SOFTWARE REQUIREMENTS SPECIFICATION

For

Tenant–Landlord Information and Management System (TLIMS)

Prepared by:

E.T.C Extra-Terrestrial Coders

Escoreal, Drexzel C.

Ticmon, Mariel M.

Corcelles, Jenneth D.

Submitted in fulfillment requirements for Integrative Programming and Technologies 2 (IT305)

August 16, 2025

Table of Contents

1. Introduction

1.1 Purpose

The Tenant–Landlord Information and Management System (TLIMS) is a web-based platform designed to simplify the process of finding and renting boarding houses, apartments, and rooms near J.H. Cerilles State College (JHCSC) - Pagadian Campus for students, while enabling landlords to post and manage their properties. The system integrates location mapping, notifications, and secure account management to enhance efficiency, transparency, and communication in the rental process. TLIMS aims to provide JHCSC-Pagadian students with a convenient way to search for accommodations near the campus and landlords with a platform to advertise their boarding houses, apartments, or rooms, facilitating direct communication and streamlined transaction management.

1.2 Intended Audience

- **Students/Tenants:** JHCSC-Pagadian students seeking boarding houses, apartments, or rooms near the campus.
- **Landlords:** Property owners near JHCSC-Pagadian who wish to rent out boarding houses, apartments, or individual rooms to students.

1.3 Product Scope

TLIMS provides a comprehensive platform for JHCSC-Pagadian students to search, filter, and view listings of boarding houses, apartments, or rooms near the campus. It offers landlords a dashboard to manage property postings, including pricing, availability, and photos. The system includes secure registration and login functionality for tenants and landlords to protect user data. Integration with Google Maps API enables accurate display of property locations, and notification features keep users informed about new listings, inquiries, bookings, and updates. TLIMS streamlines communication and transactions between tenants and landlords, making the rental process organized, transparent, and user-friendly.

1.4 Definitions, Acronyms, and Abbreviations

- **TLIMS:** Tenant–Landlord Information and Management System
- **Tenant:** A JHCSC-Pagadian student seeking to rent a boarding house, apartment, or room.
- **Landlord:** A property owner near JHCSC-Pagadian offering boarding houses, apartments, or rooms for rent.
- **API:** Application Programming Interface
- **GUI:** Graphical User Interface
- **JHCSC:** J.H. Cerilles State College - Pagadian Campus

2. Overall Description

TLIMS is a web-based application that simplifies the rental process for JHCSC-Pagadian students and landlords. It allows landlords to post listings for boarding houses, apartments, or rooms, including details such as location, price, and photos. Students can browse, search, and filter these listings based on preferences and contact landlords directly. The system enhances rental record management, and notifications ensuring an efficient and transparent experience.

2.1 User Characteristics

- **Students/Tenants:** JHCSC-Pagadian students with basic computer and internet skills, interested in renting accommodations near the campus.
- **Landlords**: Property owners familiar with posting information online, managing rental properties, and providing accurate details about their boarding houses, apartments, or rooms.

2.2 Constraints

- The system requires an internet connection for web access.
- Google Maps API usage is subject to API key limitations.
- Users must have valid email addresses to register.
- Data storage is limited by the hosting server capacity.

2.3 Assumptions and Dependencies

- Users have access to devices (e.g., desktops, laptops, or mobile devices) running modern web browsers.
- Landlords will provide accurate and up-to-date information about their properties.
- Students and landlords will actively use notification features for communication.
- System performance depends on the availability of the Google Maps API and server uptime.

3. Specific Requirements

3.1 Functional Requirements

- **User Registration & Login**: Secure account creation with role-based access for tenants and landlords.
- **Property Posting (Landlord)**: Landlords can add, edit, or delete listings for boarding houses, apartments, or rooms, including details like price, description, and photos.
- **Property Listings (Student)**: Students can browse, search, and filter listings based on preferences (e.g., price, proximity to JHCSC, availability).
- **Map Integration**: Display accurate locations of properties using Leaflet API.
- **Notifications**: Send alerts for new listings, inquiries, bookings, or updates.
- **Inquiries**: Allow students to send inquiries to landlords about specific listings.
- **Booking**: Enable students to book rooms, with landlords managing booking confirmations.

3.2 Non-functional Requirements

- **Performance**: Search and filter operations respond within 2-3 seconds.
- **Security**: Implement encrypted login (e.g., bcrypt for passwords) and secure database handling to protect user data.
- **Usability**: Provide a simple, intuitive, and responsive interface for both desktop and mobile users.
- **Reliability**: Ensure system uptime of at least 95%.
- **Scalability**: Support an increasing number of users and listings, starting with the JHCSC-Pagadian community.

3.3 External Interface Requirements

3.3.1 User Interfaces

- **Student Interface**: A web-based dashboard for students to search, filter, view property details, send inquiries, and manage bookings.
- **Landlord Interface**: A dashboard for landlords to add, update, or remove property listings and manage inquiries and bookings.
- **Graphical User Interface (GUI)**: Responsive design compatible with desktop and mobile browsers, ensuring ease of use.

3.3.2 Hardware Interfaces

- **Client Devices**: Desktops, laptops, and mobile devices with internet access and modern web browsers.
- **Server**: A web server with sufficient processing power, RAM, and storage to support concurrent users.

3.3.3 Software Interfaces

- **Operating System Compatibility**: Compatible with Windows, Linux, and Android/iOS (via browsers).
- **Database**: MySQL or similar RDBMS for structured data storage.
- **Web Technologies**: HTML, CSS, JavaScript (front-end), and PHP or Node.js (back-end).
- **External APIs**: Map Integration (Leaflet) API for location-based services.

3.3.4 Communication Protocols

- **HTTP/HTTPS**: For secure client-server communication.
- **TCP/IP**: For reliable data transmission over the internet.

3.4. System Models

The Use Case Diagram illustrates the main interactions between students and landlords with the Tenant–Landlord Information and Management System (TLIMS).

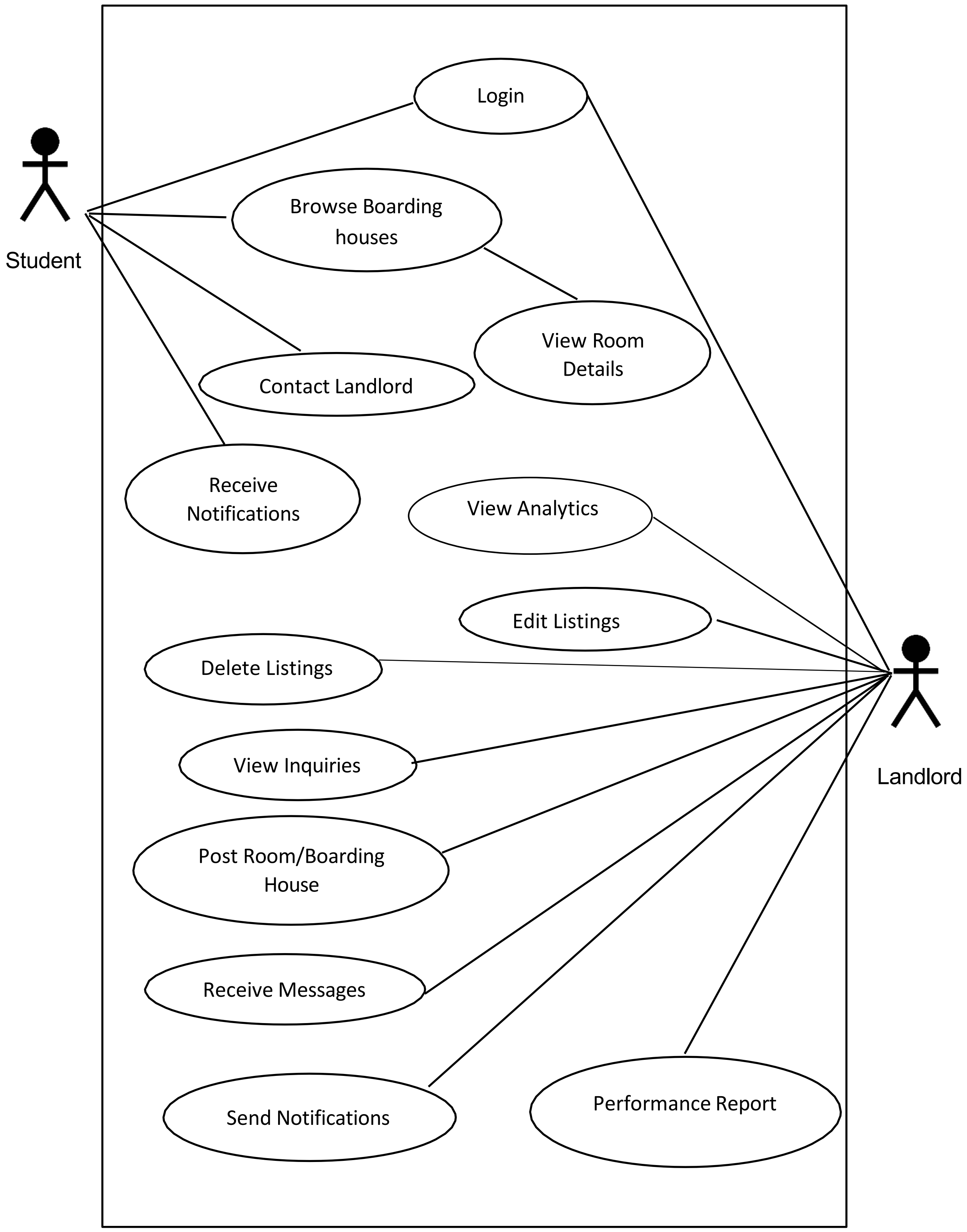

Figure 1: Use case Diagram

3.5 Database Design

The Entity Relationship Diagram (ERD) illustrates how users, properties, rooms, inquiries, bookings, and notifications are structured and connected within the Tenant–Landlord Information and Management System (TLIMS).

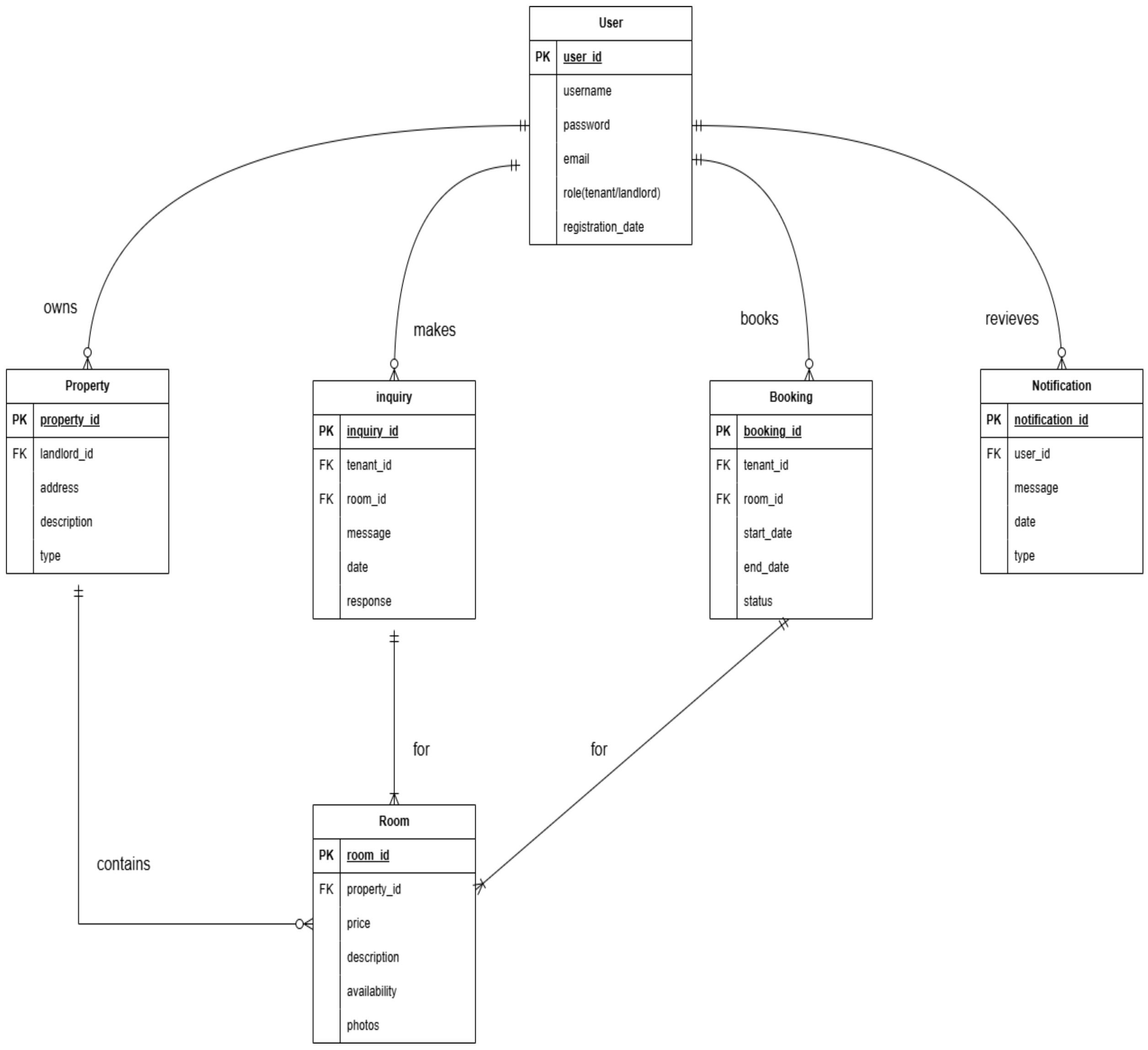

Figure 2: Entity Relationship Diagram (ERD)

3.6 Implementation

TLIMS will be developed using a full-stack web development approach. The front-end will use HTML, CSS, and JavaScript for responsive interfaces. The back-end will use PHP for server-side logic, handling requests, authentication, and API integrations. MySQL will manage the database, with tables corresponding to the ERD entities.

Key Implementation Steps:

- Set up the development environment using XAMPP (local testing) or a cloud server.
- Implement user authentication with PHP sessions and hashed passwords (e.g., bcrypt).
- Integrate Google Maps API using JavaScript SDK for property location display.
- Develop CRUD operations for property postings and listings using PHP and AJAX for seamless updates.
- Implement a notification system using WebSockets or polling for real-time alerts.
- Ensure security through HTTPS, input validation, and SQL injection prevention.
- Follow agile methodology with iterative sprints, incorporating feedback from JHCSC-Pagadian students and landlords.

3.7 Testing and Deployment

Tenant Landlord Information Management System is a web-based system where tenants search and book rental properties while landlords post and manage listings. A “user” refers to anyone performing actions on the system, a “transaction” is any operation that interacts with the database (such as search, login, or booking), and “load” refers to the number of active users or volume of stored data during testing. Our load test used JMeter to simulate concurrent student and landlord access across the key endpoints (Get Properties List, Get Notifications, Get Messages, Get Inquiries, Get Bookings) while varying the database size of property records.

Metrics Defined

Average API response time in milliseconds (aggregated across all endpoints). The SRS Section 3.2 non-functional requirement specifies that search and filter operations must respond within 2–3 seconds.

Testing Environment

Tests were conducted on a local machine using XAMPP (Apache, PHP, MySQL) on a Windows laptop using Apache JMeter 5.6+ (no cloud or distributed). All tests were performed using Google Chrome with a standard home internet connection.

3.7.1 Load Test Results

System Context: The TLIMS (Tenant–Landlord Information Management System) is a web-based platform for managing rental properties within the JHCSC–Pagadian campus community. Tenants can browse property listings, send inquiries, make bookings, and manage their profiles. Landlords can post properties, manage rooms, view bookings, respond to messages, and receive notifications. Our load test simulated concurrent tenant interactions, including loading dashboards, retrieving property listings, sending messages, and receiving notifications while gradually increasing database size.

Table 1: Load Test Performance Metrics

Database Load (records)	Average Response Time (milliseconds)	System Performance Observation
1,000 records	23.16 ms	The page loads instantly, property listings and notifications display without delay, and all interactions feel smooth and fully responsive.
3,000 records	30.47 ms	Slightly noticeable delay on tenant dashboard. Some actions take 1–2 seconds depending on server load. UI remains mostly responsive.
7,000 records	153.29 ms	The page and UI are noticeably laggy and not very responsive. Browsing properties and using search filters take mostly 3–5 seconds to respond.

10,000 records	Unresponsive	The system becomes largely unresponsive. Pages fail to load promptly, API requests time out or take very long, and the UI is practically unusable under this load.

The load test demonstrates that TLIMS system performance degrades predictably as both the database size and user activity increase. With a moderate database of 1,000–3,000 records, the system handles tenant interactions efficiently, and the UI remains mostly responsive. However, as the database scales to 7,000 records, the page and UI become noticeably laggy, with property browsing, search filters, and API calls taking 3–5 seconds to respond, indicating that query efficiency and server processing are becoming bottlenecks. At the maximum tested load of 10,000 records, the system becomes largely unresponsive, with pages failing to load promptly, API requests taking very long, and the UI practically unusable. This confirms that the primary limitation lies in the database query handling and API response management rather than the application server itself. Therefore, optimization efforts should focus on improving database indexing, implementing query caching, and using pagination to maintain acceptable performance as TLIMS scales.

3.7.2 Stress Test Results

System Context: Tenant Landlord Information Management System is a web-based platform used by tenants to search and book rental properties and by landlords to manage and update their listings. The stress test evaluates how the system behaves when both the database size and user activity exceed normal levels at the same time, simulating extreme scenarios on a local machine such as multiple tenants performing searches while landlords upload or update property listings simultaneously.

Table 2: Stress Test Performance and Breaking Point Analysis

Database Load	Concurrent Users & Actions	System Response	Observation & Failure Mode
10,000 records	50 Users Performing: • Tenant login • Messaging • Booking properties • Toggle favorite • Landlord login • Add property	Response Time: ~1368 ms	The system successfully handled 50 concurrent users with stable and consistent performance. All core modules responded within acceptable limits, with 0% error rate and reliable read/write operations.
10,000 records	500 Users Performing: • Tenant login • Messaging • Booking properties • Toggle favorite • Landlord login • Add property	Response Time: ~1368 ms	The system remained functional with 500 concurrent users and no request failures, but significant performance degradation was observed in the tenant login module due to heavy concurrent authentication load. Other modules remained mostly stable.
10,000 records	1500 Users Performing: • Tenant login • Messaging • Booking properties • Toggle favorite • Landlord login	Response Time: N/A (System Unresponsive)	The system failed to reliably handle 1500 concurrent users. High latency, timeouts, and inconsistent responses occurred. The login and booking modules became the primary bottlenecks, indicating a clear system breaking point.

Recovery Test (Return to 4,800 records & 50 users)	800 Users Performing: • Tenant login • Messaging • Booking properties • Toggle favorite • Landlord login • Add property	Response Time: ~2448 ms	The system successfully recovered to a stable state after reducing the data size and load, with 0% error rate at 800 users. However, login delays and occasional response spikes persisted, indicating remaining resource contention and performance limitations.

The stress test of the TLIMS system was conducted using Apache JMeter to simulate concurrent user activity and evaluate system performance under load. The results show that the system performs reliably under moderate load. With 10,000 database records and 50 concurrent users, the system maintained fast and stable response times with 0% error rate.

When the load increased to 500 concurrent users, the system remained operational but showed performance degradation, particularly in the tenant login process due to heavy concurrent authentication requests. At 1500 concurrent users, the system reached its breaking point, becoming unstable and unresponsive, especially in the login and booking modules.

During the recovery test using 4,800 records and 800 concurrent users, the system stabilized again with no request failures, although login operations still showed noticeable delays and occasional spikes. These findings indicate that while the system can handle medium traffic, it requires database optimization, efficient connection handling, and caching improvements before it can reliably support high-concurrency environments.

3.7.3 Beta test

Testing

The proponents have devised a comprehensive testing strategy to evaluate the accuracy and efficiency of the Tenant–Landlord Information and Management System (TLIMS), ensuring it operates as intended.

This testing plan is developed using established methodologies and procedures familiar to the proponents who are testing the system.

Table 9 - Test Plan for Tenant–Landlord Information and Management System (TLIMS)

Test Item	Tenant–Landlord Information and Management System (TLIMS). A web system that handles property listings, bookings, and tenant–landlord communication near JHCSC–Pagadian Campus.
Modules and Functions to be tested	Modules: • User Management • Property Management • Property Browsing & Search • Map Integration • Inquiry & Messaging • Booking Module • Notification System • Performance & Security • Database & System Integration
Test Environment	Software: • Windows 10 • XAMPP • MySQL • Web Browsers Hardware: • Computer/Laptop • Internet Connection • Mobile Devices (Smartphone or Tablet

Testing Criteria

Software Evaluation Sheet

Table 10 – Questioner Range and Description

RANGE	**QUANTITATIVE DESCRIPTION**
5	Strongly Agree
4	Agree
3	Neither Agree or Disagree
3	Disagree
2	Disagree
1	Strongly Disagree

Table 11 – Evaluation Criteria for Usability

Usability	**Strongly Agree (5)**	**Agree (4)**	**Neither Agree or Disagree (3)**	**Disagree (2)**	**Strongly Disagree (1)**
1. The system was easy to navigate and understand.	**13**	**14**	**3**	**0**	**0**
2. The functions of the system (searching, posting, booking, messaging) were well-integrated and worked smoothly.	**11**	**16**	**3**	**0**	**0**
3. I could learn to use the system quickly without difficulty.	**13**	**12**	**4**	**0**	**0**
4. The interface was responsive and displayed properly on different devices (computer & mobile).	**11**	**14**	**5**	**0**	**0**

5. I felt confident using the system to perform tasks such as browsing listings or managing properties.	**12**	**11**	**5**	**0**	**0**

Based on the responses of 30 respondents in the User Acceptance Testing, the Usability Evaluation of TLIMS yielded consistently high ratings across all criteria. Overall, the combined results show that approximately 82%–90% of respondents *Strongly Agreed* or *Agreed* that the system is easy to navigate, quick to learn, responsive across devices, and effective in supporting key tasks such as browsing listings and managing properties. Meanwhile, 10%–18% of respondents selected *Neither Agree nor Disagree*, indicating neutral feedback that may reflect minor usability adjustments or user familiarity. Importantly, 0% of respondents chose *Disagree* or *Strongly Disagree* for all usability indicators. Overall, the results show that TLIMS is very acceptable in terms of usability and successfully passed the User Acceptance Test.

Table 12 – Evaluation Criteria for Reliability

Reliability	**Strongly Agree (5)**	**Agree (4)**	**Neither Agree or Disagree (3)**	**Disagree (2)**	**Strongly Disagree (1)**
1. The system performed tasks consistently without errors or unexpected failures.	**12**	**10**	**7**	**0**	**0**
2. Pages and system functions loaded properly and responded within the expected time.	**8**	**19**	**3**	**0**	**0**
3. The system maintained stable performance even when multiple users accessed it.	**10**	**12**	**8**	**0**	**0**

4. Data (e.g., listings, bookings, inquiries) were saved, updated, and retrieved accurately.	**11**	**16**	**3**	**0**	**0**
5. The system remained available and accessible whenever I tried to use it.	**13**	**10**	**7**	**0**	**0**

Based on feedback from 30 respondents, the reliability of TLIMS received positive results. Around 73% to 90% of the respondents Strongly Agreed or Agreed that the system performs consistently, loads properly, maintains stable performance, accurately saves data, and remains accessible when needed. About 10% to 27% gave neutral responses, while no respondents disagreed with any reliability statements. Overall, the results show that TLIMS is reliable and stable and successfully passed the User Acceptance Test for reliability.

Table 13 – Evaluation Criteria for Interface

Interface	**Strongly Agree (5)**	**Agree (4)**	**Neither Agree or Disagree (3)**	**Disagree (2)**	**Strongly Disagree (1)**
1. The system's interface layout was clear, organized, and easy to understand.	**16**	**9**	**5**	**0**	**0**
2. The design and visual elements (buttons, forms, icons) were consistent across all pages.	**9**	**18**	**3**	**0**	**0**
3. The interface was responsive and displayed properly on both computer and mobile devices.	**14**	**12**	**4**	**0**	**0**
4. The text, labels, and instructions were readable and easy to follow.	**13**	**13**	**3**	**0**	**0**

Based on feedback from 30 respondents, the TLIMS interface received very positive results. About 80% to 90% of the respondents Strongly Agreed or Agreed that the system’s layout is clear, visually consistent, responsive on both computer and mobile devices, and easy to read and understand. Around 10% to 20% gave neutral responses, and no respondents disagreed with any interface criteria. Overall, the results show that the TLIMS interface is user-friendly, well-designed, and acceptable, successfully passing the User Acceptance Test for interface quality.