International Journal Software Engineering and Computer Science (IJSECS)
5 (3), 2025, 1260-1273
Published Online December 2025 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN: 2776-4869, E-ISSN: 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v5i3.5443.

RESEARCH ARTICLE Open Access

Development of a Proximity-Based Pet Adoption Website Application

Muhamad Akbar Susanto *
Informatics Engineering Study Program, Faculty of Information Technology, Universitas Kristen Satya Wacana, Salatiga City, Central Java Province, Indonesia.
Corresponding Email: 672021140@student.uksw.edu.

Pratyaksa Ocsa Nugraha Saian
Informatics Engineering Study Program, Faculty of Information Technology, Universitas Kristen Satya Wacana, Salatiga City, Central Java Province, Indonesia.
Email: pratyaksa.ocsa@uksw.edu.

Received: August 26, 2025; Accepted: November 15, 2025; Published: December 1, 2025.

Abstract: This study examines the development of a web-based application designed to facilitate pet adoption through practical, structured, and location-based mechanisms. The primary problem addressed is the absence of digital platforms capable of integrating adoption processes more efficiently than traditional methods. The application employs Java Spring Boot for backend architecture, ReactJS for frontend interface, and MongoDB for data management. The Haversine formula with a 50 km radius was implemented to display animals based on proximity to users. Algorithm implementation results demonstrate that Haversine effectively calculates distances and presents animals according to nearest locations, thereby enhancing search accuracy and relevance for prospective adopters. Core functionality includes a WebSocket-based real-time chat system enabling direct communication between prospective adopters and owners without page reloading. System development followed the Waterfall model encompassing requirements analysis, design, implementation, and testing phases. Evaluation through User Acceptance Testing (UAT) using a 1-4 Likert scale yielded satisfaction rates of 82% among prospective adopters and 84.5% among pet owners, both categorized as "highly satisfied." These findings validate that the application serves as a more efficient and structured alternative, with potential for further development to support expanding user bases in the future.

Keywords: Pet Adoption; Haversine; Spring Boot; ReactJS; MongoDB.

1. Introduction

Animals are living creatures created by God to coexist in the world alongside humans, and many Indonesians have the habit of keeping pets, even making it a hobby, such as dogs, cats, hamsters, various types of fish, and others. Reasons for keeping animals vary—they can serve as companions, leisure activities, or simply for enjoying their beauty; some people even care for stray animals. However, many animals are less fortunate due to irresponsible human behavior, causing them to lose their homes or even their lives due to lack of food [1], or because owners can no longer afford to care for them. One solution to address these problems is through pet adoption, but the process is often hindered by limited information and location constraints. Communities frequently struggle to find nearby animals for adoption, necessitating the development of a system that not only facilitates adoption but also recommends animals based on proximity to users' locations to enhance the effectiveness and efficiency of the process.

Pet adoption is typically pursued by individuals who wish to own pets, with information obtained through various channels such as word-of-mouth recommendations, direct visits to pet owners, pet stores, or internet searches through various communities and social media platforms like Facebook. However, irrelevant threads, duplicate posts, and spam often confuse prospective adopters, causing them to miss opportunities to obtain desired animals [2]. To address these issues, a specialized web-based application was developed to handle the pet adoption process, making it accessible to everyone regardless of the device used. The application enables prospective adopters to share information about animals available for adoption, communicate directly with pet owners, and display animals based on nearest location, thereby facilitating adopters who wish to avoid distance and access barriers. The application was built using the Java programming language with the Spring Boot framework for the backend, ReactJS for the frontend, and MongoDB as the database. ReactJS was selected because it is an open-source framework that offers advantages in speed, simplicity, and scalability. React also enables UI (User Interface) components to be more interactive, stateful, and reusable [3]. MongoDB was chosen as the database because it employs a document-based data model, eliminating the need for users to design table structures as in SQL. These documents are in JSON-like format, specifically BSON. MongoDB features a flexible schema, as it can automatically create table structures during insertion. MongoDB offers larger data storage capacity and delivers performance that is considerably better compared to MySQL [4].

The problem formulation focuses on how the system facilitates the discovery of nearby animals, optimizes communication between adopters and owners, and evaluates the effectiveness of digital applications as alternative solutions for pet adoption. The objective of this research is to build a platform that simplifies the adoption process through location search and chat features, making adoption activities easier, more efficient, and transparent. Within the research scope, the application covers only the upload and adoption processes for animals, validation by administrators, and communication between users. There is no price determination for animals, as the platform operates on a voluntary adoption basis. Data security aspects, including the use of location information and privacy of inter-user communication, as well as application performance, are limited as part of the research scope and defined as problem boundaries, ensuring that research focus remains on core feature development and user interaction.

2. Related Work

Recent studies have explored digital platforms and the Haversine algorithm for pet adoption and location-based systems, revealing both progress and persistent gaps. The mobile application "Woofland" was developed to reduce fraud in dog adoption processes in Surabaya, though its developers acknowledged the need for interface improvements and enhanced database functionality to support login, chat, and wishlist features [5]. In a different domain, a WebGIS implementation for mapping universities across West Java and Banten demonstrated the practical application of the Haversine formula (using Earth's radius of 6.371 km) combined with Google Maps API to recommend nearest campus locations, employing Rapid Application Development (RAD) methodology and GTMetrix for technical validation [6]. Another Android-based application for stray animal adoption achieved 97.78% accuracy in distance calculations compared to Google Maps using the Haversine method, though it remained confined to the Android platform without communication features or administrative management capabilities [7]. The web-based "Our Pets" platform utilized Haversine calculations for determining distances to veterinary clinics, built with PHP/Laravel and MySQL, demonstrating how information technology can integrate essential services including location search, adoption facilitation, and lost pet tracking [8]. While these previous studies—from "Woofland" mobile app to Android platforms and Haversine-based WebGIS implementations—have validated the effectiveness of distance calculations and user interaction features, they typically suffer from platform-specific limitations and lack fixed radius parameters or real-time chat services. The current research addresses these shortcomings by developing a web-based platform accessible across multiple devices, implementing the Haversine mechanism with a fixed maximum radius of 50 km, and incorporating real-time chat functionality to strengthen communication between prospective adopters and pet owners. This approach not only centralizes adoption information on a single flexible platform but also streamlines the adoption process through accurate nearest-location recommendations and direct communication channels.

The theoretical foundation encompasses several key technologies. ReactJS, developed by Facebook, facilitates the creation of interactive, stateful, and reusable component-based user interfaces, delivering speed, simplicity, and scalability for single-page applications [9]. MongoDB, as a JSON document-based NoSQL database, offers high performance, ease of data management, and schema flexibility to accommodate diverse data variations [10]. JavaScript provides dynamic logic and advanced interactive capabilities for websites, complementing HTML structure and CSS styling with complex and responsive functions [11]. The Haversine formula calculates great-circle distance between two points on Earth's surface based on latitude and longitude

coordinates, assuming Earth as a perfect sphere with a radius of 6.371 km, thereby supporting nearest-location recommendations with high precision [12]. Spring Boot was selected as the backend framework due to its open-source nature, high modularity, and lightweight performance, facilitating integration of various modules for software solution development [13]. The WebSocket protocol enables bidirectional real-time data exchange between server and client by maintaining open connections through HTTP mechanisms, minimizing session opening and closing overhead while supporting in-application chat communication [14].

This research investigates several questions: how to design a user-friendly pet adoption application capable of bridging communication between adopters and owners; whether the application can display animals based on adopter-specified regions; to what extent the nearest-distance recommendation feature assists users in finding desired animals; whether in-app chat features facilitate the adoption process; and whether using Java with Spring Boot framework and MongoDB database proves efficient for web-based pet adoption application development. The technology stack was chosen for specific advantages: Spring Boot for rapid and modular backend development, ReactJS for building interactive and responsive interfaces, and MongoDB for document-based data storage flexibility suitable for varied animal data. Based on theoretical foundations and prior research, the hypotheses propose that the application can bridge and simplify pet adoption processes for both adopters and owners; nearest-animal search features enhance user convenience in finding animals or prospective new owners near user locations; real-time chat features facilitate communication and adoption agreements; and despite its simplicity, the application can help reduce the number of stray animals. Table 1 compares previous research approaches. Most existing pet adoption applications remain limited to Android platforms, lack real-time communication features, and do not provide search radius constraints. This research offers advantages through web-based platform development accessible across devices, Haversine algorithm integration with a fixed 50 km radius to simplify searches, and real-time chat feature additions that strengthen communication between prospective adopters and pet owners. The study not only addresses limitations of previous research but also delivers more practical, structured solutions aligned with contemporary user needs.

Table 1. Research Comparison

No	Reference	Platform	Algorithm & Radius	Advantages / Notes
1	Mobile-Apps 'Woofland' Dog Adoption Platform in Surabaya	Android	No Haversine mentioned, focus on filter features	Focus on shelter-based dog adoption; requires UI and database development (authors suggest improvements)
2	WebGIS Implementation for University Mapping in West Java and Banten with Distance Recommendation Using Haversine Formula	Website (WebGIS)	Uses Haversine formula for distance recommendation	Demonstrates Haversine application in WebGIS systems; relevant as distance calculation method study
3	Design and Development of "Our Pets" Application Using Haversine Algorithm	Website (Our Pets)	Uses Haversine algorithm for distance calculation	Utilizes Haversine for distance calculation in adoption app; publication indicates no real-time chat integration
4	Implementation of Haversine Formula Method in Android-Based Stray Animal Adoption Application	Android	Uses Haversine algorithm	Calculates distance with high precision, facilitates stray animal search by nearest location. However, limited to Android with simple features (no chat, admin portal, multi-device)
5	Current Research	Website	Uses Haversine algorithm with 50 km radius	Multi-device, real-time chat (WebSocket), 50 km radius, UAT: 82%/84.5% satisfaction — integration of location features + real-time communication

3. Research Method

This research employs a qualitative approach where data is collected through in-depth interviews to evaluate the effectiveness and functionality of the developed system. The approach aims to obtain comprehensive understanding regarding user experiences and perceptions toward the pet adoption application. Interview results are analyzed descriptively to draw conclusions about the extent to which the application meets user needs and can be effectively utilized in pet adoption processes. The Waterfall model

serves as the foundation for system development methodology in this application. The design of the web-based pet adoption application involves several necessary processes, summarized in the flowchart shown in Figure 1 [15]. The initial step in this method involves conducting preliminary studies by identifying problems and formulating them into problem statements. Based on these problem formulations, data collection is conducted through interviews and literature reviews. Following data collection, the next process involves data processing. To support data processing, sequential and interconnected processes must be performed, as illustrated in Figure 1. This method depicts the stages of application development encompassing Requirements Analysis, System Design, Implementation, System Testing, and Application Maintenance.

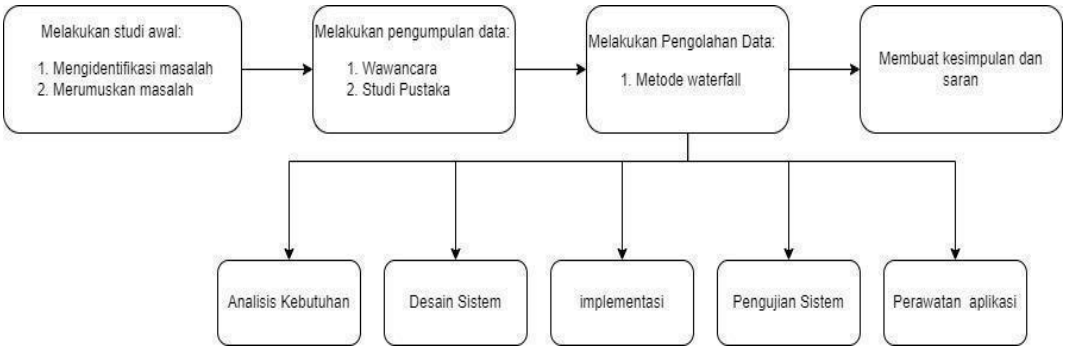

Figure 1. Research Steps

The Waterfall model is a development method that divides the process into several sequential stages including requirements analysis, design, implementation, testing, and maintenance. Each stage is completed first, and generally returning to previous stages is not recommended [16]. Although the application features real-time chat, its technical design can be determined from the design stage, thereby minimizing specification change risks. In this research, the requirements analysis stage involves interviews with prospective adopters and literature studies to formulate functional requirements (animal data CRUD operations, distance-based search, real-time chat) and non-functional requirements (ease of use, security, accessibility).

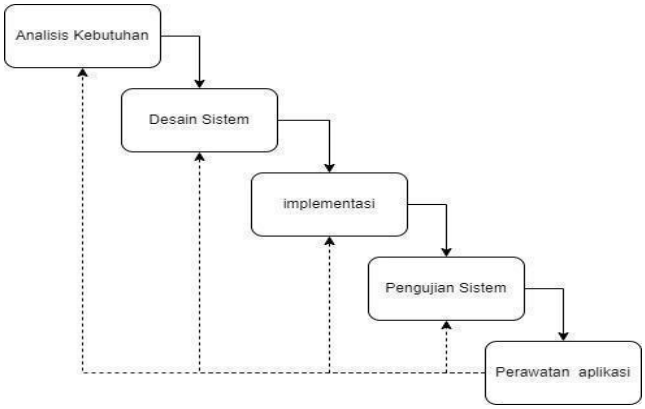

Figure 2. Waterfall Application Development Method

After requirements are identified, system design is created with MVC architecture using Spring Boot and MongoDB for the backend, and ReactJS, JavaScript, HTML, CSS, and Bootstrap for the frontend. Integration between Spring Boot and ReactJS is accomplished through REST API serving as an API gateway, enabling structured and secure data communication between backend and frontend. On the frontend side, state management uses React's built-in features (useState and useEffect) to ensure data received from the backend can be displayed and updated without disrupting application consistency. Implementation realizes business logic for animal data management, authentication, location search, real-time chat, and responsive user interfaces. Additionally, to calculate distance between user location and adoptable animals, the application employs the Haversine algorithm which converts geographic coordinates into actual distances on Earth's surface. This formula was selected because it provides accurate results by considering Earth's curvature. Detailed explanation of calculation stages and implementation is presented in the discussion section. The 50 km radius was chosen as the search limit because it is considered realistic and still easily accessible for prospective adopters in daily mobility contexts. The testing stage is then divided into functionality testing (Black Box Testing of main features) and UAT (involving prospective users to assess ease of use, effectiveness, and usage satisfaction).

The final stage encompasses maintenance through performance monitoring, error reporting, and satisfaction surveys for planning improvements and feature updates. The entire process is supported using tools such as Visual Studio Code, MongoDB, Figma, Draw.io, and responsiveness testing on mobile devices. To model system structure and workflow in a standardized manner, Unified Modeling Language (UML) is used through activity diagrams and use cases that minimize specification ambiguity and enhance design efficiency [17].

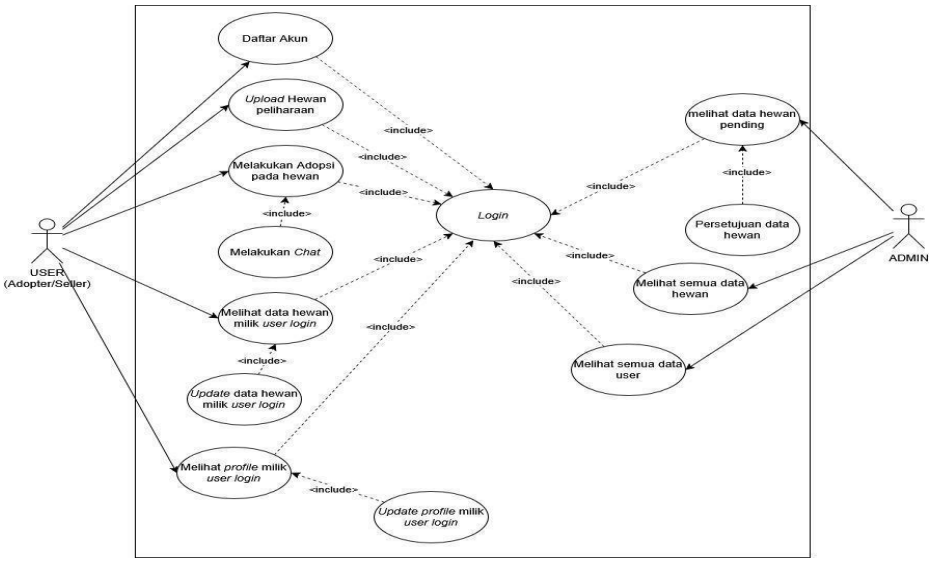

Figure 3. Use Case Diagram

Figure 3 presents a use case diagram illustrating the main functions in the pet adoption application and interaction scenarios between actors, both users and other systems, to systematically identify functional requirements [18]. In this application, there are two main actors: user and admin. The admin actor represents the system manager who has authority to approve animal uploads from users, view all animal data, and access information for all users. Meanwhile, users include prospective adopters and pet owners. Users can upload pets, perform adoptions, update uploaded animal data, modify user profiles, and utilize chat features to communicate directly between users. Beyond use cases, system modeling is complemented with activity diagrams explaining workflow processes from start to finish [19]. These diagrams help detail system functionality mechanisms and ensure that development flows align with designed requirements. The following are example activity diagrams for core processes in this application: animal upload and animal adoption.

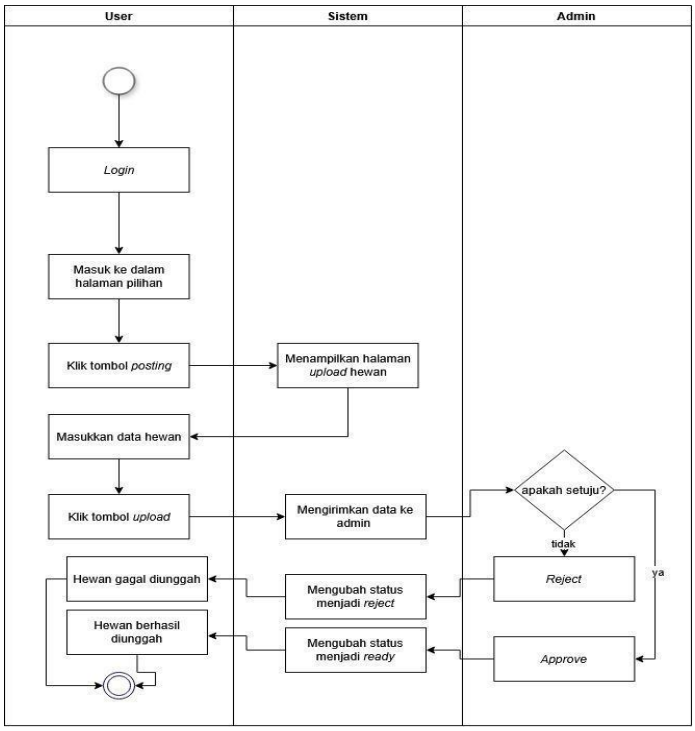

Figure 4. Animal Upload Activity Diagram

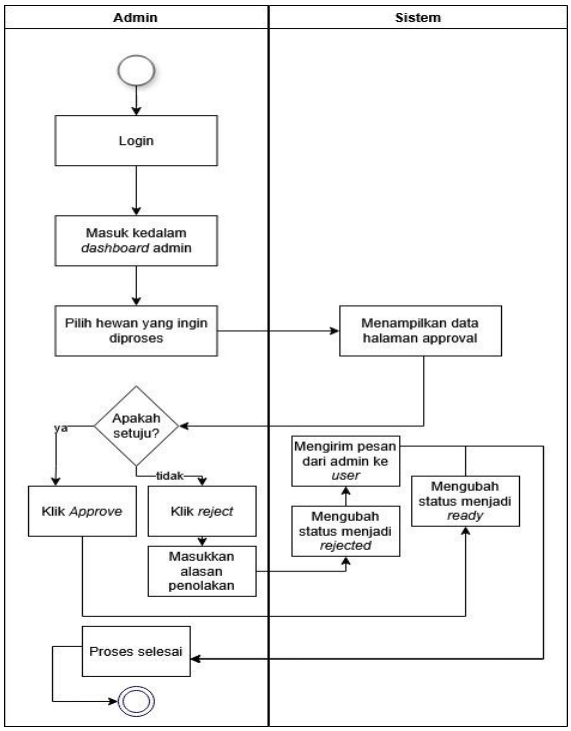

Figure 5. Validation Process Activity Diagram

Figure 4 represents the Activity Diagram for the pet upload process. In the process, after users log in and enter the options page, users can directly select the posting menu. The system then displays the animal upload page. Users can directly enter owned animal data according to fields provided by the application. After completing required animal data, users can press the upload button and data will be directly sent by the system to the admin dashboard. In the admin dashboard, the animal data will be manually processed by the admin. If the admin approves it, the system will change the pet's status to ready, thus the animal is successfully uploaded. If animal data is rejected, the animal will fail to be uploaded as shown in Figure 5.

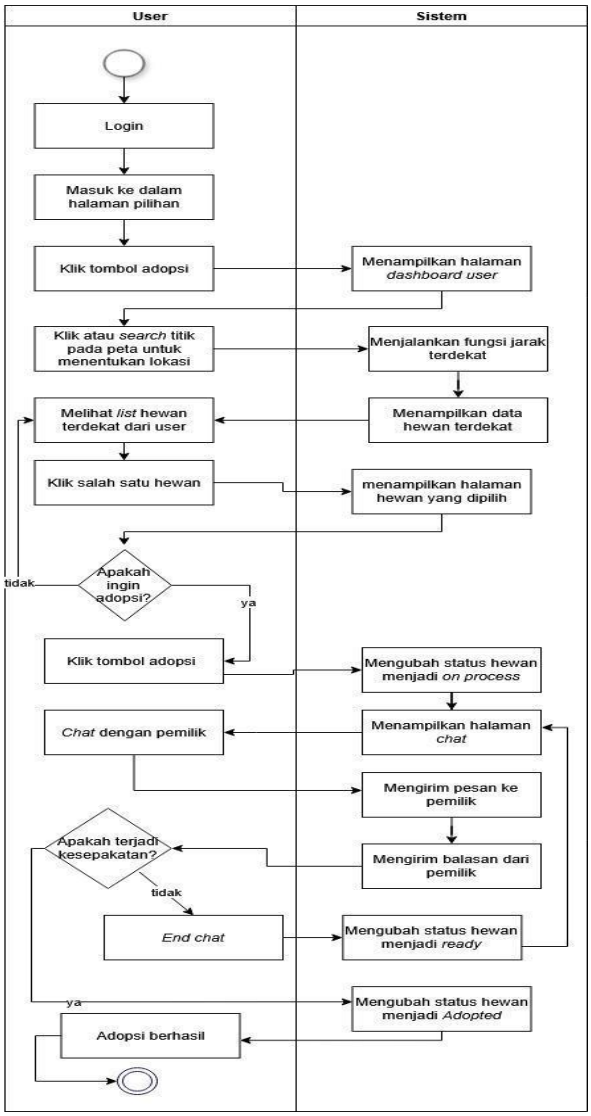

Figure 6. Animal Adoption Activity Diagram

Figure 6 displays an activity diagram illustrating the animal adoption process flow in the application. After users successfully log in, they are directed to the main page and select the adoption button. The system then displays a user dashboard containing an interactive map to determine location. Users can perform manual location searches or directly select specific points on the map. After the location point is determined, the system automatically executes distance calculation functions using the Haversine method and displays a list of animals within the nearest radius from that location. Users can select an animal they wish to adopt, then view detailed information about that animal. If users are not interested in that animal, they can return to the main page without continuing the adoption process. However, if users are interested in adopting, they can press the "Adopt" button to initiate the process. After users press that button, the system changes the animal's status to "on process" and opens a chat feature between the prospective adopter and pet owner. When the pet owner replies to messages, the system displays that response in real-time to the user. If communication proceeds smoothly and both parties reach an agreement, the system changes the animal's status to "adopted." Conversely, if no agreement is reached, users can select the end chat option, and the system returns the animal's status to "ready." This flow ensures that the adoption process proceeds interactively, efficiently, and integrated within a single platform.

4. Result and Discussion

4.1 Results

This research presents the design of a web-based pet adoption application. The application employs a nearest distance calculation mechanism to facilitate prospective adopters in finding animals within the nearest radius. By focusing on location-based features, the application aims to overcome information reach limitations and improve search time efficiency. Additionally, the platform provides real-time communication channels between prospective adopters and pet owners, enabling negotiation and adoption agreements to proceed more quickly and in a more structured manner.

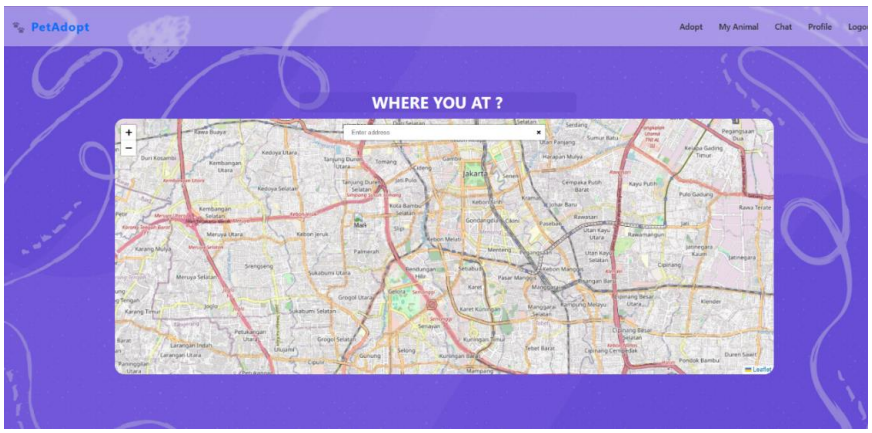

Figure 7. Animal Search Page

The application facilitates nearest animal searches by utilizing location input from users. This input can be obtained by clicking on a digital map or through location name searches. After users determine a reference point, the system converts that location into latitude and longitude coordinates. The generated coordinates are then processed by the geolocation module on the server. In the database, each animal entry also stores location coordinate information. Using the Haversine formula, the system calculates the shortest distance along Earth's surface. Subsequently, the system performs filtering to select all animals within a maximum radius of 50 km from the reference point.

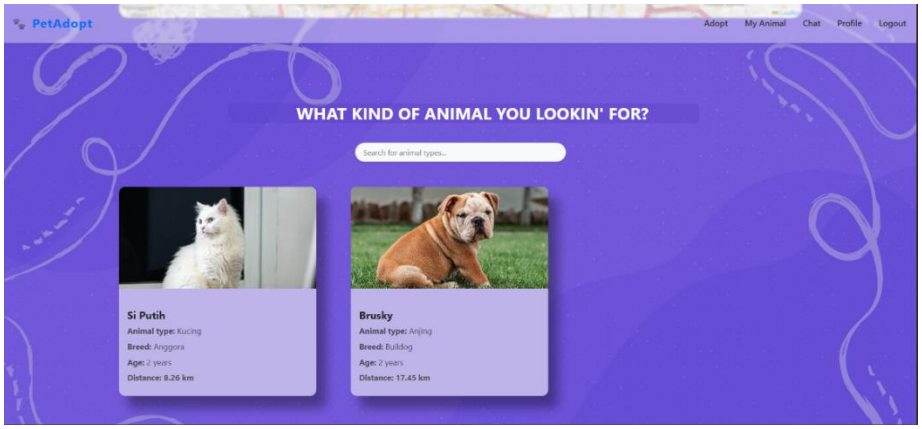

Figure 8. Animal Search Page (2)

The list of animals that pass the filtering process is sorted by nearest distance. The system then displays these results to users in the form of an animal list equipped with brief information such as photos, names, and estimated distances. Users can click on one item in the list to view detailed animal profile information and initiate conversations with owners through the chat feature. This flow ensures that the nearest adoption animal search process proceeds accurately, quickly, and is easily accessible. To obtain the nearest adoption animals accurately, the application requires precise calculations using Haversine. The first step is converting the difference in latitude coordinates (Δlat) and longitude (Δlong) into radians, then calculating the Haversine value using the formula:

$$a = sin^2(\Delta lat/2) + cos(lat1) . cos(lat2) . sin^2(\Delta long/2).$$

Subsequently, distance [d] is obtained through:

$$d = 2R.arcsin(\sqrt{a})$$

where [R] is Earth's radius valued at approximately 6.371 km. Sequentially, these mathematical stages illustrate the process of transforming angular values into actual distances on Earth's surface without relying on code-based implementation [12]. This formula is then implemented in program code to calculate distances between coordinate points used in this research, as shown in the following code snippet.

Code Program 1: Service for Calculating Distance

```
@Service
public class HitungJarakService {
    private static final double R = 6371;
    private static final double MAX_JARAK_KM = 50;

    public double hitungJarak(double lat1, double lon1, double lat2, double lon2) {
        if (!isValidCoordinate(lat1, lon1) || !isValidCoordinate(lat2, lon2)) {
            throw new IllegalArgumentException("Koordinat tidak valid.");
        }
        double dLat = Math.toRadians(lat2 - lat1);
        double dLon = Math.toRadians(lon2 - lon1);
        double a = Math.sin(dLat / 2) * Math.sin(dLat / 2)
                + Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2))
                * Math.sin(dLon / 2) * Math.sin(dLon / 2);
        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
        double jarak = R * c; // Result in kilometers
        return (jarak > MAX_JARAK_KM) ? -1 : jarak;
    }

    private boolean isValidCoordinate(double lat, double lon) {
        return lat >= -90 && lat <= 90 && lon >= -180 && lon <= 180;
    }
}
```

Code Program 1 represents backend code that automatically calculates distance between two coordinates using the Haversine formula. The calculation process also validates latitude and longitude values, then converts the difference between both values into radians. The system limits the displayed animal list to only within a 50 km radius. If the distance exceeds 50 km, the system returns a value of -1 so that only animals within that radius range are displayed.

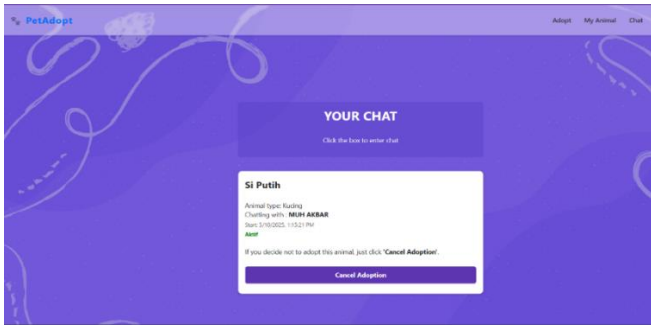

Figure 9. Adopter Chat

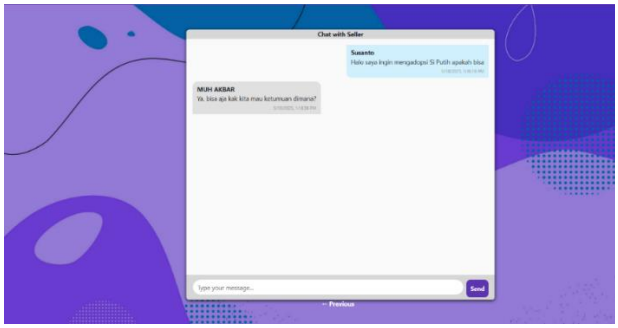

Figure 10. Chat Room

The application facilitates communication between prospective adopters and pet owners through WebSocket-based real-time chat features. When prospective adopters press the "Adopt" button on an animal profile, the system automatically opens a chat room that can be used by both parties. Within that chat room, users can exchange text messages regarding animal health conditions, behavior, or special needs without needing to reload the page. Through push mechanisms via WebSocket connections, each message is sent and received instantly, enabling discussions to proceed interactively and responsively. The system stores all message history in the MongoDB database, and access to chat rooms is restricted only to prospective adopters and pet owners involved to keep communication private and focused. After going through various stages, the next process is system testing aimed at ensuring the application runs properly and aligns with its initial formation objectives. System testing encompasses all application functions and user responses during usage. At this stage, the application is tested using two methods, one of which is the Black Box method. The Black Box method is used to test the functionality of each stage, including login, account registration, pet posting, updating and deleting animal data, buttons, and other features in the application. Black-box testing is a testing method that evaluates systems from the functional side based on interface specifications without examining internal code structure.

This approach was chosen because it enables systematic and reproducible testing at the API contract level without requiring access to source code [20]. Additionally, this testing aims to find functional errors, features that do not appear, or behaviors that do not match application specifications. Thus, this method helps ensure the application functions as intended. The following are testing results in table form for this pet adoption application.

Table 1. Black Box Testing

Menu	Test Case	Expected Result	Test Result	Status
Login	Login (correct username and password)	User successfully enters the options page	User successfully enters the options page	Pass
	Login (incorrect username or password)	Error message "Username or password is incorrect" appears	Error message "Username or password is incorrect" appears	Pass
	Enter page according to role	Admin enters admin page and user enters user page	Successfully differentiates authentication for admin and user	Pass
Register Account	Enter complete user data	Account registered and displays successful registration notification	Account registered and displays successful registration notification	Pass
	Incorrect input or missing data	Account not registered and displays failed registration notification	Account not registered and displays failed registration notification	Pass
Post Animal	Enter complete pet data	Animal posting process and displays successful posting notification	Animal posted and displays successful posting notification	Pass
	Enter incomplete data or missing data	Pet posting process fails and displays failed posting notification	Pet posting fails and displays failed posting notification	Pass
Post Option Button	Click post button	Enter pet posting page	Successfully enters pet posting page	Pass
Adopt Option Button	Click adopt button	Enter user dashboard page	Successfully enters user dashboard page	Pass
Display Nearest Animals Feature	Click on map area or search on map	Display animals within 50 km based on selected point or location	System displays animals within maximum 50 km distance	Pass
Adoption Animal Detail Page	Click on animal to adopt	Display detail page for animal selected by user for adoption	Display detail page for animal selected by user for adoption	Pass
Adopt Button on Animal Detail Page	Click Adopt button	Enter chat room with animal owner	Successfully enters chat room page with owner	Pass
		Update animal status to on process	Successfully updates animal status to on process	Pass
My Animal	Click My Animal navigation button	Display all animals owned by user	Successfully displays all animals owned by user	Pass
	Click on one animal	Display selected animal details	Display selected animal details	Pass
Update Pet	Change existing data correctly	Successfully update data for selected animal	Successfully update data for selected animal	Pass
	Incorrect input or missing data	Display failed process and must be corrected	Update process fails and data returns to original state	Pass
Delete Pet	Click Delete animal button	Animal data removed from animal database	Data in animal database deleted	Pass
Chat	Click chat navigation button	Display all chats owned by user	All user chats successfully displayed	Pass
	Click on one chat list	Display chat room according to selected list	Display chat room according to selected list	Pass

	Fill message field, click send (send message)	Can send messages between adopter and pet owner	Successfully sends messages between adopter and pet owner	Pass
Adopted Button	Click Adopted button (owner)	Change animal status from on process to adopted	Status changes from on process to adopted	Pass
Reject Button	Click Reject button (owner)	Change animal status to ready	Status changes to ready	Pass
Cancel Adoption Button	Click Cancel adoption button (adopter)	Change animal status to ready	Successfully changes animal status to ready	Pass
Profile	Click navigation button	Display user profile page	Display user profile page	Pass
Update Profile	Change existing data correctly	Successfully update user profile data	Successfully update user profile data	Pass
	Incorrect input or missing data	Display failed process and must be corrected	Update process fails and data returns to original state	Pass
Animal Approval Page	Open pending page	Display all animals with pending status	Display all animals with pending status	Pass
	Click on one existing animal	Display selected animal details	Display selected animal details	Pass
Approve Button	Click Approve button	Change animal status to ready	Animal status becomes ready	Pass
Reject Button	Click Reject button	Change animal status (rejected)	Animal status becomes rejected	Pass
Animal Page	Click Animal navigation button	Display all existing animal data	Display all existing animal data	Pass
	Click on one animal	Display selected animal data	Display selected animal data	Pass
User Page on Admin	Click User navigation button	Display all users	Display all users	Pass
	Click on one user	Display selected user data	Selected user data displayed	Pass
Logout	Logout menu	Navigate to login page	Login page displayed	Pass

4.2 Discussion

User Acceptance Testing (UAT) is the final assessment stage directly involving target users, namely prospective adopters and pet owners. Its purpose is to verify that the adoption application has met functional and non-functional requirements before launch. Besides testing technical reliability and feature completeness, UAT also assesses interface ease of use and interaction comfort. In UAT implementation, participants obtain full access to all main application modules, including adoption processes, pet data posting, chat systems, and profile updates. Testing is conducted in an environment that mimics actual operational conditions, enabling users to provide direct feedback regarding navigation, display clarity, and system response speed. Input from the UAT phase becomes the basis for final improvements, both technically and in terms of usability, before the system is officially implemented. Thus, UAT validates solution suitability from non-technical user perspectives, reduces potential obstacles after release, and maximizes satisfaction levels and application adoption among users. In this testing, several questions are used to evaluate the developed application. These questions aim to ensure that the developed application not only meets technical standards but is also useful for users. There are two types of questions: questions for adopters and questions for pet owners. The following is the list of questions:

Table 2. Adopter Question List

No	Question List	Rating			
		1	2	3	4
1	This application provides a viable alternative compared to conventional animal adoption methods.	1	2	3	4
2	The interface and workflow of the application make it easy for me to understand the adoption process as a whole.	1	2	3	4
3	The chat feature in the application makes it easier for me to communicate with pet owners.	1	2	3	4
4	The information I receive through chat is sufficient to make adoption decisions.	1	2	3	4

5	The nearest animal search feature displays results that match my location.	1	2	3	4
6	Radius parameters and coordinate points are easy for me to set when searching for animals.	1	2	3	4
7	This application increases my chances of getting the desired pet.	1	2	3	4
8	The nearest animal search distance feature is effective for finding desired animals.	1	2	3	4
9	This application has stable performance with minimal disruptions.	1	2	3	4
10	Overall, I am satisfied with the performance and features provided by the application.	1	2	3	4

Table 3. Pet Provider Question List

No	Question List	Rating			
		1	2	3	4
1	The new animal registration form is easy to understand and fill out.	1	2	3	4
2	The animal photo upload process runs smoothly without technical issues.	1	2	3	4
3	The location input feature (longitude and latitude) functions accurately and is easy to use.	1	2	3	4
4	The system provides clear notifications when animal data is successfully saved.	1	2	3	4
5	The navigation menu in the pet provider dashboard is intuitive and easy to access.	1	2	3	4
6	I can edit animal information (type, breed, age, description, etc.) quickly and easily.	1	2	3	4
7	I can clearly identify the status of uploaded animals.	1	2	3	4
8	This application increases my chances of finding prospective adopters for my pets.	1	2	3	4
9	The chat feature in the application makes it easier for me to communicate with adopters.	1	2	3	4
10	Overall, I feel the features provided meet the needs of pet providers.	1	2	3	4

Table 4. Adopter and Pet Owner Assessment

User Satisfaction Level	Rating
Strongly Disagree	0%-25.99%
Disagree	26%-50.99%
Agree	51%-75.99%
Strongly Agree	76%-100%

The Likert Scale in User Acceptance Testing (UAT) is a method to measure user attitudes and satisfaction toward systems with several answer choices, for example from "Strongly Disagree" to "Strongly Agree". Each answer is measured with a numerical value, then summed to obtain a total score [21]. This score can be analyzed statistically and converted into percentages or categories as shown in Table 4 to assess how well the system meets user needs. With this approach, researchers obtain measurable and objective data about system acceptance levels. Several calculations must be performed to achieve this objective, including the following:

$$Maximum\ Score\ =\ 4\ \mathrm{x}\ times\ number\ of\ \mathrm{respondents}\ \mathrm{x}\ times\ number\ of\ questions$$

After the total value is obtained from each answer already measured in numerical form and the maximum score is also obtained, calculation can continue by calculating the User Acceptance Testing percentage using the formula:

$$User\ Acceptance\ Testing\ = \frac{Total\ Value}{Maximum\ Score} \times 100$$

Assessment of UAT questionnaires for the Design of Nearest Distance-Based Web Pet Adoption Application will be conducted using the Likert Scale, with result interpretation as follows:

Table 5. Adopter Results

Answer	Assessment	Total
Strongly disagree	0	0
Disagree	7	14
Agree	93	279
Strongly Agree	50	200
Total Value		493
Maximum Score		600
User Satisfaction Percentage		82%

Table 6. Pet Owner Results

Answer	Assessment	Total
Strongly disagree	0	0
Disagree	10	20
Agree	42	126
Strongly Agree	48	192
Total Value		338
Maximum Score		400
User Satisfaction Percentage		84.5%

Testing results shown in Table 5 and Table 6 using User Acceptance Testing (UAT) method with Likert scale assessment demonstrate that the Design of Nearest Distance-Based Web Pet Adoption Application obtained highly satisfactory results, both from the adopter side and pet provider side. Based on questionnaires completed by adopters, the application obtained a result percentage of 82% regarding application usage. From the pet provider side, the application obtained a percentage of 84.5%. Thus, it can be concluded that the developed application has met functionality and usability criteria according to user needs and expectations, making it suitable for wide-scale implementation. Testing results also reinforce findings that utilization of digital technology in pet adoption processes provides positive contributions, particularly in facilitating prospective adopters to conduct adoption processes more efficiently and in a more directed manner. The same applies to pet providers who can find prospective adopters for their pets more efficiently. Additionally, chat features and nearest distance-based animal search features are proven capable of helping users obtain pets according to their desires. The discussion in this research explains how each feature can answer formulated problems, evaluates advantages and limitations of the developed system, and provides suggestions for further development in the future.

5. Conclusion and Recommendations

This web-based pet adoption application successfully assists users in finding available animals within a 50 km radius through coordinate calculations using the Haversine formula. The application is built using Java Spring Boot as the backend, ReactJS as the frontend, and MongoDB as the database, providing flexibility in document storage. This technology combination has proven to support stable and efficient application performance. Additionally, the Waterfall method used in the system development process ensures that each stage, from requirements analysis, design, implementation, to testing, can be executed systematically. The system is capable of displaying animal lists accurately according to user locations. WebSocket-based real-time chat features also support smooth communication between prospective adopters and pet owners, with chat rooms automatically opening when the "Adopt" button is pressed. UAT results demonstrate high satisfaction levels of 82% from adopters and 84.5% from owners, indicating that this application is effective as a more organized digital adoption alternative compared to conventional methods.

Although main functionalities have been fulfilled, further development is still necessary. Search features can be enhanced with additional filters such as animal type, age, or health conditions to increase relevance. Furthermore, to improve security and trust among users, the addition of identity verification features and post-adoption review systems is recommended. User experience can also be strengthened through integration of more interactive map APIs. This research has limitations in testing geographic scope which remains limited, so results do not yet fully represent conditions in broader regions. Moving forward, development recommendations include developing user reputation systems and animal preference matching features to make the adoption process more personalized. From the implementation side, partnership strategies with animal protection NGOs are suggested to expand platform reach and sustainability. With continuous development, this application is expected to support faster, more accurate, and more responsible adoption processes while improving animal welfare.

References

[1] Pamungkas, A. B., & Nastiti, N. E. (2020). Perancangan Media Informasi Berbasis Teknologi Mobile Tentang Tempat Penampungan Hewan Terlantar Di Bandung. *eProceedings of Art & Design*, *7*(3), 6100–6112.

[2] Septianto, D., & Windriyani, P. (2022). Pengembangan Aplikasi Adopsi Hewan Kucing Berbasis Website. *KALBISIANA Jurnal Sains, Bisnis dan Teknologi*, *8*(2), 1915-1923.

[3] Nursaid, F. F., Brata, A. H., & Kharisma, A. P. (2020). Pengembangan Sistem Informasi Pengelolaan Persediaan Barang Dengan ReactJS Dan React Native Menggunakan Prototype (Studi Kasus: Toko Uda Fajri). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *4*(1), 46-55.

[4] Maharani, A. (2022). Perancangan data base kasir dan persediaan barang menggunakan MongoDB. *Jurnal Data Mining dan Sistem Informasi*, *3*(1), 32. https://doi.org/10.33365/jdmsi.v3i1.1941

[5] Chrestella, D. V., & Cahyadi, J. (2022). Perancangan mobile-apps 'Woofland' platform adopsi anjing di Surabaya. *Jurnal DKV Adiwarna*, *1*, 1–11. https://publication.petra.ac.id/index.php/dkv/article/view/12267/10753

[6] Hasanah, N. I. (2024). *Implementasi WebGIS pemetaan perguruan tinggi di wilayah Jawa Barat dan Banten dengan fitur rekomendasi jarak menggunakan formula Haversine* [Undergraduate thesis, Universitas Nusa Putra]. Repository Universitas Nusa Putra. https://repository.nusaputra.ac.id/1326/1/NURUL%20ISLAM%20HASANAH%20.pdf

[7] Supiatma, A. S. W. (2020). *Implementasi metode Haversine formula pada aplikasi adopsi hewan terlantar berbasis Android* [Undergraduate thesis, STMIK Pradnya Paramita]. Digital Library STMIK Pradnya Paramita. https://digilib.stimata.ac.id/index.php?id=5148

[8] Amanda, N. (2022). *Rancang bangun aplikasi "Our Pets" menggunakan algoritma Haversine* [Undergraduate thesis, Universitas Mercu Buana Jakarta]. Repository Universitas Mercu Buana. http://repository.mercubuana.ac.id/id/eprint/71306

[9] Ridho, M. R., Fajrah, N., Afriana, A., & Fifi, F. (2022). Aplikasi marketplace wisata dengan community based tourism. *SUBMIT: Jurnal Ilmiah Teknologi Infomasi dan Sains*, *2*(1), 31–37. https://doi.org/10.36815/SUBMIT.V2I2.1887

[10] Rionald, V., Fadillah, A. R., & Awangga, R. M. (n.d.). *Kombinasi Golang, MongoDB dan JavaScript untuk pengembangan aplikasi pengelolaan koleksi museum*. Google Books. https://books.google.co.id/books?hl=id&lr=&id=zeCuEAAAQBAJ&oi=fnd&pg=PA1&dq=Definisi+mongodb&ots=O1dnaHbtja&sig=tgxWwzsZix1R9lk1SxiTDgZaDNc&redir_esc=y#v=onepage&q=Definisi%20mongodb&f=false

[11] Salim, R., Arisandi, D., & Hendryli, J. (2022). Pembuatan Aplikasi MOSTRANS Transporter Berbasis Mobile Menggunakan React-Native JavaScript. *Jurnal Ilmu Komputer dan Sistem Informasi*, *10*(1). https://doi.org/10.24912/jiksi.v10i1.17856.

[12] Palupi, R., Yulianna, D. A., & Winarsih, S. S. (2021). Analisa perbandingan rumus Haversine dan rumus Euclidean berbasis sistem informasi geografis menggunakan metode independent sample t-test. *JITU: Journal Informatic Technology and Communication*, *5*(1), 40–47. https://doi.org/10.36787/JTI.V14I2.270

[13] Prihantari, R. A. (2024). *Rancang bangun sistem informasi e-commerce berbasis web menggunakan framework Spring Boot pada toko GIGHA STEEL* [Undergraduate thesis].

[14] Alviando, L., Bhawiyuga, A., & Kartikasari, D. P. (2023). Penerapan Websocket pada Sistem Live Chat berbasis Web (Studi Kasus Website Kwikku. com). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *7*(2), 854-862.

[15] Dorotheoputra, S. (2023). *Implementasi sistem tender pengiriman dengan teknologi Medoo PHP berbasis website (Studi kasus: PT. Sarana Bangun Pusaka)* [Undergraduate thesis, Universitas Kristen Satya Wacana]. Repository UKSW. https://repository.uksw.edu//handle/123456789/31848

[16] Prenner, N., Unger-Windeler, C., & Schneider, K. (2021). Goals and challenges in hybrid software development approaches. *Journal of Software: Evolution and Process*, *33*(11), e2382. https://doi.org/10.1002/smr.2382

[17] Margaretha, J., & Voutama, A. (2023). Perancangan Sistem Informasi Pemesanan Tiket Konser Musik Berbasis Web Menggunakan Unified Modeling Language (UML). *JOINS (Journal of Information System)*, *8*(1), 20-31. https://doi.org/10.33633/joins.v8i1.7107

[18] Ramdany, S. (2024). Penerapan UML class diagram dalam perancangan sistem informasi perpustakaan berbasis web. *Journal of Industrial and Engineering System*, *5*(1). https://doi.org/10.31599/2E9AFP31

[19] Wayahdi, M. R., & Ruziq, F. (2023). Pemodelan sistem penerimaan anggota baru dengan Unified Modeling Language (UML) (Studi kasus: Programmer Association of Battuta). *Jurnal Minfo Polgan*, *12*(1), 1514–1521. https://doi.org/10.33395/JMP.V12I1.12870

[20] Felício, D., Simão, J., & Datia, N. (2023). RapiTest: Continuous black-box testing of RESTful web APIs. *Procedia Computer Science*, *219*, 537–545. https://doi.org/10.1016/j.procs.2023.01.322

[21] Kusuma, A. P., et al. (2024). Analysis of user acceptance testing on a shipping application to determine the quality of the system. *Antivirus: Jurnal Ilmiah Teknik Informatika*, *18*(2), 234–243. https://doi.org/10.35457/ANTIVIRUS.V18I2.4002.