HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

Building a search and visualization system for real estate data from multiple sources

ĐINH HỮU ĐẠI
dai.dh194735@sis.hust.edu.vn

Major: Information Technology

Supervisor: Ph.D Vũ Tuyết Trinh _____

Signature

Department: Computer Science

School: Information and Communications Technology

HANOI, 06/2024

ACKNOWLEDGMENTS

I am very grateful to Ms. Vũ Tuyết Trinh for her unwavering support and direction, which have greatly inspired me and helped me finish my graduation thesis. In addition, the words of support and encouragement from my devoted friends and family have been invaluable in helping me overcome obstacles in my academic career.

Thank you for everything that you have done for me in my life and profession, Ms. Vũ Tuyết Trinh, as well as to my family and close friends. The extent of my gratitude and respect for each and every one of you cannot be adequately expressed by these few words. I commit to keeping up my relentless efforts, advancing society, and making everyone happy and joyful. Thank you all so much!

ABSTRACT

The project "Building a Search and Visualization System for Real Estate Data from Multiple Sources" aims to consolidate, process, and integrate real estate data collected from various online platforms. This system provides users with up-to-date information, enabling quick searches and comprehensive visualization of the current housing market.

In the digital era, the demand for online real estate information is on the rise, particularly for individuals seeking specific residential properties. However, finding suitable properties can be challenging and time-consuming, especially with the ongoing rise in housing prices. To address this issue, the project focuses on creating user-friendly interfaces that enhance accessibility and efficiency and need a robust approach to collecting, integrating real estate data. Users will be able to access real estate data, utilize search functions, and receive additional support for a thorough assessment of the housing market, providing a broader perspective and more options to find the most suitable property.

The primary contribution of this thesis is the successful development and implementation of a system that collects and processes real estate data from multiple online sources into a unified provisioning system. The key achievement is the creation of a system that delivers continuously updated real estate information, integrates fast and accurate search capabilities, and offers data-driven analytics to provide users with a comprehensive view of the market trends and current housing demands. This facilitates users in saving time and gaining insights while exploring suitable property options.

Looking ahead, the project has the potential to expand into a platform that allows users to post real estate listings and engage with brokers. This expansion could include features such as listing functionalities for brokers, creating an open social network that facilitates postings, application management, and real estate data searches.

Student

(Signature and full name)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abriviation	Full Expression
API	Giao diện lập trình ứng dụng (Application Programming Interface)
HTML	Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language)

CHAPTER 1. INTRODUCTION

1.1 Motivation

The real estate market stands as a cornerstone of economic vitality, influencing investment decisions, urban planning initiatives, and financial forecasts. Yet, amid the wealth of information available online, navigating the complexities of the real estate landscape has become increasingly daunting. Traditional search methods are struggling to keep pace with the evolving demands for timely, accurate data, leading to inefficiencies and missed opportunities.

In response to these challenges, there is a pressing need for a holistic solution that can seamlessly aggregate, refine, and present real estate data from diverse sources. This project emerges as a response to this imperative, driven by the vision to bridge the gap between data abundance and actionable insights. By harnessing the power of advanced technologies, such as data integration, cleansing algorithms, and intuitive visualization tools, the project aims to empower stakeholders with a comprehensive understanding of the real estate market dynamics.

Moreover, the project endeavors to not only compile data from multiple sources but also to deliver it in a manner that is accessible and user-friendly. By prioritizing ease of use and intuitive design, catering to a broad spectrum of users, from seasoned investors to first-time home buyers. Ultimately, the goal is enabling stakeholders to make informed decisions with confidence and clarity.

1.2 Objectives and scope of the graduation thesis

Existing real estate platforms often struggle with various limitations, including incomplete data, lack of integration, and inadequate visualization capabilities. To address these challenges, this project aims to develop a comprehensive solution in the form of a web application.

The primary objectives of this project include:

Data Collection: The system needs to be able to gather information from a variety of sources; for this project, real estate websites are my main focus. This entails setting up systems and frameworks to efficiently gather data on real estate addresses, costs, areas, kinds, and comprehensive descriptions. In addition to handling various site structures, the data gathering procedure must guarantee that the information gathered is correct and current.

Data Integration and Cleaning: Data from different sources need to be inte-

grated and standardized to ensure consistency and usability. This requires merging datasets from various sources and aligning formats to create a unified dataset. This involves standardizing formats, resolving discrepancies, and removing duplicates to enhance the quality of the dataset.

Database Storage: Establish a structured database to efficiently store the cleaned and integrated real estate data. Proper database design is crucial for optimizing data retrieval and management processes. The database should be designed to efficiently manage and query real estate information, supporting various types of searches and analyses

Data Visualization: Implement advanced visualization techniques to present the real estate data in an intuitive and interactive manner. The focal point of visualization will be an map, allowing users to explore properties spatially and interactively. The visualizations should be dynamic, allowing users to filter and manipulate the data.

The project is oriented to solve the challenges of collecting and displaying real estate data through a web application. **However, the scope of the project includes the following limitations to accommodate implementation capabilities and specific purposes:**

Advanced Predictive Analytics: The project focuses on data visualization and basic search functionalities. Advanced predictive analytics, such as forecasting property prices using machine learning algorithms or assessing market trends based on historical data, are not within the scope of this thesis.

User Authentication and Management: Although essential for a complete web application, user authentication, authorization, and management functionalities are not included. The project concentrates solely on data handling and visualization without implementing user accounts, roles, or permissions.

Mobile Application Development: The thesis focuses on a web-based application accessible via desktop and mobile browsers. Native mobile application development for platforms such as iOS and Android is not covered in this project.

Real-time Data Updates: The system does not include real-time data fetching or updating. There is no requirement for real-time synchronization with external data sources when the collection execution shows the property state as of the most recent update.

Search suggestions based on user location: The thesis does not incorporate the use of the user's location to offer recommendations regarding the real estate

distance between their current location and the real estate.

By achieving these objectives and adhering to the defined scope, the project aims to address the shortcomings of existing real estate platforms and provide users with a powerful tool for accessing and analyzing real estate data effectively.

1.3 Tentative solution

Building upon the objectives outlined above, the proposed solution follows a structured approach:

Identification of Solution Direction: The project will explore various methods, techniques, and technologies to address the identified challenges in real estate data management and visualization.

Description of Proposed Solution: The solution will involve the development of a web application equipped with robust data visualization features. It will leverage modern technologies and best practices to ensure efficiency and usability.

Primary Contributions: The project's primary contribution lies in providing a comprehensive solution to the challenges faced by existing real estate platforms. By offering enhanced data collection, integration, and visualization capabilities, the solution aims to empower users with valuable insights and facilitate informed decision-making in the real estate domain.

1.4 Thesis organization

The thesis comprises fives chapters, each addressing a specific aspect of the project:

Chapter 1 introduces the project, outlining its motivation, objectives, scope and tentative solution. It also provides an overview of the thesis structure.

Chapter 2 analyzes the overall system, indentifies its challenges in data integration and visualization; discusses about system architecture

Chapter 3 delves into data integration, discussing the technologies employed for data collection and integration. It elaborates on the solutions to challenges outlined in Chapter 2, including methods for data cleaning and standardization and after that is storage process

Chapter 4 focuses on data visualization, presenting the technologies used for this purpose. It describes the processes and methods utilized to create interactive maps and integrate them into the website. Additionally, it covers the implementation of search functionality.

Chapter 5 offers a conclusion, summarizing the project's outcomes, reflecting

on its achievements and limitations, and proposing potential avenues for future work.

The necessity for efficient data administration and presentation has been brought to light by the quick developments in online real estate apps. In order to address these issues, our project has created an extensive online application that unifies, purifies, and displays real estate data. Our goals are to collect data from dependable sources, ensure data consistency through integration and cleansing, and display the data in an interactive map style.

CHAPTER 2. OVERALL SYSTEM

In the dynamic real estate market, access to reliable data is paramount. The goal of this thesis is to the gathering, combining, and displaying of real estate data from multiple sources. Chapter 2 will include a detailed analysis of the requirements, a discussion of the difficulties encountered during the data collection, integration, and visualization processes, as well as an overview of the system architecture design

2.1 Problem Challenges

2.1.1 Data Integration Challenges

a, Collect data from from Different Sources

There are now a lot of real estate websites developed and in use in Vietnam. Among the trustworthy websites that were examined are meedyland, nhdat24h, alonhadat, and batdongsan.com.vn. These websites are autonomous, but their primary purpose is to host real estate listings. Depepending my scope and servey, I am only concerned in estate data and how it is used. The result draws attention to an important problem: the inconsistent and fragmented data from many sources. Estate Information are usually restricted to the websites where they are published, making it more difficult for users to find . In order to solve this, my idea combines data into a single, current system, removing the need for consumers to visit many websites.

Through the survey, it was found that the structure in all 5 above website uses pagination. Despite its benefit for user experience, causes some complications for web scraping efforts. Batdongsan.com uses dynamic content loading as users navigate through pages, requiring the scraper to process JavaScript to ensure that all data is fetched correctly, which is more complicated than scraping HTML on the other 4 sites.

I also need to handle multiple pages: Navigating through these pages requires careful handling to ensure full data. This involves constructing the correct URLs for each page. Example On muaban.net, listings are spread across pages that can be accessed via URLs like ‘https://muaban.net/batdongsan/bannhadatchungcu#page=1‘, ‘#page=2‘

Incomplete Data Links: Links to detailed property information after crawl often need to be constructed by appending relative URLs to a base URL: Example: A

relative link like ‘/ban-nha-dat/chung-cu-123‘ on muaban.net must be combined with the base URL ‘https://muaban.net‘.

Different websites often have varying HTML structures and naming conventions for their elements. This requires parsing logic for each site. Example: On nhadat24h, property prices might be contained in a ‘‘ tag with a class ‘price‘, while on alonhadat, the same information could be in a ‘ ‘ tag with a class ‘listing-price‘. Logic must be written for each website to correctly extract the data.

b, Inconsistent and Incomplete Data

Data collected from five different real estate websites is saved to a JSON file. Based on the detailed survey and analysis above, it was found that these data fields have inconsistencies and errors in units, making it difficult to search and compare real estate properties. To solve these problems, data needs to be standardized.

Inconsistency in price:

The price field is stored inconsistently across the 5 websites. These discrepancies occur not only between different sites but also within the same site. The issues identified include varying units, decimal symbols, and non-numeric representations of prices.

On alonhadat.com, the price might be listed as ’1.5 tỷ VND,’ while another source might represent the same value as ’1 tỷ 500 triệu VND.’ These need to be standardized to a single format such as ’1.2’

On muaban.net, the price could be represented as ’1,2 tỷ VND’ (using commas), while on nhadat24h, it might be ’1.2 tỷ VND’ (using a decimal point). These should be normalized to use a consistent decimal point representation like ’1.2’.

Special Cases: Occasionally, the price is listed as "thỏa thuận" (negotiable), which should be handled appropriately, possibly by flagging it for manual review or setting it as a special category.

Conflict area and width:

The area and width fields also exhibit inconsistencies in units and decimal points across all of 5 websites.

For example, the page batdongsan.com area is: ’12.5m2’ to represent the real estate price, while another source uses ’12.5 m@’. This data needs to be renormalized as ’12.5’

In addition, the meedyland page has width = 4.2 m, the nhadat24h page records

it as 4.2, which needs to be standardized as ’4.2

Inconsistent in bathroom, bedroom, floor:

To facilitate searching and unify the data, the fields for the number of bathrooms, bedrooms, and floors need to be converted into numeric values.

On muaban.net and batdongsan.com, bedrooms might be listed as ’2 phòng ’. The term ’phòng’ should be removed to standardize it to ’2.’

On batdongsan.com, bathrooms might be listed as ’3 phòng’, and on muaban.net as ’3 WC.’ The terms ’phòng’ and ’WC’ should be removed to standardize it to ’3.’

On batdongsan.com, floors might be listed as ’3 tầng’, and on nhadat24h as ’3 (tầng).’ The terms ’tầng’ and ’(tầng)’ should be removed to standardize it to ’3.’

Address formats: For real estate searches to be effective, it is necessary to extract two additional fields: Province/City and District. After that, these fields also need to be extracted from the address and stripped of any administrative units.

Example: An address on batdongsan.com might be listed as "Ngoc Khanh Street, Kim Ma Ward, Ba Dinh District, Hanoi." After processing, it should be broken down into:
Province/City: "Hanoi" and District: "Ba Dinh". This needs to be done with all 5 real estate websites.

2.1.2 Data Visualization

a, Address in Text Format

Real estate addresses are typically stored as text strings, which lack the geographical coordinates necessary for mapping and spatial analysis. This limitation poses a challenge for visualizing property locations accurately on maps. Geocoding, the process of converting addresses into latitude and longitude coordinates, is essential to overcome this challenge.

For instance, consider the address ’123 Main Street, City, Country.’ as a text string. However, through the process of geocoding, this address can be converted into specific geographical coordinates, such as latitude: 40.7128° N and longitude: 74.0060° W. . This transformation is crucial for creating detailed and informative maps that can display the distribution of properties, analyze spatial patterns, and provide users with an intuitive way to explore real estate data.

b, Difficulty in Filtering Data for Searching

Efficiently filtering real estate data based on user-defined criteria such as price, area, and property type requires optimized search functionality. Database indexing, query optimization, and user interface design play crucial roles in facilitating responsive and accurate data retrieval.

Example: Implementing a search feature that allows users to filter properties by price range involves optimizing database indexes for the 'price' field and designing intuitive user interfaces for specifying price ranges.

c, Difficulty in Integrating maps

Integrating maps into real estate application interfaces poses several challenges, particularly in managing data visualization effectively. For instance, handling large datasets of property listings and ensuring smooth interaction while displaying markers, overlays, and additional data layers can strain performance and user experience.Intricate UI/UX design is needed to balance comprehensive property information with map interactivity, such as filtering by location or property type, avoiding overwhelming users.

For instance, when consumers are looking for real estate, the precise location as well as some of the property's most crucial features—such as its picture, price, amenities, and area—must be displayed. This combination allows for seamless property exploration and filtering on the map without compromising speed for users.

By addressing these challenges with appropriate strategies and technologies, the real estate data system can achieve seamless integration, robust visualization, and insightful analysis, empowering users with valuable information for informed decision-making.

2.2 System overall architecture

In the previous sections, I analyzed the requirements, outlined the challenges, and identified the limits of the scope of this project. This section will outline the system's overall design and explain each component's function in detail, building on this application. This methodical strategy guarantees a smooth system development process, establishing a platform that is easy to use for obtaining and comprehending real estate data and laying the groundwork for setting the stage for processes and detailed solutions are discussed in the following chapters.

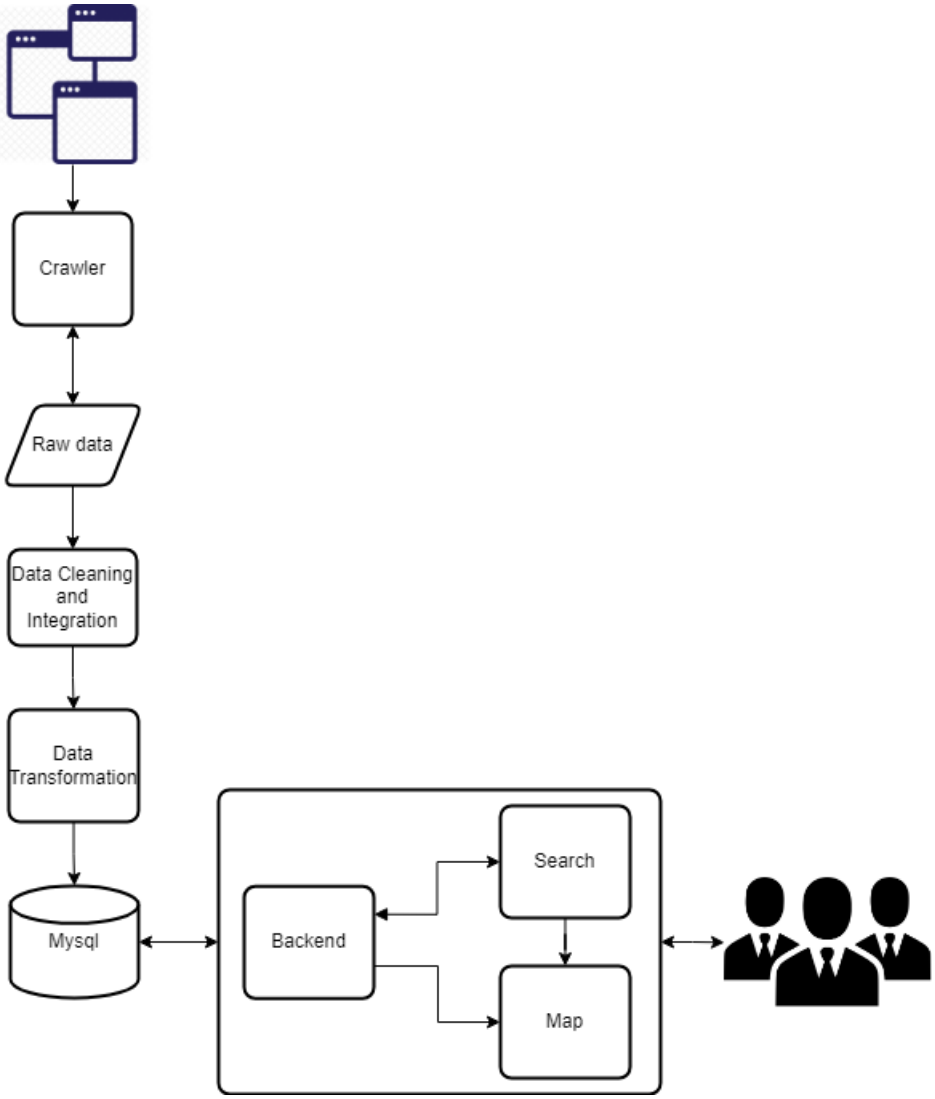

Figure 2.1: Overall Architecture

Collection and Integration Data

First, I will carefully select the data sources from Vietnamese real estate websites. The websites that are selected should primarily provide a large and varied selection of data, covering a broad spectrum of property types, locations, and price points. It's also critical that the most recent real estate listings and market trends are consistently added to these websites. It is imperative that these websites do not contain any features that restrict or forbid access to data collecting. These limitations may make it more difficult to collect thorough data and jeopardize the dataset's accuracy and completeness.

The crawler is a crucial component of this system, tasked with the important role of collecting data and properties of real estate from various websites. This process

involves systematically navigating through web pages to gather relevant information such as addresses, prices, sizes, and other attributes. The crawler ensures the system's dataset complete and up to date by constantly gathering and updating this data. This component is crucial in order to provide a strong basis for further data processing and analysis.

The collected raw data is stored in a component known as Raw Data. The raw data is separated into two categories: txt files and JSON files, as a result of the structure of websites and the data crawling procedure. The txt files contain links of individual real estate listings but do not include detailed information. These links serve as pointers for further data extraction.Conversely, each real estate property's specific characteristics and attributes, like its address, price, size, and other pertinent information, are stored in JSON files. This structured format allows for easy access and manipulation of the data during the processing and normalization stages.

The data cleaning component plays a critical role in ensuring the quality and reliability of the collected real estate data in file JSON of Raw data. The cleaning process includes tasks such as removing duplicates, filling in missing values, correcting erroneous entries, and standardizing formats across different data types. The data integration component is responsible for combining the cleaned data from various sources into a cohesive and unified dataset. During integration, data from multiple sources is reconciled to ensure consistency and completeness into a file CSV. To implement this component, I use the geopy library, the opencagedata.com api to coordinate addresses, as well as use functions of the pandas library to process JSON data.

The data transformation component use pymysql library to convert data from CSV files, which have been cleaned and integrated, into the MySQL database. This process ensures that the data is formatted correctly for storage and retrieval in MySQL, with careful attention to constraints between tables to avoid duplicates and maintain database integrity.

Database component: The main database used by the data transformer component to transform and push in data from CSV files is MySQL. The four main tables that make up the database's structure are specifically designed to accommodate various facets of the website's search capabilities. These tables facilitate effective data management and query performance by managing many kinds of real estate data, including property details, locations, pricing, and qualities. The system queries these tables in response to a user's search request, providing fast and pre-

cise retrieval of pertinent data and enabling sophisticated searches, filtering, and sorting.

Data Visualization

The backend component uses Django to provide APIs that take user requests from the frontend and use these APIs to pull the required data out of the MySQL database and send it back to the frontend.

The Map component is a crucial part of the system, utilizing the Leaflet library in React to visually represent the locations of real estate properties on a map. When users interact with the icons representing these properties, basic information such as address, price, area, and photos are displayed. Users can choose to view detailed pages if they are interested. Additionally, the component features a heatmap function to visualize density of property folowing search component.

The search component is a critical frontend element built with ReactJS.It functions by recording user selections and submitting requests for data retrieval to the backend. The search component obtains the pertinent data and incorporates it into the frontend when the backend processes these queries.Furthermore, according to the search demands, the component easily integrates a map element that improves user experience by visualizing property locations and perhaps density.

In this section, I have outlined the difficulties and challenges that must be addressed during the processes of data collection, integration, and visualization. Furthermore, I have provided an overview of the design of the overall system architecture.

CHAPTER 3. DATA COLLECTION AND INTEGRATION

3.1 Technology

Modern data-driven projects rely heavily on a suite of technologies to collect, process, and analyze data efficiently. In this section, we discuss the key technologies utilized in our project for data extraction, manipulation, and storage. Each technology plays a crucial role in streamlining the data pipeline, from web scraping to database management, enabling us to effectively gather and utilize data for our project's objectives.

3.1.1 Beautifulsoup library

Beautiful Soup is a Python library designed for parsing HTML and XML documents, making it indispensable for extracting structured data from web pages. It integrates seamlessly with popular parsers like lxml and html5lib, providing Pythonic methods for navigating, searching, and manipulating the parse tree.

Developers rely on Beautiful Soup to automate the extraction of data, saving significant time compared to manual parsing. Its ability to handle malformed HTML gracefully enhances its utility in real-world web scraping tasks. By simplifying the process of locating specific elements and extracting data, Beautiful Soup ensures consistency and reliability in data retrieval from static web pages.

3.1.2 Selenium Library

Selenium is a powerful tool for automating web browsers, essential for scenarios requiring interaction with dynamic web elements and complex user interactions. It supports multiple browsers and can execute JavaScript, making it ideal for scraping websites that dynamically load content or rely on client-side scripting frameworks like React or Angular.

In this project, Beautiful Soup complements Selenium by parsing the initial HTML structure fetched by Selenium. Selenium handles tasks such as simulating user actions (clicks, form submissions) and navigating through AJAX-loaded content or pages with infinite scrolling. This combination ensures comprehensive data extraction from modern web applications, where static HTML parsing alone may not suffice.

The integration of Beautiful Soup and Selenium optimizes web scraping for building a robust real estate application. Beautiful Soup efficiently parses static HTML content, while Selenium automates interactions with dynamic elements,

ensuring thorough data extraction from diverse web page architectures.

3.1.3 Requests library

3.1.2. Requests Requests, a Python library, streamlines the process of sending HTTP requests and handling responses. Its simple yet powerful API simplifies interactions with web services, making it an essential tool for fetching web pages and data from URLs. Requests, a Python library, plays a vital role in sending HTTP requests and managing responses. Its straightforward API facilitates seamless communication with web services, enabling effortless retrieval of web pages and data from URLs in various web-related tasks.

3.1.4 Pandas library

Pandas is a versatile Python library essential for data manipulation and analysis, offering powerful data structures like Series and DataFrame. It excels in efficiently handling large datasets and supports operations such as data cleaning, transformation, and analysis. Pandas reads data from various formats like CSV, Excel, and SQL databases, enhancing its flexibility. With its comprehensive functions for merging, reshaping, and aggregating data, Pandas simplifies complex data tasks, speeding up data preparation and enabling effective analysis and visualization.

In this project, Pandas plays a crucial role in the normalization of real estate data. Data normalization involves structuring data to reduce redundancy and improve data integrity. Pandas facilitates this process through its robust functionalities for data cleaning and transformation. By using Pandas, we can efficiently handle missing values, standardize data formats, and ensure consistency across the dataset. This ensures that the real estate data collected from various sources is uniform and ready for integration into the MySQL database, ultimately enhancing the reliability and usability of the data in our application.

3.1.5 Geopy Library

Geopy is a Python library tailored for geocoding and reverse geocoding tasks, essential in transforming addresses into geographic coordinates and vice versa. It supports seamless integration with various map providers, enabling applications to perform location-based analyses and services effectively. The library simplifies geocoding operations by providing a straightforward interface to retrieve latitude and longitude coordinates from addresses or place names. Similarly, reverse geocoding allows applications to obtain detailed address information from geographic coordinates. These functionalities are crucial for applications requiring

spatial data processing, such as mapping real estate listings or providing location-aware features to users.

In addition to Geopy, the project incorporates the OpenCageData API for enhanced geocoding capabilities. This API expands the project's functionality by offering advanced search capabilities and precise geocoding results. Developers can leverage its features to perform complex location queries and retrieve comprehensive address details, which are crucial for accurate mapping and spatial analysis tasks. However, it's important to note that the OpenCageData API operates under a free-tier model with limited request quotas. This constraint requires developers to manage API usage efficiently, potentially considering higher-tier options for increased capacity and additional features as the project scales.

Together, Geopy and the OpenCageData API enhance the project's geographic functionalities. They enable seamless conversion between addresses and geographic coordinates, crucial for mapping real estate properties and providing location-based services to users.

3.1.6 MySQL

MySQL is a widely-used open-source relational database management system (RDBMS). Developed, distributed, and supported by Oracle Corporation, MySQL is known for its reliability, performance, and ease of use. It is the database system of choice for many web-based applications, including popular platforms like WordPress, Drupal, and Joomla. MySQL uses Structured Query Language (SQL) for accessing, managing, and manipulating databases, which is a standard language for relational database management.

Mysql is free and open-source, fostering a large community for support and development. Known for high performance, MySQL can handle a high volume of queries and transactions efficiently. It ensures reliability and robustness with strong data protection mechanisms, supporting ACID properties for transaction processing. MySQL's scalability allows it to manage a large number of simultaneous connections and adapt to both small and large applications. Its flexibility across various platforms and support for multiple storage engines make it versatile. Additionally, MySQL's advanced security features protect data through user authentication, encryption, and SSL support. Lastly, its straightforward syntax and comprehensive documentation, along with graphical tools like MySQL Workbench, make it user-friendly for both beginners and experienced developers.

In this Thesis, MySQL integrates seamlessly with Django, the web framework

used for developing the real estate application. Django's ORM (Object-Relational Mapping) provides a high-level abstraction for interacting with the MySQL database, simplifying database operations. This integration ensures efficient data storage, retrieval, and management, allowing the application to handle large volumes of real estate data smoothly.

This introduction sets the stage for discussing the technologies used, while the conclusion summarizes their significance in the project. If you have any further questions or need additional assistance, feel free to ask.

3.2 Collection and Integration Data Process

In this section, I present the steps from collecting data, integrating and standardizing the data and storing it back into the database

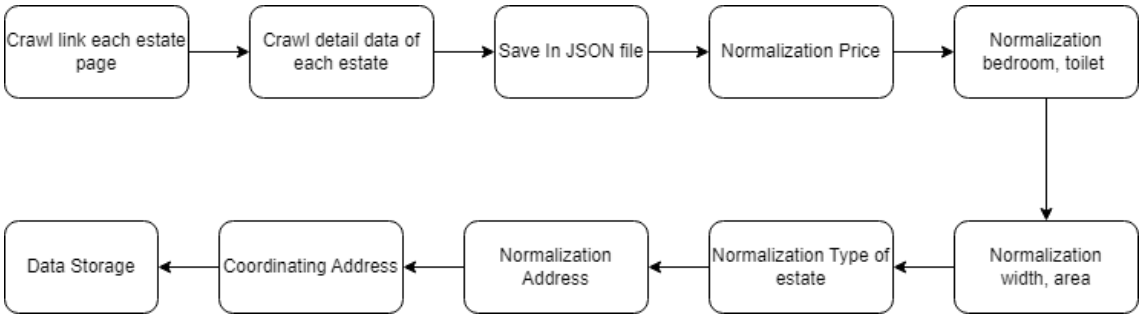

Figure 3.1: Integration and Collection data process

3.2.1 Crawl data from website

The process of crawling and scraping real estate data from websites involves systematically gathering detailed information about real estate. This information includes attributes such as title, price, area, type classification, bedroom count, description, address details, and images.

Based on my survey and analysis of various real estate websites, I have identified differing technical implementations HTML structure. Specifically, websites like muaban.net, nhadat24h, alonhadat, and meedyland.com predominantly use straightforward HTML and employ pagination techniques. For these sites, I will utilize BeautifulSoup to scrape data efficiently. In contrast, batdongsan.com utilizes JavaScript extensively, affecting how its content is dynamically loaded and displayed. Therefore, I have opted to employ Selenium for scraping batdongsan.com. Selenium's capabilities in handling JavaScript will enable me to effectively navigate and extract data from this particular website, ensuring comprehensive data

collection. The urls of real estate listing pages must be browsed using a loop since pagination is used by all websites.

Setting Up the Development Environment

Before starting the data crawling process, ensure that the development environment is set up correctly:

Python Environment: Begin by installing Python on your system, ensuring compatibility with required libraries such as BeautifulSoup, requests and Selenium library. Library Installation: Execute the following commands in your terminal to install essential libraries:

```
pip install beautifulsoup4
pip install selenium
```

Figure 3.2: Install library

Crawl link process

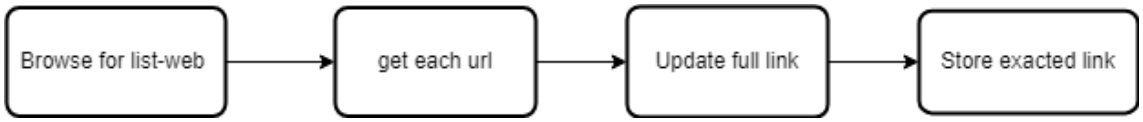

Figure 3.3: Crawl link process

In this process to collect links from real estate websites, the method involves iterating through pagination of each website. Firstly, an urls[] list need to be initialized to store the real estate links. For websites like muaban.net, nhadat24h, alonhadat, and meedyland.com, BeautifulSoup and requests are used to send GET requests to each page URL of the pagination. Then, BeautifulSoup parses the HTML structure and extracts real estate links from <a> tags with the href attribute. For batdongsan.com, which uses JavaScript for dynamic content loading, Selenium is essential. Selenium automates web browser control to gather links from this website.

To gather data from multiple real estate listings, you need to scrape information from individual pages. Start by accessing the search pages, such as
"https://batdongsan.com.vn/nha-dat-cho-thue/p1"
"https://alonhadat.com.vn/nha-dat/can-ban/trang1.html"
In these URLs, the "pvalue" or "trangvalue" segments are pagination values that allow you to navigate through the pages of listings.

To automate this process, I use a for loop to replace these pagination values, enabling you to traverse through the subsequent pages. Each of these sites typically contains a list of job listings, usually ranging from 20-30 per page. This method allows you to systematically collect all the required data from multiple pages of real estate listings.

Each found link is checked and appended with the base URL of the website to form a complete link. Ensure to construct full URLs if necessary by appending the base URL. In some case I need add the base URL:

full_url = 'https://nhadat24h.net' + href_link

Storing Extracted Links: Extracted property URLs are systematically stored in a designated text file (links.txt). Each URL serves as a unique identifier for a specific property listing, essential for subsequent data extraction and analysis tasks. This structured approach facilitates efficient data retrieval and utilization in real estate research and application development

```
with open("link.txt", "w") as file:
for url in all_urls:
    file.write(url + "\n")
```

Crawling Property Details

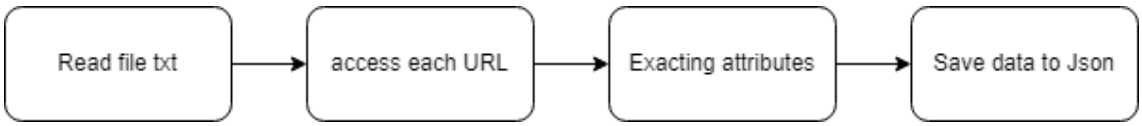

Figure 3.4: Crawl detail data process

Read url in file txt: Begin by iterating through the list of property links stored in links.txt. Each link corresponds to a unique property listing page on the real estate website.

Extracting Key Attributes: For each URL, an HTTP GET request is made to retrieve the HTML content. BeautifulSoup is then utilized to parse the HTML and extract necessary details and store in a dictionary for each real estate listing and appended to the data_list.With batdongsan.com, each URL from txt is opened using a Chrome WebDriver. The script waits until the required elements are visible using WebDriverWait to ensure complete loading. Selenium's methods are then also used to find and extract data added to the data_list.

Saving Data to JSON: Using JSON Format for Data Storage, data_list store the

scraped property data into a structured JSON file named data.json which is ease of access for further processing and analysis. Each real estate object is represented as a JSON object, containing details such as title, price, area, address, detailed description, property type, legal documents, number of floors, number of bedrooms, number of bathrooms, house direction, frontage width, and image links.

```
import json
with open('topcv.json', 'w', encoding='utf-8') as file:
json.dump(data_list,json_file,ensure_ascii=False,indent=4)
```

3.2.2 Integration data

In this subsection, we will focus on the critical process of normalizing the data stored in JSON files. Data normalization is essential for ensuring consistency, accuracy, and uniformity across the dataset, which in turn facilitates seamless data analysis and integration. The issues addressed in this section are aligned with our overall goal of creating a clean and well-structured database. To achieve these goals, we will utilize the powerful data manipulation capabilities of the pandas library to handle various data conversions and standardization processes.

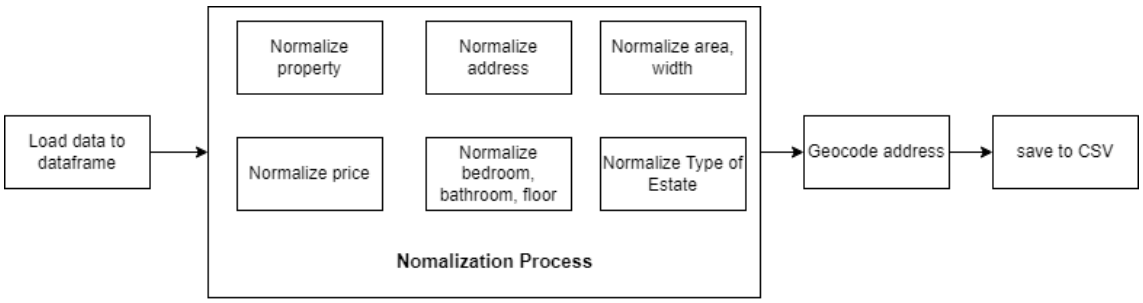

Figure 3.5: Normalization data process

Loading the Data

The first step in the normalization process involves loading the JSON file containing the collected property data into a pandas DataFrame. By converting the JSON data into a pandas DataFrame, we can easily access, modify, and analyze the data using pandas' extensive library of functions and methods.

```
import pandas as pd
import json
data_df = pd.read_json('data.json')
```

Normalization Address

Seprate district and province: This involves parsing the address string to create two new fields: district and city_province. By separating these elements, users can search for real estate listings by province or city, making the search process more

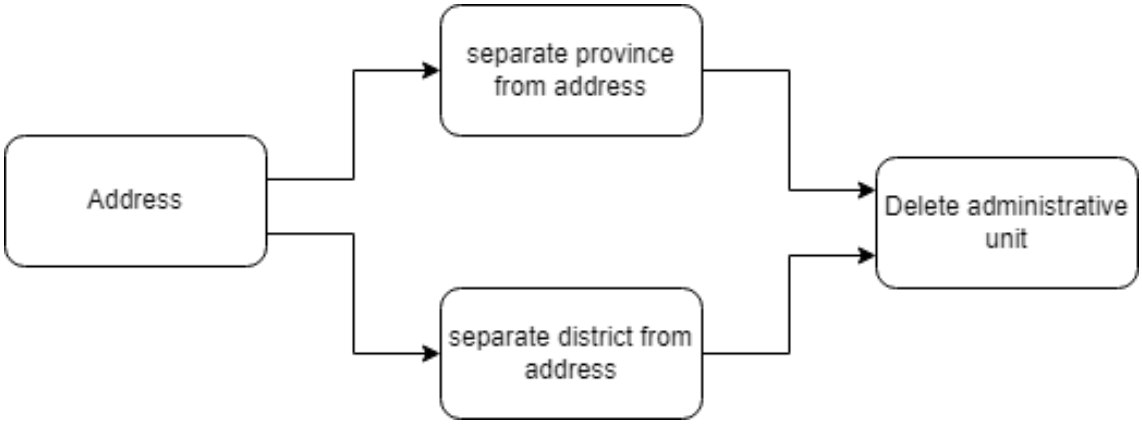

Figure 3.6: Normalization address process

efficient and user-friendly.Example: function extract_district_and_province(address): for get two part after split(',')

Delete admibistrative unit: To maintain consistency in geographic data, it is crucial to standardize the district and city_province fields. This involves removing any extraneous information and ensuring that only the standardized names are retained. Due to inconsistent data sources, the same district or city might be referred to by various names (e.g., "Hai Ba Trung District," "Hai Ba Trung," "Hai Bà Trưng"). Example:

```
district.replace('Quận','').replace('H.','').strip()
province.replace('Tỉnh','').replace('Tp.','').strip()
```

Normalizing Area and width Measurements

Another important step in the normalization process is ensuring that the values in the area and width field are in a unified unit of measurement, specifically square meters (m²). This involves converting any non-standard area measurements into square meters to ensure comparability. This step is crucial for maintaining uniformity in property size specifications, enabling accurate comparisons and analyses of property sizes across the dataset. For example:

```
area.replace('m2','').replace(',','.').strip()
width.replace('m','').replace(',','.').strip()
```

Standardizing Price

In standardizing property prices for both sale and rental properties, to ensure consistency in representing property prices across diverse sources, I employed the following method:

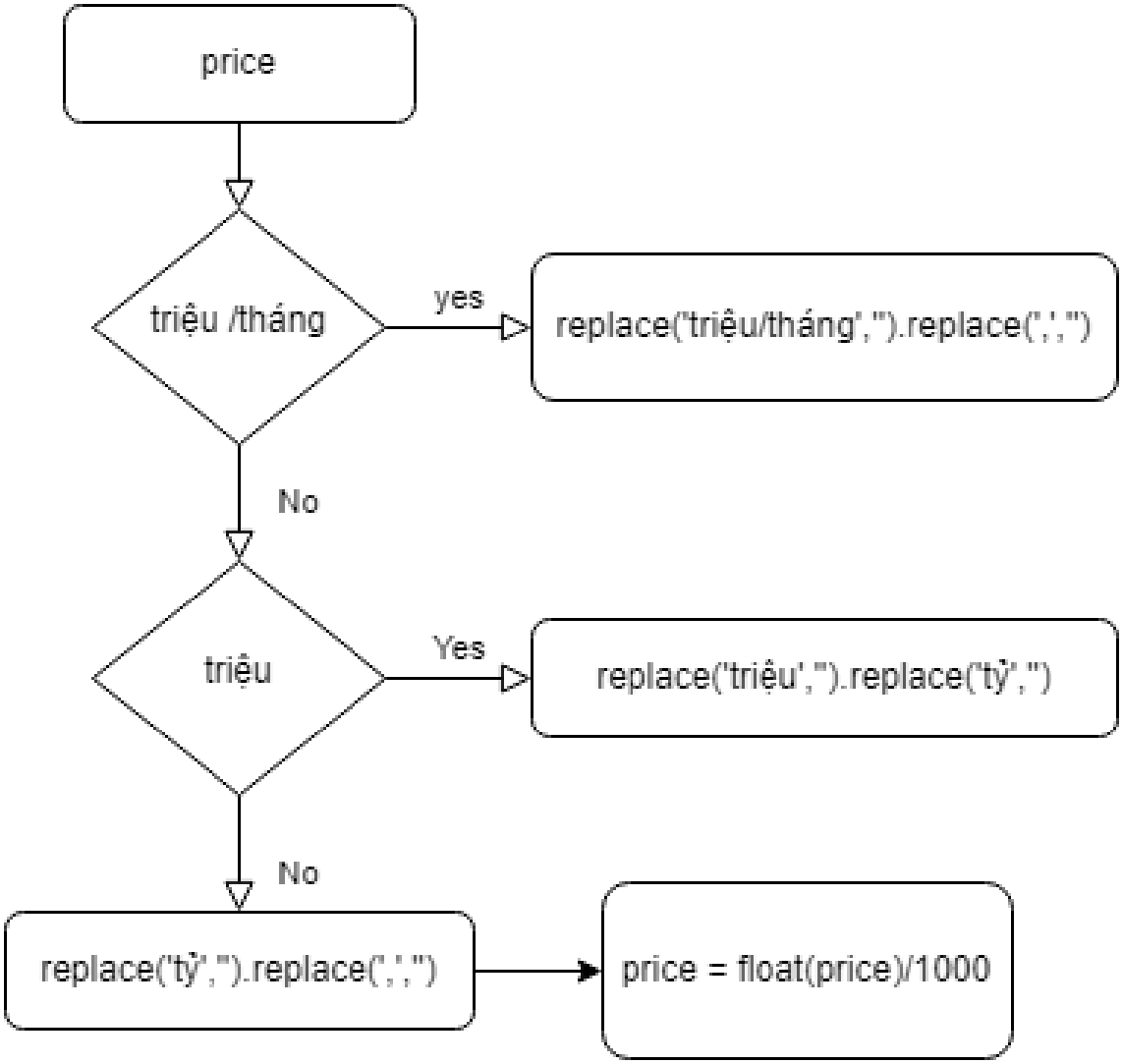

Figure 3.7: Normalization price process

Identifying Key Terms: I identified common terms and units that were inconsistently used across different listings, such as "triệu" (million), "tỷ" (billion), and "triệu/tháng" (million per month).

String Manipulation: Using string manipulation techniques, I systematically removed these identified terms from the price fields of the data. For instance:

Removed "triệu" and "tỷ" to normalize numeric values and removed "triệu/tháng" from rental prices to standardize monthly rental costs. Example:
price.replace('triệu,'').replace('tỷ','').replace('triêu/tháng','')

Handling Special Cases: I addressed special cases, such as listings where the price was denoted as "thỏa thuận" (negotiable). In this cases, the price field was by setting it to a default value 0.

Conversion to Numeric Values: After cleaning and normalizing the strings representing prices, I converted the processed values into numerical formats (floats) for consistency and ease of comparison. Example: price.replace(',','.')

Removing Units from Numeric Fields

Another critical aspect of data normalization is removing any unit descriptions from the price, area, bedroom, and toilet fields. Different data sources may use various formats for these fields, resulting in inconsistencies. By removing unit descriptions, we ensure that these fields contain purely numeric values, simplifying data manipulation, analysis, and integration. For example, a property with an area listed as "120 m²" should have the area value stored simply as "120." This approach ensures consistency and facilitates more straightforward numerical operations on these fields.

Normalizing Property Type Names

Finally, it is essential to harmonize the type_of_BDS field across different data sources. Different sources may use different names for the same property type, leading to inconsistencies. This step involves mapping these different names to a standardized set of property type names, ensuring uniformity across the dataset. For instance, properties referred to as "apartment," "flat," and "condo" should be standardized to a single term, such as "apartment." This normalization ensures that all properties of the same type are categorized uniformly, making it easier to search, filter, and analyze the data based on property types. IN this case I only need replace function of pandas.

By following these steps, we ensure that the data is normalized and ready for further analysis. Data normalization is a critical step in preparing the dataset for accurate and efficient querying, analysis, and integration with other data sources.

3.2.3 Geocoding address

In this subsection, we focus on the process of geocoding addresses, which involves converting address data into geographic coordinates (latitude and longitude).

Geocoding with Geopy Geopy is a robust library that provides an easy interface to access various geocoding services. It is particularly effective for standard addresses that follow a clear and consistent format, which is typical of many property listing websites. For instance, addresses from sites like nhadat24h.com often include details such as the street name, ward/commune, district, and city. This structure allows Geopy to quickly and accurately find the corresponding geographic coordinates.

Steps to Geocode Standard Addresses with Geopy:
Data Preparation: Ensure that the address data is clean and follows a consistent

format.

Geocoding Process: Use Geopy's Nominatim or another suitable geocoding service to process the address. For each address, the geocoding service will return latitude and longitude coordinates.

Handling Complex Addresses with OpenCageData API

While Geopy handles standard addresses effectively, some addresses, particularly those referring to large projects or developments, present challenges. penCageData offers more sophisticated geocoding capabilities, capable of handling complex and non-standard address formats. It aggregates data from multiple sources to provide precise coordinates, making it an essential tool for ensuring comprehensive geocoding coverage.

Steps to Geocode Standard Addresses with Geopy:

API Integration: Integrate the OpenCageData API within the geocoding workflow. Ensure that API keys are securely managed and configured. Geocoding Process: For each complex address, send a request to the OpenCageData API. The API returns geographic coordinates along with additional metadata, such as confidence scores and address components.

3.2.4 Data Storage

In this subsection, we will detail the process of storing the normalized and geocoded data into a MySQL database using the pymysql library. This process involves creating a database schema, ensuring proper encoding for Vietnamese characters, and efficiently inserting the data into the respective tables.

a, Database Design

The Relationship diagram serves as a foundational tool in this regard, providing a visual representation of the database schema. This diagram defines the relationships between entities, their attributes, and the constraints that govern them.

I also created the four tables below to provide more information and clarity about the four tables in the database above so that you could better grasp the responsibilities, definitions, types, and restrictions:

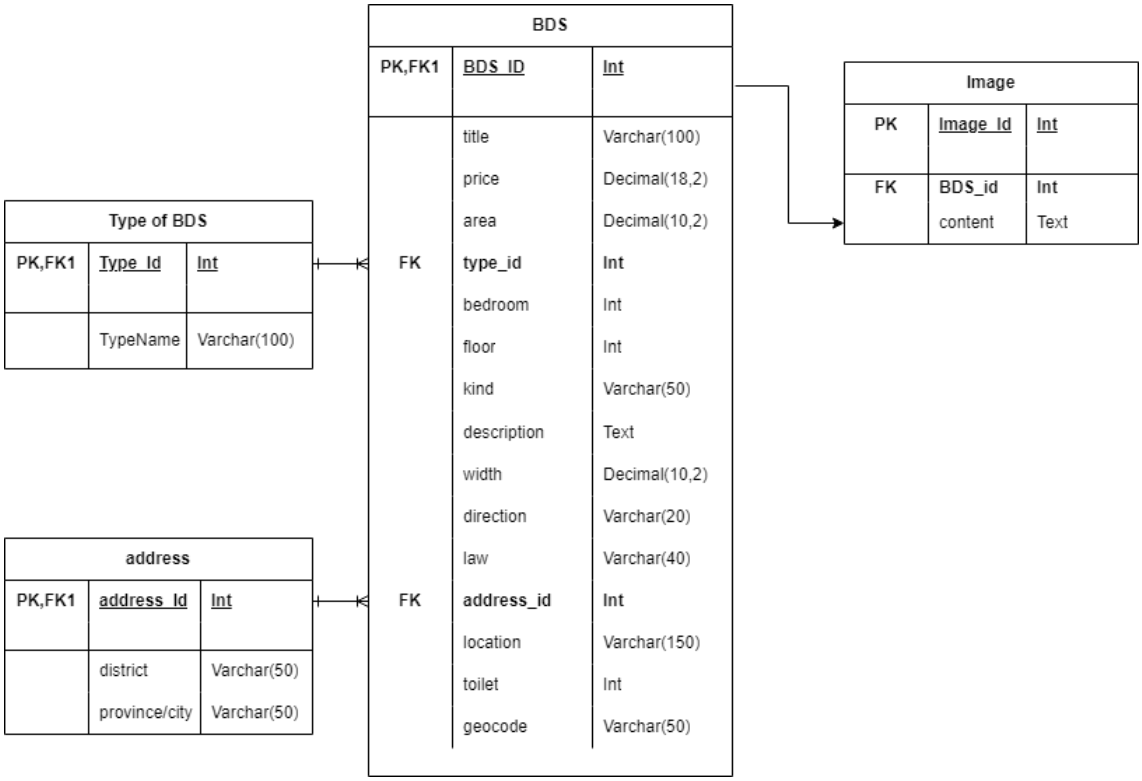

Figure 3.8: Relation Diagram

Table 3.1: Table of type of real estate

Field Name	Data Type	Constrain	Description
Type_Id	INT	Primary Key	Unique identifier for each type of real estate
Type_Name	VARCHAR(100)		Name of the estate type

Table 3.2: Table of Image

Field Name	Data Type	Constrain	Description
Image_Id	INT	Primary Key	Unique identifier for each image associated with a real estate
BDS_id	INT	Foreign Key (BDS)	Reference to the real estate entry
content	TEXT		image link

Table 3.3: table of Address

Field Name	Data Type	Constrain	Description
address_Id	INT	Primary Key	Unique identifier for address
district	VARCHAR(50)		District of the address
province/city	VARCHAR(50)		Province or city of the address

Table 3.4: table BDS

Field Name	Data Type	Constrain	Description
BDS_ID	INT	Primary Key	Unique identifier for each real estate entry
title	VARCHAR(100)		Overall content of the real estate
price	DECIMAL(18,2)		Price of the real estate
area	DECIMAL(10,2)		Size of the real estate (m2)
type_id	INT	Foreign Key	Reference table typeofBDS
bedroom	INT		Number of bedrooms in the real estate
floor	INT		Number of floor in the real estate
kind	VARCHAR(50)		Category of the real estate
description	TEXT		Detailed description of estate
width	DECIMAL(10,2)		Width dimension of the real estate (m)
direction	VARCHAR(20)		direction of the real estate
direction	VARCHAR(3)		legal documents of the real estate
address_id	INT	Foreign Key	Reference to address table
location	VARCHAR(150)		Specific address of estate
toilet	INT		Number of toilets in estate
geocode	VARCHAR(50)		Coordinates of the estate

Data storage

After completing the data integration process as well as defining data tables and data fields, I started pushing data into the mysql database according to the following process:

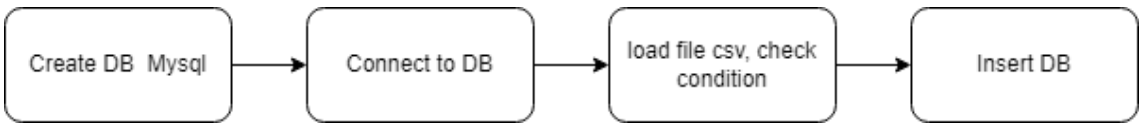

Figure 3.9: Data Storage Process

Below is the pesuacode that basically describes the implementation process:

```
Create the Database:
    Create database named 'estate' with utf8mb4 character set
    Create four tables: BDS, Image, TypeofBDS, Address

Connect to the Database:
    Connect using pymysql.connect with necessary parameters

Open the CSV file to read Data and Insert into Database:
    For each row in the CSV:
        Extract district and province_city from the row
        Check if the address exists in the 'pools_address' table:
            If address exists: Get address_id
            Else: Skip the row
        Check if the property type exists in the 'pools_typeofbds' table:
            If type exists: Get type_id
            Else: Skip the row
        Extract other property details (title, price, area, kind, law, bedroom, description,
        width, direction, toilet, location, geocode)
        If geocode is None or empty: Skip the row
        Insert data into 'pools_bds' table with the extracted and processed values
        Insert image_link into 'pools_image' table with BDS_id

Close Database Connection
```

Figure 3.10: Pesuacode of data storage

Data Accuracy in Integration Phase

The data integration process is a critical part of the project. Throughout this process, I utilized the data_normalization function to standardize the data and the get_geocode function to geocode addresses. After implementing and testing these functions, I found that the data_normalization function was highly effective, successfully normalizing fields such as price, area, bedroom, floor, toilet, and width. The data results, after being stored in the MySQL database, confirmed that the data processed through the data_normalization function were accurate.

Given that the five data source websites have different structures, the normalization function also varied, leading to different processing times, as illustrated in the table below:

Table 3.5: Table of Time in Normalization process

Web Name	Number of Objects	Time
Batdongsan.com	1000	90s
alonhadat	1000	75s
muaban.net	1000	100s
nhadat24h	1000	60s
meedyland	1000	85s

The coordinating process is crucial for visualizing data on maps. After applying the get_geocode function, I evaluated the number of None values in the geo_code column of the DataFrame to determine the effectiveness of the geocoding process.

This evaluation showed that the get_geocode function successfully processed most addresses, with only a few failures, demonstrating its overall efficiency.

```
def evaluate_geocode_accuracy(df):
    total_addresses = len(df)
    successful_geocodes = df['geo_code'].count()
    success_rate =(successful_geocodes/total_addresses)*100
    failure_rate = 100 - success_ratesuccess)")
```

The results are as following table:

Table 3.6: Table of Effective of Geocoding process

Web Name	**Efficiency**
Batdongsan.com	90%
alonhadat	nearly 100%
muaban.net	95%
nhadat24h	92%
meedyland	90%

This assessment highlights the efficiency of the data normalization and geocoding processes within the system, ensuring accurate data integration and dependable geographic visualization.

In conclusion, using tools like Beautiful Soup, Selenium, and the requests library, we efficiently gather data from multiple real estate websites. The normalization process ensures data consistency by standardizing some important attribute such as: price units, area measurements, and address details,... . Geocoding converts address information into geographical coordinates for visualizing by map in next chapter. The processed data is securely stored in a MySQL database using the pymysql library, ensuring proper encoding for Vietnamese characters and maintaining data integrity. The data collection and integration process will be an important precursor to the subsequent data visualization process

CHAPTER 4. DATA VISUALIZATION

Data visualization is essential in a real estate search system as it converts raw data into user-friendly graphical formats. The main goals of the data visualization module are to enhance the user experience, aid in informed decision-making, and identify market trends. By offering an interactive interface, users can easily engage with the data, making the search process more informative. Visualization also helps real estate companies understand market dynamics, price ranges, and user demand. Ultimately, this module aims to facilitate straightforward property searches across various attributes for all users.

4.1 Technology

In this section, I will simply present the technologies that I will apply in the data visualization process.

Effective data visualization in modern web applications relies on a robust combination of technologies that enable seamless data collection, processing, and presentation. In this section, we discuss the key technologies employed in our project for building a dynamic and interactive real estate web application. Each technology plays a vital role in ensuring that data is efficiently retrieved, processed, and displayed in an intuitive manner, ultimately enhancing user experience and decision-making.

4.1.1 Django

Django is a high-level Python framework that promotes rapid development and clean, pragmatic design. Created by experienced developers, Django handles many of the complexities of web development, allowing you to focus on building your web application without needing to start from scratch.Being written in Python, Django is cross-platform and not tied to any specific server platform, with strong support from many hosting providers who offer infrastructure and documentation for hosting Django web applications. It is both free and open-source.

Django offers several significant advantages. It is comprehensive, following the "Batteries included" philosophy, providing everything a developer needs out of the box, which allows developers to focus on their product as all components work seamlessly together. Django's versatility allows it to be used to build a wide range of websites, from content management systems to social networks and news sites, integrating well with client-side frameworks and supporting various content formats such as HTML, RESS, JSON, and XML. Security is a priority in Django,

helping developers avoid common security pitfalls by providing a framework that includes best practices for protecting websites, such as managing account and password security, avoiding storing session information in cookies, and encrypting passwords. Django's scalable shared-nothing architecture allows it to effectively handle increased traffic by adding hardware at any level, including caching, servers, database servers, or application servers, making it suitable for high-traffic applications. The maintainability of Django is ensured by its adherence to design principles and patterns that encourage code reuse, following the "Don't Repeat Yourself" (DRY) principle to minimize unnecessary repetition and reduce code volume.

4.1.2 Reactjs

ReactJS, developed by Facebook, is a powerful JavaScript library widely used for building interactive user interfaces in web applications. It employs a component-based architecture, allowing developers to create reusable UI components that enhance code maintainability and scalability.

In our real estate web application project, ReactJS serves as the frontend framework responsible for creating dynamic and responsive user interfaces. It communicates with Django, a robust backend framework, through RESTful APIs. ReactJS efficiently renders data fetched from Django's backend, ensuring a seamless user experience with real-time updates and interactions.

ReactJS integrates seamlessly with Django, leveraging its capabilities for data retrieval and business logic handling. Django, as the backend framework, manages the MySQL database where real estate data is stored and processed. ReactJS, on the frontend, interacts with Django APIs to fetch and display property listings, perform searches, and present data visualization components such as maps.

React-Leaflet is a React wrapper for Leaflet, a leading open-source JavaScript library for interactive maps. It provides React components that simplify the integration of maps into React applications. React-Leaflet supports features such as markers, layers, popups, and interactions, making it ideal for displaying geographic data dynamically.

In our project, React-Leaflet enhances the user interface by integrating interactive maps. It allows users to visualize property locations, filter listings based on geographic criteria, and interact with map elements seamlessly. The integration of React-Leaflet with ReactJS enables the application to provide intuitive and engaging map functionalities, enhancing the overall user experience.

4.2 Data Visualization Process

In the real estate applications, effective data visualization is paramount for converting intricate property datasets into accessible and insightful information that resonates with users. This chapter explores methodologies, implementation strategies, and the overarching approach employed to visualize real estate data by searching component and integrating map

4.2.1 Search Component

The search and filter functionality in this application allows users to specify criteria such as province, district, price range, area range, transaction type (buy or rent), and property type to find properties that match their preferences. This section details how these search parameters are handled on the ReactJS frontend, how they interact with the backend Django server via API calls, and the subsequent display of relevant property listings.

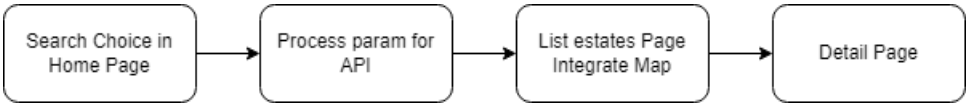

Figure 4.1: Search Process

a, Search Home Page

On the main page, users have the option to filter the results based on various property attributes. Once the user makes their selections and initiates the search, the search results will be displayed according to the chosen criteria.

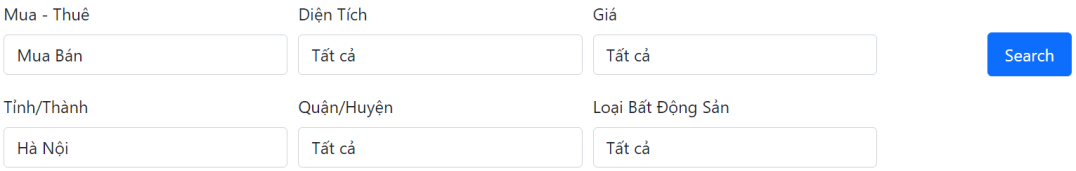

Figure 4.2: Search Form

On the homepage, there is a dropdown attribute for selecting a province. Each time the homepage is opened, an API call is automatically made to fetch all the provinces from the database. **Endpoint: /pools/provinces/** Description: Retrieves all provinces in database, facilitating geographical navigation for users.
HTTP Method: GET
URL: http://127.0.0.1:8000/provinces

On the homepage, there is a dropdown attribute for selecting a district. However this attribute depends province selection. so that when user choose a province se-

lection, an API call is automatically made to fetch all the disitrct in thí province from the database.

Endpoint: /pools/address/<province>/ Description: Retrieves a list of districts within a specified province, facilitating geographical navigation for users.
HTTP Method: GET
URL: http://127.0.0.1:8000/pools/address/<province>/
Parameters: <province> (Path parameter) - Specifies the province for which districts are to be retrieved.

After choosing all attributes in search form and choose search, system send request to backiend and call /pools/properties-with-conditions/ for get all data.

Endpoint: /pools/properties-with-conditions/
The /pools/properties-with-conditions/ API endpoint is designed to retrieve properties from the database based on user-defined search criteria. These criteria include parameters such as province, district, price range, area range, transaction type (buy or rent), and property type. This endpoint utilizes the HTTP GET method to fetch data from the server, accommodating straightforward queries by appending parameters directly to the URL.
HTTP Method: GET

b, Process param Data for API

The API endpoint accepts the following parameters as query parameters in the URL:

province: (optional) Specifies the province where the property is located.
district: (optional) Specifies the district within the province where the property is located.
max_price: (optional) Specifies the maximum price limit for the property.

min_price: (optional) Specifies the minimum price limit for the property.
Example:you choose price: 1 - 2 tỷ so the maximum price is 2 and minimum price is 1 max_area: (optional) Specifies the maximum area (size) of the property.
min_area: (optional) Specifies the minimum area (size) of the property.

Example:you choose price: 100 -150 m2 so the maximum area is 150m2 and minimum area is 100m2 kind: (optional) Specifies the transaction type of the property (e.g., buy or rent).
type: (optional) Specifies the type of the property (e.g., house, apartment, land).

In my database mysql, both attributes: price, area is in decimal type so it can be easy to compare with min, max price; min, max area to get full necessary data

c, Listing Page

After the data is fetched by the API and stored, it is passed to the /list-estate page. Here, the data is retrieved again and a geocode variable is built to serve as an input parameter for the map.

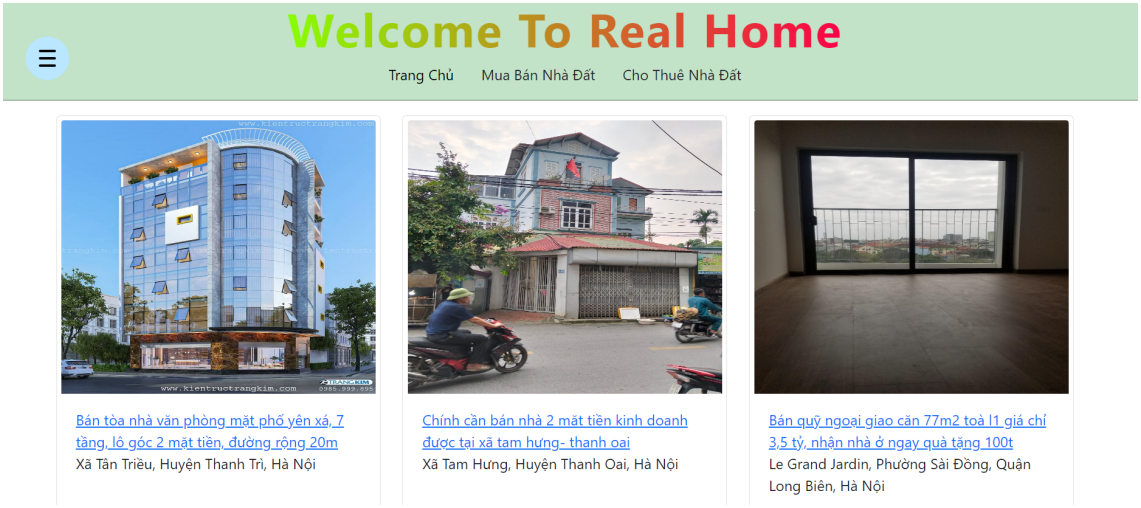

Figure 4.3: Listing page.

d, Detail Page

After selecting a property to view its details on the /list-estate page, the user will be redirected to the /listing page where an API call will be made to fetch the detailed information of the selected property: Endpoint: /pools/property/<id>/

Detail page with image of estate, position map, and some attributes:

We also have more Data In Detail page with detail description, some information such as: number of floors,number of bedrooms,number of bathrooms, main direction, wide of estate, legal status

Figure 4.4: Detail page.

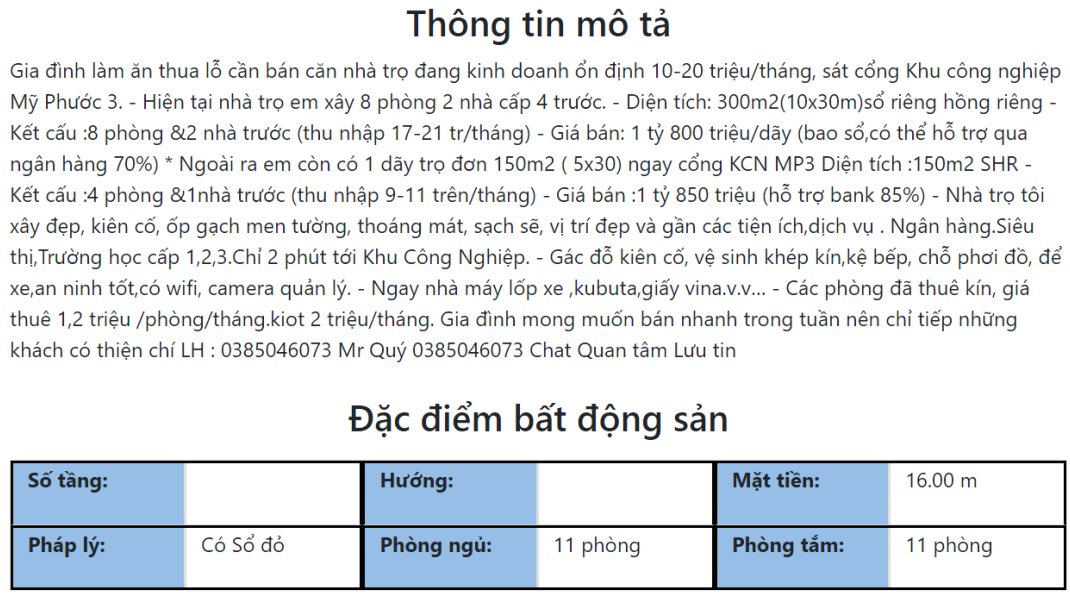

Figure 4.5: More Data in detail page.

Endpoint: /pools/property/<id>/

The /pools/property/<id>/ API endpoint retrieves detailed information about a specific property based on its unique identifier (id). This endpoint utilizes the HTTP GET method to fetch data from the server, providing comprehensive details of the property stored in the database.

HTTP Method: GET

URL: http://127.0.0.1:8000/pools/property/56/

Parameters: <id> (Path parameter) - Specifies the unique identifier (ID) of the property to retrieve.

4.2.2 Integrating Maps

Integrating interactive maps into the real estate web application enhances user experience by providing a visual representation of property locations. This section discusses the implementation of interactive maps, focusing on functionalities such as displaying property markers, clustering markers for dense areas, and providing info windows with detailed property information.

Prepare Data

IN previous chapter, I used geopy library or API of OpenCageData to transform address textdata into a geocoding data. Geospatial data is an input to interactive map features and provides users with visual insights into asset location and distribution. Before calling the Map component, it needs to be preceded by a "geocode" parameter. The location coordinates of each property, along with necessary details such as price, area and image, title, are meticulously processed and structured into 'geocode' parameters. This parameter ensures that all data points are accurately represented on the map.

Interactive Maps Integration

To integrate interactive maps, we utilize the react-leaflet library, which is built on top of Leaflet, a widely used open-source JavaScript library for interactive maps. react-leaflet allows embedding interactive maps within React applications efficiently and effectively.

First map will call "geocode" parameter. Each time the user performs a search on the main page, along with the list of properties that match the search criteria, a map showing the positions and a heatmap of these properties will also be displayed. I can see some base information, or click to see detail information by icon in Position Map

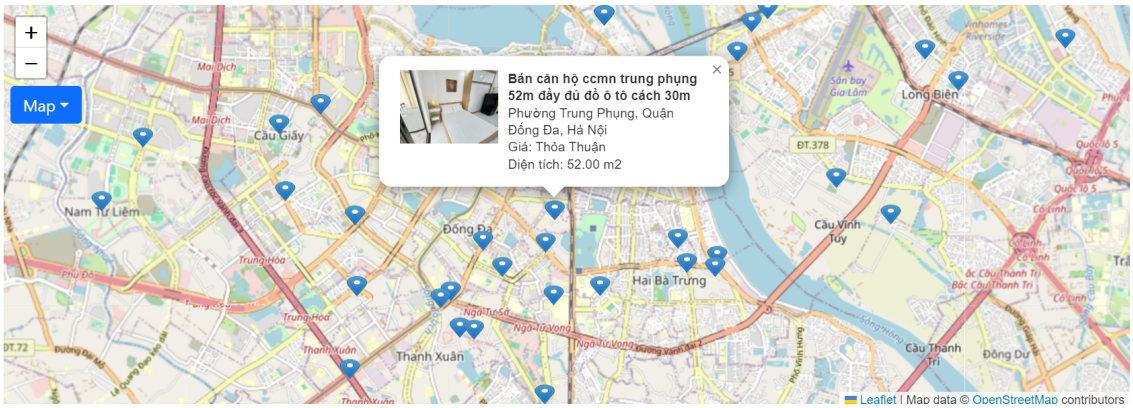

Figure 4.6: Position Map

I can switch into heatmap after choosing in dropdown

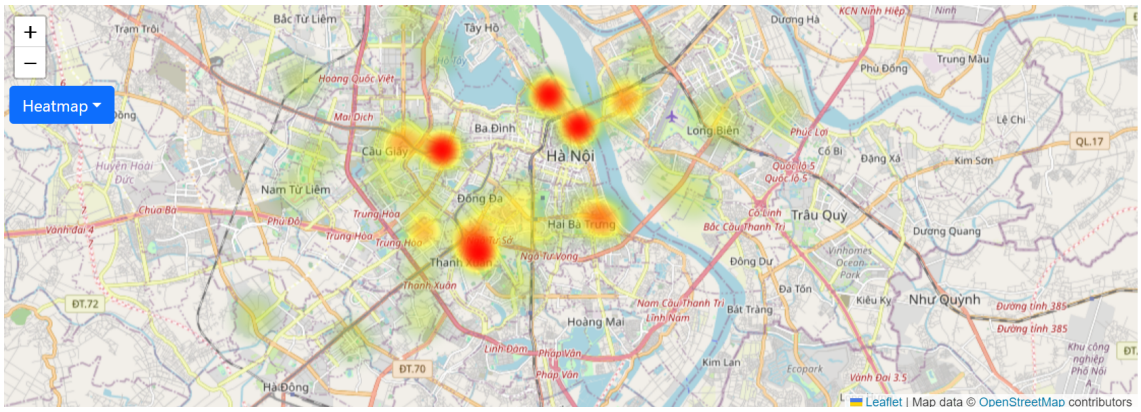

Figure 4.7: Heat Map.

After applying and testing, the processing speed of calculations is quite fast. From the moment a user sends a request to when the user interface receives it, it takes an average of only 368ms per request. This speed is impressive and meets the initial performance requirements.

Data visualization is a crucial component of our real estate system, providing users with a clear and interactive way to explore property information. By leveraging mapping and charting tools, we enable users to gain insights into property locations, prices, and other attributes at a glance. Through these visualizations, users can make more informed decisions based on comprehensive data presented in an easily digestible format.

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

Throughout the process of completing this graduation project, Ihave developed a robust system for the collection, integration, cleaning, storage, and visualization of real estate data from multiple sources. The results have been compared with similar research and products, demonstrating effectiveness in providing accurate, standardized, and easily accessible real estate information.

I have successfully integrated data from multiple sources with different structures, employed rigorous data cleaning and standardization techniques to ensure consistency, and offered user-friendly data visualization tools that enhance user interaction and data analysis capabilities. However, during the implementation, I encountered some limitations, such as challenges in processing free text data and dealing with missing information in certain data fields, time for collection and procession is long.

Despite these challenges, this project include the development of a comprehensive and efficient system for real estate data management, the creation of a standardized real estate database, and the provision of an intuitive web platform that allows users to easily search, filter, and analyze real estate data. Through this project, I have gained valuable insights into handling big data, optimizing database performance, and developing user-friendly interfaces.

Furthermore, the project has highlighted the importance of continuous improvement and adaptation to new technologies. The ability to collect and process large volumes of data from diverse sources and present them in a meaningful way is crucial for decision-making in the real estate market. Our system provides a foundation for future enhancements and scalability, ensuring it remains relevant and useful in a rapidly evolving market.

5.2 Future Work

In the future, I plan to continue refining and enhancing this system. Firstly, I will improve the existing functionalities by optimizing the data collection and cleaning processes to ensure the accuracy, fast and completeness of the data. A significant future direction is to integrate artificial intelligence and machine learning technologies. By incorporating predictive analytics, we can automatically analyze and forecast real estate market trends, providing users with deeper insights and helping

them make more informed decisions.

I also plan to enhance the user interface to support more interactive and dynamic visualizations. This includes developing advanced mapping features that allow users to visualize property locations and related attributes, such as price ranges, area, and property types, on interactive maps. Moreover, we will implement tools that enable users to perform complex queries and filter data based on multiple criteria, improving the overall user experience.

Expanding the system to support more geographic areas is another priority. This involves adapting the system to handle region-specific data formats and requirements. Finally, we will continue researching and applying the latest methods in data processing and visualization to keep the system up-to-date and responsive to user needs. This includes exploring new data sources, such as social media to enhance the richness and accuracy of the data. By continuously evolving and improving the system, wIaim to maintain its usefulness in real estate market.

REFERENCE

[1] React Leaflet. [Online]. Available: https://react-leaflet.js.org/ (visited on 15/6/2024).

[2] Geopy Libarary. [Online]. Available: https://pypi.org/project/geopy/ (visited on 5/6/2024).