Rentistan

Developed By:

Raja Abdul Basit **UOC-IT-F2020-147**
Muhammad Ehsan Ul Haq **UOC-IT-F2020-178**
Saif Ullah khan **UOC-IT-F2020-150**

Supervised By

Mr. Fahim Abid

Department of Computer Science & Information Technology
Faculty of Computing & Information Technology
The University of Chakwal

Bachelor of Science in Computer Science / Information Technology
(2020-2024)

DECLARATION

We hereby declare that this project, titled **Rentistan - Home Rental App**, has been developed entirely by our team based on our personal efforts and dedication. Throughout the project, we have adhered to academic integrity and professional standards. All components, including the design, code, documentation, and implementation, are original and have been created by us.

We confirm that no part of this project has been copied, reproduced, or sourced from any other material, unless otherwise cited and referenced. We have strictly followed ethical guidelines in conducting our research, development, and testing phases. The data and information used in this project have been obtained legally and ethically.

Furthermore, this project has not been submitted to any other institution or for any other academic or professional purpose. We take full responsibility for the authenticity and originality of the content presented in this documentation and the developed application.

RAJA ABDUL BASIT	EHSAN UL HAQ	SAIF ULLAH KHAN
_____	_____	_____

CERTIFICATE OF APPROVAL

It is to certify that the final year project of BS (IT) "**Rentistan: Home Rental App**" was developed by "**RAJA ABDUL BASIT, UOC-IT-F2020-147"**, "**MUHAMMAD EHSAN UL HAQ, UOC-IT-F2020-178"** AND "**SAIF ULLAH KHAN, UOC-IT-F2020-150"** under the supervision of "**FAHIM ABID**" and that in their opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Information Technology.

FAHIM ABID
Supervisor

Mr. ___
External Examiner

Dr. Rashid Amin
HOD

Abstract

Rentistan is a revolutionary property rental application designed to redefine the rental experience by integrating user-centric design, robust security, scalable technology, and advanced sentiment analysis on reviews. The application addresses the complex needs of both tenants and property managers, providing an intuitive, secure, and efficient platform for managing rental properties. This extended abstract provides an in-depth exploration of **Rentistan's** key features, technological stack, user experience design, and the comprehensive solutions it offers to streamline property rentals.

Introduction

In the contemporary real estate market, the demand for efficient and accessible property rental solutions has grown significantly. Traditional methods often involve cumbersome processes, lack of transparency, and insufficient security measures. **Rentistan** addresses these challenges by leveraging modern technology to create a seamless rental experience. The application is designed with a focus on three core principles: ease of use, security, and scalability. This abstract delves into each of these aspects, illustrating how **Rentistan** successfully integrates them into a cohesive platform.

Ease of Use

Rentistan prioritizes user experience by offering an interface that is both intuitive and user-friendly. The application's design is centered around guiding users through every stage of the rental process with minimal effort. Key features include:

- **User Interface Design**: The frontend of **Rentistan** employs a clean and straightforward layout that ensures users can easily navigate through property listings, application forms, and account settings. The use of clear icons, simple navigation menus, and responsive design elements contributes to a seamless user experience.
- **Search and Filters**: To enhance the property search experience, **Rentistan** includes advanced search functionality with customizable filters. Users can search for properties based on location, price range, number of bedrooms, and other criteria, allowing them to find suitable options quickly.
- **Interactive Maps**: Integration with interactive maps provides users with a visual representation of property locations. This feature helps users understand the geographical context of available properties and plan their visits more effectively.
- **Application and Booking Process**: The application process is streamlined with pre-filled forms and user-friendly input methods. Tenants can apply for properties with a few taps, and property managers can review applications and manage bookings through a centralized dashboard.

Security

Security is a fundamental concern for **Rentistan**, and the application employs several measures to protect user data and ensure secure interactions:

- **Authentication and Authorization**: **Rentistan** uses Firebase Authentication to handle user login and registration. This service provides secure authentication methods, including email/password login, and social media integration. Multi-factor authentication (MFA) is implemented to add an additional layer of security.
- **Data Encryption**: All user data, including personal information and rental agreements, is encrypted both in transit and at rest. This encryption ensures that sensitive information remains confidential and protected from unauthorized access.
- **Secure Transactions**: Although **Rentistan** does not handle payments directly, it provides a secure platform for users to submit payment details and verify transactions. The application supports manual payment confirmation processes, allowing users to provide proof of payment for verification.
- **Fraud Prevention**: The app includes mechanisms for detecting and reporting fraudulent activities. Users can report suspicious activities, and a dedicated support team addresses these reports to prevent fraudulent transactions and ensure a secure environment.

Scalability

To accommodate future growth and ensure consistent performance, **Rentistan** is designed with scalability in mind:

- **Architecture**: The application's architecture supports scaling both horizontally and vertically. By using cloud-based services like Firebase, **Rentistan** can efficiently manage increased traffic and data volume without degrading performance.
- **Database Management**: Firestore, a NoSQL database, is utilized for its ability to handle large datasets and provide real-time updates. Firestore's scalability ensures that property listings, user data, and application information remain accessible and up-to-date even as the user base grows.
- **Performance Optimization**: **Rentistan** incorporates performance optimization techniques, including caching strategies and efficient data querying, to maintain responsiveness and speed. The application is designed to handle peak usage times without compromising user experience.

Technological Stack

Rentistan leverages a modern technological stack to deliver a high-quality user experience:

- **Frontend Development**: The frontend is developed using React Native, a framework that enables the creation of cross-platform mobile applications with a single codebase. React Native's component-based architecture allows for reusable UI elements, accelerating development and ensuring consistency across devices.

- **Backend Services**: Firebase serves as the backend platform, providing essential services such as real-time data synchronization, cloud storage, and analytics. Firebase's comprehensive suite of tools supports the app's functionality and enhances its ability to deliver real-time updates and notifications.
- **Notification System**: To keep users informed, **Rentistan** integrates Expo Notifications for timely alerts and updates. The notification system is designed to deliver relevant information, such as new messages and application updates, directly to users' devices.

User Experience and Design

The design philosophy of **Rentistan** focuses on creating a positive user experience through thoughtful design and functionality:

- **Aesthetic Design**: The application features a modern and visually appealing design that enhances usability. Consistent use of colors, fonts, and icons contributes to a cohesive and professional appearance.
- **User Feedback**: The application includes mechanisms for collecting user feedback and suggestions. This feedback is used to continuously improve the app's features and address any issues that arise.
- **Sentiment Analysis on Reviews**: One of the innovative features of **Rentistan** is the use of sentiment analysis to evaluate user reviews. This technology analyzes the sentiment behind user feedback, categorizing reviews as positive, negative, or neutral. This insight helps property owners and managers understand tenant satisfaction and identify areas for improvement. Sentiment analysis is powered by machine learning algorithms that process natural language data, providing accurate and actionable insights. This feature not only enhances the user experience but also helps in maintaining a high standard of service quality across the platform.

Admin Panel

The web-based admin panel for **Rentistan** is a powerful tool for property managers and administrators. Built with React and Firebase, it offers a comprehensive suite of features for managing users, properties, bookings, and more. The admin panel includes functionalities such as user management, property management, booking management, chat management, notifications management, and analytics & reports. The user management feature allows administrators to view, edit, and remove user accounts as needed. Property management enables the addition, modification, and removal of property listings. Booking management provides a centralized view of all bookings, making it easy to track and manage reservations. The chat management feature allows administrators to monitor and moderate conversations between users, ensuring a safe and respectful communication environment. Notifications management helps in sending out important alerts and updates to users. The analytics & reports feature provides detailed insights into the app's performance, user activity, and other key metrics.

Manual Payment Confirmation

Rentistan does not handle payments within the app; users will make payments outside the app. To manage external payments effectively, we have implemented a robust system for manual payment confirmation. This allows users to manually input payment details (like transaction IDs or screenshots) into the app for verification. Tenants can submit proof of payment for property owners to review, ensuring transparency and accountability in the payment process. This system also helps in keeping track of payment histories, making it easier for both tenants and property owners to manage their financial records.

Automated Notifications

To ensure timely payment confirmations and enhance user experience, **Rentistan** sets up automated notifications to remind users to verify payments. Tenants receive prompts to confirm payments in the app, reducing the chances of missed or delayed payments. These notifications are customizable, allowing users to set their preferences for when and how they receive reminders.

Reporting and Support

Rentistan provides a robust system for users to report suspected fraud or payment issues. This feature allows users to flag suspicious activities and seek assistance from our support team. Our support team is equipped to handle disputes and verify transactions manually, ensuring that any issues are resolved promptly and fairly. This system helps in maintaining the integrity of the platform and ensures that users can trust the services provided by **Rentistan**.

Conclusion

Rentistan represents a significant advancement in property rental technology, combining ease of use, robust security, and scalability into a single platform. By leveraging modern technologies and focusing on user-centric design, **Rentistan** offers a comprehensive solution for property management and rental. The application is well-positioned to adapt to future developments and continue delivering value to its users.

Acknowledgement

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor "**FAHIM ABID**" for personal supervision, advice, valuable guidance and completion of this project. We are deeply indebted to him for encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

RAJA ABDUL BASIT EHSAN UL HAQ SAIF ULLAH KHAN

_____ _____ _____

Abbreviations

SRS	Software Requirement Specification
AI	Artificial Intelligence
NFR	Non Functional Requirements
UI	User Interface
UX	User Experience
NoSQL	Not Only SQL
CRUD	Create, Read, Update, Delete
API	Application Programming Interface
JSON	JavaScript Object Notation
SDK	Software Development Kit

Table of Contents

Chapter 1: Introduction....1
1.1. Brief....1
1.2. Project Background....2
1.3. Methodology and Software Lifecycle for This Project....3
1.4. Rationale behind Selected Methodology....3
1.5. Project objectives....4
1.6. Project Scope....5
1.7. Tools and Technologies....6
1.8. Project Summary....7
Chapter 2: Literature Review & Problem Definition....8
2.1. Literature Review....8
2.2. Analysis from Literature Review....9
2.3. Problem Statement....10
2.4. Product Functions....10
2.5. Chapter Summary....11
Chapter 3: Requirement Analysis....12
3.1. Functional Requirements....12
3.2. Non Functional Requirements....14
3.3. Use Cases model....16
3.4. Chapter Summary....27
Chapter 4: Design and Architecture....28
4.1. Proposed Architecture....28
4.2. Deliverables and Development Requirements....29
4.4. Assumptions and Dependencies....31
4.5. UML Diagrams....32
Chapter Summary:....39
Chapter 5: Project Implementation....40
5.1. Programming Language....40
5.2. Framework....40

5.3. System Requirement....41
5.4. Chapter Summary....41
Chapter 6: Software Testing & Maintenance....42
6.1 Deriving Test....42
6.2 Test Environment....43
6.3 Testing identification....44
6.4 Test Procedure....45
6.5 Testing techniques....46
6.6 Test Cases....47
6.7 Chapter Summary....49
Chapter 7: Conclusion and Future work....50
7.1. Discussion....50
7.2. Conclusion....50
7.3. Limitations....50
7.4. Future work....51
References....52

List of Figures

Fig 3.1 Use Case Diagram 16
Fig 3.2 Use Case Diagram 17
Fig 3.3 Use Case Diagram 17
Fig 4.1 Sequence Diagram 33
Fig 4.2 Sequence Diagram 34
Fig 4.3 Sequence Diagram 34
Fig 4.4 Sequence Diagram 35
Fig 4.5 Sequence Diagram 35
Fig 4.6 Sequence Diagram 36
Fig 4.7 Sequence Diagram 36
Fig 4.8 Data Flow Diagram 37
Fig 4.9 Block Diagram 38

List of Tables

Table 3 Use Case Description Table 8
Table 3.1 Signup 18
Table 3.2 Login 18
Table 3.3 View Property.....19
Table 3.4 Search and Filter Property 19
Table 3.5 Contact Renter 19
Table 3.6 Submit Review 20
Table 3.7 Application Status 20
Table 3.8 Update profile 20
Table 3.12 Manage Property Listing 21
Table 3.13 Respond to tenant Inquiries 21
Table 3.9 Review and Approve Applications 21
Table 3.21 View All Users 25
Table 3.22 Manage Users 25
Table 3.24 Monitor and Resolve Issues 26
Table 3.26 Manage Chats 27
Table 3.27 Logout 27

Chapter 1: Introduction

1.1. Brief

Rentistan is an advanced property rental application designed to simplify and enhance the rental process for both tenants and property managers. In response to the growing demand for efficient, transparent, and secure rental solutions, **Rentistan** leverages cutting-edge technology to address these needs comprehensively. The application offers a user-centric interface, robust security measures, and scalable technology to support a seamless rental experience.

The core functionality of **Rentistan** revolves around providing a streamlined and intuitive platform for property management and rental transactions. Users can easily browse property listings, apply for rentals, and manage their bookings through a straightforward, user-friendly interface. For property managers, **Rentistan** offers powerful tools to oversee and manage their listings, applications, and communications with tenants.

Security and data integrity are central to **Rentistan**'s design. By employing Firebase Authentication for secure logins and using encryption methods to protect user data, **Rentistan** ensures a high level of security. Additionally, the application's integration with real-time data synchronization services helps maintain the accuracy and reliability of property listings and user interactions.

Scalability is another key feature of **Rentistan**. The app's architecture is designed to handle an expanding user base and increasing data volume efficiently, ensuring that performance remains consistent as the platform grows. The use of React Native for cross-platform development and Firebase for backend services allows **Rentistan** to offer a robust, scalable, and high-performing rental management solution.

In summary, **Rentistan** represents a significant advancement in property rental technology, combining ease of use, security, and scalability into a cohesive platform. Its design and technological choices reflect a commitment to delivering a superior rental experience for users across various devices and platforms.

.

1.2. Project Background

The **Rentistan** project was initiated in response to the growing complexities and inefficiencies associated with traditional property rental processes. The conventional methods often involve extensive paperwork, lack of transparency, and difficulties in managing communications between tenants and property managers. **Rentistan** was conceived as a solution to these challenges, aiming to modernize and streamline the rental experience through the integration of advanced technology.

The real estate market has increasingly shifted towards digital solutions, reflecting a broader trend of leveraging technology to enhance user experiences and operational efficiency. Recognizing this trend, the development team set out to create an application that not only addresses the pain points of existing rental processes but also introduces innovative features to improve the overall rental experience.

The project team, consisting of **Raja Abdul Basit, Muhammad Ehsan Ul Haq, and Saif Ullah Khan,** along with their supervisor **Mr. Fahim Abid**, embarked on this initiative with a vision to deliver a platform that caters to the needs of both tenants and property managers. **Rentistan** was designed with a focus on user experience, security, and scalability, ensuring that it meets the demands of a dynamic rental market.

Key motivations for the project include:

- **Simplification of Rental Processes**: Traditional rental processes often involve multiple steps and interactions, which can be cumbersome and time-consuming. **Rentistan** simplifies these processes by offering a centralized platform where users can manage all aspects of their rental experience.
- **Enhanced Transparency**: By providing real-time updates and comprehensive information about properties, **Rentistan** aims to improve transparency and help users make informed decisions.
- **Improved Security**: Addressing security concerns related to personal data and transactions, **Rentistan** incorporates robust measures to protect user information and ensure safe interactions.
- **Scalability**: As the rental market evolves, **Rentistan** is designed to scale with it, accommodating an increasing number of users and properties without compromising performance.

The development of **Rentistan** involved careful planning and execution, including the selection of appropriate technologies and the implementation of a user-friendly interface. The use of React Native for frontend development and Firebase for backend services ensures a seamless and efficient user experience across different devices and platforms.

Overall, **Rentistan** represents a significant advancement in property rental technology, combining modern design principles with cutting-edge technology to deliver a comprehensive and effective solution for the rental market.

1.3. Methodology and Software Lifecycle for This Project

The project will follow an Agile software development methodology. Agile is known for its iterative and incremental approach, which is particularly beneficial for projects involving complex user interactions and ongoing enhancements. This methodology involves breaking the project into smaller, manageable units called sprints, allowing for continuous improvement and adaptation to changing requirements. Regular feedback loops and collaboration with stakeholders ensure that the project stays aligned with its goals and can quickly adapt to any new insights or changes in direction.

1.4. Rationale behind Selected Methodology

The Agile methodology has been chosen for the **Rentistan** project due to its inherent flexibility, iterative nature, and focus on collaboration. This approach aligns with the project's goals of adapting to user feedback and evolving requirements. Agile is particularly beneficial for projects where requirements are likely to change or become more refined as development progresses, which is typical in the dynamic landscape of property rental applications. By adopting Agile, the development team can deliver incremental updates and improvements, ensuring that each iteration incorporates user feedback and adjusts to emerging needs. This iterative process facilitates continuous engagement with stakeholders, allowing for regular reassessment of priorities and ensuring that the final product meets user expectations and business objectives. Agile's emphasis on collaboration and adaptability supports a responsive development environment, making it well-suited for **Rentistan**'s development and ensuring a product that evolves in alignment with user and market demands.

1.5. Project objectives

The primary objectives of the **Rentistan** project are to develop a comprehensive property rental application that enhances the rental experience for both tenants and property managers. Key objectives include:

1. **User-Centric Design**: To create an intuitive and user-friendly interface that simplifies property searches, applications, and management processes. The design aims to provide a seamless experience, making it accessible to users with varying levels of technical expertise.
2. **Robust Security**: To implement stringent security measures to protect user data and ensure secure transactions. This includes integrating secure authentication methods, data encryption, and fraud prevention mechanisms to safeguard personal and financial information.
3. **Scalability**: To design the application architecture to support future growth and accommodate a growing user base. This involves leveraging scalable technologies and optimizing performance to handle increasing traffic and data volume without compromising user experience.
4. **Real-Time Updates**: To provide real-time synchronization of property listings and user data. This ensures that users have access to the most current information regarding property availability, applications, and notifications.
5. **Sentiment Analysis Integration**: To incorporate sentiment analysis into the review system to evaluate and categorize user feedback. This feature aims to assess the quality of reviews and provide insights into user satisfaction, helping property managers and tenants make informed decisions.
6. **Efficient Property Management**: To offer property managers tools for efficient management of listings, applications, and bookings. This includes features for reviewing applications, managing reservations, and communicating with tenants.
7. **Enhanced Communication**: To facilitate smooth communication between tenants and property managers through integrated messaging and notification systems. This ensures that important updates and messages are delivered promptly.
8. **Automated and Manual Payment Verification**: Although **Rentistan** does not handle payments directly, it will provide mechanisms for users to manually verify payments and report issues, ensuring a reliable payment confirmation process.

By achieving these objectives, **Rentistan** aims to deliver a high-quality, secure, and scalable property rental platform that meets the needs of users and stakeholders while providing a seamless rental experience.

1.6. Project Scope

The primary objectives of the **Rentistan** project are to develop a comprehensive property rental application that enhances the rental experience for both tenants and property managers. Key objectives include:

1. **User-Centric Design**: To create an intuitive and user-friendly interface that simplifies property searches, applications, and management processes. The design aims to provide a seamless experience, making it accessible to users with varying levels of technical expertise.
2. **Robust Security**: To implement stringent security measures to protect user data and ensure secure transactions. This includes integrating secure authentication methods, data encryption, and fraud prevention mechanisms to safeguard personal and financial information.
3. **Scalability**: To design the application architecture to support future growth and accommodate a growing user base. This involves leveraging scalable technologies and optimizing performance to handle increasing traffic and data volume without compromising user experience.
4. **Real-Time Updates**: To provide real-time synchronization of property listings and user data. This ensures that users have access to the most current information regarding property availability, applications, and notifications.
5. **Sentiment Analysis Integration**: To incorporate sentiment analysis into the review system to evaluate and categorize user feedback. This feature aims to assess the quality of reviews and provide insights into user satisfaction, helping property managers and tenants make informed decisions.
6. **Efficient Property Management**: To offer property managers tools for efficient management of listings, applications, and bookings. This includes features for reviewing applications, managing reservations, and communicating with tenants.
7. **Enhanced Communication**: To facilitate smooth communication between tenants and property managers through integrated messaging and notification systems. This ensures that important updates and messages are delivered promptly.
8. **Automated and Manual Payment Verification**: Although **Rentistan** does not handle payments directly, it will provide mechanisms for users to manually verify payments and report issues, ensuring a reliable payment confirmation process.

By achieving these objectives, **Rentistan** aims to deliver a high-quality, secure, and scalable property rental platform that meets the needs of users and stakeholders while providing a seamless rental experience.

1.7. Tools and Technologies

Rentistan utilizes various tools and technologies, classified by their use as follows:

Frontend Development:

- **React Native**: Enables the creation of a cross-platform mobile app with a single codebase, ensuring a seamless user experience across iOS and Android devices.
- **Expo**: Provides a development environment and tools to streamline the build and deployment process for React Native applications.

Backend Services:

- **Firebase**: Offers a suite of backend services including:
    - **Firestore**: A NoSQL cloud database used for real-time data management and storage.
    - **Firebase Authentication**: Manages secure user login and registration processes.

Data Analysis:

- **Sentiment Analysis Tools**: Employed to evaluate and categorize user reviews based on sentiment, helping maintain the quality of user feedback and ensure positive engagement.

Admin Panel:

- **React**: Used for developing the web-based admin panel interface.
- **Firebase**: Powers backend functionalities for the admin panel, including data management and user administration.

These technologies collectively ensure that **Rentistan** operates efficiently, securely, and provides a high-quality user experience.

1.8. Project Summary

Rentistan is an advanced property rental application designed to simplify and enhance the rental experience for both tenants and property managers. The app integrates a user-friendly interface with robust security measures and scalable technology to meet the needs of a diverse user base.

The frontend of **Rentistan** is built using React Native, allowing for a unified app experience across both iOS and Android platforms. This choice facilitates a consistent and seamless user experience, while Expo streamlines the development and deployment process.

On the backend, **Rentistan** leverages Firebase to provide real-time data synchronization, secure user authentication, and comprehensive cloud storage. Firebase's Firestore database supports efficient data management, while Firebase Authentication ensures secure login and registration. The app also integrates sentiment analysis to monitor and evaluate user reviews, helping maintain the quality of feedback and engagement.

Rentistan includes a web-based admin panel developed with React, which is powered by Firebase for backend functionalities. This panel enables administrators to manage users, properties, bookings, and communications effectively.

Overall, **Rentistan** aims to provide a reliable and intuitive platform for property rentals, focusing on ease of use, security, and scalability. The project is designed to accommodate future growth and continuously improve the user experience through modern technology and thoughtful design.

Chapter 2: Literature Review & Problem Definition

In the evolving landscape of property rentals, this chapter explores the current state of rental management solutions and identifies the gaps that exist in the market. By reviewing recent studies and technological advancements, the chapter aims to highlight the challenges faced by tenants and property managers. It sets the stage for understanding how **Rentistan** addresses these issues, offering innovative solutions to enhance the property rental experience and improve overall efficiency in property management.

.

2.1. Literature Review

Reviewing existing rental management platforms reveals several limitations related to real-time updates, user authentication, communication efficiency, user interface design, and review analysis. **Rentistan** aims to address these challenges through innovative features and technology integration.

Existing System 1: OLX:
OLX is a widely used platform for property rentals and sales. Despite its popularity, OLX faces challenges with real-time updates, as property listings can often be outdated due to the lack of real-time synchronization. This issue leads to discrepancies between available properties and their actual status, causing inconvenience for users and reducing trust in the platform.

Existing System 2: Zameen.com
Zameen.com is a prominent property portal that provides comprehensive property listings. However, it has faced issues with user authentication, as its security measures are not as robust as they could be. This leaves the platform vulnerable to unauthorized access and potential fraud. Additionally, Zameen.com's communication tools have been critiqued for not being fully optimized for real-time interactions, which can delay negotiations and information exchange between landlords and tenants.

Existing System 3: Johnson et al. (2019)
Johnson et al. developed a rental management system that included basic user authentication methods. However, the system lacked advanced security measures, such as multi-factor authentication, making it susceptible to unauthorized access and potential fraud.

Existing System 4: Davis et al. (2021)
Davis et al. introduced a rental platform with a complex user interface that users found

challenging to navigate. The design lacked intuitive elements, making tasks such as searching for properties and managing applications cumbersome.

Existing System 5: Patel et al. (2022)
Patel et al. highlighted a deficiency in analyzing user feedback in rental platforms. Many systems did not effectively utilize sentiment analysis on reviews, missing out on valuable insights into user satisfaction and areas for improvement.

Existing System 6: Williams et al. (2020)
Williams et al. developed a rental platform with integrated communication tools but faced challenges with efficiency. The system's messaging features were not optimized for real-time interactions, leading to delays in communication between landlords and tenants.

The literature indicates a clear need for advancements in rental management systems to overcome these limitations. **Rentistan** aims to address these issues through real-time updates, enhanced security features, efficient communication tools, a user-friendly interface, and comprehensive sentiment analysis of reviews. By leveraging modern technologies and focusing on user-centric design, **Rentistan** seeks to provide a more efficient, secure, and satisfying rental experience.

2.2. Analysis from Literature Review

Existing rental platforms exhibit several critical issues that **Rentistan** aims to resolve. Many platforms suffer from outdated information due to a lack of real-time updates, leading to user frustration. Security is another concern, with many systems facing challenges in ensuring secure user authentication. Communication tools are often inadequate, making interactions between landlords and tenants difficult. Additionally, complex and unintuitive user interfaces negatively impact the user experience. Sentiment analysis of user reviews reveals widespread dissatisfaction with communication features and usability, underscoring the need for improvements in these areas.

Key Findings:

- **Outdated Information**: Lack of real-time updates leads to outdated property details.
- **Security Concerns**: Insufficient secure user authentication.
- **Inefficient Communication**: Inadequate tools for landlord-tenant interactions.
- **Complex User Interfaces**: Poor usability due to complex interfaces.
- **Sentiment Analysis**: User dissatisfaction with communication and interface.

2.3. Problem Statement

The current rental market is underserved by efficient digital platforms for managing rental properties, leading to significant challenges for both landlords and tenants. Many existing platforms suffer from outdated property information, which results in discrepancies and user frustration due to the lack of real-time updates. Security vulnerabilities are also prevalent, with inadequate user authentication mechanisms exposing platforms to unauthorized access and potential fraud. Furthermore, many rental apps lack efficient communication tools, making it difficult for landlords and tenants to interact and negotiate effectively. The complexity and non-intuitive design of these platforms exacerbate the problem, creating a poor user experience and hampering effective navigation.

These issues contribute to inefficiencies in the rental process, leading to a suboptimal experience for users. Rentistan aims to address these challenges by providing a comprehensive platform that offers real-time updates, robust security measures, and a user-friendly interface. By focusing on these key areas, Rentistan seeks to fill the current gap in the market and deliver a solution that enhances the rental experience for all users.

2.4. Product Functions

Rentistan offers a range of functions designed to streamline the property rental process and enhance the user experience. Key functions include:

- **Real-Time Property Listings**: Users can browse and view up-to-date property listings with current availability, pricing, and detailed descriptions. This ensures that users have access to the most accurate information.
- **Advanced Search and Filters**: The app provides robust search capabilities with customizable filters, allowing users to find properties based on location, price range, number of bedrooms, and other criteria tailored to their needs.
- **User Authentication and Security**: Secure user authentication methods are implemented to ensure that user data and interactions are protected. This includes encrypted communication and secure login processes.
- **Interactive Maps**: Integration with interactive maps enables users to view property locations and assess nearby amenities, helping them make informed decisions about their potential rental.

- **Application and Booking Management**: The app simplifies the application and booking process with pre-filled forms and a centralized management system for property managers. Users can easily apply for properties, and landlords can review and manage applications efficiently.
- **Sentiment Analysis on Reviews**: **Rentistan** utilizes sentiment analysis to assess the quality of reviews. This feature helps users gauge the overall sentiment of reviews and make better-informed decisions about properties and landlords.
- **Notifications and Alerts**: Users receive timely notifications and alerts about new messages, application updates, and important changes. This keeps users informed and engaged throughout the rental process.
- **Manual Payment Confirmation**: Although **Rentistan** does not handle payments directly, it supports a manual payment confirmation process where users can submit proof of payment for verification.

These functions are designed to enhance the efficiency, security, and usability of the rental experience, addressing common pain points and providing a comprehensive solution for property management.

2.5. Chapter Summary

This chapter presented a detailed literature review and analyzed the current deficiencies in rental platforms that **Rentistan** aims to overcome. It identified critical issues such as outdated information, security vulnerabilities, inefficient communication, and complicated user interfaces prevalent in existing systems. The analysis underscored the necessity for real-time updates, robust security measures, effective communication tools, and an intuitive user experience. The problem statement clarified the challenges and dissatisfaction experienced by both landlords and tenants due to these shortcomings. The product functions section detailed how **Rentistan** will address these issues by offering a comprehensive, secure, and user-friendly platform designed to enhance the rental process, providing an efficient and satisfactory experience for all users.

Chapter 3: Requirement Analysis

3.1. Functional Requirements

User Authentication and Registration:

- **Tenant and Property Manager Registration:** Users can register as tenants or property managers using email and password. Registration includes entering personal details and creating a secure password.
- **Login and Logout:** Registered users can log in using their email and password. They can also log out, which securely ends their session.
- **Password Recovery:** Users can recover their passwords through a password reset link sent to their registered email if they forget their password.

User Profiles:

- **Tenant Profile:** Tenants can create and manage their profiles, including personal information, preferred locations, and rental preferences. They can also update their contact details.
- **Property Manager Profile:** Property managers can create and maintain profiles that include their contact information, property listings, and other relevant details.

Property Listings:

- **Create and Manage Listings:** Property managers can create new property listings, including details such as property type, location, price, and availability. They can also update or remove existing listings.
- **Search and Filter Properties:** Tenants can search for properties based on criteria such as location, price range, number of bedrooms, and other relevant factors. They can apply filters to refine search results.

Interactive Maps:

- **View Property Locations:** Tenants can view the location of properties on an interactive map integrated into the app. This feature provides a visual representation of property locations relative to nearby landmarks and amenities.
- **Map Filters:** Users can filter properties on the map based on criteria like price range and number of bedrooms, helping them to identify suitable options more easily.

Application and Booking:

- **Contact Property Manager:** Tenants can contact property managers through multiple channels—phone call, email, WhatsApp, or in-app message. This enables them to inquire about the property and discuss booking details directly.
- **Mutual Agreement on Booking:** Bookings are conducted through direct communication between the tenant and property manager. Once a tenant expresses interest in a property, they initiate contact via the contact button. Further discussions are handled outside the app, based on mutual agreement between the tenant and the property manager.

Notifications:

- **In-App Notifications:** Users receive notifications within the app for new messages, application updates, and other relevant events. These notifications keep users informed about their interactions and updates on property listings.

Review and Sentiment Analysis:

- **Submit Reviews:** Tenants can submit reviews for properties they have rented. Reviews include ratings and comments on their experience.
- **Sentiment Analysis:** Reviews are analyzed to determine sentiment—whether positive, negative, or neutral. This analysis helps provide insights into tenant satisfaction and property quality.

Data Security:

- **Encryption:** All user data, including personal information and communication details, is encrypted to ensure privacy and security.
- **Secure Authentication:** The app uses secure authentication methods to protect user accounts and prevent unauthorized access.

Integration and Compatibility:

- **Cross-Platform Access:** The app is designed to be compatible with both iOS and Android devices, ensuring a consistent experience across different platforms.
- **Real-Time Updates:** Property availability and other relevant data are updated in real-time, providing accurate and current information to users.

3.2. Non Functional Requirements

Non-functional requirements specify the attributes that the system must have, such as performance, security, and usability. For **Rentistan**, these requirements ensure that the app not only functions correctly but also performs well and meets user expectations.

Performance:

- **Response Time:** The app should provide quick responses to user interactions. For instance, search results should load within 2 seconds, and property details should be accessible instantly.
- **Scalability:** The system should handle an increasing number of users and property listings without degradation in performance. It must be able to scale horizontally to accommodate growing data and user loads.

Reliability:

- **Availability:** The app should be available 24/7 with minimal downtime. Scheduled maintenance should be communicated to users in advance, and any unscheduled outages should be resolved promptly.
- **Data Integrity:** The system must ensure that all user data and property information are accurate and consistent. Any data inconsistencies or errors should be identified and corrected swiftly.

Usability:

- **User Interface:** The app should have an intuitive and user-friendly interface, allowing users to navigate easily through different sections, such as property listings, profiles, and communication features.
- **Accessibility:** The application should be accessible to users with disabilities. This includes support for screen readers and other assistive technologies to ensure an inclusive experience.

Security:

- **Data Protection:** User data, including personal information and communication details, should be protected using encryption both in transit and at rest.
- **Authentication and Authorization:** Secure authentication mechanisms must be in place to prevent unauthorized access. Multi-factor authentication should be used to enhance account security.
- **Fraud Prevention:** The system should include features to detect and prevent fraudulent activities, such as suspicious behavior monitoring and fraud reporting mechanisms.

Compatibility:

- **Cross-Platform Functionality:** The app must be compatible with both iOS and Android devices, providing a consistent experience across different platforms.
- **Device and Browser Support:** The application should work efficiently on a range of devices, including smartphones and tablets, and should be compatible with the latest versions of popular web browsers if accessed via a web interface.

Availability:

- **Disaster Recovery:** The system should have a disaster recovery plan in place to ensure data backup and recovery in case of system failures or data loss.
- **Backup and Restore:** Regular backups should be performed to ensure that data can be restored quickly in the event of corruption or loss.

Maintainability:

- **Code Quality:** The application's code should be well-documented and adhere to best practices to facilitate ease of maintenance and future updates.
- **Error Handling:** The system should handle errors gracefully, providing meaningful error messages and options for users to resolve issues or seek support.

Benefits of non-functional requirements:

- **Enhanced User Experience:** Ensures that the app is intuitive, responsive, and accessible, improving overall user satisfaction and ease of use.
- **Increased Reliability:** Guarantees that the system is consistently available, performs well under load, and maintains data integrity, which builds user trust and reliability.
- **Improved Security:** Protects user data and system resources from unauthorized access and malicious activities, reducing the risk of data breaches and fraud.
- **Scalability:** Allows the app to handle increased user loads and data volumes without performance degradation, supporting future growth and expansion.
- **Efficient Performance:** Ensures quick response times and efficient processing, leading to a smooth and seamless experience for users.
- **Cross-Platform Compatibility:** Enables the app to work across different devices and platforms, reaching a wider audience and providing a consistent experience.
- **Maintainability:** Facilitates easier updates and bug fixes, reducing the cost and effort associated with ongoing maintenance and support.
- **Accessibility:** Ensures that the app is usable by individuals with disabilities, promoting inclusivity and compliance with accessibility standards.

3.3. Use Cases model

3.3.1 Use Case Diagram

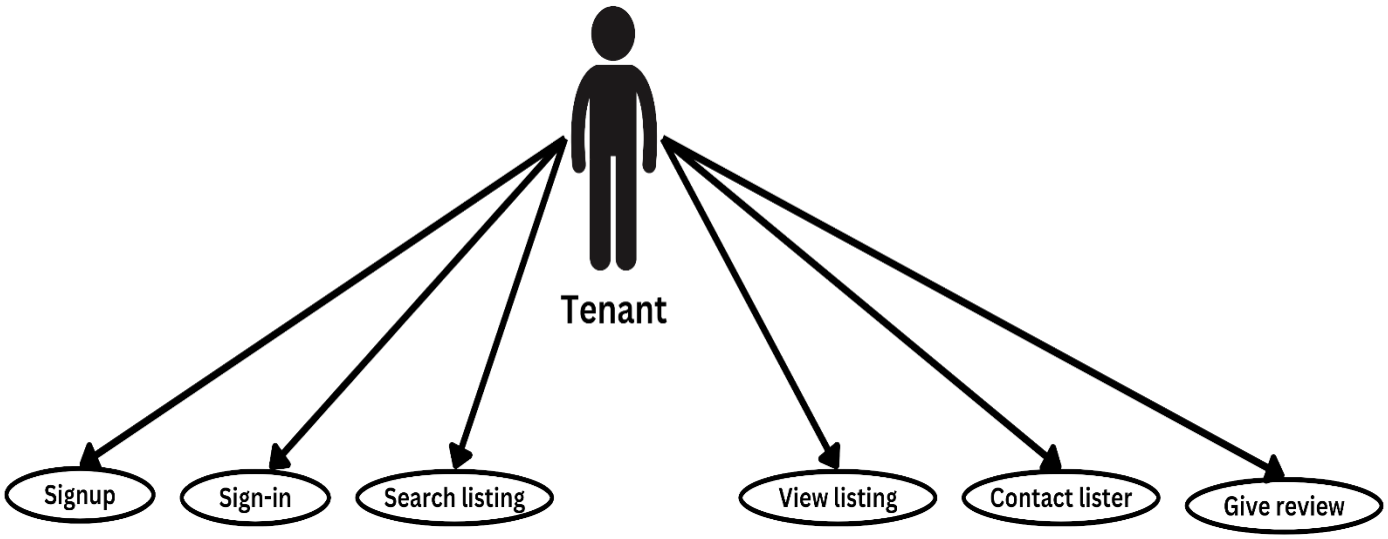

Figure 3.1: Use Case Diagram (Tenant)

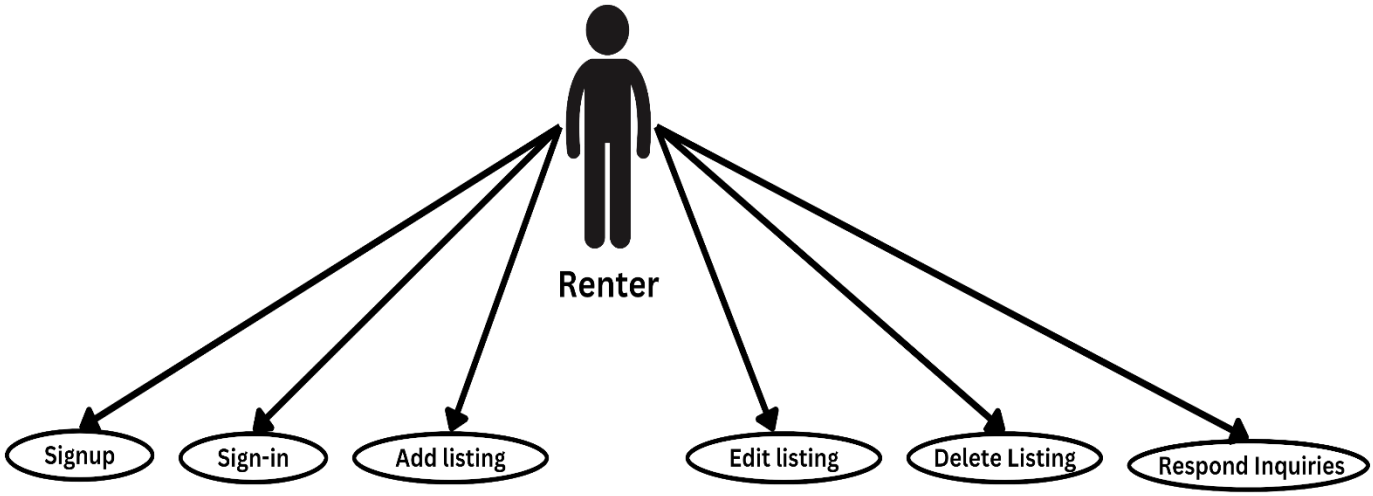

Figure 3.2: Use Case Diagram (Renter)

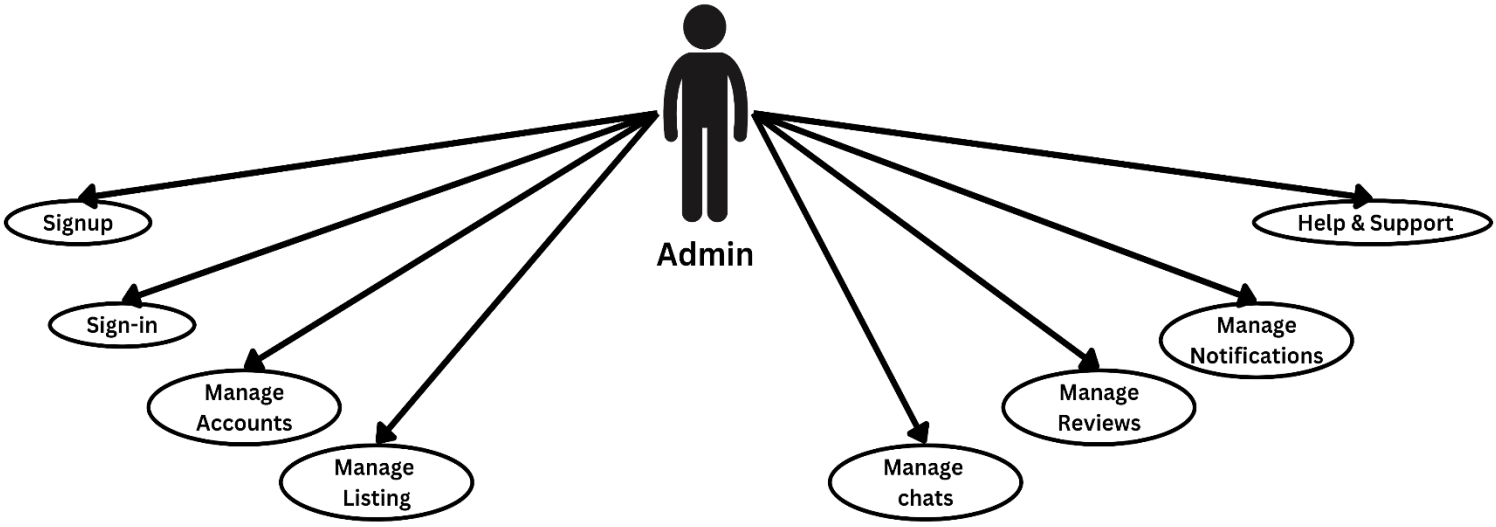

Figure 3.3: Use Case Diagram (Admin)

3.3.2 Use Case description:

Use Case Description (Tenant)

Table 3.1: Sign Up

Use Case ID:	3.1
Use Case Name:	Sign Up
Actors:	Tenant
Description:	The user signs up by providing necessary details such as email, password, name, city, and province.
Preconditions:	Tenant has not already signed up.
Post conditions:	Tenant account is created and stored in Firebase.

Table 3.2: Login

Use Case ID:	3.2
Use Case Name:	Login
Actors:	Tenant
Description:	The Tenant logs in by entering their email and password.
Preconditions:	Tenant must have a registered account.
Post conditions:	Tenant is authenticated and got access to app.

Table 3.3: View Property Listings

Use Case ID:	3.3
Use Case Name:	View Property Listings
Actors:	Tenant
Description:	The tenant browses available properties with details such as location, price, and description.
Preconditions:	Tenant is logged in.
Post conditions:	Tenant views property listings.

Table 3.4: Search and Filter Properties

Use Case ID:	3.4
Use Case Name:	Search and Filter Properties
Actors:	Tenant
Description:	The tenant uses search functionality and filters to find properties based on location, price, and number of bedrooms.
Preconditions:	Tenant is logged in and searching for properties.
Post conditions:	Tenant finds relevant properties.

Table 3.5: Contact Renter

Use Case ID:	3.5
Use Case Name:	Contact Renter
Actors:	Tenant
Description:	The tenant contacts a property renter via call, mail, WhatsApp, or message.
Preconditions:	Tenant has selected a property listing.
Post conditions:	Tenant initiates communication with the renter.

Table 3.6: Submit Reviews

Use Case ID:	3.6
Use Case Name:	Submit Reviews
Actors:	Tenant
Description:	The tenant posts reviews and rates properties based on rental experience.
Preconditions:	Tenant has completed a rental transaction.
Post conditions:	Review is submitted and visible on the platform.

Table 3.7: View Application Status

Use Case ID:	3.7
Use Case Name:	View Application Status
Actors:	User
Description:	The tenant checks the status of their rental applications and any updates from the property renter.
Preconditions:	Tenant has submitted an application.
Post conditions:	Tenant views the status and any updates.

Table 3.8: Update Profile

Use Case ID:	3.8
Use Case Name:	Update Profile
Actors:	Tenant
Description:	The tenant edits personal information, including contact details and preferences.
Preconditions:	Tenant is logged in.
Post conditions:	Tenant's profile is updated

Table 3.9: Logout

Use Case ID:	3.9
Use Case Name:	Log out
Actors:	Tenant
Description:	The tenant Log outs.
Preconditions:	Tenant is logged in.
Post conditions:	Tenant's profile is logged out

Use Case Description (Renter)

Table 3.10: Signup

Use Case ID:	3.10
Use Case Name:	Renter Sign Up
Actors:	Renter
Description:	The user signs up by providing necessary details such as email, password, name, city, and province.
Preconditions:	Renter has not already signed up.
Post conditions:	Renter account is created and stored in Firebase.

Table 3.11: Login

Use Case ID:	3.11
Use Case Name:	Login
Actors:	Renter
Description:	The Renter logs in by entering their email and password.
Preconditions:	Renter must have a registered account.
Post conditions:	Renter is authenticated and got access to app.

Table 3.12: Manage Property Listings

Use Case ID:	3.12
Use Case Name:	Manage Property Listings
Actors:	Renter
Description:	The renter adds, updates, or removes property listings from the platform.
Preconditions:	Renter is logged in.
Post conditions:	Property listings are updated on the platform.

Table 3.13: Respond to Tenant Inquiries

Use Case ID:	3.13
Use Case Name:	Respond to Tenant Inquiries
Actors:	Renter
Description:	The renter communicates with tenants who contact them through calls, mails, WhatsApp, or messages.
Preconditions:	Tenant has initiated contact.
Post conditions:	Renter responds to tenant inquiries.

Table 3.14: Review and Approve Applications

Use Case ID:	3.14
Use Case Name:	Review and Approve Applications
Actors:	Renter
Description:	The renter reviews tenant applications and approves or rejects them.
Preconditions:	Tenant applications are submitted.
Post conditions:	Application status is updated.

Table 3.15: View Tenant Reviews

Use Case ID:	3.15
Use Case Name:	View Tenant Reviews
Actors:	Renter
Description:	The renter reads reviews left by tenants on their properties.
Preconditions:	Reviews are submitted.
Post conditions:	Renter views tenant reviews..

Table 3.16: View Application Status

Use Case ID:	3.16
Use Case Name:	View Application Status
Actors:	User
Description:	The tenant checks the status of their rental applications and any updates from the property renter.
Preconditions:	Tenant has submitted an application.
Post conditions:	Tenant views the status and any updates.

Table 3.17: Update Profile

Use Case ID:	3.17
Use Case Name:	Update Profile
Actors:	Renter
Description:	The Renter edits personal information, including contact details and preferences.
Preconditions:	Renter is logged in.
Post conditions:	Renter 's profile is updated

Table 3.18: Logout

Use Case ID:	3.18
Use Case Name:	Log out
Actors:	Renter
Description:	The Renter Log outs.
Preconditions:	Renter is logged in.
Post conditions:	Renter 's profile is logged out

Use Case Description (Admin)

Table 31.9: Sign up

Use Case ID:	3.19
Use Case Name:	Sign Up
Actors:	Admin
Description:	The admin registers an account with credentials and verification details
Preconditions:	Admin has not already signed up.
Post conditions:	Admin account is created and stored in Firebase.

Table 3.20: Login

Use Case ID:	3.20
Use Case Name:	Login
Actors:	Admin
Description:	The admin logs in using admin credentials.
Preconditions:	Admin has a registered account.
Post conditions:	Admin is authenticated and granted access to the admin panel.

Table 3.21: View All Users

Use Case ID:	3.21
Use Case Name:	View All Users
Actors:	Admin
Description:	The admin views a list of all registered users.
Preconditions:	Admin is logged in.
Post conditions:	List of all users is retrieved from Firebase and displayed.

Table 3.22: Manage Users

Use Case ID:	3.22
Use Case Name:	Manage Users
Actors:	Admin
Description:	The admin approves, blocks, or deletes tenant and renter accounts.
Preconditions:	Admin is logged in.
Post conditions:	User accounts are managed as per admin actions.

Table 3.23: Manage Property Listings

Use Case ID:	3.23
Use Case Name:	Manage Property Listings
Actors:	Admin
Description:	The admin oversees property listings, including additions and removals by renters.
Preconditions:	Property listings exist.
Post conditions:	Listings are managed and updated as required.

Table 3.24: Monitor and Resolve Issues

Use Case ID:	3.24
Use Case Name:	Monitor and Resolve Issues
Actors:	Admin
Description:	The admin addresses user complaints, disputes, or issues reported by tenants or renters.
Preconditions:	Issues are reported.
Post conditions:	Issues are resolved or escalated as necessary.

Table 3.25: View and Analyze Reviews

Use Case ID:	3.25
Use Case Name:	View and Analyze Reviews
Actors:	Admin
Description:	The admin monitors and analyzes reviews submitted by tenants to ensure quality and address any issues.
Preconditions:	Reviews are submitted.
Post conditions:	Reviews are analyzed, and actions are taken as necessary.

Table 3.26: Manage Chats

Use Case ID:	3.26
Use Case Name:	Manage Chats
Actors:	Admin
Description:	The admin oversees and manages the chat system between tenants and renters to ensure compliance and resolve disputes..
Preconditions:	Chats are active on the platform.
Post conditions:	Chats are monitored, and any necessary administrative actions are taken.

Table 3.27: Logout

Use Case ID:	3.27
Use Case Name:	Log out
Actors:	Admin
Description:	The Admin Log outs.
Preconditions:	Admin is logged in.
Post conditions:	Admin 's profile is logged out

3.4. Chapter Summary

This chapter provided an in-depth analysis of the requirements necessary for the development and implementation of **Rentistan**. It detailed both the functional and non-functional requirements of the system, ensuring that all aspects of the app are addressed comprehensively. The functional requirements covered the core functionalities for tenants, renters, and admins, including account management, property listings, communication methods, and user reviews. Non-functional requirements focused on aspects such as performance, scalability, and security, emphasizing the importance of a robust and reliable platform. Additionally, the use case model illustrated the sequence of actions for tenants, renters, and admins, providing a clear and structured overview of the interactions within the system. This thorough analysis ensures that **Rentistan** will meet the needs and expectations of its users, delivering a seamless and efficient rental experience.

Chapter 4: Design and Architecture

4.1. Proposed Architecture

The Rentistan home rental application follows a modern architecture designed to ensure scalability, maintainability, and efficiency. The proposed architecture comprises the following components:

- **Frontend**: Implemented using React Native, providing a cross-platform user interface that works seamlessly on both iOS and Android devices.
- **Backend**: Powered by Firebase, which offers real-time database services, authentication, and cloud functions to handle backend logic.
- **Database**: Firebase Firestore is used for real-time data storage and synchronization.
- **Authentication**: Firebase Authentication manages user sign-up, login, and access control.
- **Communication Tools:** In-app messaging system for direct communication between tenants and property managers. Options for calling, emailing, WhatsApp, or messaging tenants through contact buttons on listings.
- **Storage**: Firebase Storage is used for storing images and other media files uploaded by users.
- **Notification System**: Integrates Expo Notifications for real-time alerts and updates.
- **Sentiment Analysis**: Incorporates sentiment analysis for user reviews to assess the quality of feedback.
- **Mapping and Geolocation**: Integration with interactive maps (e.g., Google Maps) for property location visualization. Geolocation services to provide users with nearby property suggestions.

4.2. Deliverables and Development Requirements

The project aims to deliver the following key outcomes:

Deliverables:

1. **Mobile Application (Rentistan):**

    - A fully functional mobile application developed using React Native, available for both iOS and Android platforms.
    - Features include user registration, property listings, search and filter options, booking management, in-app messaging, and sentiment analysis for reviews.

2. **Web Admin Panel:**

    - A web-based admin panel developed using React and Firebase.
    - Features include user management, property management, booking management, chat management, notifications management, and analytics & reporting.

3. **Backend Services:**

    - A robust backend infrastructure powered by Firebase.
    - Real-time database (Firestore), authentication, cloud storage, and analytics.

4. **Source Code:**

    - Full source code for the mobile application, web admin panel, and backend services.
    - Version control managed through a Git repository.

5. **Deployment Scripts:**

    - Scripts and instructions for deploying the application on cloud platforms and app stores.

Development Requirements:

Hardware Requirements:

- Development machines with sufficient processing power, memory, and storage to run development tools and emulators/simulators.
- Mobile devices (iOS and Android) for testing purposes.

Software Requirements:

- **Development IDEs:** Visual Studio Code for coding and testing.
- **Node.js** for server-side development.
- **Firebase CLI** for managing Firebase services.
- **Expo CLI** for React Native development and testing.

Programming Languages:

- **TypeScript:** Optionally used for type safety in JavaScript code.
- **Python:** For any backend logic or cloud functions that require more complex processing (optional).

Libraries, Technologies, and Frameworks:

Frontend:

- **React Native:** For developing cross-platform mobile applications.
- **React:** For the web-based admin panel.
- **Expo:** For managing React Native projects and dependencies.

Backend:

- **Firebase:** For real-time database (Firestore), authentication, cloud storage, and analytics.
- **Node.js**: For server-side logic and API development.
- **Firebase Cloud Functions:** For serverless backend logic.

Additional Libraries:

- **Firebase SDK:** For integration with Firebase services such as Firestore, Authentication, Cloud Functions, and Cloud Messaging.

- **Redux:** For state management in React Native applications (optional).
- **Axios:** For making API requests.

By adhering to these deliverables and development requirements, **Rentistan** aims to deliver a high-quality, secure, and scalable property rental application that meets the needs of both tenants and property managers.

4.4. Assumptions and Dependencies

Assumptions:

1. **User Familiarity:**
    - Users are familiar with basic smartphone operations and internet browsing, which is essential for interacting with the mobile application and web admin panel.
2. **Stable Internet Connection:**
    - A stable internet connection is assumed for the proper functioning of real-time updates and communication features within the application.
3. **Device Compatibility:**
    - The application assumes that users will access it on compatible devices (iOS and Android smartphones) that meet the minimum system requirements for running React Native applications.
4. **User Data:**
    - It is assumed that user data provided during registration is accurate and up-to-date, which is crucial for effective communication and property management.

Dependencies:

1. **Firebase Services:**
    - The application depends on Firebase for real-time database management (Firestore), authentication, cloud storage, and analytics. Any changes or disruptions in Firebase services could impact application functionality.
2. **React Native and Expo:**
    - The development and functionality of the mobile application are dependent on React Native and Expo. Any updates or changes in these frameworks may require adjustments in the application code.
3. **Third-Party Libraries:**
    - The application uses additional libraries such as Axios and Redux. Dependencies on these libraries mean that their updates or changes could affect the application’s performance and functionality.

4. **Development Tools:**
    - The project relies on tools like Visual Studio Code, Node.js, and Firebase CLI. Availability and compatibility of these tools are crucial for development and deployment processes.

5. **Device and OS Compatibility:**
    - The application must be compatible with the latest versions of iOS and Android operating systems. Changes in device or OS requirements may affect the app's performance and compatibility.

By identifying these assumptions and dependencies, the development team can better prepare for potential challenges and ensure a smoother project execution.

4.5. UML Diagrams

UML is known as unified model language and used to made static structured diagrams which will show how system will work. There are many UML diagrams but some of them are:

SEQUENCE Diagram:

A sequence diagram is a valuable tool in software development, specifically within the field of Unified Modeling Language (UML). It presents a visual representation of the interactions and chronological order of messages exchanged between objects or components in a system. By illustrating the dynamic behavior of the system, sequence diagrams capture the flow of control and sequence of events during the execution of a particular scenario or use case.

Purpose:

- Detail the flow of operations in a specific scenario.
- Show the sequence of events and interactions between system components.
- Highlight the dynamic behavior of the system.

User Registration:

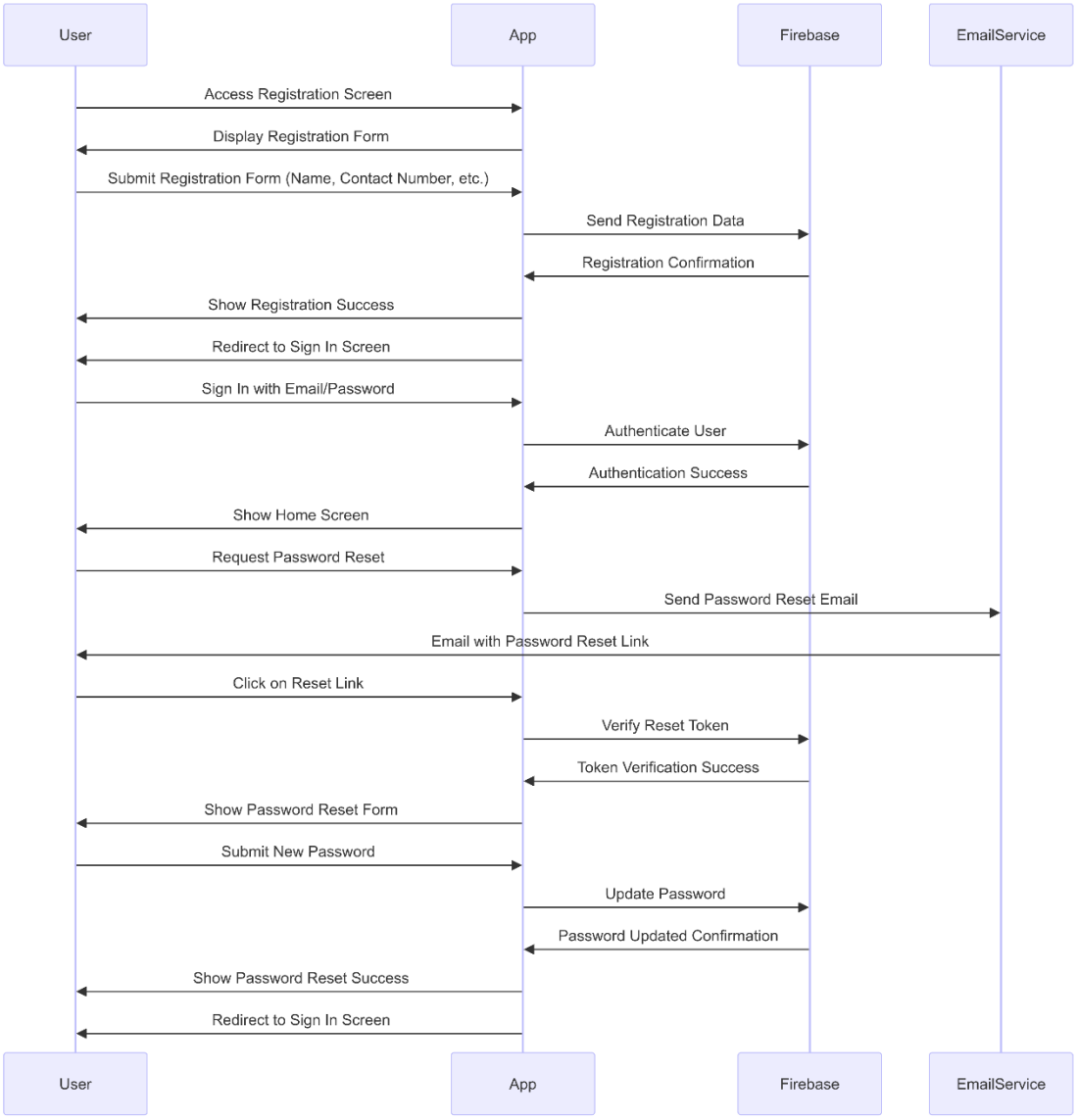

Figure 4.1: Sequence diagram Diagram

Property Listing Creation:

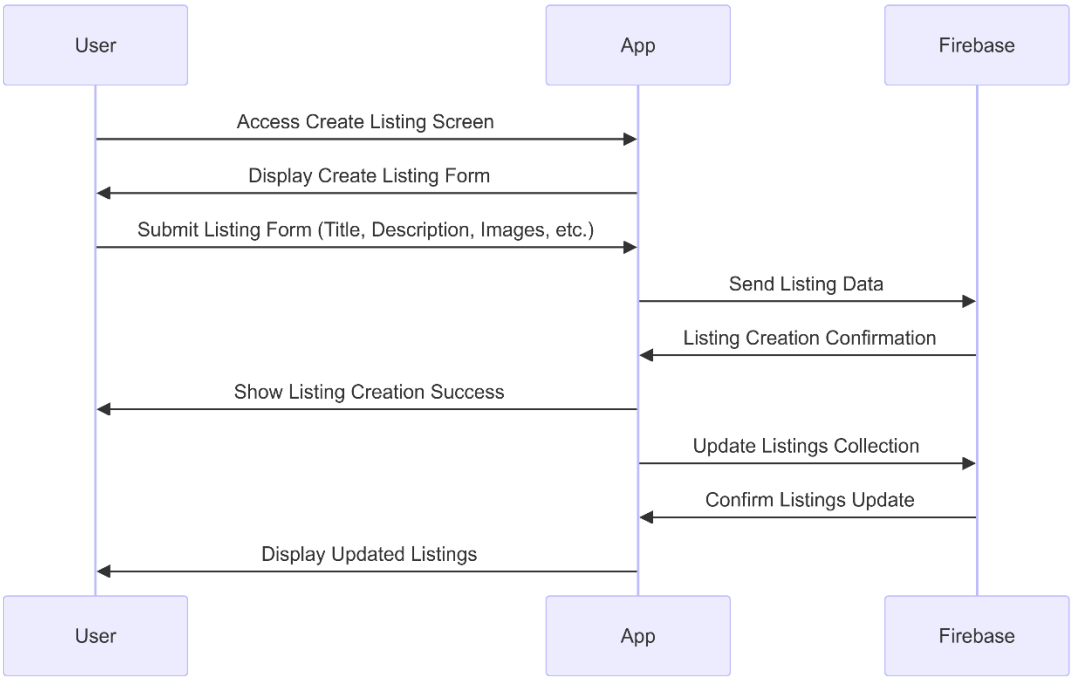

Figure 4.2: Sequence diagram Diagram

Search Listings:

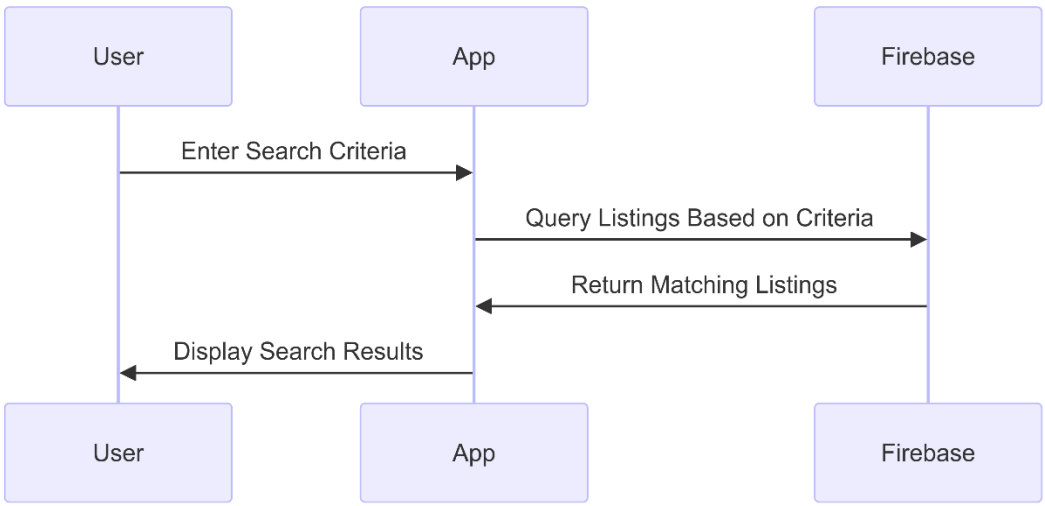

Figure 4.3: Sequence diagram Diagram

Combined Filter and Sort Listings:

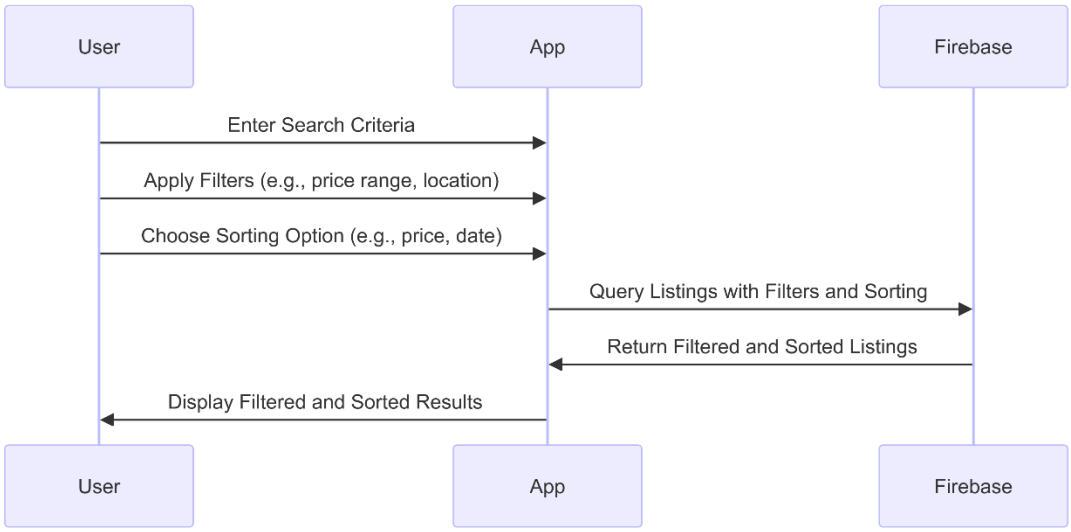

Figure 4.4: Sequence diagram Diagram

Booking Request Sequence Diagram:

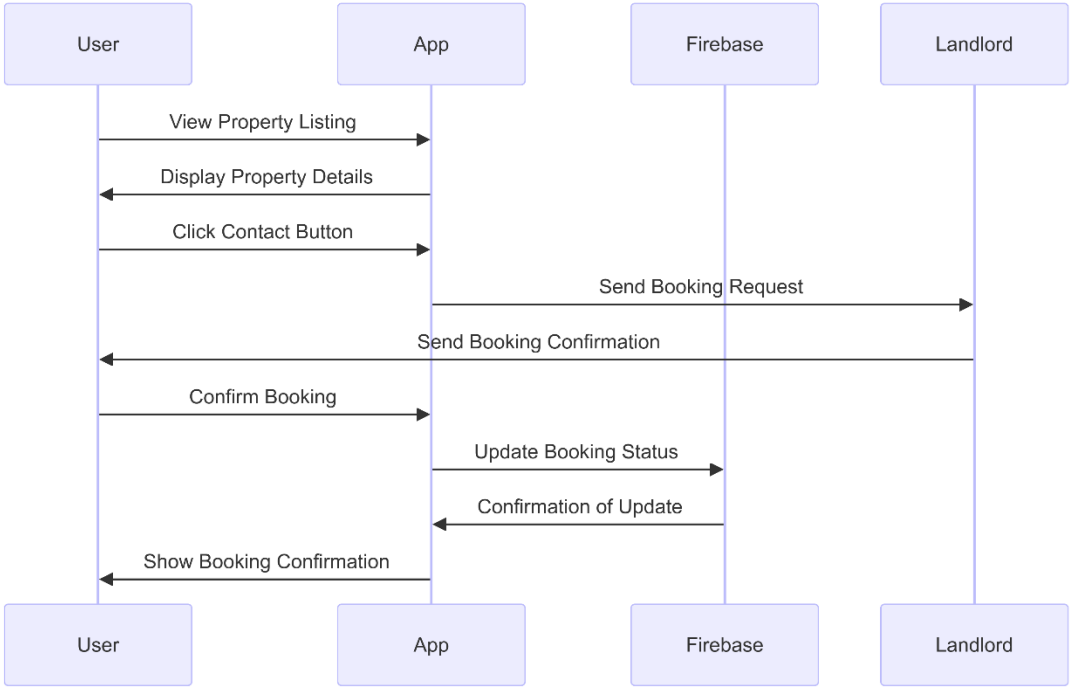

Figure 4.5: Sequence diagram Diagram

Sending a Message Sequence Diagram:

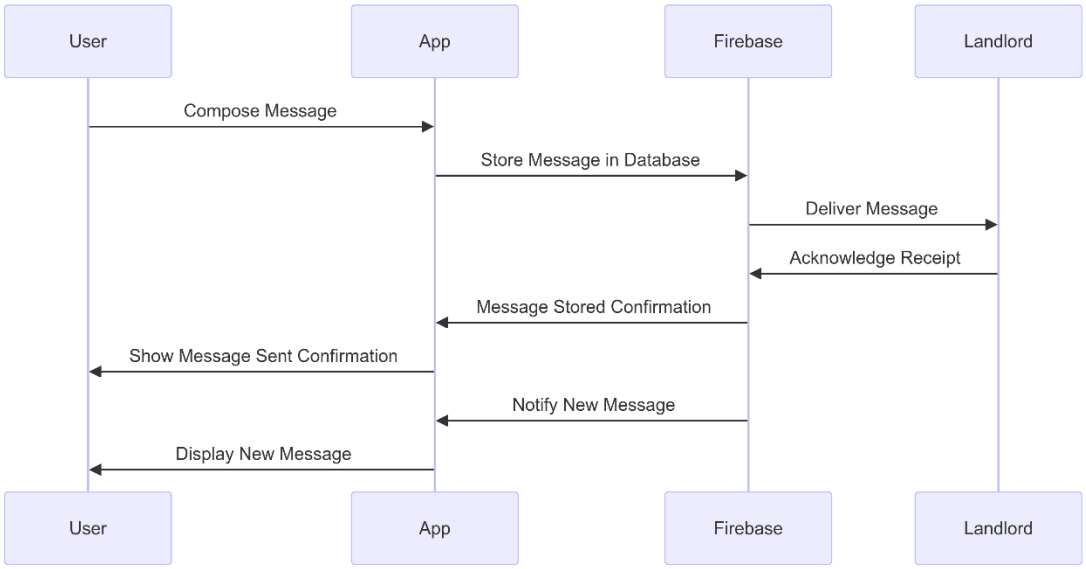

Figure 4.6: Sequence diagram Diagram

Sequence Diagram for Managing Listings:

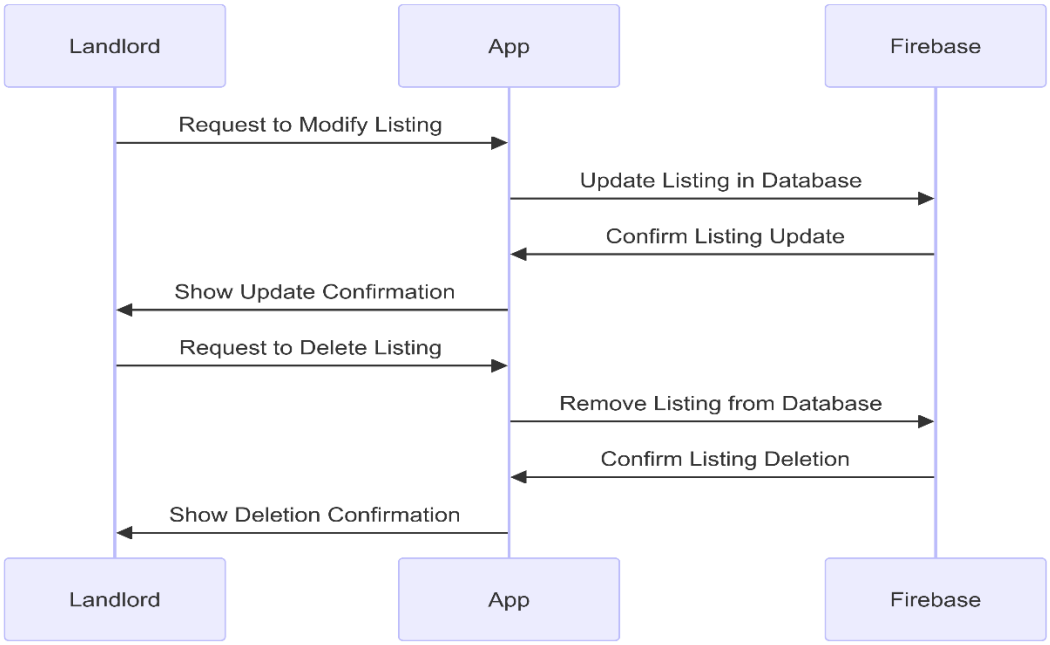

Figure 4.7: Sequence diagram Diagram

Data flow Diagram

DFD (Data Flow Diagram) is a diagram that represents the flow of data within a system. It shows how data moves from one process to another, where it's stored, and how it's processed. DFDs are often used in systems analysis and design to model the flow of information within a system or between systems. They are useful for understanding the overall system architecture and identifying data inputs, outputs, and processes.

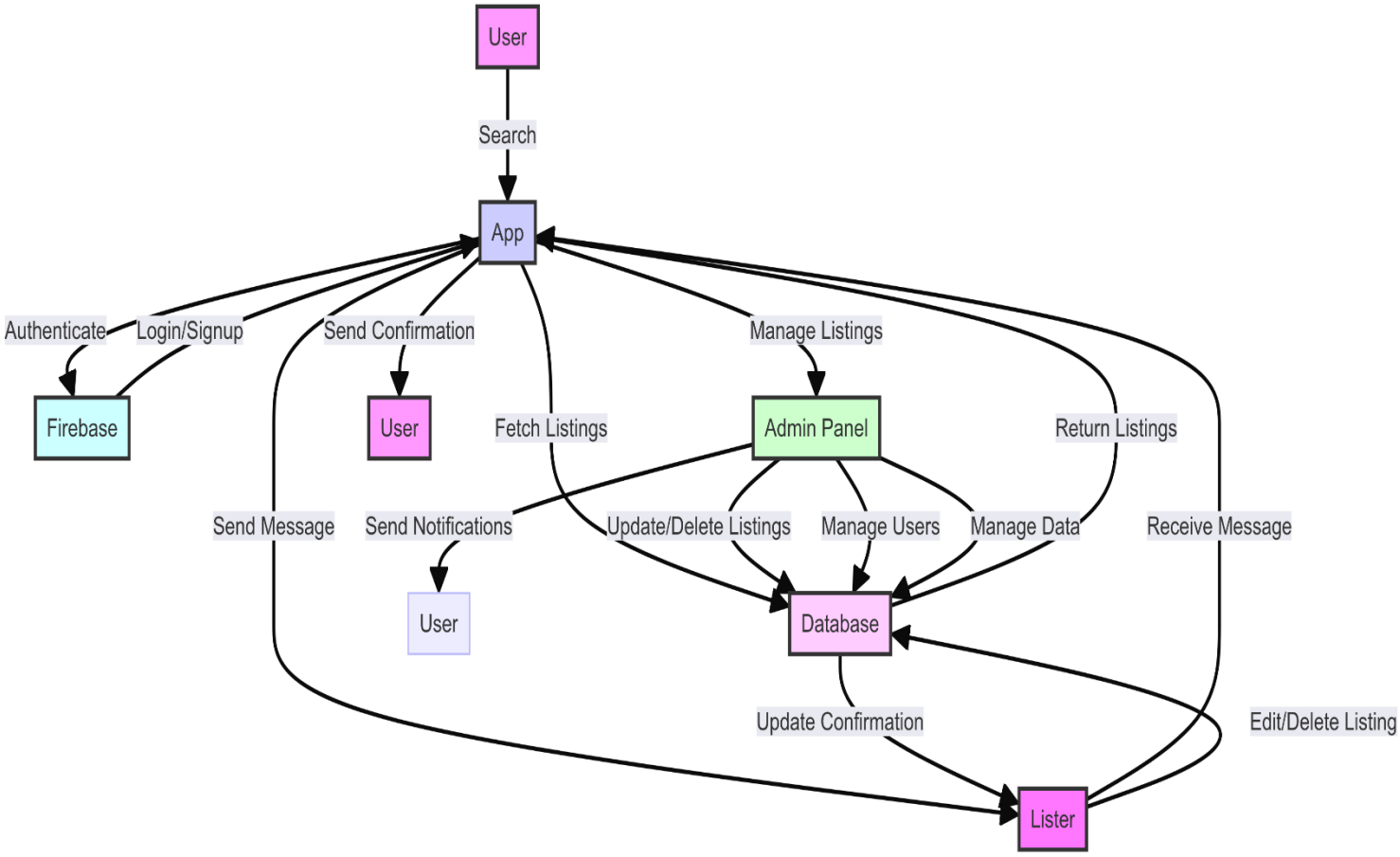

Figure 4.8: data flow Diagram

UML Block Diagram:

A block diagram provides a high-level overview of a system's architecture, showing the main components and their interactions. For the Rentistan app, the block diagram will illustrate the major system components, including the frontend, backend, database, and external services.

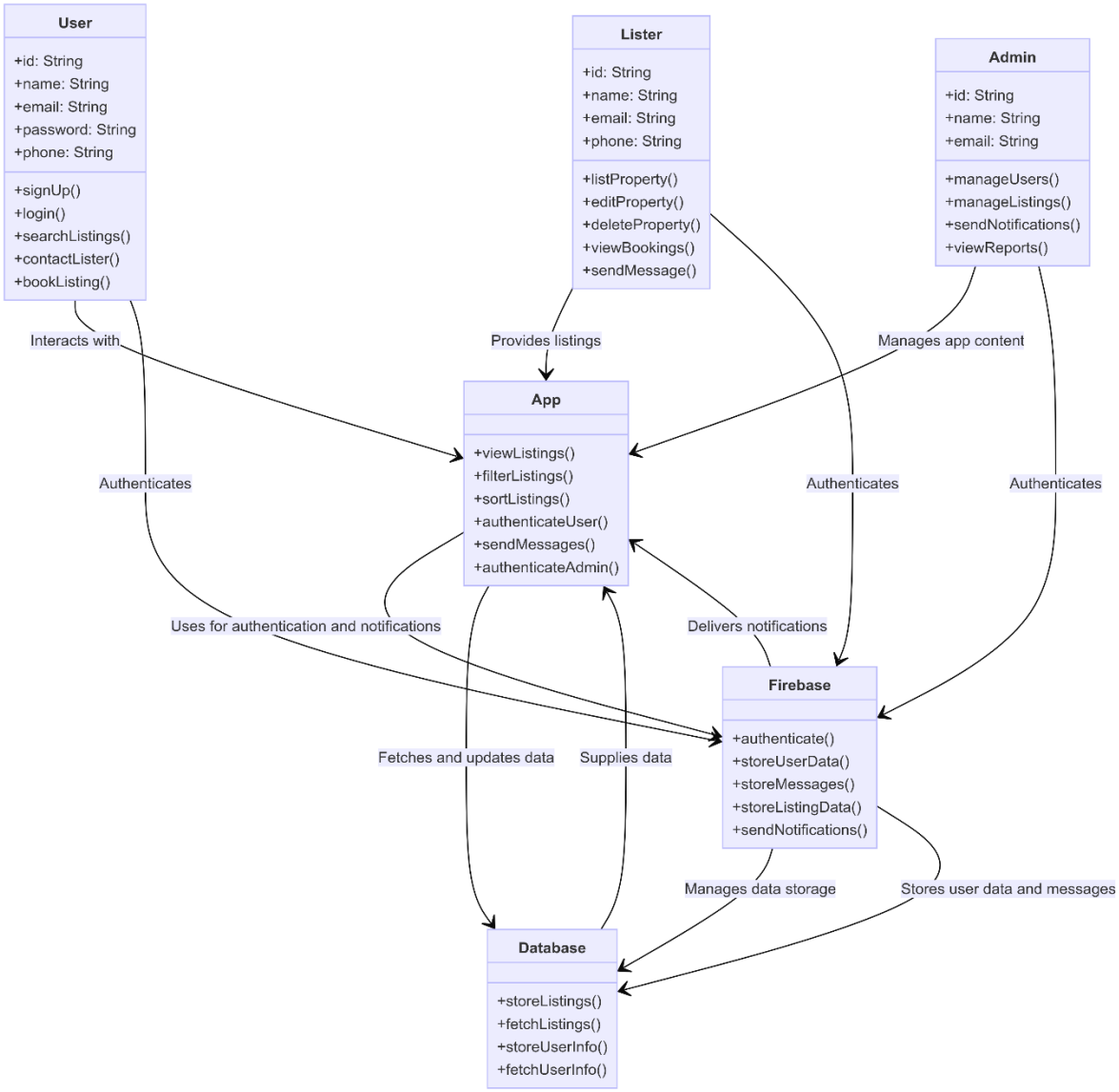

Figure 4.9: Block Diagram

Chapter Summary:

This chapter detailed the proposed architecture, outlining the various components and their interactions within **Rentistan**. The architecture includes the user-facing mobile application, the web-based admin panel, and the backend services powered by Firebase. The deliverables and development requirements were specified, including the necessary hardware, software, and programming languages. We also illustrated the data flow and sequence diagrams, showcasing the user interactions and processes within the app, such as registration, listing management, and messaging. This comprehensive design framework aims to ensure a high-quality, secure, and scalable rental application that addresses the needs of both tenants and listers.

Chapter 5: Project Implementation

This chapter details the step-by-step process of implementing the **Rentistan** project. It includes the development environment setup, coding standards, and the tools and technologies used. We will also cover the implementation of key features and functionalities, testing methodologies, and the deployment process.

5.1. Programming Language

Rentistan utilizes several programming languages to ensure robust functionality and seamless performance across platforms:

- **JavaScript**: The primary language used for both frontend and backend development.
- **TypeScript**: Used optionally for type safety in JavaScript code.
- **Python**: Utilized for backend logic and serverless functions when more complex processing is required.

5.2. Framework

Rentistan is built using a variety of frameworks to leverage their strengths for different components of the application:

- **React Native**: For developing the cross-platform mobile application, enabling a consistent user experience on both iOS and Android devices.
- **React**: Used for the web-based admin panel, providing a responsive and interactive user interface.
- **Expo**: A set of tools and services built around React Native, used for managing the development workflow, including building, deploying, and testing the mobile application.
- **Node.js**: For server-side logic and API development, allowing efficient handling of asynchronous operations.
- **Firebase**: Powers the backend services including real-time database (Firestore), authentication, cloud storage, and analytics.

5.3. System Requirement

To develop and deploy **Rentistan**, certain hardware and software requirements must be met:

Hardware Requirements:

- Development machines with at least 4GB RAM and an Intel i5/i7 processor.
- Mobile devices (iOS and Android) for testing purposes.

Software Requirements:

- **Operating System**: Windows 10 or macOS for development.
- **Development IDEs**: Visual Studio Code for coding and testing.
- **Node.js**: For server-side development and managing dependencies.
- **Firebase CLI**: For managing Firebase services.
- **Expo CLI**: For React Native development and testing.

5.4. Chapter Summary

This chapter provided an overview of the programming languages, frameworks, and system requirements essential for the development and deployment of **Rentistan**. The project leverages a combination of JavaScript, TypeScript, and Python, alongside powerful frameworks like React Native, React, Expo, and Firebase. The system requirements outlined ensure that both the development and testing environments are adequately equipped to handle the demands of the project, facilitating a smooth and efficient development process.

Chapter 6: Software Testing & Maintenance

Effective software testing and maintenance are crucial to ensure the reliability, security, and performance of **Rentistan**. This chapter discusses the various testing methodologies employed during the development phase and the strategies for ongoing maintenance to keep the application running smoothly and efficiently.

6.1 Deriving Test

Deriving tests for Rentistan involves identifying critical functionalities that need to be validated to ensure the application operates as expected. These tests will be derived based on user stories, functional requirements, and non-functional requirements. Each test will focus on validating the system's capabilities, such as user authentication, property listings, search functionalities, booking management, and messaging.

Functional Tests:

Ensure the system performs its intended functions:

- **User Authentication**:
    - **Sign Up**: Verify new user registration with email, password, name, and contact number.
    - **Login:** Verify registered user login.
    - **Password Reset:** Verify password reset via email.
    - **Google Sign-In:** Verify Google account sign-in.
- **Property Listings Management**:
    - **Add Listing:** Verify lister can add new listings.
    - **Edit Listing:** Verify lister can edit existing listings.
    - **Delete Listing:** Verify lister can delete listings.
- **Communication**:
    - **Sending Messages:** Verify users can message listers.
    - **Receiving Messages:** Verify listers can receive messages.

- **Booking Management**:
    - **Quotation Requests**: Verify users can send requests.
    - **Booking Confirmation:** Verify listers can confirm bookings.
- **Search and Filter Listings**:
    - Search Listings: Verify users can search properties.
    - Filter Listings: Verify users can apply filters.

Non-Functional Tests:

Validate performance attributes:

- **Security**: Ensure secure user authentication and data handling.
- **Usability**: Verify user-friendly navigation.
- **Reliability**: Ensure consistent performance.
- **Compatibility**: Verify seamless operation across iOS, Android, and web.

These tests ensure **Rentistan** delivers a secure, reliable, and user-friendly property rental application.

.

6.2 Test Environment

The test environment is a setup that mirrors the production environment to ensure that testing results are accurate and relevant. The environment includes the necessary hardware, software, and configurations required to execute test cases effectively.

Hardware:

- Development machine with at least 4GB RAM, Intel i5/i7 processor.
- Android and iOS devices or emulators with varying specifications to test the mobile application’s performance across different devices.

Software:

- **Operating System**: Windows 10, macOS.
- **Development IDEs**:
  - Visual Studio Code: For coding and testing.
  - Android Studio: For testing the Android application.
  - Xcode: For testing the iOS application.
- **Firebase**: For testing backend services and database interactions.
- **Expo**: For testing React Native applications.

Network Configuration:

- A stable internet connection for accessing Firebase and cloud-based services.
- Network configurations to test real-time alerts, messaging, and data synchronization.

By replicating the production environment, the test environment ensures that all components of **Rentistan** function correctly under real-world conditions, providing accurate and reliable test results.

6.3 Testing identification

Testing identification involves pinpointing critical areas of the **Rentistan** app for thorough testing:

1. **User Authentication**: Verify sign-up, login, password reset, and Google sign-in functionalities.
2. **Property Listings Management**: Test creation, modification, deletion, and search/filter functionalities.
3. **Messaging System**: Ensure messages can be sent, received, and stored correctly.
4. **Booking Process**: Validate the booking request, confirmation, and status updates.
5. **Admin Panel**: Test functionalities for user and property management, chat management, and notifications.
6. **Sentiment Analysis**: Verify the accuracy of sentiment analysis on reviews.
7. **Performance and Security**: Conduct tests for performance, security, and user experience.

6.3.1 Functional Tests

- **User Authentication**: Sign-up, login, and password reset.
- **Property Listings**: Create, modify, delete, and search listings.
- **Messaging**: Send, receive, and store messages.
- **Booking**: Request, confirm, and update bookings.
- **Admin Panel**: Manage users, properties, and notifications.

6.3.2 Non-Functional Tests

- **Performance**: Ensure efficient handling of real-time updates.
- **Security**: Test for secure authentication and data protection.
- **Usability**: Verify user-friendly interface and experience.
- **Reliability**: Ensure consistent operation under various conditions.
- **Compatibility**: Test across different devices and platforms.

By focusing on these areas, **Rentistan** can be ensured to provide a robust, secure, and user-friendly experience.

6.4 Test Procedure

The test procedure involves executing a series of tests to verify that the system meets its requirements and functions correctly. It includes the following steps:

Test Planning

- Define test objectives, criteria, and scenarios based on functional and non-functional requirements.
- Prepare test cases and scripts for each scenario.

Test Execution

- **Functional Testing**:
    - Verify user registration, login, and authentication processes.
    - Test property listing creation, modification, and deletion.
    - Ensure that search and filter functionalities work as expected.
    - Validate messaging and communication between users and listers.
- **Non-Functional Testing**:
    - Assess performance metrics like response time and scalability.
    - Test security measures including data encryption and secure authentication.
    - Check usability to ensure a user-friendly experience.
    - Verify system reliability under various conditions.

Issue Reporting

- Document any issues or defects found during testing.
- Assign severity levels and track the resolution process.

Test Review

- Evaluate test results to ensure all criteria are met.

- Review test coverage to confirm that all functionalities and requirements have been tested.

Regression Testing

- Re-test the system after fixes or changes to ensure that new code does not adversely affect existing functionalities.

Final Validation

- Perform a final round of testing to confirm that the system is ready for production deployment.

By following these steps, the test procedure ensures that **Rentistan** is thoroughly tested and validated, providing a robust and reliable platform for users.

6.5 Testing techniques

Several testing techniques will be employed to ensure comprehensive coverage, including:

Functional Testing

- **Unit Testing**: Verify individual components.
- **Integration Testing**: Check interactions between components.
- **System Testing**: Validate the complete system.

Non-Functional Testing

- **Performance Testing**: Assess speed and scalability.
    - **Load Testing**: Measure performance under load.
    - **Stress Testing**: Test limits under extreme conditions.
- **Security Testing**: Identify vulnerabilities.
    - **Penetration Testing**: Simulate attacks.
    - **Vulnerability Scanning**: Automated issue detection.
- **Usability Testing**: Ensure user-friendliness.
- **Compatibility Testing**: Verify across devices and browsers.
- **Reliability Testing**: Test stability and recovery.

These techniques will ensure **Rentistan** meets all requirements and delivers a high-quality user experience.

6.6 Test Cases

User Registration:

- **Test Case ID**: TC001
- **Description**: Verify user registration with valid details.
- **Steps**:
    1. Navigate to the registration page.
    2. Enter valid details (name, email, password).
    3. Submit the form.
- **Expected Result**: User account is created, and a confirmation message is displayed.

User Login

- **Test Case ID**: TC002
- **Description**: Verify login with valid credentials.
- **Steps**:
    1. Navigate to the login page.
    2. Enter valid email and password.
    3. Click the login button.
- **Expected Result**: User is logged in and redirected to the home screen.

Property Listing:

- **Test Case ID**: TC003
- **Description**: Verify creation of a property listing.
- **Steps**:
    1. Navigate to the listing page.
    2. Enter property details (title, description, images).
    3. Submit the listing.
- **Expected Result**: Property is listed and visible in the app.

Contact Lister:

- **Test Case ID**: TC004
- **Description**: Verify the ability to contact a lister.
- **Steps**:
    1. View a property listing.
    2. Click the contact button.

3. Choose contact method (call, mail, message).

- **Expected Result**: Communication is initiated, and lister receives the message.

Search Functionality:

- **Test Case ID**: TC005
- **Description**: Verify search and filtering of property listings.
- **Steps**:
    1. Enter search criteria (location, price range).
    2. Apply filters.
- **Expected Result**: Search results match criteria and filters are applied correctly.

Admin Listing Management:

- **Test Case ID**: TC006
- **Description**: Verify admin's ability to manage listings.
- **Steps**:
    1. Log in as an admin.
    2. Navigate to the listings management page.
    3. Modify or delete a listing.
- **Expected Result**: Listing is updated or removed as per action.

Message Handling:

- **Test Case ID**: TC007
- **Description**: Verify sending and receiving messages.
- **Steps**:
    1. Send a message to a lister.
    2. Check if the message is received.
- **Expected Result**: Message is sent and stored correctly, and recipient receives it.

These test cases cover core functionalities of **Rentistan** to ensure the application operates as expected.

6.7 Chapter Summary

This chapter outlines the testing strategies for **Rentistan**, including the derivation of tests, test environments, procedures, techniques, and specific test cases. It emphasizes:

- **Deriving Tests**: Tests are derived to verify both functional aspects (like user authentication and property management) and non-functional attributes (such as security and usability).
- **Test Environment**: A setup mimicking the production environment is essential, encompassing hardware (development machines and testing devices), software (development tools and services), and network configurations.
- **Test Procedure**: Detailed procedures are followed to ensure each test case is executed systematically, covering various functionalities of the application.
- **Testing Techniques**: Various techniques such as black-box and white-box testing are utilized to ensure comprehensive coverage of the application's features.
- **Test Cases**: Specific scenarios, including user registration, login, property listing, and message handling, are tested to validate the application's performance and correctness.

The chapter provides a structured approach to ensuring **Rentistan** meets its requirements and functions reliably in real-world conditions.

Chapter 7: Conclusion and Future work

7.1. Discussion

This chapter discusses the key outcomes of the **Rentistan** project, reflecting on its achievements and challenges. The successful implementation of a cross-platform rental application that integrates essential features like property management, user communication, and sentiment analysis demonstrates its capability to address significant gaps in existing rental platforms. The application leverages Firebase for robust backend services and React Native for a seamless mobile experience, meeting the core requirements of both tenants and listers. Challenges such as ensuring real-time data synchronization and implementing user-friendly interfaces have been effectively managed, though they required substantial effort and iterative development

7.2. Conclusion

Rentistan successfully delivers a comprehensive and user-friendly rental management solution. By addressing critical pain points identified in the literature review, such as outdated information, security concerns, and inefficient communication, the application provides an enhanced user experience for tenants and listers. The integration of features like real-time property updates, secure user authentication, and streamlined communication supports the project's goal of creating a more efficient and satisfying rental process.

7.3. Limitations

Despite its advancements, **Rentistan** faces some limitations. Current versions of the application do not include in-app notifications via SMS or email, which could enhance user engagement. The system's scalability could be further tested to handle larger volumes of users and properties. Additionally, the application’s reliance on third-party services like Firebase introduces potential risks related to service availability and data privacy.

7.4. Future work

Future development for **Rentistan** includes several potential enhancements:

- **In-App Notifications**: Implementing SMS and email notifications to improve communication and reminders.
- **Enhanced Scalability**: Optimizing backend infrastructure to better handle increased user and property volumes.
- **Advanced Analytics**: Integrating more sophisticated analytics tools to provide deeper insights into user behavior and system performance.
- **Expanded Platform Support**: Exploring additional features and integrations to extend the app's functionality and reach.
- **User Feedback Integration**: Collecting and incorporating user feedback to refine and enhance the application's features and user experience.

By addressing these areas, **Rentistan** aims to continue evolving and improving, ensuring it remains a valuable tool in the rental market.

References