International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 5, pp 304-311, May-2023

https://doi.org/10.47001/IRJIET/2023.705043

Enhancing Code Quality and Feature Functionality through Refactoring of the Property Listing Feature in the Hotel Property Management System

[1]*Patricia Evericho Mountaines, [2]Zulfa Fatah Akbar Ahmad

[1,2]Department of Computer Engineering, Diponegoro University, Jl. Prof. Soedarto, S.H., Semarang, Central Java, 50275, Indonesia

*Corresponding Author's E-mail: evericho@ce.undip.ac.id

***Abstract* - With the rapid advancement of technology, it is inevitable that nearly every aspect of our lives involves technology, particularly when accessing information. Through the utilization of sophisticated technology, we can conveniently and practically access various types of information or data using mobile devices. This has led to the development of web applications that encompass diverse data within organizations. The purpose of creating web app is to streamline team management processes, enabling faster and more efficient work. However, quite often we find features in the web apps that can still be enhanced in terms of program code and functionality. Refactoring offers a method to enhance both of the code quality and the feature functionality. It aims to improve the internal quality of a feature, making the system easier to maintain and free from errors or bugs. Using the Test-Driven Development approach in this research, a feature has been successfully refactored, resulting in a more user-friendly experience and facilitating easier maintenance for developers.**

Keywords: Refactoring, Code Quality, Feature Functionality, Test-Driven Development, Property Listing Feature.

I. INTRODUCTION

Digitalization is currently experiencing rapid growth. Furthermore, developers continue to innovate with information technology-based solutions to ensure users can reap the benefits of technology in their daily lives. Companies are also embracing the development of their own information technology products to streamline their operations. For instance, hotel management service companies have introduced web applications designed to simplify employee tasks, including guest data management, employee data management, transaction data processing, and property details viewing. However, despite these features, there is still room for improvement in terms of their quality.

One approach to enhance these features is through code refactoring. Refactoring involves modifying the structure of software without altering its behavior [1]. According to Martin Fowler's book [2], refactoring is the process of transforming a software system without changing its external behavior. The primary objective of refactoring is to enhance the internal structure quality of the software. Refactoring serves as a means to "clean up" the program code and minimize the occurrence of software bugs in the implementation. In essence, when we engage in refactoring, we are effectively enhancing the design quality of our software [3].

In the context of hotel property management system, the property listing feature plays a crucial role in attracting potential guest, facilitating reservations, and providing accurate information about available amenities and pricing. However, as software system evolve over time, the property listing feature can become burdened, with legacy code, inconsistencies, and limitations that hamper performance and impede the addition of new functionalities. Therefore, a systematic and well-planned refactoring process becomes essential to ensure the continued success of the system.

This study focuses on refactoring the property listing feature within the hotel property management system. Several reasons justify the need for refactoring this feature, such as messy code, excessive code repetition, inefficient data filtering in property listings, lengthy data loading times impeding team productivity, and inadequate usability for the development team.

II. METHODOLOGY

One of the most widely used techniques for code refactoring is Test-Driven Development (TDD), also known as Red-Green-Refactor approach, which is commonly employed in Agile test-based development. TDD is a software development approach that emphasizes writing automated test before writing the actual code. This approach was selected due to its significant advantages in enhancing both code quality and

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 5, pp 304-311, May-2023

https://doi.org/10.47001/IRJIET/2023.705043

feature functionality while preserving the same functionality before and after the refactoring process. When applying this method, developers break down the refactoring process into five distinct steps [4]:

1) Stop and consider what needs to be developed (Red),
2) Run through all lines of code and see what fails,
3) Perform a little development or write code to pass basic testing (Green),
4) Re-run the code and ensure that all tests pass,
5) Implement refactoring by optimizing and cleaning the code without adding functionality (Refactor).

After completing these five steps, the developer can return to the first step to achieve cleaner or neater code. The TDD cycle can be seen in Figure 1.

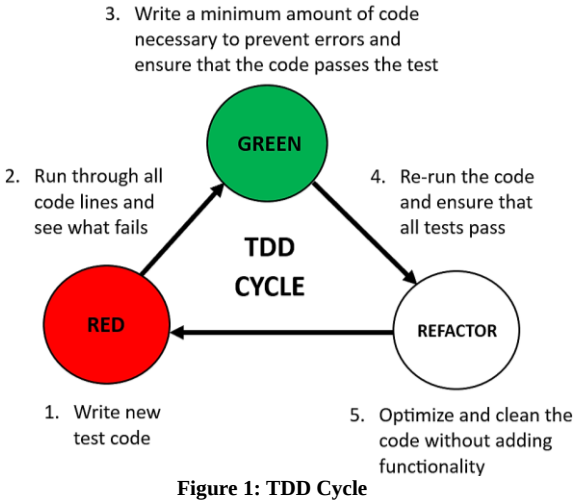


Figure 1: TDD Cycle

III. RESULTS AND DISCUSSIONS

3.1 Web Application

The Hotel Property Management System has been developed as a web-based solution to facilitate access, viewing, and management of essential data for teams within a company. For instance, in a hotel management company, a web app can be built using the ReactJS library, incorporating features such as a dashboard, guest list, payment list, property list, employee list, and partner list. Among these features, property data plays a crucial role as it represents the properties that the company collaborates with. This data is integrated into features and displayed on property listing pages within the web app.

The property listing feature within the web app serves the purpose of presenting the property data managed by the company. It includes information, i.e. the property name, property contract date, property type, location area, property partner status, property Google Maps link, and property WhatsApp group link. Additionally, the property list feature provides filtering options to refine property data based on specific criteria and allows users to search for properties by their desired names.

3.2 Designing the Property Listing Feature

The design of a feature is essential prior to conducting a refactoring process to ensure that the resulting improvements truly enhance the feature's quality. This stage aims to clarify the requirements for refactoring the feature and provides an overview of how the feature will be transformed once the refactoring is complete.

3.2.1 Feature Problems and Solutions

The refactoring of the property listing feature was undertaken due to several issues encountered. These problems are outlined as follows:

a) The front-end program code was complex, making it difficult for developers to comprehend and maintain due to excessive code repetition,
b) The table components lacked reusability for other features,
c) The filter components were not user-friendly and lacked reusability,
d) Excessive data retrieval from the back-end resulted in lengthy loading times,
e) The display features could be further enhanced in terms of aesthetics and user comfort.

Based on the issues, a plan or solution was devised for refactoring the property list features, including:

a) Refactoring the program code to minimize code repetition and facilitate easier maintenance by developers,
b) Developing a new table component that can be utilized in various features,
c) Creating a new user-friendly filter component that can be applied to different features,
d) Optimizing data retrieval by retrieving only the necessary data,
e) Redesigning the UI/UX of the property list feature to improve its visual appeal and user experience.

The implementation of these solutions was carried out in this research in order to successfully transformed the property listing feature, enabling the refactoring process to effectively address and overcome its initial challenges and limitations.

3.2.2 Functional Requirements

Functional requirements are essential for understanding how a feature responds to specific inputs and behaves in

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 7, Issue 5, pp 304-311, May-2023
https://doi.org/10.47001/IRJIET/2023.705043

various situations. Through discussions involving Product Owners, Front-End Developers, Back-End Developers, and UI/UX Designers, a comprehensive list of functional requirements for the property listing feature in the Hotel Property Management System was compiled. This list includes user categories, Software Requirements Specification Code (SRSC-Id), description of requirements, and priorities, as presented in Table 1.

Table 1: The functional requirements list

No.	SRSC-Id	Requirements Description	Priority
1.	SRSC-HPMS-0001	View the property list	High
2.	SRSC-HPMS-0002	Search for a specific property	High
3.	SRSC-HPMS-0003	Filter property data based on available filters	High
4.	SRSC-HPMS-0004	Sort property data by name or date	Medium
5.	SRSC-HPMS-0005	Sort property data in ascending or descending order	Medium
6.	SRSC-HPMS-0006	Download property data in CSV format	Medium

3.2.3 Non-Functional Requirements

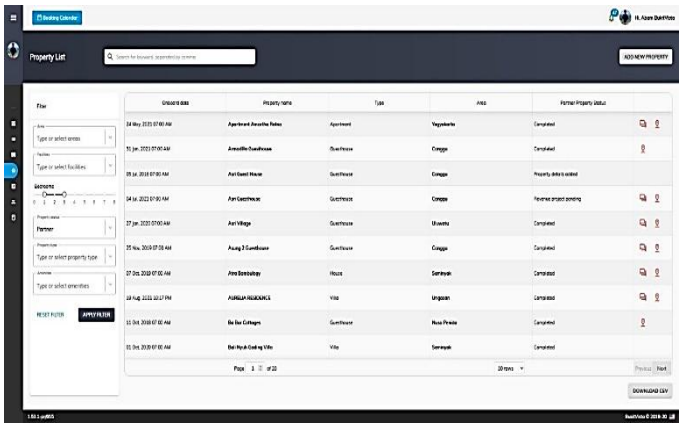
Non-functional requirements are limitations on services or functions offered by features, including time constraints, process development limitations, standardization, and others. From discussions between Product Owners, Front-end Developers, Back-end Developers, and UI/UX Designers, a list of non-functional requirements for the property listing feature in the Hotel Property Management System was obtained. The list of non-functional requirements includes Software Requirements Specification Code (SRSC Id), parameters, and requirements description, as shown in Table 2.

Table 2: The non-functional requirements list

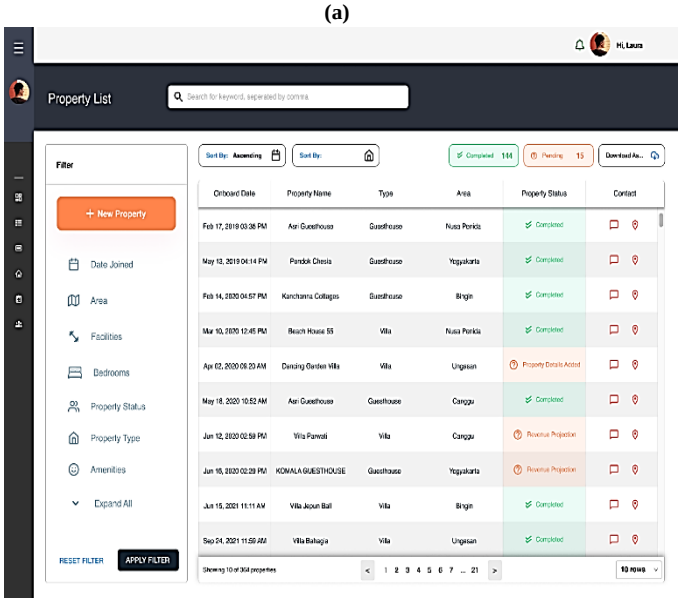
No.	SRSC-Id	Parameter	Requirements Description
1.	SRSC-HPMS-0007	Availability	24 hours a day
2.	SRSC-HPMS-0008	Reliability	Never fails
3.	SRSC-HPMS-0009	Portability	Can be accessed using any browser and must have an account
4.	SRSC-HPMS-0010	Communication	English and Indonesian

3.2.4 Feature Design

The property listing feature underwent a redesign process led by User Interface / User Experience (UI/UX) Designers. These designers collaborated with the Product Development team to evaluate the potential improvements in the feature's quality and assess its feasibility for implementation by developers. Being a critical component of the system, the property listing feature required a fresh approach to enhance its usability, visual appeal, and overall user experience. By leveraging the expertise of UI/UX Designers and closely collaborating with the Product Development team, the goal was to address usability issues, enhance navigation, and optimize the visual presentation. This was achieved through the identification of pain points, gathering user feedback, and ensuring that the design enhancements could be effectively translated into actionable development tasks. The outcome of this collaborative redesign effort drives the evolution of the property listing features towards improved quality and user satisfaction, resulting in a visually appealing and user-friendly interface while maintaining the core functionality of the property listing feature. The successful implementation of the redesign demonstrated in the Figure 2.



(a)



(b)

Figure 2: The (a) Old Design and (b) New Design of Property Listing Feature

In the latest design, as shown in Figure 2 (b), several additional small features have been incorporated. These include: (1) buttons for sorting data by name or date in ascending or descending order, (2) a date joined filter to facilitate data filtering based on contract date, and (3) an

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 5, pp 304-311, May-2023

https://doi.org/10.47001/IRJIET/2023.705043

indicator for property partner status, displayed as “completed” or “pending”. It should be noted that during the development stage, certain changes may occur based on considerations arising from encountered issues during the refactoring process. Consequently, the final appearance may differ from the existing design, while maintaining the same functionality.

3.3 Code Refactoring of the Property Listing Feature

After completing the refactoring design process, the functional requirements for the property list feature are obtained, along with a new design for the property listing. These two elements enable developers to proceed with refactoring the feature.

3.3.1 The Data inside Property List

Within the property listing features, the property data will be displayed according to the discussed specifications and the new design of the feature. Please refer to Table 3 for the data.

Table 3: Data inside the Property List

No.	Data Name	Data Type
1.	Onboard Date	Date
2.	Property Name	String
3.	Type	String
4.	Area	String
5.	Property Status	String
6.	Google Maps Link	Link
7.	WhatsApp Link	Link

The property data within the property listing feature is retrieved from the Back-end in the initial form of JSON (JavaScript Object Notation). Subsequently, the JSON data is processed on the Front-end to facilitate its display within the property list table.

3.3.2 Creating Table Component

The table component is created to display a list of properties along with their corresponding data. This table component is also reusable, which means it can be utilized for similar features that require data to be displayed in a table format. By making it a reusable component, it helps reduce code duplication and promotes code efficiency.

A table component named Dynamic Table is created using the ReactJS functional component. The props parameter is later used to receive property data that is passed into this table component. This enables the table component to display the property list data. The program code to retrieve the property list data is implemented based on the provided props, as shown in the following code snippet.

```
<Dynamic Table
handleGetTrProps = {handleShowProperty}
filteredObjectKeys = {allObjectKeysPropertyList}
cell = {colCell}
data = {tableData}
count = {totalItemsFetched}
page = {currentPage}
onPageChange = {handlePageChange}
rowsPerPage = {rowsPerPage}
onRowsPerPageChange = {handleChangeRowsPerPage}
loading = {fetchPropertyListLoading}
noDataText = "No property found, please check
the filter"
/>
```

3.3.3 Creating Filter Component

The filter component is created to display various filters that can be utilized to filter the property list data. The creation of the filter component involves using components from Material UI, including Accordion, Checkbox, KeyboardDatePicker, and Textfield. Additionally, the filter component is designed to be reusable, allowing other features to use this component as well.

```
const DynamicPropertyFilter =
allPropertyFiltersData.map((filter) =>{ return
(
<DynamicFilter
key={filter.label}
filterType={filter.type}
     label={filter.label}
     value={filter.value}
onChange={filter.onChange}
placeholder={filter.placeholder}
searchOption={filter.searchOption}
onChangeSearchOption=
{filter.onChangeSearchOption}
optionList={filter.optionList}
startDate={filter.startDate}
endDate={filter.endDate}
onChangeStartDate=
{filter.onChangeStartDate}
onChangeEndDate={filter.onChangeEndDate}
isClearable={filter.label === "property
statuses" ? false : true}
     />
  );
});
```

The filter component consists of four types of filters: single select, multiple select, date, and range. In the property listing feature, the filter types used include the date filter for the ‘Date Joined’ filter option, the single select filter for the ‘Property Statuses’, and the multiple select filter for Areas, Property Types, and Unit Amenities.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 7, Issue 5, pp 304-311, May-2023
https://doi.org/10.47001/IRJIET/2023.705043

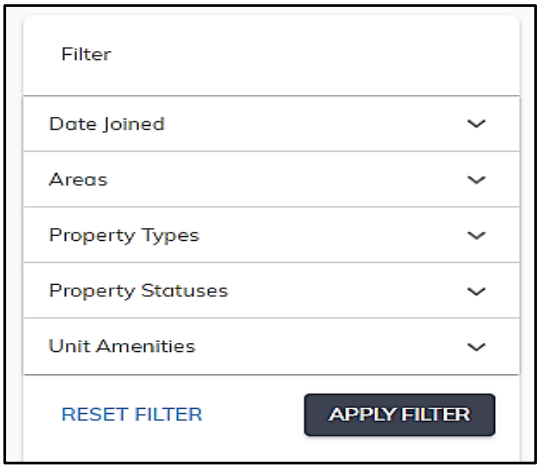


Figure 3: Design of the Property List Filter

The filter type forms become visible when the down arrow symbol is clicked. The forms for the date, single select, and multiple select filter types can be observed in Figure 4.

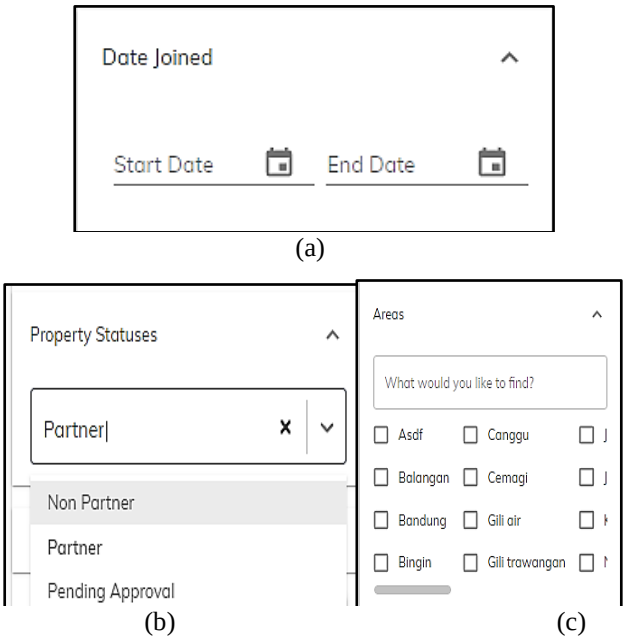


Figure 4: Filter type of (a) date, (b) single select, and (c) multiple select

3.3.4 Creating Property Search Component

The property search component serves as a field where users can search for specific property names by typing their desired keywords. The implementation of this component is relatively simple, using three components from Material UI: Paper, Icon, and Input. UI of the property search input field can be seen in Figure 5and the program code for this component is shown below.

```
<Paper  className="flex  p-4  items-center  w-full
max-w-512 px-8 py-4" elevation={1} >
<Icon className="mr-8"color="action">
Search</Icon>
<Input
      placeholder="Search for property name"
className="flex flex-1"disableUnderline
      fullwidth value={searchHeader}
inputProps={{"aria-label": "Search",}}
onChange={handleSearchHeader}/>
</Paper>
```

Figure 5: Input field of the property search

3.3.5 Creating Sort and Order Components

The sort and order components are implemented using the Chip component from Material UI. These two components serve the purpose of sorting and changing the order of data within the property list. The sorting functionality can be based on either the property name or the property contract date, while the order can be arranged in ascending or descending fashion. UI of the sort and order components can be seen in Figure 6.

Figure 6: The sort and order components

```
<div className="flex gap-8">
<Chipvariant="outlined"
className={classes.chip}
        label={`Sort     By:     ${sortBy     ===
"property_name" ? "Name" : "Date"}`}
        clickableonClick={handleSortBy}/>
<Chip variant="outlined"
className={classes.chip}
        label={`Order: ${sortOrder === "ASC" ?
"Ascending ↑" : "Descending ↓" }`}
        clickable onClick={handleOrder}/>
</div>
```

3.3.6 Creating Indicator Component and CSV File Download Button

The indicator component and the CSV file download button are implemented using the Chip component from Material UI, with only a difference in color. The indicator component displays the status of property partners, distinguishing between those that are complete and those that are still pending or incomplete. On the other hand, the CSV file download button enables users to download the property list data as a CSV or Excel file, allowing them to save it locally. The program code is presented below.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 5, pp 304-311, May-2023

https://doi.org/10.47001/IRJIET/2023.705043

```
<div className="flex gap-8">
<Chipvariant="outlined"
className={`${classes.chip}${classes.chipGreen}`
}
icon={<DoneAllIconclassName="text-green-dark"
/>}
label={`Completed ${completedProperty}`}/>
<Chipvariant="outlined"
className={`${classes.chip}
${classes.chipOrange}`}
      icon={<HelpOutlineIconclassName="text-
orange-dark" />}
      label={`Pending ${pendingProperty}`}/>
<Chipvariant="outlined"
className={classes.chip}
      label="Download CSV"
  icon={<CloudDownloadIconclassName="text-blue"
/>}
      clickable                          onClick={()
=>handleDownloadCsvFile(propListCsv,    "Property
List", user)}/>
</div>
```

UI of the indicator component and download button can be seen in Figure 7.

Figure 7: Input field of the property search

3.3.7 Combining All Components

After completing the creation of the components, they are then integrated into a single property list feature with interconnected functionalities. This stage ends all steps.

3.4 Deployment of the Property Listing Feature

After completing the refactoring stage, the development process is initiated to implement the refactored property list feature in the Hotel Property Management System. The deployment process involves merging the refactored program code into the GitHub repository. The deployment process utilizes three servers: development server for testing, staging server to assess application behavior before going into production, and production server for user access. The system automatically updates the servers, enabling users to utilize and test the property list feature.

Once the deployment process is completed, the next stage is the implementation phase. This phase will provide an explanation of the functionalities and operations of the components within the refactored property listing feature.

3.4.1 Property List Table

The property list table serves as the primary component of the property listing feature. This table presents various information including the Contract Signing Date, Property Name, Property Type, Area Name, Partner Property Status, Google Maps Link, and Whatsapp Link. By default, the property list table displays 10 properties on the first page. Nonetheless, it incorporates a pagination feature with buttons and page indicators located at the bottom right of the table. This allows users to navigate to the next or previous page, enabling them to view another set of 10 properties. Moreover, users have the flexibility to choose the number of properties displayed per page, as illustrated in Figure 8.

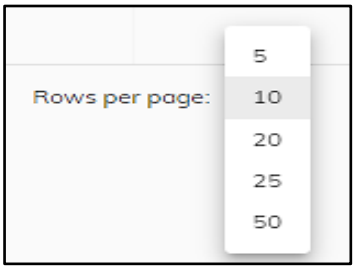


Figure 8: Option to determine the number of data displayed per page

3.4.2 Property List Filter

The purpose of using filters is to narrow down the properties based on specific criteria. To utilize the property list filters, simply select the desired filter, click on the downward arrow to display the filter options, make the appropriate selection, and finally, click the 'Apply Filter' button. The system will then transmit the filter data parameters to the back-end, and the data will be retrieved based on the applied filters. These filters can also be used in combination. For example, by filtering the Canggu area with the Apartment property type, only properties in the Canggu area with the Apartment property type will be displayed.

3.4.3 Property List Data Search

Searching for property data follows a similar process to that of the filter component, albeit with the sole purpose of searching for property names. Users can input keywords or specific property names they wish to search for. For instance, if the keyword "test" is used, any property names containing the word "test" will be displayed. The search functionality can also be combined with the filter components. For example, users can search with the keyword "test" and apply a filter for the property type "Villa".

It is important to note that searching for property names may not always yield result, as the keywords entered might not match any property names within the data list in the system. For instance, when searching with the keyword "prop", if no properties are displayed, it signifies that there are no property names containing the word "prop".

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 5, pp 304-311, May-2023

https://doi.org/10.47001/IRJIET/2023.705043

3.4.4 Property List Data Sort and Order

The property list data is initially sorted by name in ascending order, following an alphabetical arrangement from A to Z. However, users have the flexibility to customize the sorting based on their preferences and requirements. To sort the data by date, users simply need to click on the sort component, which will rearrange the data accordingly. Similarly, the order of the data can be modified by clicking on the order component, enabling users to choose between descending or ascending order. It is worth noting that sorting and sequencing the data can also be applied in conjunction with filtering and property name searching.

3.4.5 Downloading the Property List Data

When users wish to save the property list data as a CSV or Excel file, they can do so by clicking on the 'Download CSV' button located at the top right corner of the table. Upon downloading, the property list data will be saved as a CSV file and will display the same data as presented in the property list table.

3.5 Testing

Once the refactoring process is finished and the features are deployed on the development server, Quality Assurance (QA) testing is necessary before proceeding to the production stage. The testing phase aims to verify that the feature operates in accordance with the intended functional requirements and is free from any bugs that may lead to malfunctions when utilized. The testing process should be conducted iteratively, rather than just once, until no bugs are detected in the feature. This iterative approach ensures that the property list feature is suitable for deployment on production servers.

Table 4: Initial Testing Results

No.	Parameter	Test Results
1.	The property list data is displayed correctly	Success
2.	Pagination in the table is functioning properly	Success
3.	The property list filter is working properly, providing accurate data	Bugs still found when filtering area
4.	The property name search feature is functioning properly, yielding accurate results	Success
5.	It is possible to sort the data by name or date	Success
6.	Both ascending and descending sorting options are available and work properly	Success
7.	The option to download property data as a CSV file is provided and can function properly	Success

Table 4 displays the outcomes of the initial test, revealing the presence of bugs in the property list filter specifically related to area filtering. Consequently, developers are required to perform bug fixes to address the identified issues with the area filter. Subsequently, a retest was conducted and the corresponding results are presented in Table 5.

Table 5: Retest results obtained after bug fixes

No.	Parameter	Test Results
1.	The property list data is displayed correctly	Success
2.	Pagination in the table is functioning properly	Success
3.	The property list filter is working properly, providing accurate data	Success
4.	The property name search feature is functioning properly, yielding accurate results	Success
5.	It is possible to sort the data by name or date	Success
6.	Both ascending and descending sorting options are available and work properly	Success
7.	The option to download property data as a CSV file is provided and can function properly	Success

IV. CONCLUSION

Based on the results of refactoring the property list feature in the hotel property management system, it can be concluded that this study has showed the effectiveness of refactoring in enhancing both the code quality and functional aspects of a feature. Refactoring improves the code structure, making it more organized, readable, understandable, and easier to maintain for developers. The utilization of ReactJS, a component-based JavaScript library, greatly supports the refactoring process for the property listing features, minimizing code repetition. Furthermore, the creation of reusable table components and filter components proved to be valuable in implementing similar table and filter features. These components can be utilized with different datasets, enhancing code reusability, and reducing development time.

For future development, it is essential to carefully redesign the system during the refactoring process to ensure the desired outcomes are achieved. Additionally, in creating components in ReactJS, prioritizing reusability is highly recommended to avoid unnecessary code duplication. Finally, before deploying the feature, thorough testing should be conducted by developers to identify and resolve any remaining errors or bugs. This ensures that the refactored property listing feature delivers an improved user experience compared to its previous state.

REFERENCES

[1] E. Murphy-Hill, C. Parnin,and A. P. Black, "How We Refactor and How We Know It", *IEEE Transactions on Software Engineering*, vol. 38, no.1, pp. 5-18, 2012, doi: 10.1109/tse.2011.41.

[2] M. Fowler, "Refactoring: Improving the Design of Existing Code," Canada, Addison Wesley Longman, Inc., 1999.

ISSN (online): 2581-3048
Volume 7, Issue 5, pp 304-311, May-2023
https://doi.org/10.47001/IRJIET/2023.705043

[3] T. R. Pradana, "Refactoring", https://medium.com/ppl-sutopo/refactoring-6e0e047c285e, 2019.
[4] S. Hammond and D. Umphress, "Test Driven Development: The State of the Practice," ACM-SE '12: *Proceedings of the 50th Annual Southeast Regional Conference*, pp. 158-163, 2012, doi: 10.1145/2184512.2184550.

Citation of this Article:

Patricia Evericho Mountaines, Zulfa Fatah Akbar Ahmad, "Enhancing Code Quality and Feature Functionality through Refactoring of the Property Listing Feature in the Hotel Property Management System" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 5, pp 304-311, May 2023. https://doi.org/10.47001/IRJIET/2023.705043

****

