
     [I 2024-06-09 10:40:36,699] Trial 42 finished with value: 0.6886241978379356 and parameters: {'lambda_l1': 8.993075062201724, 'lambda_l2': 8.124054941259239, 'nu
     [I 2024-06-09 10:40:39,185] Trial 43 finished with value: 0.6927485318461218 and parameters: {'lambda_l1': 9.891564141313356, 'lambda_l2': 8.976141398463849, 'nu
     [I 2024-06-09 10:40:40,491] Trial 44 finished with value: 0.6892707849945763 and parameters: {'lambda_l1': 4.964334939496506, 'lambda_l2': 7.424670381193298, 'nu
     [I 2024-06-09 10:40:41,100] Trial 45 finished with value: 0.6897436390910243 and parameters: {'lambda_l1': 5.861671406953008, 'lambda_l2': 6.783453989609653, 'nu
     [I 2024-06-09 10:40:41,690] Trial 46 finished with value: 0.6835569473128011 and parameters: {'lambda_l1': 8.993445590436737, 'lambda_l2': 6.215637520194947, 'nu
     [I 2024-06-09 10:40:42,375] Trial 47 finished with value: 0.6936556351229247 and parameters: {'lambda_l1': 7.882770001403595, 'lambda_l2': 8.279822480466697, 'nu
     [I 2024-06-09 10:40:43,123] Trial 48 finished with value: 0.6890879074995125 and parameters: {'lambda_l1': 6.808020117063667, 'lambda_l2': 2.5705758762394604, 'nu
     [I 2024-06-09 10:40:43,858] Trial 49 finished with value: 0.695167851613147 and parameters: {'lambda_l1': 9.494042037126146, 'lambda_l2': 7.171763574401794, 'num
     Best hyperparameters:  {'lambda_l1': 6.171418504574549, 'lambda_l2': 4.248114395566523, 'num_leaves': 77, 'feature_fraction': 0.5910238536004729, 'bagging_fracti
     Best RMSE:  0.6960393381016239
```

Connected to Python 3 Google Compute Engine backend

Figure(64)

- **Objective Function**:
  - The `objective` function is defined to minimize the validation R^2 score by tuning hyper parameters.
  - It takes a `trial` object as input, which suggests hyper parameter values to be evaluated.
- **Hyper parameter Tuning**:
  - The hyper parameters to be optimized are specified within the `param` dictionary.
  - `Trial suggest float` and `trial suggest INT` are used to suggest float and integer values for hyper parameters within specified ranges.
- **Model Training**:
  - A Light GBM model is created and trained using the suggested hyper parameters.
  - Training and validation sets are used with categorical features specified.
- **Model Evaluation**:
  - The trained model is used to make predictions on the validation set.
  - The R^2 score is calculated between the predicted and actual values.
- **Op tuna Study**:
  - An Op tuna `study` object is created with the optimization direction set to 'maximize' since we want to maximize the validation R^2 score.
- **Optimization**:
  - The `study optimize` function is called with the `objective` function and the number of trials (`n trials`) to perform the optimization.
  - Op tuna will run the objective function with different sets of hyper parameters, seeking the best R^2 score.
- **Best Hyper parameters**:
  - The best hyper parameters and corresponding R^2 score are printed after optimization.

What the Code Does:

- This code performs hyper parameter optimization using Optuna for a Light GBM model.
- It tunes various hyper parameters such as regularization parameters, number of leaves, and feature fraction.
- The objective is to maximize the R^2 score on a validation set.

- After optimization, the best hyper parameters and the corresponding R^2 score are printed.

In summary, this code automates the process of hyper parameter tuning to optimize the performance of a Light GBM model.

```
[11] X_train_final = pd.concat([X_train, X_valid], axis=0)
     y_train_final = pd.concat([y_train, y_valid], axis=0)

     train_set = lgb.Dataset(X_train_final, y_train_final, categorical_feature=cat_feats)
```

Figure(65)

- **Combining Training and Validation Data**:
  - `X_train` and `X_valid` contain features for the training and validation sets respectively.
  - Similarly, `y_train` and `y_valid` contain target values for the training and validation sets.
  - `pd.concat` function is used to concatenate (combine) `X_train` and `X_valid` along axis 0 (rows), resulting in `X_train_final`.
  - Similarly, `y_train` and `y_valid` are concatenated to create `y_train_final`.

- **Creating LightGBM Dataset**:

- `train_set` is created using `lgb.Dataset`, which is a Light GBM-specific data structure for training.
- It takes `X_train_final` and `y_train_final` as input, along with `categorical_feature=cat_feats`.
- `Categorical_feature=cat_feats` specifies the categorical features in the dataset, ensuring they are treated as categorical during training.

What the Code Does:

- This code combines the training and validation data to create a larger dataset for training.
- The features (`X_train_final`) and target (`y_train_final`) are concatenated along rows to create a single training dataset.
- The concatenated dataset is then used to create a Light GBM training dataset (`train_set`) for model training.

In summary, this code prepares the final training dataset and creates a Light GBM dataset object for training the model.

```
model = lgb.train(study.best_trial.params, train_set)
preds = model.predict(X_test, num_iteration=model.best_iteration)

# Calculate RMSE
r2 = r2_score(y_test, preds)

print('R2 Score of Final Test Data: ', r2)
```

```
[LightGBM] [Warning] Categorical features with more bins than the configured maximum bin number found.
[LightGBM] [Warning] For categorical features, max_bin and max_bin_by_feature may be ignored with a large number of categories.
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000465 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 802
[LightGBM] [Info] Number of data points in the train set: 21354, number of used features: 10
[LightGBM] [Info] Start training from score 4393631.647326
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
R2 Score of Final Test Data:  0.6686650451054772
```

Figure(66)

Model Training:

- The Light GBM model is trained using the best hyper parameters obtained from the Op tuna study (`study.best_trial.params`).
- The `train_set` dataset, which contains the concatenated training and validation data, is used for training.

Making Predictions:

- The trained model is used to make predictions on the test set (`X_test`) using the `predict` method.
- `num_iteration=model.best_iteration` specifies the number of boosting iterations to use for making predictions, which is set to the best iteration found during training.

Model Evaluation:

- The R^2 score is calculated between the predicted values (`preds`) and the actual target values (`y_test`) using the `r2_score` function.
- **Printing Results**:
  - The R^2 score of the final test data is printed to evaluate the performance of the model on unseen data.

What the Code Does:

- This code trains the Light GBM model using the best hyper parameters obtained from the Op tuna study.
- It then makes predictions on the test set and evaluates the model performance using the R^2 score.
- Finally, it prints the R^2 score to assess how well the model generalizes to unseen data.

In summary, this code completes the training and evaluation of the Light GBM model and provides the final evaluation metric on the test data.

```
[15] model.save_model('final_model.txt')

<lightgbm.basic.Booster at 0x7849bdf96170>
```

```
[16] def load_label_encoders(filename):
         with open(filename, 'rb') as file:
             label_encoders = pickle.load(file)
         return label_encoders

     def load_prediction_model(filename):
         loaded_model = lgb.Booster(model_file=filename)
         return loaded_model

     def predict_house_price(model, label_encoders, input_data):
         """
         Predict the price of a house using a machine learning model.

         Parameters:
         - model: The trained machine learning model (should implement the sklearn interface).
         - label_encoders: A dictionary of sklearn.preprocessing.LabelEncoder objects for categorical features.
         - input_data: A dictionary containing the house features.

         Returns:
         - float: The predicted house price.
         """
         # Define the features based on the input requirement
         features = ['City', 'Type', 'Area', 'Bedrooms', 'Bathrooms', 'Furnished', 'Level',
                     'Compound', 'Payment_Option', 'Delivery_Date', 'Delivery_Term']

         # Create a DataFrame from the input data
         input_df = pd.DataFrame([input_data], columns=features)

         # Encode categorical features
         categorical_features = ['City', 'Type', 'Furnished', 'Level', 'Compound', 'Payment_Option',
                                 'Delivery_Date', 'Delivery_Term']
         for feature in categorical_features:
             if feature in label_encoders:
                 # Encode the feature using the corresponding label encoder
                 input_df[feature] = label_encoders[feature].transform(input_df[feature])
             else:
                 raise ValueError(f"Label encoder for {feature} is missing in the provided 'label_encoders' dictionary.")



         # Predict the house price
         predicted_price = model.predict(input_df)

         return predicted_price[0]


     # Example usage:
     label_encoders = load_label_encoders(filename='features_encoders.pkl')
     model = load_prediction_model(filename='final_model.txt')
```

Figure(67)

```
[16]  model = load_prediction_model(filename='final_model.txt')
      input_features = {'City': 'Camp Caesar',
                        'Type': 'Apartment',
                        'Area': 160,
                        'Bedrooms': 3,
                        'Bathrooms': 3,
                        'Furnished': 'No',
                        'Level': '10',
                        'Compound': 'Unknown',
                        'Payment_Option': 'Cash',
                        'Delivery_Date': 'Ready to move',
                        'Delivery_Term': 'Finished'}
      predicted_price = predict_house_price(model, label_encoders, input_features)
      print(f"The predicted price of the house is: ${predicted_price}")

      The predicted price of the house is: $2406994.142018563
```

Figure(68)

`load_label_encoders` Function:

- This function loads label encoders from a pickle file.
- It takes the filename as input and returns a dictionary containing the label encoders.

`load_prediction_model` Function:

- This function loads a LightGBM prediction model from a file.
- It takes the filename as input and returns the loaded model.

`predict_house_price` Function:

- This function predicts the price of a house using a trained machine learning model.
- It takes the model, label encoders, and input data as inputs.
- It returns the predicted house price as a float.

What the Code Does:

- These functions are designed to load a trained machine learning model, label encoders, and make predictions on new data.
- `load_label_encoders` loads label encoders from a file containing a dictionary of label encoders.
- `load_prediction_model` loads a LightGBM prediction model from a file.

- `predict_house_price` takes the loaded model, label encoders, and input data, and returns the predicted house price.

In summary, these functions facilitate the process of loading trained models and making predictions on new data.

Summary

In summary, We talked about the machine and how to benefit from it on our website to meet the user's purposes and the types of machines. We also talked about the model used and we extracted the results and compared them with another model to know the extent of the efficiency of the model and the algorithm used. Based on the results of all previous algorithms, it appeared to us that the best performance and effectiveness in terms of accuracy is the Light GBM algorithm, so we used it to build our website for real estate price prediction
We hope that we have explained to you all the details regarding choosing the algorithm used to predict prices on our website. In the end, thank you

Chapter 7
PROJECT CONCLUSION AND FUTURE WORK

Introduction

In this chapter, we will mention the strengths and weaknesses of the site. We will also talk about what we would like to do in the future to develop the site

Overall Strengths

Automation: The model can automate the price forecasting process, saving time and effort compared to manual assessments.

Scalability: With the right design and infrastructure, the model can be scaled to handle a larger volume of real estate data and forecasts.

Data-driven insights: The model can identify patterns and relationships in data that may not be readily apparent through traditional methods, which can provide valuable insights into market trends.

Customization: The form can be customized to include specific user requirements or evaluation criteria.

Overall Weaknesses

Market Volatility: The model may not accurately capture sudden or unexpected changes in the real estate market, such as an economic downturn or rising interest rates. Consider incorporating economic indicators or using time series forecasting techniques.

Location specificity: Model performance may be limited to the specific geographic region used for training. It may not generalize well to other sites with different market dynamics. Train the model on data from a wider region or develop separate models for different regions.

Future Work

Data acquisition: Explore ways to obtain more comprehensive, high-quality data, perhaps from public records, real estate websites, or collaboration with real estate agents.

Model Improvement: We will experiment with different model architectures, such as ensemble methods or deep learning techniques, to enhance prediction accuracy.

Real-time integration: Consider integrating the model with real estate platforms or apps to provide real-time price estimates to users.

Market Sensitivity: Explore incorporating economic indicators or time series forecasting techniques to improve the model's sensitivity to market fluctuations.

Location expansion: Train the model on data from a wider geographic area or develop separate models for different locations to improve generalizability.

Error Analysis: Continuously monitor and analyze model performance to identify and address any sources of errors or biases.

Summary

In summary, We learned about the strengths that characterize our site, as well as the weak points, and what we will do in future work to develop our site and make it suitable for all parties around the world.

REFERENCES

colab. (n.d.). *colab*. Retrieved novamber 13, 2023, from colab: https://colab.research.google.com/drive/1StA008Jbyvc4v492JcpZGr06_9W50gpB?usp=sharing

kaggel. (n.d.). *kaggel*. Retrieved novamber 9, 2023, from https://www.kaggle.com/datasets/ahmedsamirafattah/egypt-houses-price

lightGBM. (n.d.). Retrieved Decamber 23, 2023, from https://scholar.google.com/scholar?hl=ar&as_sdt=0%2C5&as_ylo=2020&q=A+Linear+Regression+Model+for+House+Price+Prediction&btnG=#d=gs_qabs&t=1715610050167&u=%23p%3DABVui2jkB3oJ

Regressor, r. (n.d.). Retrieved Decamber 22, 2023, from https://esajournals.onlinelibrary.wiley.com/doi/full/10.1890/07-0539.1

regrestion, l. (n.d.). Retrieved decamber 24, 2023, from https://scholar.google.com/scholar?hl=ar&as_sdt=0%2C5&as_ylo=2020&q=A+Linear+Regression+Model+for+House+Price+Prediction&btnG=#d=gs_qabs&t=1715610050167&u=%23p%3DABVui2jkB3oJ

YARDI. (2023, 10 15). RETRIEVED FROM YARDI : HTTPS://WWW.YARDI.COM/

ZILLOW. (2023, 10 15). RETRIEVED FROM ZILLOW : HTTPS://WWW.ZILLOW.COM/

AQARATIKOM. (2023, 10 20). RETRIEVED FROM AQARATIKOM: HTTPS://WWW. AQARATIKOM.COM/

KAGGLE. (2023, 10 25). RETRIEVED FROM KAGGLE: HTTPS://WWW.KAGGLE.COM/DATASETS/AHMEDSHAHRIARSAKIB/USA-REAL-ESTATE-DATASET

CANVA. (2023, 11 2). RETRIEVED FROM CANVA : HTTPS://WWW.CANVA.COM/