WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

PROJECT TITLE: - BROKER SERVICE MANAGEMENT SYSTEM

PREPARED BY

Name	ID
1. Melkamu Abraraw	NSR/1002/12
2. Nebiyat Hassen	NSR/1144/12
3. Gelila Belachew	NSR/0646/12
4. Zelalem Daniel	NSR/1604/12

SUBMITTED TO: Department of Software Engineering

PROJECT ADVISOR: Mr. Sime A.

MAY 10, 2024

WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

PROJECT TITLE: - BROKER SERVICE MANAGEMENT SYSTEM

SUBMITTED TO DEPARTMENT OF SOFTWARE ENGINEERING

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR

THE DEGREE OF BACHLER OF SCIENCE IN SOFTWARE ENGINEERING

PREPARED BY

Name	ID
1. Melkamu Abraraw	NSR/1002/12
2. Nebiyat Hassen	NSR/1144/12
3. Gelila Belachew	NSR/0646/12
4. Zelalem Daniel	NSR/1604/12

MAY 10, 2024

Wolkite University, Wolkite, Ethiopia

DECLARATION

This is to declare that this project work which is done under the supervision of Mr Sime and having the title of Broker Service Management System is the sole contribution of:

Melkamu Abrarw,

Nebiyat Hassen,

Gelila Belachew and

Zelalem Daniel

No part of the project work has been reproduced illegally (copy and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: _____

Group Members:

Full Name	Signature
_____	_____
_____	_____
_____	_____
_____	_____

APPROVAL FORM

This is to confirm that the project report entitled Broker Service Management System submitted to **Wolkite University, College of Computing and Informatics Department of Software Engineering** by:

Melkamu Abrarw,

Nebiyat Hassen,

Gelila Belachew and

Zelalem Daniel approved for submission.

Advisor Name	Signature	Date
Department Head Name	Signature	Date
Examiner 1 Name	Signature	Date
Examiner 2 Name	Signature	Date
Examiner 3 Name	Signature	Date

ACKNOWLEDGEMENT

First and foremost, we extend our gratitude to the Almighty God, the source of our strength and success. We are deeply thankful to our families for their indescribable support throughout this journey and unwavering support have been pivotal to our development and accomplishments. we would like to extend our gratitude to broker service companies for helping us by providing the necessary information we required.

We would like to extend our heartfelt gratitude to our advisor, Mr. Sime for his exceptional guidance and advising throughout the duration of this project. His unwavering support and expertise have played a pivotal role in shaping the project's direction and success.

LIST FIGURES AND TABLES

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS & ABBREVIATIONS

API	Application Programming Interface
BSMS	Broker Service Management System
CSS	Cascading Style Sheet
E-SIGNATURE	Electronic signature
GUI	Graphical User Interface
JS	JavaScript
NPM	Node Package Manager
OOA	Object Oriented Analysis
OOD	Object Oriented Design
OOSAD	Object Oriented System Analysis and Design
RAM	Random Access Memory
UML	Unified Modelling Language
URL	Uniform Resource Locator

ABSTRACT

The Broker Service Management System was developed to modernize the brokerage industry, connecting buyers, sellers, lessors, lessees, and brokers seamlessly. Our approach was grounded in thorough market research, including user interviews, and observation, to ensure alignment with user needs and preference. Leveraging modern development tools like iterative practices and secure communication protocols, our goal was to create a scalable, user-friendly solution. The system simplifies property viewings, payment processing, and communication through intuitive interfaces and messaging systems which increased efficiency, improved negotiation processes, and enhanced overall transactional experiences in the brokerage industry.

CHAPTER ONE

1. INTRODUCTION

People nowadays use contemporary technology to facilitate their work in their day-to-day operations. Those technologies are created or developed by a single individual or group for the purpose of business or to tackle a specific problem.

There are now several market exchanges in Ethiopia. Those markets may be tough and complicated to obtain service in a timely manner, therefore selling, buying, or renting materials, houses, and products may necessitate the use of an agent who makes the market simple and accessible. Brokers are agents who make the market simple and accessible. Brokers handle all of the selling, buying, and renting of materials and properties in Ethiopia, particularly in cities and towns. Additionally, brokers work with job seekers. Brokers form a corporation and obtain a government license in order to provide quick and accurate broker service. They should exercise their own rights and obligations. We are highly motivated to change this semi manual system in to computerized system to increase the quality of the service, decrease workload, man power, cost and store the data in the database to change the current paper-based data record process.

A Broker Service Management System, often abbreviated as BSMS, is a comprehensive software solution tailored to meet the unique needs of brokerage firms and agencies. In a world where intermediaries play a crucial role in connecting buyers and sellers across various industries, BSMS software serves as the backbone for managing, streamlining, and optimizing the operations of these entities.

At its core, a BSMS is designed to simplify and automate the diverse tasks and processes associated with brokerages. BSMS not only enhances the effectiveness and productivity of brokers but also improves the overall experience for their clients.

1.1. BACKGROUND OF THE PROJECT

The Broker Service Management System (BSMS) project was initiated to address the inefficiencies and limitations of the existing manual system in managing broker services. In today's rapidly evolving market industry, brokers play a pivotal role in connecting buyers and sellers across various markets. However, the conventional manual processes have become increasingly inadequate in meeting the demands of this dynamic industry**.**

In order to solve the challenges posed by the manual method, the Broker Service Management System provides significant a step towards modernizing and improving broker services. The BSMS intends to change the way broker services are managed by leveraging automation and state-of-the-art technology. This will ultimately result in increased efficiency, security, and customer focus for all stakeholders.

1.2. STATEMENT OF THE PROBLEM

In the past, customers had to directly interact with brokers or conduct their own searches to fulfil their needs. They needed to communicate with service providers or sellers to determine if the services met their requirements. However, in today's fast-paced world, people are busier than ever, seeking simplification in various industries, from shopping to buying cars, renting house, and more. As a result, regular customers encounter multiple inconveniences in their quest for what they need. These inconveniences include waiting, differences in demand, and time-consuming processes. Currently, most customers manually search for their needs, such as homes and vehicles, which consumes their time and financial resources. It is also challenging to find the specific products or services of interest. The broker system remains non-digitalized, and as a result, several challenges persist in manual processes. These issues are outlined below:

- Communication between brokers and customers often suffers from delays and is subject to human error, affecting the overall customer experience.
- Manual broker operations consume substantial human resources and physical materials, resulting in higher operational costs.
- Manual processes often result in inefficiencies, such as the slow processing of paperwork and the need for extensive manual checks and verifications.
- Customer Dissatisfaction Due to Lengthy Service Times: Because it's difficult to serve clients quickly, customers become tired and disinterested, ultimately resulting in their dissatisfaction.

- Accessibility Barrier: The broker's office's fixed physical location creates a hindrance for clients residing at a distance. Only those in close proximity can easily register for selling, renting, or buying services. However, digitalization can break down these geographical barriers, enabling people from everywhere to access the services with ease.
- Inadequate Data Security: The reliance on a paper-based system makes the data vulnerable to unauthorized access and improper use, as there are limited safeguards in place to protect against such breaches.
- Lack of Organized Property Descriptions
- Data Redundancy

1.3. OBJECTIVES OF THE PROJECT

1.3.1. GENERAL OBJECTIVE

The main objective of Broker Service Management System is to design and develop web-based application for brokerage operations aimed at optimizing the efficiency, security and reliability for effective management system.

1.3.2. SPECIFIC OBJECTIVES

To achieve the above stated general objective, we formulate the following specific objectives:

- To design user friendly interface for easy navigation, ease of access and interaction with the BSMS
- To implement robust data security measures to protect sensitive client and property information's
- To implement efficient method for real-time communication between clients and brokers by implementing a secure and responsive messaging system.
- To create and integrate a centralized property database for the system, optimizing data retrieval and ensuring seamless interoperability.
- To implement an integrated payment processing system to improve financial transactions for efficient broker service management.
- To implement electronic signature functionality in the Broker Service Management System to enhance trust and document approvals.

1.4. FEASIBILITY ANALYSIS

The project's feasibility assessment is essential to ascertain whether it is worthwhile and aligns with the planned processes and objectives. In line with our project team's goals, this project addresses multiple challenges.

The fundamental criteria for determining the feasibility of our proposed system include:

- Ensuring user acceptance.
- Enhancing user trust and confidence in its reliability.
- Ensuring it effectively resolves the problems of the existing system.

Hence, the project teams are conducting several feasibility studies, including technical feasibility, economic feasibility, and operational feasibility.

1.4.1 OPERATIONAL FEASIBILITY

Operational feasibility focuses on ensuring our broker service management system not only meets its initial goals but also seamlessly integrates with existing processes, offering a superior user experience. It operates within the current system, creates a positive user environment. As we roll out the system, our dedicated team provides comprehensive training, enabling users to navigate and leverage its capabilities for a user-friendly experience. Our system effectively addresses issues, optimizes opportunities, and aligns with analysis phase requirements, rooted in providing timely, accurate, and user-friendly information, reliable services, and simplified data retrieval and management.

1.4.2 ECONOMIC FEASIBILITY

We're assessing the financial viability of our system by comparing costs to benefits. Using cost-effective software and hardware, our online service replaces the need for paper, reduces transportation costs, and overall offers greater benefits than costs, making it economically worthwhile for brokers and users. The expected benefits are greater than the expected costs, which means our system we're building is worth it economically.

1.4.3 TECHNICAL FEASIBILITY

A technical feasibility analysis helps us determine whether our proposed system can be supported by our existing technology or if it needs significant upgrades. The current manual process relies heavily on human effort and a deep understanding of paper-based tasks, making customer service a challenge. However, our system is designed with user-friendliness, it easy for the employees of the brokerage business to learn and use.

Our plan is to build the system using the Windows operating system and a familiar programming language. We assume that the necessary hardware and software resources are readily available for developing and implementing the system. Therefore, from a technical standpoint, the project is feasible and can be successfully carried out.

1.5. SCOPE AND LIMITATION OF THE PROJECT

1.5.1. SCOPE OF THE PROJECT

The Broker Service Management System (BSMS) is a web-based application development project aimed at enhancing communication and interaction between brokers or agents and their clients. The system will cover the functions listed below:

- User and property related information management
- Client communication management
- User account management
- Payment Processing
- Online agreement signing
- Reporting and analytics

Our system focuses on optimizing the management of information, communication, accounts, payments, agreement processes, and data analysis for brokers and their clients through a user-friendly web-based application.

1.5.2. LIMITATION OF THE PROJECT

- The system may not be fully accessible to people with disabilities, such as those with visual impairments**.**
- The system will not conduct property inspections or appraisals. it will assist in scheduling and tracking appointments. But not perform the actual inspections

- The system doesn’t work offline.

1.6. SIGNIFICANCE OF THE PROJECT

The significance of this project is to create efficient and effective working system for all the Broker and the target users. Generally, as we are under information and technology age using computerized system is more advantageous than that of using manual system to decrease the complexity of life. The following are some of the significances of the system:

- It creates a user-friendly web application for both brokers/agents and clients to facilitate efficient and transparent communication
- The system enables clients to connect with their brokers/agents easily, view property listings, and access important information related to the property
- Equip agents with tools to manage client interactions, property listings, appointments, and communication in one unified platform
- Provide clients with access to property information, appointment scheduling, and document management
- Provide real-time notifications for clients and agents to stay updated on property-related activities and appointments.
- Allow users to create and update their profiles, including contact information, preferences, and property requirements.
- Enable the uploading, storage, and retrieval of property-related documents, agreements, and contracts.
- Enables customers to refine their search criteria and select suitable employees from the list provided by brokers for potential hiring.

1.7. BENEFICIARY OF THE PROJECT

The Broker Service Management System (BSMS) will make buying, selling, and renting properties easier and more efficient. It will have a positive impact on the country's economy and the companies involved. Once the project is done, it will have various important benefits. Our project has been important from the start because it saves time and resources. It's a web app, so it helps in many ways. There are two main groups that benefit from our digital system.

Customers:

- The customer can easily see the list of houses, vehicles and other products and save them from extra expense for finding properties. Then this will facilitate its working process.
- As a result, they can see all information of their need in the system, they maximize service choices
- It saves time and money at the same time because information is easily available
- It increases the loyalty and truthfulness on the service

Brokers:

- It facilitates the working process and reduces the time needed to complete the task.
- Since it is a computerized system, it decreases their workload
- It saves time and energy loose
- It can promote properties easily
- Be profitable since they get the correct commissioning and the system is available for everyone.
- It avoids redundancy of documents that was happened in paper based
- It reduces economy spend for buying papers and pen.

1.8 METHODOLOGY OF THE PROJECT

1.8.1 DATA COLLECTION TOOL/TECHNIQUES

To do this project, we used different fact-finding techniques to gather information about the current system and to collect the necessary information that is needed to develop the project. In order to know how the existing system work and what problem are there we used the following fact-finding techniques.

- **Observation:** We observed how brokers do their work, how they handle tasks, communicate with clients, and how documentation is managed, including contracts, agreements, and legal paperwork. We also observed how they collect, store, and access their data.

- **Document analysis**: to get more information related to our project we analyzed different documents to understand the current system and make improvement in the proposed system. Going through existing documents makes the information trustworthily, documents are attached in appendix.
- **Interview**: we gathered required and relevant data through interview with different legal brokers in wolkite city and Addis Ababa.

1.8.2 SYSTEM ANALYSIS AND DESIGN APPROACH

In the system analysis and design phase of a project, we use the **OOSAD** (Object Oriented System Analysis and Design) approach. Because it is a better way to construct, manage and assemble objects that are implemented in our system. This technique has several phases, some of them are: -

1.8.2.1. OBJECT-ORIENTED ANALYSIS (OOA)

During this phase, the team use to model the function of the system (use case modeling), find and identify the business objects, organize the objects and identify the relationship between them and finally model the behavior of the objects in detail.

1.8.2.2. OBJECT-ORIENTED DESIGN (OOD)

During this phase, our team used draw-io and Visio software to refine the use case model and rational rose for designing the sequence, activity diagrams and to model object interactions and behavior that support the use case scenario.

The reason why we have selected OOSAD (Object Oriented System Analysis and Design) method specifically UML (Unified Modelling Language) model is because of the following advantages: -

- To enable a high degree of reusability of designs. To decrease the cost of software maintenance.
- To reduce maintenance burden.
- To Increased consistency among analysis, design, and programming activities.
- Improved communication among users, analysis, design, and programming.

1.8.3. SYSTEM DEVELOPMENT MODEL

In our project we will use the iterative model because of the following reasons:

- Iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.
- The life cycle model does not attempt to start with a full specification of requirements.
- This process is then repeated, producing a new version of the software at the end of each iteration of the model.
- The progress is easily measurable
- Generates working software quickly and early during the software life cycle. More flexible less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.

1.8.4. SYSTEM TESTING METHODOLOGY

- **Unit testing**

Since the designed system is in an object-oriented method the team firstly tested the system at the individual class level. For this we will use a tool called Jasmine

- **Integration Test**

To check whether the individual unit of the system is working together correctly or not, we will do Integration testing using a tool called postman.

- **System testing**

In this phase we will evaluate and test entire software system as a whole.

- **User Acceptance testing**

In this phase we will invite system end-users to determine whether it meets their specified requirements and if it is ready for production deployment

1.8.5. DEVELOPMENT TOOLS AND TECHNOLOGY

1.8.5.1. FRONT END TECHNOLOGY:

- **Next JS** is preferable for its ease of use, performance optimization features, and strong integration with React, making it a robust framework for building modern web applications that is why we choose it.
- Tailwind CSS
- JavaScript as our main programming language

1.8.5.2. BACK-END TECHNOLOGY:

- **NodeJS (with Express framework)** has a vast and active ecosystem of libraries and packages available through npm (Node Package Manager), providing developers with a wide range of tools and modules for various functionalities.
- **MongoDB (Document-oriented database**) is a NoSQL database that uses a flexible, JSON-like document structure.
- JavaScript as our main programming language

1.8.5.3. DOCUMENTATION AND MODELING TOOLS:

- **Documentation tools**

  - MS word 2021: We use for the purpose of writing documentation.
  - Microsoft PowerPoint 2021: We will use for making presentation.

- **Modeling Tools**

  - Figma: we used to design system user interface
  - draw-io: we used to draw use case diagrams, class diagrams and etc.

1.8.5.4. DEPLOYMENT ENVIRONMENT

For deployment environment we must have

- Desktop and personal computer
- Internet Connection
- Storage devices like hard disk, flash disk

1.9. DOCUMENT ORGANIZATION

The system document contains the following chapter each chapter describe in the following manner:

Chapter one:

We introduced the existing system, its problem, the project's objective (both general and specific), its scope, significance, benefits, limitations, timetable, budget, and the methods we used to collect and analyze the data.

Chapter two:

We described what the existing system looks like in detail (who uses the existing system with major function, business rule and drawback of the existing system will describe in this chapter in detail).

Chapter three:

We described the functional and non-functional requirements of proposed system

Chapter four:

We discussed about use case model, object model and dynamic models of proposed system.

Chapter five:

We designed goals, current and proposed software architecture, Hardware/software mapping, Persistent data management and Access control and security.

Chapter six:

In this chapter, we will discuss about proposed system implementation.

CHAPTER TWO

2. DESCRIPTION OF THE EXISTING SYSTEM

2.1. INTRODUCTION

The current broker service operates as a manual brokerage activity carried out through human involvement, without relying on automated or digital processes. These services involve direct interaction among brokers, buyers, and sellers. The brokers actively seek properties available for sale, purchase, or rent, as well as search for potential renters and announce their services to their clients. Then they manually record various property types based on their intended uses, such as for sale, rent. Following this, the brokers make these properties accessible to consumers if there is compatibility between the buyer/lessee and seller/lessor, and vice versa. Customers also go to the broker and register their available properties, which must be preserved within the existed system. The broker then provides services accordingly.

Brokers operate under government licensing and comply with legal regulations to ensure they offer legal and appropriate services to the society. These regulations outline the permissible actions, restrictions, and fees charged to clients giving obligations towards the government.

The existing system main activities:

- ✓ **Customer Search**

They look for available resources or properties in the town by asking owners if they need a broker, or by reading announcements, in which the seller/lessor advertises available properties on his building or on public notice boards, and the brokers look at the information and try to find the customer who wants to buy or rent.

- ✓ **Property Registration**

The current recording of customer properties is handled in a tedious manner. It is a process which involves manual entry of property details, which is prone to errors and consumes significant manpower. This manual approach often leads to delays in updating property information and cause challenges in maintaining an accurate and up-to-date data.

- ✓ **Conducting customers**

Considering available resource and customer satisfaction, both parties need to engage in negotiations to come to an agreement. Consequently, the broker meets with the two clients through different methods such as contact them through phone, meeting in person and such.

- ✓ **Make an agreement**

Following their discussion, they reach an agreement based on their specific requirements, including payment terms.

2.2. PLAYERS IN EXISTING SYSTEM

Brokers: The agent Activities like selling, buying, and renting vehicles, properties, houses and so on are accomplished by the brokers.

Users:

- ✓ Buyers: They buy houses, vehicles, lands, and some other properties.
- ✓ Seller: They sell houses, vehicles, lands, and some other properties.
- ✓ Lessors: They rent houses, vehicles, lands, and some other properties.
- ✓ Lessee: They lees houses, vehicles, lands, and some other properties.

Manager: Manages the overall activities of the company

Secretory: Organizing files, scheduling appointments, answering phone calls, preparing documents and so on.

2.3. BUSINESS RULES

The business rules of a broker system are guidelines or regulations that govern the operations, interactions, and transactions with which the proposed system operates accordingly. It deals with access control issues. It often pertains to access control issues, operating policies and principles of the organization.

Broker systems have the following principles in the existing system which includes:

For brokers:

- ✓ They should have a license from the government
- ✓ They should pay to the government the right and expected tax.

- ✓ They should only participate in licensed activities.
- ✓ They should confirm that the accessible properties are indeed available and lawful.
- ✓ They must follow the rules that have been established following an agreement.

For customers:

- ✓ The customer should have to register their properties
- ✓ They should provide all of their details for legal purposes.
- ✓ They must pay the requisite fee both before and after the agreement is reached.
- ✓ They should give proper information while property registration.

2.4. REPORT, FORMS GENERATED IN THE EXISTING SYSTEM

This system uses forms for registration, contract documents, agreement and others essential paperwork necessary for operational processes within the system.

2.5. CHALLENGES WITH THE CURRENT SYSTEM

Issues commonly found in the current broker service system include:

- ✓ High consumption of resources like paper, manpower, and time.
- ✓ Inefficient operations due to manual processes, impacting overall system effectiveness.
- ✓ Slow workflow caused by challenges in accessing data from the stored document, resulting in greater time and energy consumption.
- ✓ Longer waiting time for customers as the number of clients increases.
- ✓ Lack of authentication or authorization mechanism, causing difficulties in controlling and securing customer records.
- ✓ The workplace necessitates a significant number of filing cabinets. As a result, the current system's costs are high.
- ✓ Redundant storage of data leading to duplicated information being generated.

2.6. PRACTICES TO BE PRESERVED FROM EXISTING SYSTEM

This project aims to create a solution that addresses the current system's challenges, enhancing the effectiveness and efficiency of the service. It will consider the constraints of limited resources while utilizing the strengths of the existing system.

2.7. PROPOSED SYSTEM

We decided to develop a web-based broker service system that enables clients to access information at any time. The new system is aimed at benefiting broker and customers by saving time and money. To address current challenges, the proposed system will leverage on the major functionalities of the existing system, aiming to enhance system speed, security, and reliability.

Some components of the new system that solves the existing system are:

- ✓ It reduces the time taken that was consumed in the manual system.
- ✓ This system enables customer to search and view details of their need that is posted and will generate and forward announcement and the need notifications for the owner.
- ✓ Create a seller Agreement via eSignature and making this agreement document visible for the buyer to create more trust.

Our proposed system has the following purpose:

- ✓ The working process will be fast, attractive and accurate.
- ✓ Our system enhances security by assigning individual email and passwords based on their privileges.
- ✓ User login is protected through secure user authentication.
- ✓ Ensure data accessibility, confidentiality, and integrity.
- ✓ As the system relies on web-based automation, it reduces workload and manpower requirements.

CHAPTER THREE

3. PROPOSED SYSTEM

We are developing a web-based broker service management system that will make it easier and more convenient for our clients to access information. This new system will overcome the challenges of the current manual system and save our clients time and money.

The existing manual system was slow, resource consuming and unreliable. Our proposed system is designed to address all of these problems.

3.1. FUNCTIONAL REQUIREMENTS

Functional requirements capture the intended behavior of a broker service management system and it defines what a system is supposed to do. To achieve this, the following functional requirements have been established for the new proposed system.

- The system should be able to allow a system admin to create, update, and delete broker manager account.
- The system should enable the broker manager to create an account for brokers.
- The system should enable the broker manager to have the capability to create, modify, and remove posted properties.
- The system should be able to allow the broker to have the authority to approve properties.
- The system should enable the broker manager to assign brokers to properties.
- The system should enable the broker manager with the capability to add, update, and remove system users.
- The system should enable the broker manager to handle feedback received from users.
- The system should grant brokers the capability to add, modify and remove properties.
- The system shall facilitate brokers to communicate with users.
- The system should enable users to have the ability to create an account.
- The system should allow users to search for properties.
- The system should enable users to view posted properties.
- The system should allow users to add properties to the system.
- The system should provide users to communicate with brokers.
- The System should provide users to send feedback.
- The system should provide users to sign agreements by using e signature

- The system should enable users to make payments.
- The system should enable users to view listed employee information.
- The system should provide user with profile management capabilities.
- The system should provide user to download documents as soon as they make payments.
- The system should enable user to have the ability to add necessary property related documents to the system.
- The system should enable a broker to manage employee especially house-made and security Gard

3.2. NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirements are the constraints or the requirements imposed on the system. It specifies the quality attribute of the software. Non-functional requirements for a Broker Service Management System covers various aspects to ensure the system's performance, security, usability, and other qualities. The proposed system has the following nonfunctional requirements.

3.2.1. USER INTERFACE AND HUMAN FACTORS

- The system should provide an intuitive and user-friendly interface for brokers and users.

3.2.2. HARDWARE CONSIDERATION

- The system should be compatible with commonly used hardware configurations.

3.2.3. SECURITY ISSUES

- The system shall implement robust security measures to protect against internal and external intrusions.
- Access controls shall be implemented based on user roles and responsibilities. This ensures that only authorized users can access sensitive data and functionality based on their roles and responsibilities.
- Security algorithms, such as encryption standards, shall be specified. This ensures that the system uses strong and well-established security algorithms to protect data.

3.2.4. PERFORMANCE CONSIDERATION

- The system should be fast, reliable, and able to handle a lot of users.
- The system should be responsive, with low latency in processing broker service system.
- **Processing Time: -** Since the system will be, develop with efficient programming language like JavaScript and database upon request for user's Activities, the system under normal condition will process the request quickly and respond quickly.

3.2.5. ERROR HANDLING AND VALIDATION

- The system shall handle exceptions gracefully, preventing system crashes and data loss.
- The system shall validate user inputs to prevent data inconsistencies and errors. This ensures that only valid and properly formatted data is entered into the system, preventing data corruption and inconsistencies. Input validation should be implemented at various levels, including data entry forms, API requests, and data manipulation routines.

3.2.6. QUALITY ISSUES

- **Availability: -** The system will be available for 24 hours, if there is local area network or wide area network.
- **User operability: -**The system will offer simple navigation any user with basic computer knowledge so, can operate function.
- **Reliability: -** The proposed system minimize crash during its runtime, since more than one user could use the system simultaneously.

3.2.7. BACKUP AND RECOVERY

- The system shall maintain a comprehensive backup and recovery plan to ensure data integrity and business continuity.

3.2.8. PHYSICAL ENVIRONMENT

- The system shall be deployed in a secure and reliable data center environment with adequate hardware and infrastructure to support its operation.

3.2.9. RESOURCE ISSUES

- The system shall efficiently utilize hardware and network resources to optimize performance and minimize costs.

3.2.10. DOCUMENTATION

- The system shall provide comprehensive documentation to support its users and maintainers.
- This requirement emphasizes the importance of clear and accessible documentation for both users and technical personnel.

CHAPTER FOUR

4. SYSTEM ANALYSIS

4.1. SYSTEM MODEL

The system model for a Broker Service Management System encompasses various components and their interactions, providing a comprehensive overview of the system's structure and behavior. we used different object-oriented diagrams like system model use case diagram and object model class diagram and data directory and finally dynamic model like state chart, sequence and activity diagram in order to represent our system.

4.1.1. USE CASE MODEL

Creating a use case model involves identifying actors, defining use cases, and illustrating their relationships in a use case diagram. The actors and the use case that participate in our projects are:

Actor: System Admin

Use case:

- ✓ Login
- ✓ Logout
- ✓ Mange Broker manager
    - Create
    - Delete
    - Update
    - Read

Actor: Broker Manager

Use case:

- ✓ Login
- ✓ Assign brokers to the properties
- ✓ Manage brokers (CRUD brokers information)
- ✓ Manage Clients (CRUD client's information)
- ✓ Logout

Actor: Brokers

Use case:

- ✓ Manage employee (CRUD employee's information)
- ✓ Upload agreement document
- ✓ Approve properties
- ✓ Chat
- ✓ Login
- ✓ Logout

Actor: Client

Use cases:

- ✓ Create Account
- ✓ View listed properties
- ✓ Sign agreement
- ✓ Make payment
- ✓ Download document
- ✓ Chat
- ✓ view Listed employee
- ✓ Login
- ✓ Logout

4.1.1.1. USE CASE DIAGRAM

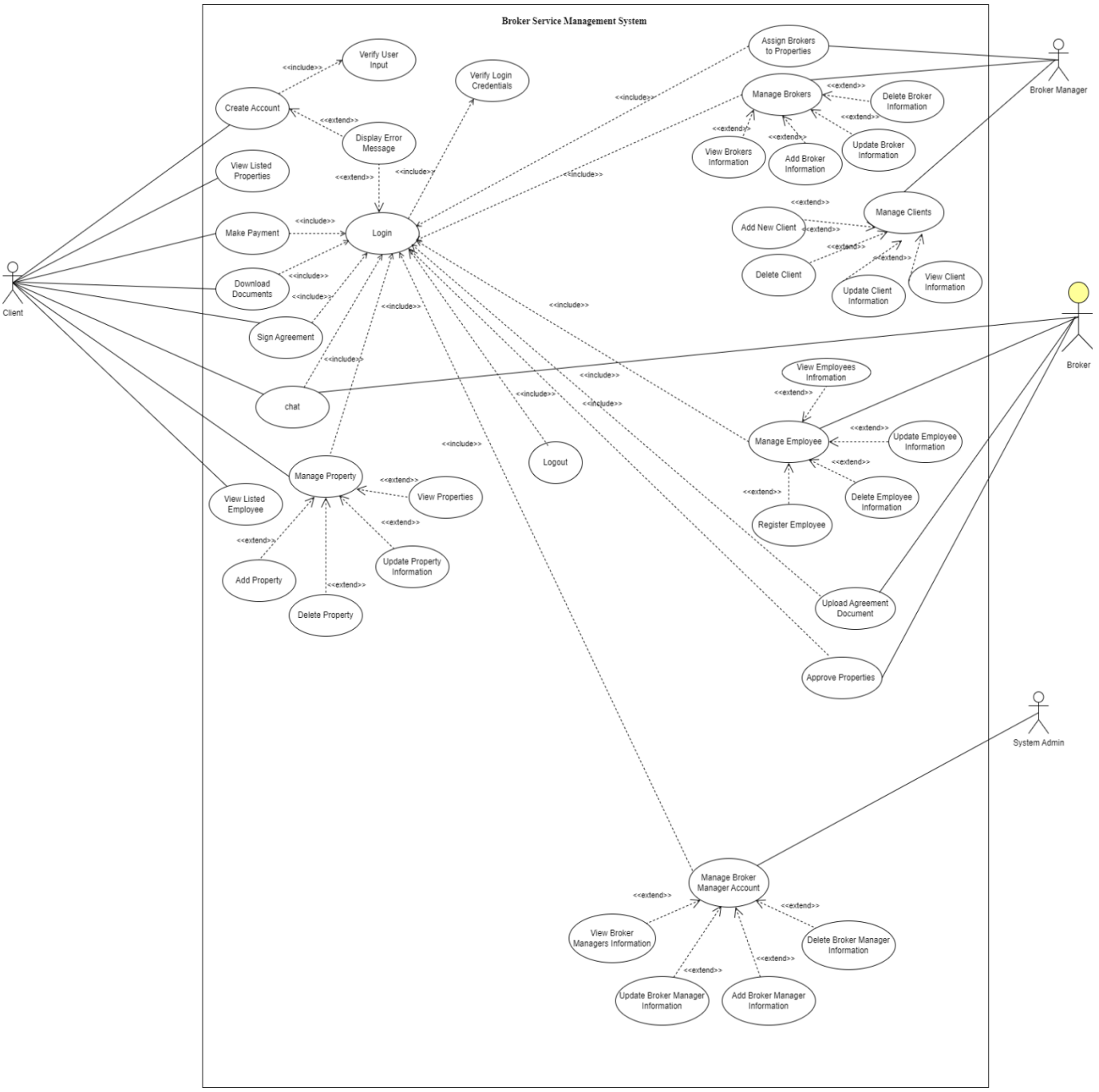


Figure 4. 1 :Use Case diagram

4.1.1.2. USE CASE DIAGRAM DESCRIPTION

The following tables show use cases description based on the system business or context of our system.

Table 4 1: Use Case Description for Login

<table>
<tr><td>Use case name</td><td colspan="2">Login</td></tr>
<tr><td>Use case id</td><td colspan="2">UC-01</td></tr>
<tr><td>Use case description</td><td colspan="2">A system allows users to log in and access their privileges within the system</td></tr>
<tr><td>Actors</td><td colspan="2">Client, Broker, Broker Manager, System Admin</td></tr>
<tr><td>Precondition</td><td colspan="2">The System must be accessible using web
The user, broker, broker manager, system admin must have an account</td></tr>
<tr><td rowspan="2">Basic flow</td><td>Actor action</td><td>System response</td></tr>
<tr><td>Step1: The client Open system web-page

Step3: The client clicks the login button on the landing page.

Step5: The client type email and password, then click the login button</td><td>Step2: The System displays the landing page

Step4: The System displays login interface for the user.

Step6: The System confirms the users email and password. [Alt. Course A]

Step7: The System displays a page with privileges of a client’s use case end. [Alt. Course B]</td></tr>
<tr><td>Alternative course of action</td><td colspan="2">Alt. Course A: If the entered information (email and password) is incorrect. The system returns to step 4 after displaying an error message.

Alt. Course B: if a client wants to exit from the system. Step 8: a client clicks on logout button then use case end.</td></tr>
<tr><td>Post condition</td><td colspan="2">Client successfully login into the system</td></tr>
</table>

Table 4 2: Use Case Description for Create Account

Use case name	Create Account	
Use case id	UC-02	
Use case description	The system allows users to create user account for them self's	
Actors	Client	
Precondition	client is on the account creation webpage or interface. The system is operational and able to accept new account creation requests.	
Basic flow	Actor action	System response
	Step1: The client opens system web-page **Step3:** The client clicks the signup button on the landing page. **Step5:** The client enters required information such as: - First Name - Last Name - Phone Number - Email Address - Password - Gender	**Step2:** The System displays the landing page **Step4:** The System displays registration interface for the clients. **Step6:** The System validates the entered information. [Alt. Course A] **Step7:** The System verifies the uniqueness of the provided email address to ensure it does not already exist within the system database. [Alt. Course B] **Step8:** The System displays a confirmation message indicating successful account creation and redirect the user to login page

Alternative course of action	Alt. Course A: If any entered information is invalid or missing, the system prompts the customer to correct the errors and resubmit the form. Alt. Course B: If the provided email already exists in the system, the system alerts the customer to use a different email or prompts them to proceed with a login if they have an existing account.
Post condition	A new User account is created in the system database with associated information

Table 4 3: Use Case Description for View Listed Properties

Use case name	View Listed Properties	
Use case id	UC-03	
Use case description	The system allows client to view listed properties on the landing page	
Actors	Client	
Precondition	The System must be accessible using web The System has access to a database of listed properties.	
Basic flow	Actor action	System response
	Step1: The client open system web-page **Step3:** The client navigates to the section of the system that displays listed properties **Step5:** The client selects a specific property from the list.	**Step2:** The System displays the landing page **Step4:** The system presents a comprehensive list of all properties. **Step6:** The system displays detailed information about the selected property.

Alternative course of action	In case of system downtime or technical issues, the user may not be able to access property details until the system functionality is restored.
Post condition	The client can efficiently navigate through listed properties and access their details.

Table 4 4: Use Case Description for Sign Agreement

Use case name	Sign Agreement	
Use case id	UC-04	
Use case description	The system allows to electronically sign an agreement document within the broker service management system using an e-signature feature.	
Actors	Client	
Precondition	The client, broker has logged into the broker service management system. The client, broker has access to the specific agreement document requiring signature.	
Basic flow	Actor action	System response
	Step1: The client open system web-page **Step3:** The client logs into the broker service management system using their email and password. **Step5:** The client navigates to the document management section **Step6:** The client identifies the agreement requiring signature from the list of available documents. **Step7:** The client selects the agreement document to begin the signing process.	**Step2:** The System displays the landing page **Step4:** Upon successful login, the System directs the client to user- dashboard. **Step8:** The System opens the document within a secure e-signature platform integrated into the system. **Step10:** The System prompts the ser to place their electronic

	Step9: The client scrolls through the document pages and sections to ensure its accuracy and completeness. **Step11:** The client utilizes a digital signature tool to sign the document using a mouse or touchscreen. **Step12:** After signing, the client confirms their signature placement and intent to sign the document.	signature at specified signature fields or designated areas within the document. **Step13:** Upon successful confirmation, The System applies the user's e-signature to the document.
Alternative course of action	In case of signature rejection or error, the system allows the client to rectify or cancel the signing process before final submission	
Post condition	The agreement document is digitally signed and stored securely within the broker service management system	

Table 4 5: Use Case Description for Make Payment

Use case name	Make Payment	
Use case id	UC-05	
Use case description	The system enables secure payment processing by integrating a third-party payment API into the broker service management system.	
Actors	Client	
Precondition	The client is logged into the broker service management system. The third-party payment API is integrated and accessible within the system	
Basic flow	Actor action	System response
	Step1: The client clicks on the Pay button **Step2:** The client provides payment details.	**Step3:** Upon submission of payment details, the System utilizes the integrated third-party payment API.

	Step5: The client is redirected to the payment gateway provided by the third-party API **Step6:** The client follows the prompts provided by the payment gateway to authenticate the payment.	**Step4:** The System securely sends the payment request along with necessary transaction information to the third-party API. **Step7:** Upon successful authentication and authorization, the payment gateway processes the transaction securely. **Step8:** The third-party API communicates the transaction status (success or failure) back to the System. **Step9:** The broker service management System generates a payment confirmation and updates the payment status. **Step10:** The System redirects the user to success page acknowledging the completed payment.
Alternative course of action	• If the payment fails due to technical issues, insufficient funds, or any other reason, appropriate error handling messages are displayed to the user. • In case of payment gateway errors or timeouts, the system provides guidance and instructions for the user to retry or seek assistance.	
Post condition	The payment has been securely processed through the third-party payment API, and the broker service management system has updated records of the transaction.	

Table 4 6: Use Case Description for Download Documents

Use case name	Download Documents	
Use case id	UC-06	
Use case description	The system allows users who have completed a payment to access and download the signed agreement document	
Actors	Client	
Preconditions	The client is logged into the system. The client has successfully completed the payment process.	
Basic flow	Actor action	System response
	Step2: The client by clicking on a Download button on a Download Signed agreement section initiates the downloading process.	**Step1:** After payment confirmation, the System directs the user to a section titled Download Signed Agreement indicating that the signed property agreement documents are available for download. **Step3:** The System prompts the client to save the downloaded file to a specified location on their device **Step4:** The System shows download progress to the client, ensuring they are informed of the download status. **Step5:** Upon successful completion of the download, the System displays a confirmation message, confirming the successful download of the signed property agreement documents.
Alternative course of action	In cases where the payment is not completed, the client must need to complete to gain access to the signed property agreement document.	
Post condition	The client successfully downloads the signed property agreement document.	

Table 4 7: Use Case Description for Chat

Use case name	Chat	
Use case id	UC-07	
Use case description	The system allows real-time communication and interaction between client and brokers within the broker service management system.	
Actors	Client, Broker	
Precondition	Both the client and broker are registered and logged into the broker service management system. The client has initiated a request for communication with a broker. The broker is available to engage in a conversation with the user	
Basic flow	Actor action	System response
	Step1: The client initiates a chat session by clicking in the chat with broker button on the desired property item. **Step4:** The Broker accepts the chat request and engages in conversation with the client.	**Step2:** The System opens a chat interface, allowing the client to send messages and engage in real-time conversation with the selected broker. **Step3:** The System, notifies the corresponding broker of the property about the incoming message. **Step5:** Upon successful completion of the download, the System displays a confirmation message, confirming the successful download of the signed property agreement documents.
Alternative course of action	If the selected broker is offline, the system must allow client to leave a message.	
Post condition	A successful exchange of information occurs between the client and the broker within the chat session. Both parties have access to the chat history for reference or follow-up purposes.	

Table 4 8: Use Case Description for View Listed Employee

Use case name	View Listed Employee	
Use case id	UC-08	
Use case description	The system allows employers to access and view the details of employees listed within the broker service management system.	
Actors	Client	
Precondition	• The client is logged into their account with in the broker service management system.	
Basic flow	Actor action	System response
	Step1: Upon logging into the broker service management system, The client navigates to a section on the system that contain employee list.	**Step2:** The System displays a list of employees available for the employer to view. **Step3:** The System allows the employer to apply filters, search by specific criteria or sort the employee listings based on desired parameters. **Step:4** Upon selecting an employee from the list, The System presents detailed information about the chosen employee.
Alternative course of action	In cases where specific employee details are not available or incomplete, the employer may choose to explore other employees or revisit the employee listings later	
Post Condition	The client successfully accesses and views information about listed employees within the broker service management system.	

Table 4 9: Use Case Description for Manage Property

Use case name	Manage Property	
Use case id	UC-9	
Use case description	The system allow user efficiently manage and update properties listed on the platform, ensuring accurate and up-to-date information for potential buyers or renters.	
Actors	Client	
Precondition	The client is logged into their account within the broker service management system. The client has the necessary permissions or access rights to manage properties.	
Basic flow	Actor action	System response
	Step1: Upon logging into the system, the client, Broker navigates to the property management section within the platform. **Step3:** The client, Broker selects a specific property from the list that requires management, modification, or updates. **Step4:** The client, broker uploads images or documents associated with the property to enhance the listing's visual representation.	**Step2:** Within the property management interface, the system displays a list of properties managed by client, Broker **Step5:** The System presents options to edit or modify various details of the selected property. **Step:7** After saving changes, the system acknowledges the successful update of property details. **Step8:** The updated information becomes available within the property listing on the platform.

Alternative Course of Action	In cases where mandatory details are missing or incomplete, the system may prompt the user, broker to provide the necessary information before allowing submission
Post condition	The client, Broker successfully manages and updates the details of a property within the broker service management system, ensuring accurate and relevant information for potential buyers or renters.

Table 4 10: Use case description for Upload agreement document

Use case name	Upload agreement document	
Use case id	UC-10	
Use case description	The system enables a broker to upload agreement document for the user	
Actors	Broker manager	
Precondition	The Broker has a valid email and password for authentication. The Broker has successfully logged into the BSMS. The system is operational and accessible. The Broker has the necessary permissions to upload agreement documents.	
Flow events	Actor Action	System response
	Step 1: The Broker navigates BSMS web page. **Step 3:** The Broker logs into the Broker Service Management System with valid email and password. **Step 6:** Broker Selects Upload Agreement Document option. **Step 8:** The Broker clicks the "Browse" or "Choose File" button.	**Step 2** The system renders and presents the BSMS web page to the broker. **Step 4:** The system validates the broker credentials and grants access. **Step 5:** The system displays a dashboard. **Step 7:** The system presents options

	Step 10: The Broker chooses the agreement document from their local files. **Step 12:** Broker Confirms Upload. **Step 15:** The broker chooses to log out from the system.	**Step 9:** The system provides a button for selecting the agreement document to upload. **Step 11:** System Validates Document Format and Size. **Step 13:** The system processes the uploaded agreement document. **Step 14**: The system returns to the main agreement management interface or dashboard. **Step 16:** The system ends the user session and returns to the login screen.
Alternative flow events/course of action	If a broker enters a wrong/invalid email and password. The system displays please enter valid email Address and password. If the document format or size is invalid, the system displays an error message and prompts the Broker to choose a valid document.	
Postcondition	The agreement document is successfully uploaded by Broker	

Table 4 11: Use case description for Manage Brokers

Use case name	Mange Brokers
Use case id	UC-11
Use case description	The system allows a broker manager to delete, update and create brokers/agents
Actors	Broker Manager
Precondition	The broker manager has a valid email and password for authentication. The broker manager has successfully logged into the system.

	The system is operational and accessible.	
Flow events	Actor Action	System response
	Step 1: The broker manager navigates to the BSMS web page. **Step 3:** The broker manager logs into the Broker Service Management System with valid email and password. **Step 5:** The broker manager chooses from the available actions: "Create Broker," "Update," or "Delete Broker." **Step 9:** The broker manager chooses to log out from the system.	**Step 2:** The system renders and presents the BSMS web page to the broker manager. **Step 4:** The system validates the broker manager credentials and grant access. **Step 7:** The system presents options. **Step 8:** The system processes the submitted information and creates a new broker. The system processes the submitted changes and updates the information for the selected broker. The system removes the selected broker from the system. **Step 10:** The system ends the user session and returns to the login screen.
Alternative flow events/course of action	If a broker manager enters a wrong/invalid email and password.	

	The system displays please enter valid email Address and password.
Postcondition	The brokers information's are successfully managed.

Table 4 12: Use case description for assign brokers to properties

Use case name	Assign brokers to properties	
Use case id	UC-12	
Use case description	Broker Managers within the Broker Service Management System, enabling efficient allocation and management of responsibilities among their members through the assign of brokers to specific properties.	
Actors	Broker Manager	
Precondition	The broker manager has a valid email and password for authentication. The broker manager has successfully logged into the system. The system is operational and accessible. Properties and brokers are already registered in the system.	
Flow events	Actor Action	System response
	Step 1: The broker manager navigates to the BSMS web page. **Step 3:** The broker manager logs into the Broker Service Management System with valid email and password. **Step 5:** Broker Manager Selects Assign Brokers to Properties option	**Step 2:** The system renders and presents the BSMS web page to the broker manager. **Step 4:** The system validates the broker manager credentials and grant access. **Step 7:** The system shows a list of available brokers and properties that can be assigned.

	Step 6: The Broker Manager selects a specific property to which brokers will be assigned. The Broker Manager adds one or more brokers to the selected property. **Step 10:** The broker manager chooses to log out from the system.	**Step 8:** The system provides options to add or remove brokers from the selected property. **Step 9:** Notifications sent to the assigned brokers about their new responsibilities. **Step 11:** The system ends the user session and returns to the login screen.
Alternative flow events/course of action	If a broker manager enters a wrong/invalid email and password. The system displays please enter valid email Address and password. The system validates the assign process, ensuring that each broker is not assigned to multiple properties simultaneously. If it not displays an error. If the Broker Manager cancels the assigning process, the system returns to the list of available brokers and properties without making any changes.	
Postcondition	Brokers are successfully assigned to the selected property.	

Table 4 13: Use case description for approve properties

Use case name	Approve properties
Use case id	UC-13
Use case description	The Broker Service Management System's capability to empower a broker to review and approve various types of properties, including vehicles, land, and houses.
Actors	Broker
Precondition	The broker has a valid email and password for authentication. The broker has successfully logged into the system. The system is operational and accessible.

Flow events	Actor Action	System response
	Step 1: The broker navigates to the BSMS web page. **Step 3:** The broker logs into the Broker Service Management System with valid email and password. **Step 5:** The broker navigates to approval section **Step 7:** The Broker selects a property from the list for detailed review. **Step 9:** The Broker decides whether to approve or reject the property. **Step 10:** The broker chooses to log out from the system.	**Step 2:** The system renders and presents the BSMS web page to the broker. **Step 4:** The system validates the broker credentials and grant access. **Step 6:** The system presents a list of properties, including vehicles, land, and houses, that are awaiting approval. **Step 8:** The system displays detailed information about the selected property, including relevant details and documentation. **Step 11:** The system ends the user session and returns to the login screen.
Alternative flow events/course of action	If a broker manager enters a wrong/invalid email and password. The system displays please enter valid email Address and password. The Broker decides to reject the property, our system prompts for a reason for rejection.	
Postcondition	The selected properties (vehicles, land, houses) have been successfully reviewed and approved by the Broker,	

Table 4 14: Use case description for manage employee

Use case name	Manage employee
Use case id	UC-14
Use case description	The system that allows broker manager to oversee and administer employee (house made and security guard)
Actors	Broker

Precondition	The broker has a valid email and password for authentication. The broker has successfully logged into the system. The system is operational and accessible.	
Flow events	Actor Action	System response
	Step 1: The broker navigates to the BSMS web page. **Step 3:** The broker logs into the Broker Service Management System with valid email and password. **Step 5:** The broker navigates to employee management section **Step 7:** The broker adds, modify, delete employee details. **Step 9:** The broker chooses to log out from the system.	**Step 2:** The system renders and presents the BSMS web page to the broker manager. **Step 4:** The system validates the broker credentials and grant access. **Step 6:** The system presents a list of forms, including details such as phone number, roles, skill, experience and contact information. **Step 8:** The system records any modifications, additions, or removals of employee. **Step 10:** The system ends the user session and returns to the login screen.
Alternative flow events/course of action	If a broker enters a wrong/invalid email and password. The system displays please enter valid email Address and password.	
Postcondition	Employee information's is effectively managed, and any changes are recorded in the system.	

Table 4 15: Use case description for Manage Clients

Use case name	Mange Clients	
Use case id	UC-15	
Use case description	The system allows a broker manager to delete, update and create clients	
Actors	Broker Manager	
Precondition	The broker manager has a valid email and password for authentication. The broker manager has successfully logged into the system. The system is operational and accessible.	
Flow events	Actor Action	System response
	Step 1: The broker manager navigates to the BSMS web page. **Step 3:** The broker manager logs into the Broker Service Management System with valid email and password. **Step 5:** The broker manager chooses from the available actions: Create Client, Update, or Delete Client. **Step 9:** The broker manager chooses to log out from the system.	**Step 2:** The system renders and presents the BSMS web page to the broker manager. **Step 4:** The system validates the broker manager credentials and grant access. **Step 7:** The system presents options. **Step 8:** The system processes the submitted information and creates a new client. The system processes the submitted changes and updates the information for the selected client. The system removes the selected client from the system. **Step 10:** The system ends the user session and returns to the login screen.
Alternative flow	If a broker manager enters a wrong/invalid email and password. The system displays please enter valid email Address and password.	

events/course of action	
Postcondition	The client's information's are successfully managed.

Table 4 16: Use case description for Logout

Use case name	Logout	
Use case id	UC-16	
Use case description	The system enables a system user to clear session data and ensure security.	
Actors	System Admin, Client, Broker, Broker Manager	
Precondition	The System Users must be logged into the system.	
Flow events	Actor Action	System response
	Step 1: The system users click a logout button. **Step 3:** The system users confirm the logout in the prompt.	**Step 2:** The system renders and presents confirmation prompts to the system users. **Step 4:** The system clears user session data.
Alternative flow events/course of action	If the system users cancel the logout, the system remains in the current state, and no session data is cleared. The system user continues their session without logging out.	
Postcondition	The System user is successfully logged out, and their session is terminated.	

4.1.1.3. USE CASE SCENARIO

A use case scenario is a detailed narrative description of how a system interacts with an external entity, typically a user or another system, to accomplish a specific goal. It outlines the sequence of events and interactions that occur when a user interacts with a system to achieve a particular outcome.

Use Case Name: Login

Actors:

1. **System users:** The individual who needs to access the Broker Service Management System.

Preconditions:

- The Broker Service Management System is running.
- The system user has a valid account with the necessary credentials.

Basic Flow:

Step 1: The system users navigate to the landing page of the Broker Service Management System.

Step 2: The system users navigate to the login page of the Broker Service Management System.

Step3: The system users enter their email and password into the designated fields on the login page.

Step 4: The system checks the entered credentials against the stored user database.

Step 5: If the entered credentials are valid, the system grants access to the Broker Service Management System.

Step 6: Upon successful authentication, the system redirects the user to the home page of the Broker Service Management System.

Alternative Flow:

1. **Invalid credentials:**
    - If the entered credentials are invalid, the system displays an error message.

2. **Forgot Password:**
    - If the user forgets their password, they can click on a Forgot Password link.
    - The system provides a password recovery process, such as sending a reset link to the user's registered email address.

Postconditions:

- The user is successfully logged into the Broker Service Management System.

Use Case Name: View Listed Properties

Actors: client

Preconditions:

- The system is running and has a database of listed properties.

Basic Flow:

1. **Navigate to Property Listings:**
    - The client is presented user homepage to navigate to the "Property Listings" section.
2. **Search or Browse Properties:**
    - The client uses a search feature to find specific properties or browse through the general list of available properties.
3. **View Property Details:**
    - The system displays a list of properties.
    - The client clicks on a specific property to view detailed information.
4. **Property Details:**
    - The system presents detailed information about the selected property, including images, description, specifications, price, and contact details of the broker or agent.
5. **Navigate Back to Listings:**

- The client has the option to go back to the list of properties or continue browsing other listings.

Alternative Flow:

1. **Search Filters:**
    - Instead of browsing, the customer uses search filters to narrow down the list based on specific criteria such as location, price range, or property type.

Postconditions:

- The client has successfully viewed detailed information about one or more listed properties.

Exceptional Conditions:

- Network issues may disrupt the loading of property details.
- If the system experiences a temporary glitch, the user needs to refresh a system.

Use Case Name: Sign Agreement

Actors:

- Client

Preconditions:

- Client are logged into their account.

Basic Flow:

1. Client navigates to the Agreements section.
2. Client selects the specific agreement requiring a signature.
3. Client reviews the agreement terms.
4. Client digitally signs the agreement.

Alternative Flow:

1. Agreement requires modifications.
2. Client requests changes, and the system notifies the other party.

Postconditions:

- The agreement is signed and stored in the system.

Use Case Name: Make Payment

Actors: Client

Preconditions:

- Client is logged into their account.
- There is an outstanding payment for a property.

Basic Flow:

1. Client navigates to the Payments section.
2. Client selects the property or invoice for payment.
3. Client chooses a payment method.
4. Client enters payment details.
5. Client confirms and submits the payment.

Alternative Flow:

1. Payment fails due to insufficient funds.
2. Client is notified and prompted to update payment information.

Postconditions:

- The payment is processed, and the client receives a payment confirmation.

Use Case Name: Download Document

Actors:

- Client

Preconditions:

- Client is logged into their account.
- There are documents available for download.

Basic Flow:

1. Client navigates to the Documents section.
2. Client selects the document they want to download.
3. Client initiates the download.

Postconditions:

- The document is downloaded to the client's device.

Use Case Name: Chat

Actors:

- Client, Broker

Preconditions:

- Both parties (client and broker) are logged into their accounts.

Basic Flow:

1. Client or broker navigate to the Chat section.
2. Client or broker select the user they want to chat with.
3. Client or broker send a message.
4. The other party receives and responds to the message.

Postconditions:

- A chat history is maintained in the system.

Use Case Name: Manage Property

Actors:

- Client

Preconditions:

- Client is logged into their account on the Broker Service Management System with valid Email and password.

Basic Flow:

1. Client navigates to the Manage Property section.
2. Client sees a list of their listed property
3. Client selects a property to view or modify.
4. Client can view and edit property details, including price, description, and images.
5. Client saves any changes made to the property.

Alternative Flow:

1. Client wants to remove a property from the listings.
2. Client selects the property to remove.
3. Client confirms the removal, and the property is no longer listed.

Postconditions:

- The property details are updated or removed from the system, reflecting any changes made by the user.

Use Case Name: View Listed Employee

Actor:

- Client

Preconditions:

- Client is logged into their account on the Broker Service Management System
- There are employees listed in the system.

Basic Flow:

1. **Navigate to Employee Listings:**
    - Client logs into the Broker Service Management System.
    - Client navigates to the "View Employees" or "Employee Listings" section.
2. **View Employee Details:**
    - Client clicks on a specific employee to view detailed information.

3. **Navigate Back to Employee Listings:**
    - After viewing an employee's details, the employer has the option to go back to the list of employees or continue browsing.

Alternative Flow:

1. **Search Filters:**
    - Instead of browsing, the employer uses search filters to narrow down the list based on specific criteria.

Postconditions:

- The client has successfully viewed detailed information about one or more listed employees.

Exceptional Conditions:

- Network issues may disrupt the loading of employee details.

Use Case Name: Upload Agreement Document

Actor:

- Broker

Preconditions:

- Broker is logged into their account on the Broker Service Management System with valid Email and Password.

Basic Flow:

1. Broker navigates to the Upload Agreement Document section.
2. Broker selects the option to upload a new agreement document.
3. Broker uploads the document file.
4. The system validates and stores the uploaded agreement document.

Postconditions:

- The agreement document is successfully uploaded and stored in the system.

Use Case Name: Assign Brokers to Properties

Actor:

- Broker Manager

Basic Flow:

- Broker Manager navigates to the Assign Brokers section.
- Broker Manager selects a property.
- Broker Manager assigns one or more brokers to the selected property.

Postconditions:

- Brokers are successfully assigned to the specified property.

Use Case Name: Manage Brokers

Actor:

- Broker Manager

Basic Flow:

1. Broker Manager navigates to the Manage Brokers section.
2. Broker Manager perform CRUD operations on broker information:
    - **Create:** Broker Manager adds new broker profiles.
    - **Read:** Broker Manager views details of existing brokers.
    - **Update:** Broker Manager modifies broker information.
    - **Delete:** Broker Manager removes broker profiles/information.

Postconditions:

- Broker information is created, updated, or deleted as per the Broker Manager's actions.

Use Case Name: Approve Properties

Actor:

- Broker

Basic Flow:

1. Broker navigates to the Approve Properties section.
2. Broker reviews and approves pending property listings.

Postconditions:

- Pending property listings are approved and become visible to users.

Brokers Use Case Scenario

1. **Manage Employee**

Use Case Name: Manage Employee

Actor:

- Brokers

Basic Flow:

1. Brokers navigates to the Manage Employee section.
2. Brokers perform actions related to managing employees:
    - **Create:** Brokers creates new employee profiles.
    - **Read:** Brokers views details of existing employees.
    - **Update:** Brokers modifies employee information.
    - **Delete:** Brokers removes employee profiles.

Postconditions:

- Employee information is created, updated, or deleted as per the Brokers/Agent's actions.

Manage Broker Managers

Use Case Name: Manage Broker Managers

Actor:

- System Admin

Basic Flow:

1. System Admin navigates to the Manage Broker Manger section.
2. System Admin perform actions related to managing broker manager:
    - **Create:** System admin creates new broker manager.
    - **Update:** System Admin modifies broker manager.
    - **Delete:** System admin removes broker manager.

Postconditions:

- Broker manager information is successfully created, updated, or deleted as per the System Admin's actions.

Manage Clients (CRUD Clients Information)

Use Case Name: Manage Clients

Actor:

- Broker Manager

Basic Flow:

1. Broker Manager navigates to the Manage Clients section.
2. Broker Manager perform CRUD operations on client information:
    - **Create:** Broker Manager adds new client profiles.
    - **Read:** Broker Manager views details of existing clients.
    - **Update:** Broker Manager modifies client information.
    - **Delete:** Broker Manager removes client profiles/information.

Postconditions:

- Client information is created, updated, or deleted as per the Broker Manager's actions.

Logout

Use case Name: Logout

Actor: System Admin, Client, Broker, Broker Manager

Basic Flow:

1. The system users click a logout button.
2. The system renders and presents confirmation prompts to the system users.
3. The system clears user session data.

Post Condition:

- The System user is successfully logged out, and their session is terminated.

4.2. OBJECT MODEL

An object model is a conceptual representation of a system that captures the structure and behaviour of the system's entities using object-oriented principles.

4.2.1. CLASS DIAGRAM

A class diagram is a type of diagram from the Unified Modelling Language (UML) that represents the static structure of a system by depicting the classes, their attributes, methods, and the relationships among them. It provides a blueprint for understanding and designing the structure of a software system from an object-oriented perspective.

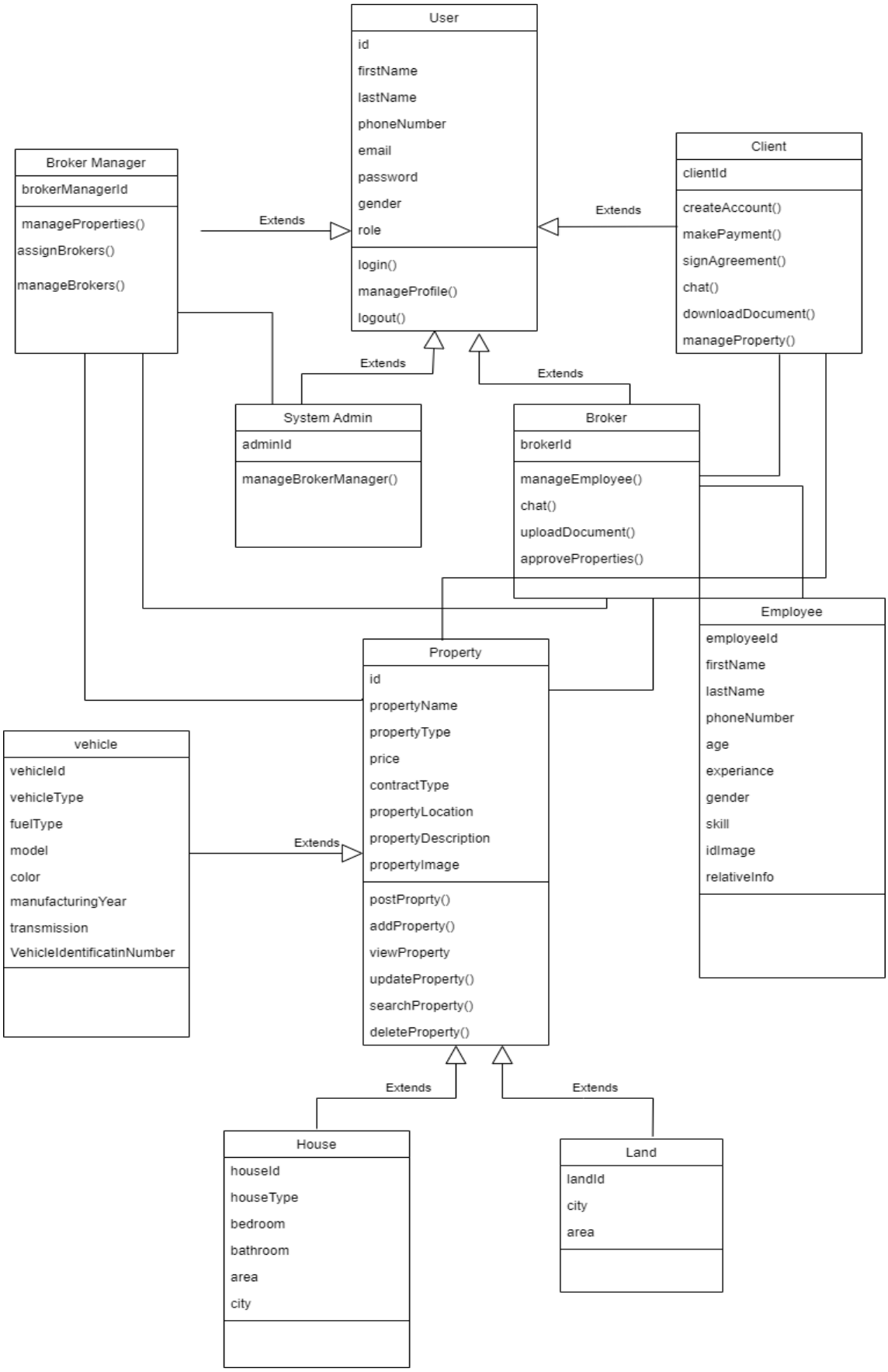


Figure 4. 2: Class Diagram

4.2.2. DATA DICTIONARY

A Data Dictionary for a Broker Service Management System is a structured documentation that defines and describes the data elements used in the system. It includes information about names, definitions, attributes, and relationships of various data entities.

Table 4 17: Data dictionary for Employee

Field Name	**Data Type**	**Size**	**Description**
Id	ObjectId	24	Unique Identifier of the employee
FirstName	String	30	The FirstName of the employee
LastName	String	30	The lastName of the Employee
Phone Number	String	15	Phone Number of the Employee
DOB	String	10	The Birthday of the employee
Experience	String	100	The experience of the employee
Gender	String	10	Gender of the employee
Job Type	String	50	The type of the job either House made or squirty Guard
Skill	String	100	Represents a specific skill of the employee
ID Image	String	100	The URL image of the Identification Card
Description	String	255	It provides Detail description of the employee

Table 4 18: Data dictionary for User

Field Name	**Data Type**	**Size**	**Description**
Id	ObjectId	24	Unique Identification of user
FirstName	String	30	The first Name of the user
LastName	String	30	The last Name of the user
Phone Number	String	15	Phone Number of the user
Email	String	50	Email Address of the user
Password	String	50	Password given to a user to login
Gender	String	6	Sex of the user
Role: ✓ Broker ✓ Broker Manger ✓ System Admin ✓ Client	String	30	It represents the role or type of user within the system. It defines a user permission or access level or functionalities

Table 4 19: Data dictionary for Property/Houses

Field Name	**Data Type**	**Size**	**Description**
Images	String	100	It stores the URL images the house
Contract Type	String	50	For sale or for rent
Price	Double	-	The price information for the house.

House Type	String	50	Specifies the type of house, such as single-family, multi-family, condominium, townhouse, etc.
Bedroom	Integer	20	Number of bedrooms in the house
Bathroom	Integer	6	Number of bathrooms in the house
Area	Integer	100	The total area or size of the house
City	String	50	The city where the house is located.
Location	Object		The location of the house
Description	String	255	A detailed textual description of the house
HouseId	ObjectId	24	A unique identification

Table 4 20: Data dictionary for Vehicles

Field Name	**Data Type**	**Size**	**Description**
Images	Array	5	It stores a reference of images of vehicles
Contract Type	String	50	It specifies a Vehicle for sell or for rent
Price	Double		It holds the price information for the vehicle
Vehicles Type	String	50	It specifies the type or category of the vehicle

Model	String	50	The model's name of vehicles
Fuel Type	String	50	Indicates the type of fuel used by the vehicle, such as gasoline, diesel, electric, etc.
Description	String	255	A textual description of the vehicle, providing additional details beyond the model's name or number.
Colour	String	50	A colour of Vehicles.
Transmission	String	100	It specifies transmission type of Vehicles, automatic or Manual
VIN (Vehicle Identification Number)	Integer	30	Serves as a unique identifier for the vehicle.
Manufacturing year	Integer	4	Manufacturing year of Vehicles
Vechicle Id	ObjectId	24	Unique Id

Table 4 21: Data Dictionary for Land

Field Name	**Data Type**	**Size**	**Description**
Images	Array	5	It stores a reference of images of land
Contract Type	String	50	It specifies land for sell or for rent
Price	Double		Price information of Land
Area	Double		It specifies the total area or size of the land

Location	Object		Represents the geographical location of the land
Description	String	255	A textual description of the Land, providing additional details.
City	String	100	It represents exact location of land
Land Id	ObjectId	24	Unique ID

Table 4 22: Data dictionary for employee relative(ተዛማጅ)

Field Name	**Data Type**	**Size**	**Description**
First Name	String	30	The first Name for employee Relative
Last Name	String	30	The last Name for employee Relative
Relative	String	30	A various Relationship between the employee
Address	Object		The location of the employee relative
Phone Number	String	15	Contact number of the employee Relative
Identification Card image	Array	5	It stores the URL images the employee relative Identification card
Id	ObjectId	12	Unique id

4.3. DYNAMIC MODEL

A dynamic model is a representation of a system's behaviours over time, emphasizing how components interact and respond to various events.

4.3.1. SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

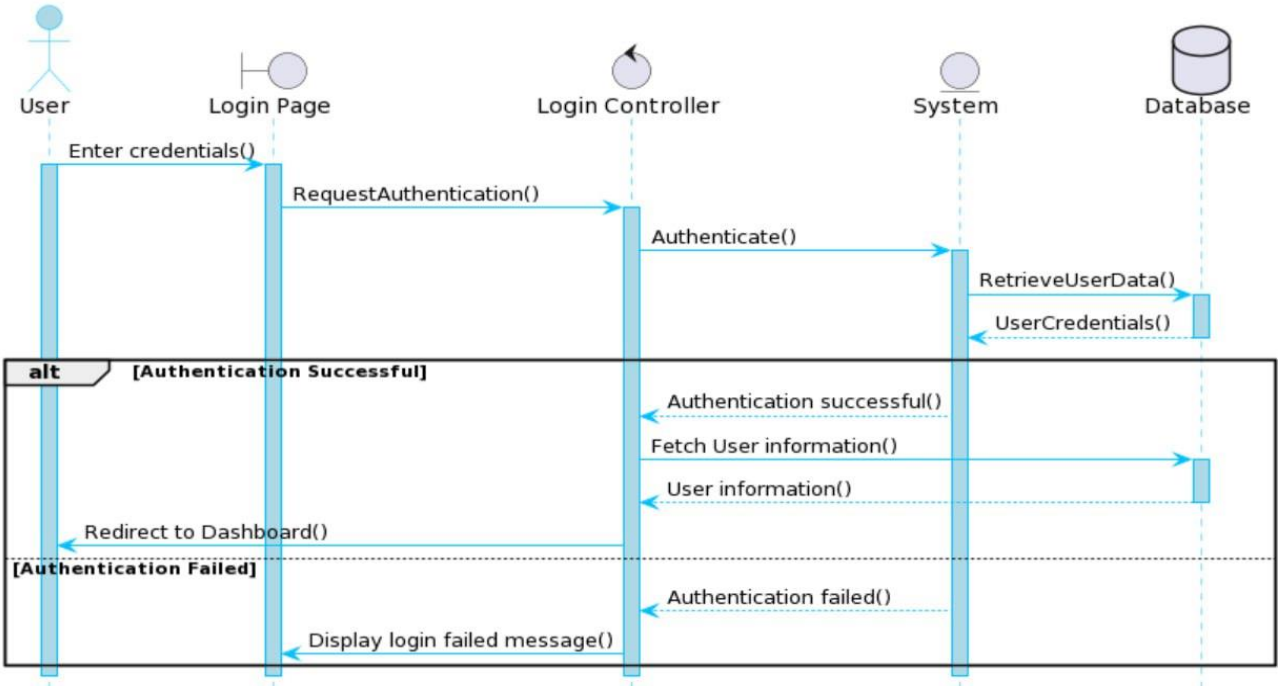


Figure 4. 3: Login Sequence Diagram

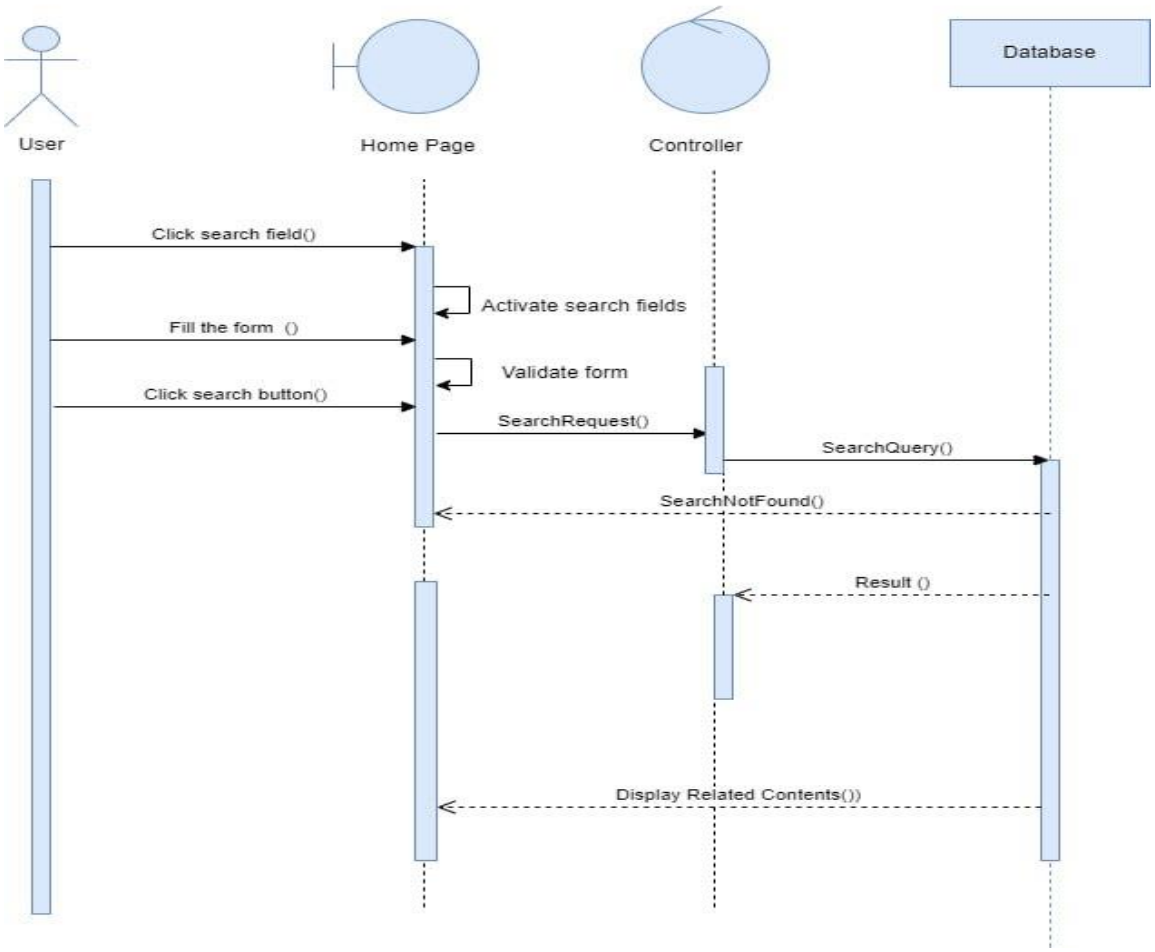


Figure 4. 4: Search Sequence Diagram

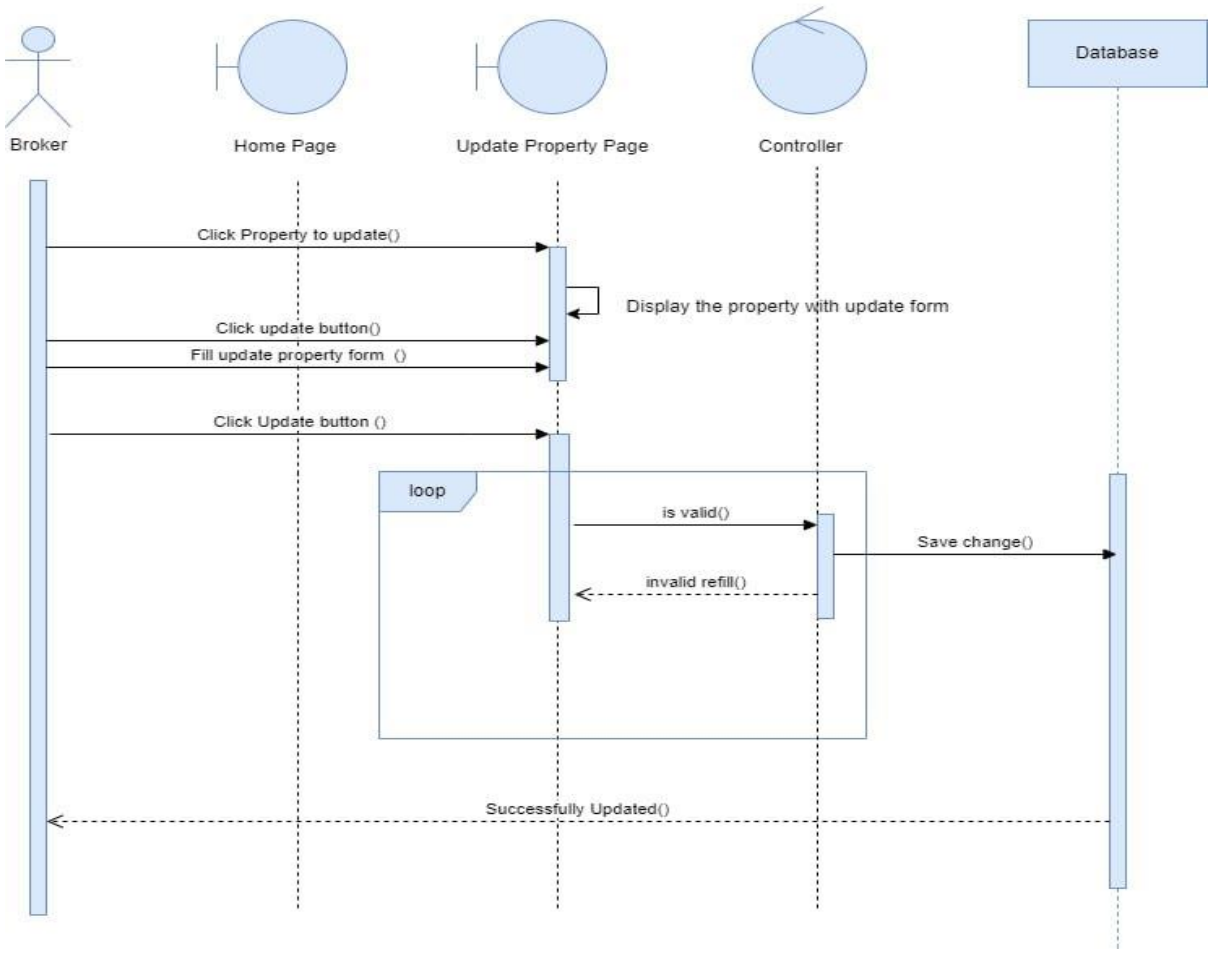


Figure 4. 5: update Sequence Diagram

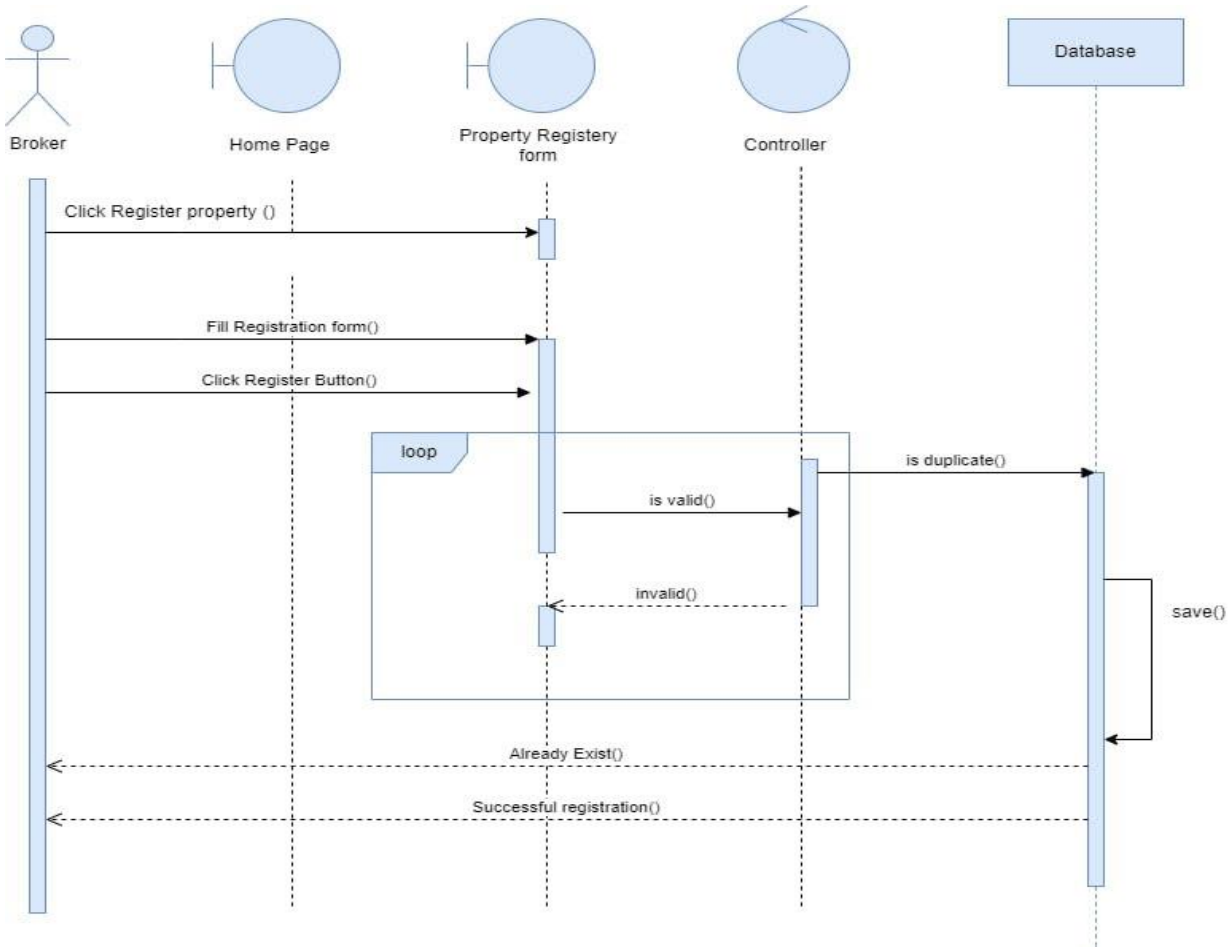


Figure 4. 6: Register Property Sequence Diagram

4.3.2. ACTIVITY DIAGRAM

Below figure shows in the system that visually represents the flow of activities, actions, and transitions within a system, process, or use case.

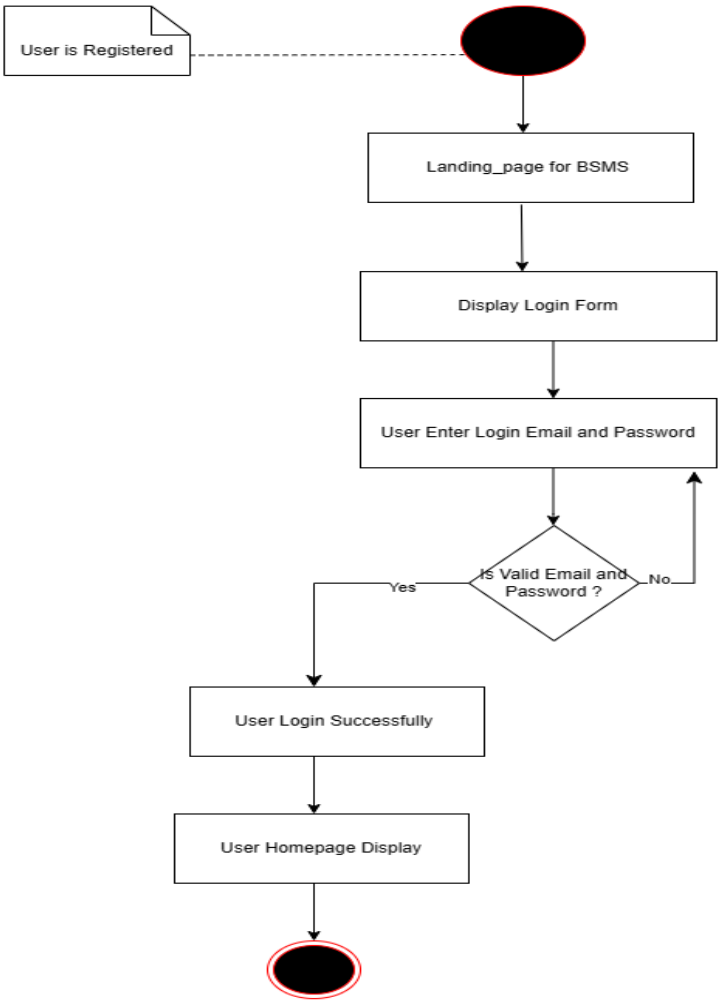


Figure 4. 7: *Login Activity Diagram*

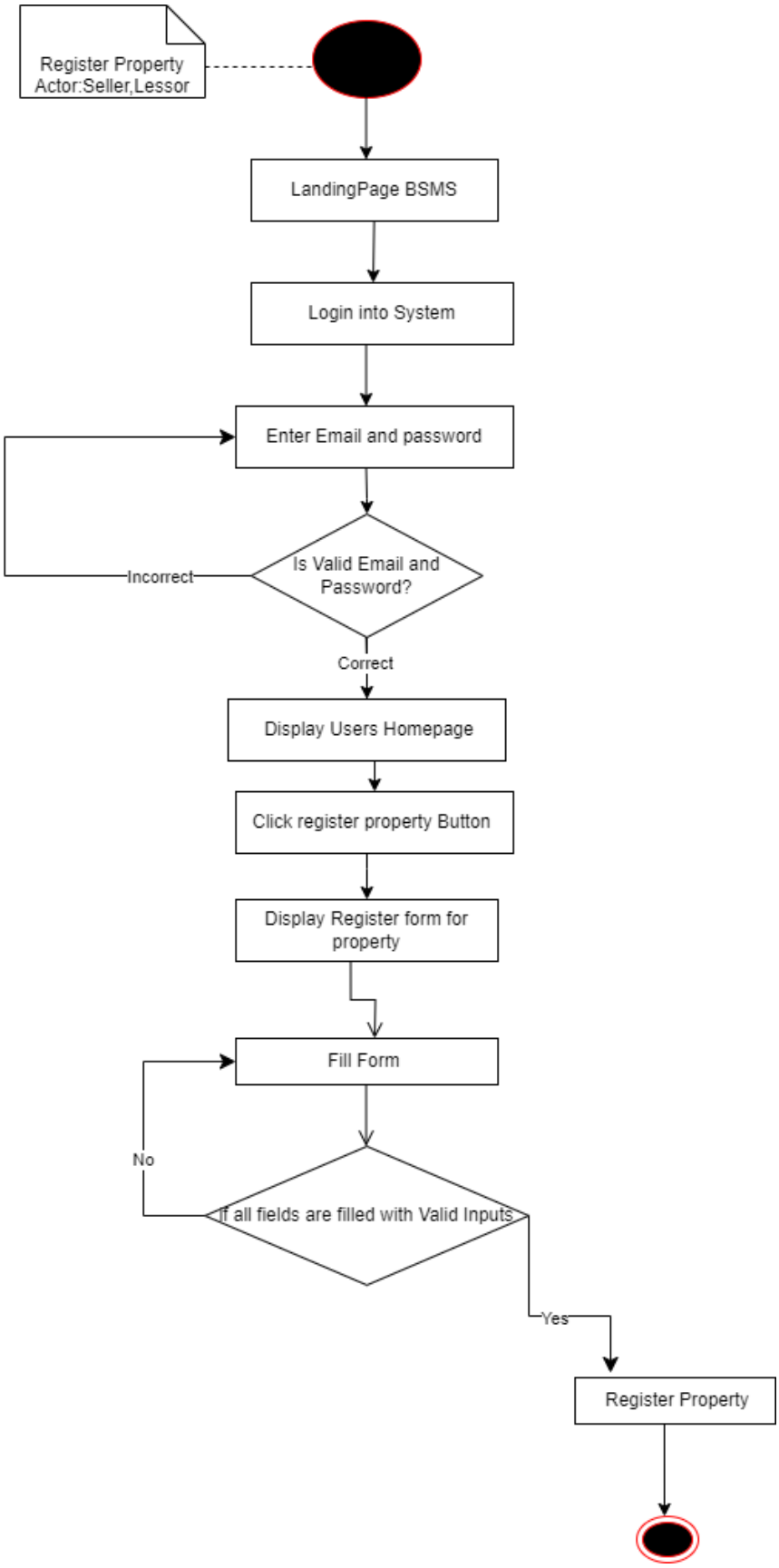


Figure 4. 8: Register property Activity Diagram

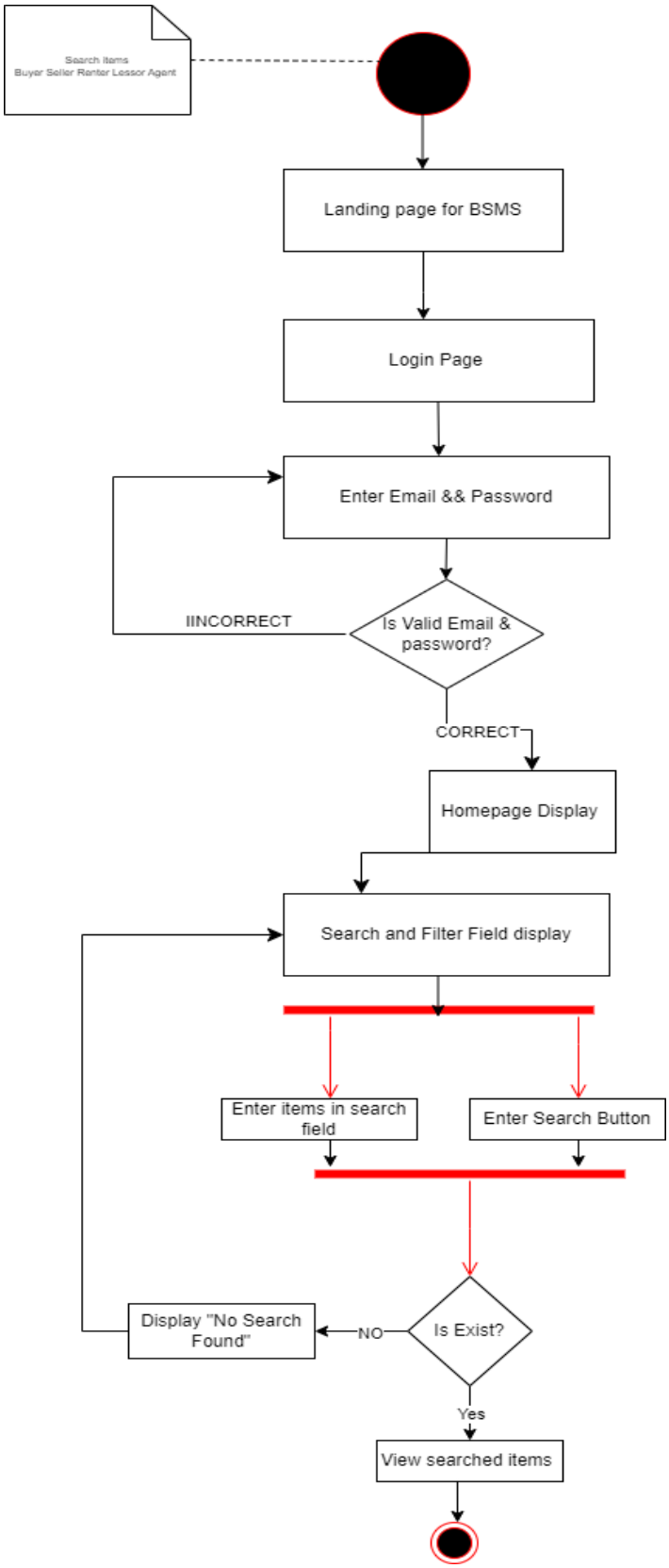


Figure 4. 9: Search & filter Activity Diagram

4.3.3. STATE CHART DIAGRAM

State Chart Diagram is system dynamic model used to model the dynamic behavior of a class in response to time and changing external stimuli. As shown below

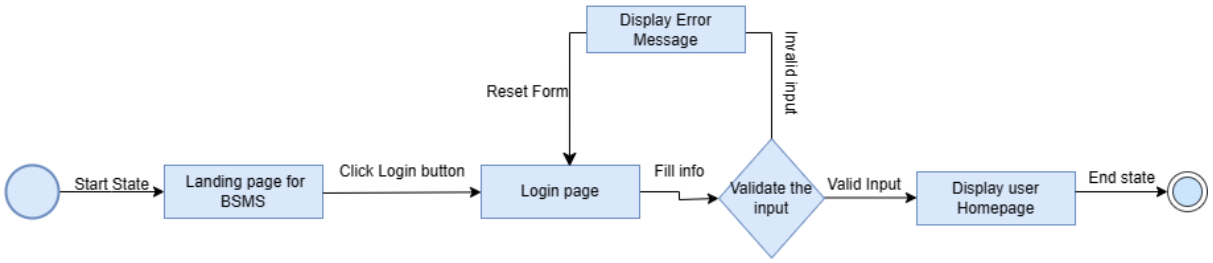


Figure 4. 10：Login state chart Diagram

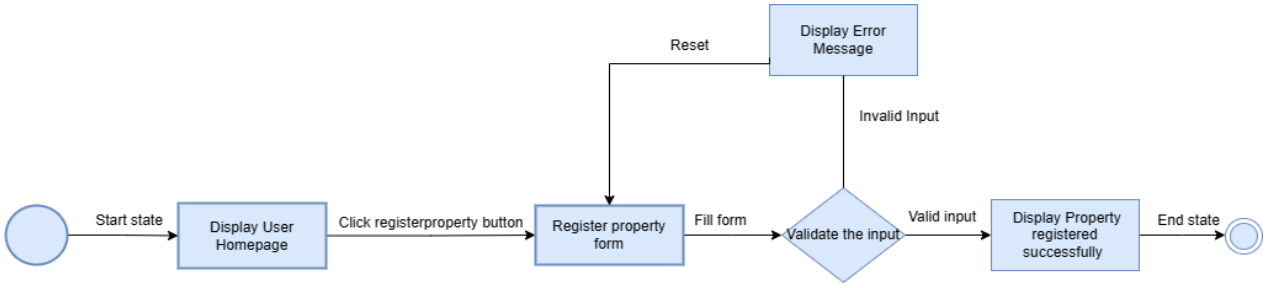


Figure 4. 11：register property state chart Diagram

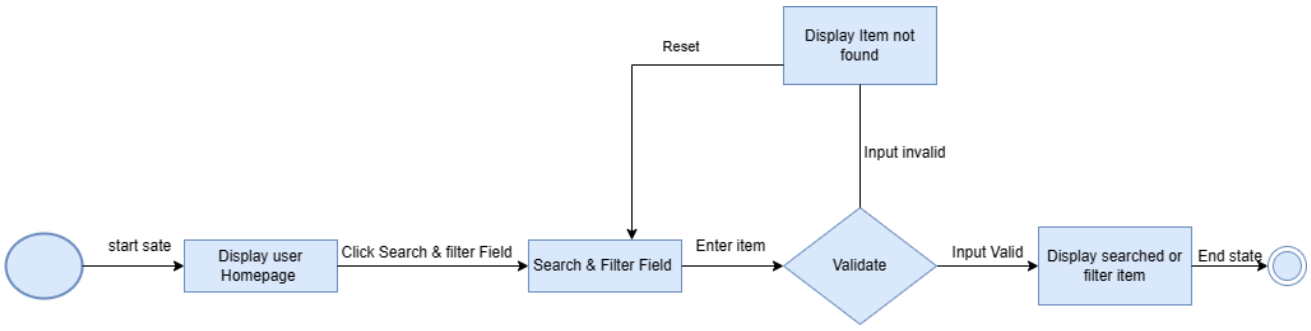


Figure 4. 12：search state chart Diagram

CHAPTER FIVE

5. SYSTEM DESIGN

System design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. The success of implementing a high-quality system depends on the nature of the design. If a system is well-designed, it becomes easier to make changes or adjustments once it's in use. The main aim of system design is to simplify and organize the complexity by breaking the system into smaller, more understandable parts. To design the system, we used various diagrams, such as: hard-ware software mapping, current and proposed software architecture, persistent data management, and access control and security.

5.1. DESIGN GOAL

User Interface and Human Factors:

This focuses on creating an intuitive and user-friendly interface for both brokers and users. It involves designing layouts, navigation, and features that enhance user experience. A user-friendly graphical user interface (GUI) will be provided for the system, enabling simple interactions with it. The proposed system interface will contain different buttons which guides the users on what they need to click and fill form to enter user's information.

Security Issues:

Security is a crucial goal, aiming to protect sensitive data and ensure the system is resilient against unauthorized access. The proposed security measures involve implementing robust authentication and authorization mechanisms to prevent unauthorized access. User interactions with the system, such as entering personal information, are facilitated through a secure interface with clear instructions provided by buttons and forms.

Performance

The following performance requirements will be met by the system.

Response time: The system, which we will fully create, will have a quick response time when users use it since we use MongoDB.

Throughput: The system will perform well in terms of throughput since we will concentrate on concurrency control approaches when implementing it and use MongoDB, which is excellent at handling locks and ensuring consistency.

Memory: Because we implement our analysis simply and make good use of memory, our system should operate in optimized RAM.

Error Handling

Errors could happen when a user interacts with the system. Different user-friendly messages will be generated by the system to control this kind of inaccuracy. To prevent these errors, we've organized the buttons in a way that follows how users typically use the system. We've also set up the system to respond differently based on what you do. We use a programming language called JavaScript to handle these issues and fix different types of errors.

Quality Issues

Availability: The system will be operational and able to provide users with relevant services every time the users need service

Reliability: The system minimize crash during its runtime, since more than one user could use the system simultaneously. This will be done by Using Validation Parameters for user input will be done, avoid incorrect storage of record

User operability: Simple navigation will be provided by the system, making it functional for any user with rudimentary computer skills.

Back Up and Recovery

backing up data is like making a copy of your important information and keeping it in a safe place. If something goes wrong, we use this copy to quickly recover everything. In our system, we use tools like mongo dump to take the backup and mongo restore to bring back the saved data. It's like having a spare key for your information, ensuring it stays safe even if there are any issues.

5.2. PROPOSED SYSTEM ARCHITECTURE

The proposed system is consisting of 3-tier architecture model. The top layer is where you, the user, interact with the system. This is called the presentation layer. It takes information when user type or click on things. The second layer or middle layer, known as the business logic layer. This layer is like the middleman, passing information between what you see (presentation layer) and the place where data is kept (data layer). It acts as the mediator between presentation layer and data layer. The last layer, the data layer, is where all the information you give gets stored. When you use the system through a browser, like on the internet, and type or click something, the middle layer makes sure your request is sent to the data layer where the information is stored.

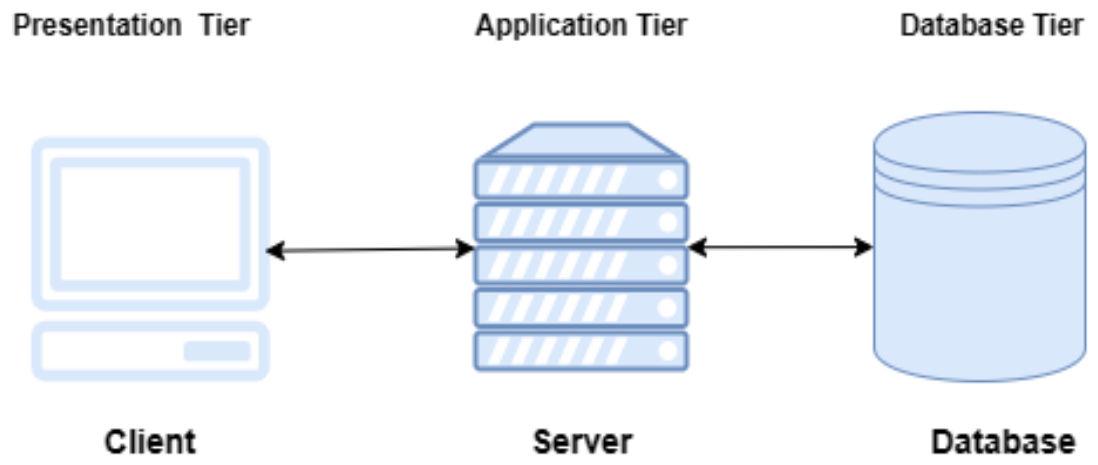


Figure 5 1: Proposed system architecture

5.2.1. SUBSYSTEM DECOMPOSITION AND DESCRIPTION

Subsystem decomposition is the process of breaking down the proposed system's analytical model into manageable components. The goal of decomposition is to divide the class of the system into substantial and coherent components while reducing the complexity of the design model. Typically, components in a system are either compute units or data storage units. A component has a name, which is typically chosen to reflect the function or role the component serves. While a system is being used, its many components are likely to interact to provide the service that is expected of it. It is specified by component diagram.

Subsystem Descriptions:

3. User Management Subsystem

The User Management Subsystem is a crucial component within the Broker Service Management System, which is responsible for the management of user-related activities. It comprises several classes, including Client, Broker Manager, Broker and System Admin.

- In Generally the following activities are done under this Subsystem
  - User Registration
  - User Authentication and Authorization
  - User Account Management
  - User Profile Management
  - Handling Users Role
  - Security and access control

4. Property Management Subsystem

The Property Management Subsystem serves as a main element within the system, focused on handling of property-related activities.

- The Following activities are done under this Subsystem
  - Property Listings Management
  - Property Search and Filtering
  - Property Listing Approval

5. Brokerage Management Subsystem

The Brokerage Management Subsystem serves as a foundational element within the system, specifically focused on coordinating the various functions associated with brokerage operations.

- This Subsystem performs the following activities
  - Broker Management
  - Client Management
  - Assign Broker to Property
  - Feedback and review management

6. Financial Management Subsystem

The Financial Management Subsystem within a Broker Service Management System handles the financial aspects and payment integration to brokerage operations.

- This subsystem encompasses several key activities
  - Handling Payment Process
  - Commission Management

7. The Chat Management Subsystem

The Chat Management Subsystem is a vital component within the system, responsible for managing activities related to chat functionalities.

- This Subsystem preforms the following activities
  - Users Conversation Management
  - Enable users to share Document

8. The Employee Management Subsystem

The Employee Management Subsystem is a fundamental module within the system, tasked with handling employee-related functionalities. It encompasses Employee class, that plays a specific role in ensuring the efficient management of employees.

- This Subsystem preforms the following activities
  - Managing employees and all their information's

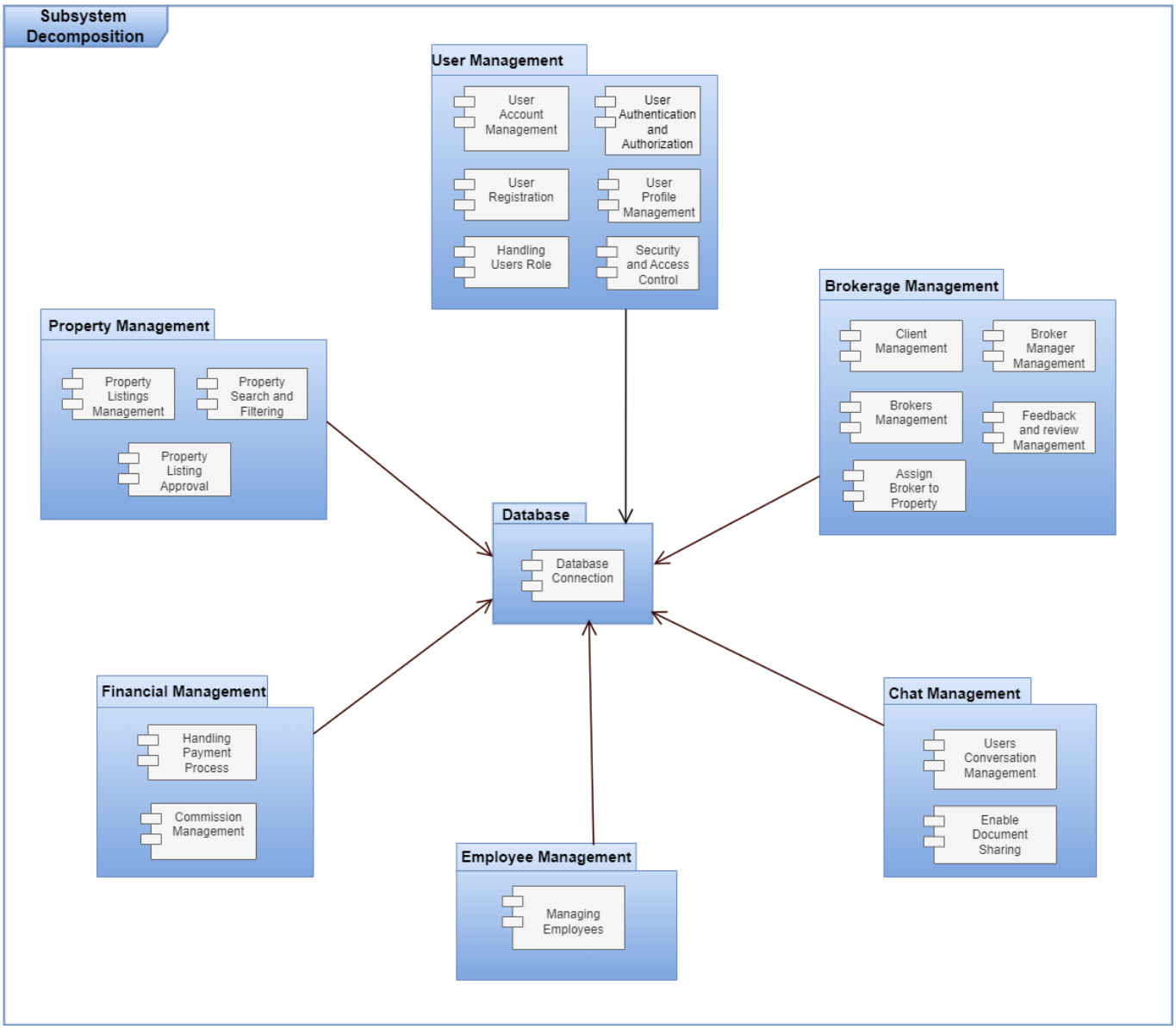


Figure 5 2: Component Diagram

5.2.2. HARDWARE/SOFTWARE MAPPING

Deployment diagram depicts a static view of the run-time configuration of processing nodes and the components that run on those nodes. In this mapping we are going to show the hard ware configuration during deployment of this proposed system. This is used to connect separate machines which are used for configuration purpose. Essentially, this diagram acts as a virtual bridge connecting separate machines for the explicit purpose of configuration. It's a visual road-map revealing how hardware and software seamlessly collaborate. The deployment diagram is not just a blueprint, it is a detailed tapestry, illustrating the Collaboration of hardware and software in action, each component playing a crucial role in the seamless operation of our proposed system.

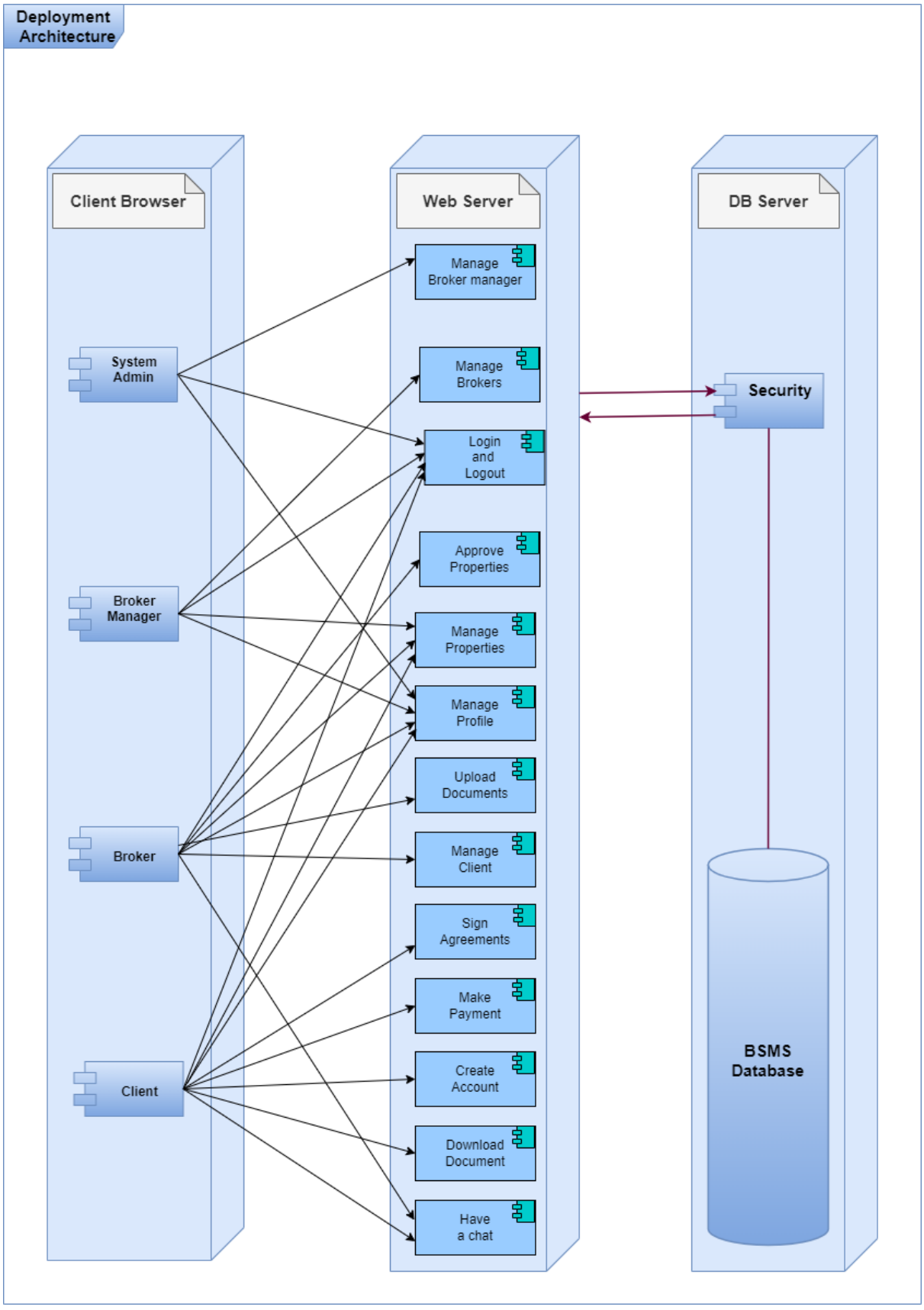


Figure 5 3: Deployment Diagram

5.2.3. DETAILED CLASS DIAGRAM

The class diagram is one of the types of UML diagrams which is used to represent the static diagram by mapping the structure of the systems using classes, attributes, relations, and operations between the various objects

In this section we are going to demonstrate classes, their attributes, methods and visibility of attributes and methods also the relationship between different classes.

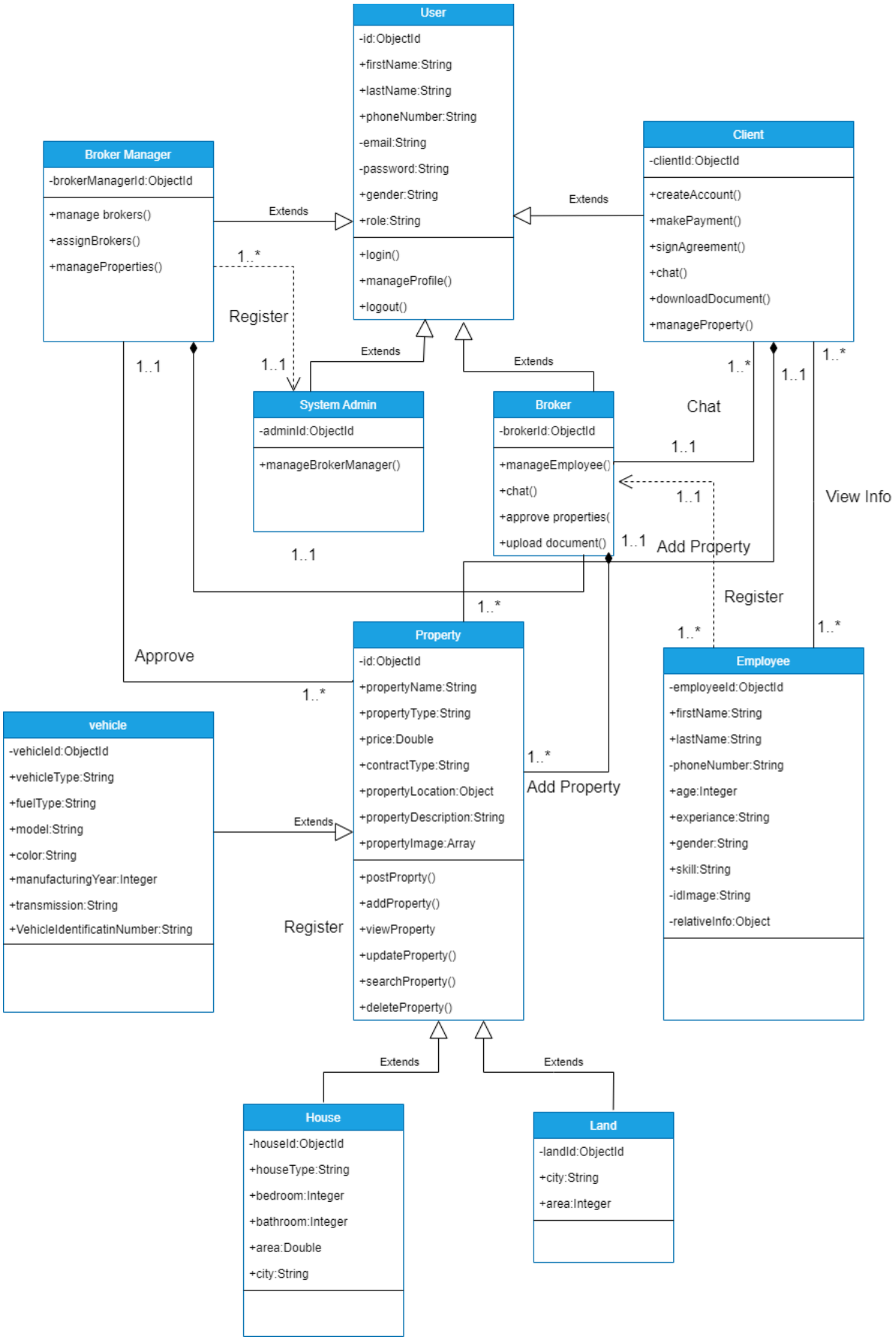


Figure 5 4: Detail Class Diagram

5.2.3. PERSISTENT DATA MANAGEMENT

Persistent data management within a Broker Service Management System refers to how the system store and handle data over time, describing both the storage of persistent data and the necessary data management.

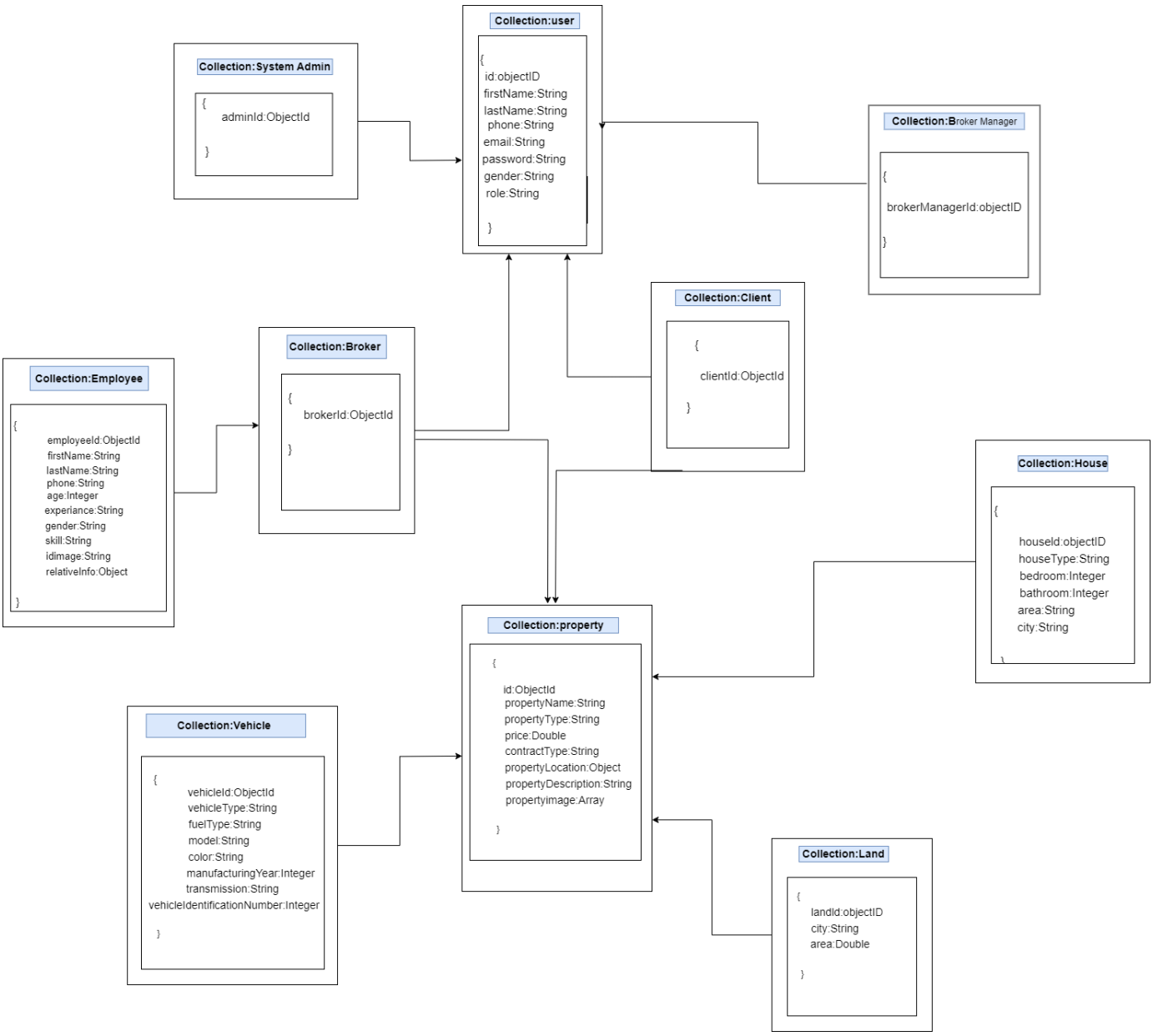


Figure 5 5: Persistent Data Management Diagram

5.2.4. ACCESS CONTROL AND SECURITY

Access control is a fundamental data security mechanism, used as a special lock for the information in a system. Access control and security main objective in the system is to check and confirms the identity of users and then decides what parts of the system they can use and what actions they're allowed to perform. This access control is verified by email and password. Then the system will have:

Authorization: Authorization adds an extra layer of security to the authentication process by specifying access rights and permissions.

Integrity: Ensures that data remains accurate and consistent which add limits on who can change the data in the system.

Confidentiality: Ensures that access to sensitive information is restricted to authorized individuals, preventing unauthorized access.

5.3. PACKAGES

Packages is used to represent the organization and arrangement of various elements within a system. This section explains how subsystems are broken down into packages and how the code is organized into files. This comprises a summary of every package, information on how it depends on other packages, and usage expectations.

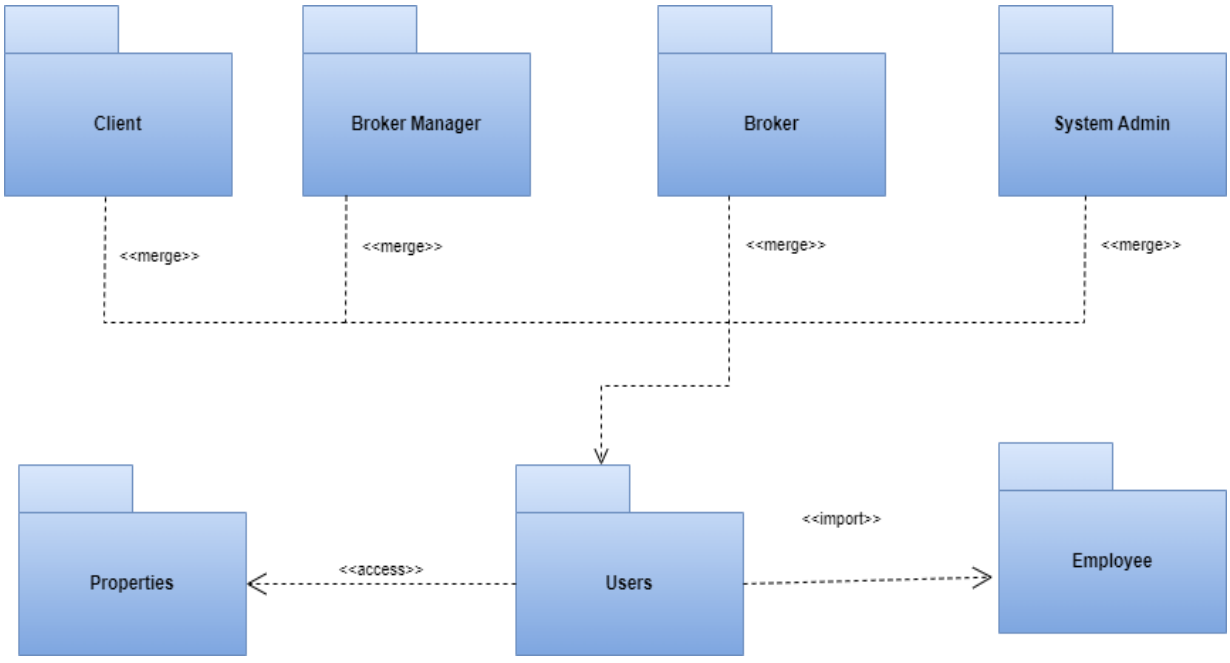


Figure 5 6: Package Diagram

5.4. ALGORITHM DESIGN

Algorithm design refers to the process of creating step-by-step procedures or sets of rules for solving a specific problem or accomplishing a particular task. Designing an algorithm for a broker service management system involves creating a set of procedures and rules to handle various tasks such as user authentication and authorization,

Login

- System displays Landing Page.
- Client clicks on the login option from the navigation menu.
- System redirects the client to the Login Page.
- Client enters email and password.
- Client clicks on login button

If email and password are correct, then the system displays the Home Page:

Else email and password are not correct, then the system displays the wrong email or password message and stays in the login page.

```
start
    Landing Page
    Click Login Option
    Display Login Page
    If(get.email===entered.email and get.password===entered.password)
        Display Home Page
    Else
        Display Login Failed Error Message
        Redirect Client to Login Page
end
```

Add Property

- System Display Client Home Page
- Client clicks on add property option from navbar options
- System redirect's client to property adding page
- Client fills the necessary information about the property
- Client clicks on submit button

If all required information filled fully and meet necessary condition display success message:

Else if information is not filled fully system display fill all information fully message:

Else if the information filled not fulfill the required criteria system display Error message

```
Start
    Client Home Page
    Client Clicks on Add Property Option from navbar options
    System Redirects Client to Property Adding Page
    Client Fills in the Necessary Information About the Property
    Client Clicks on Submit Button

    If (all required information is filled fully and meets necessary conditions)
        Display Success Message
    Else If (information is not filled fully)
        System Displays "Please Fill in All Information" Message
    Else If (the filled information does not fulfill the required criteria)
        System Displays Error Message
End
```

5.5. USER INTERFACE DESIGN

User Interface (UI) Design involves crafting the visual elements and interactive features that users interact with in our system. It's the process of creating interfaces that are not just visually appealing but also intuitive, functional, and user-friendly.

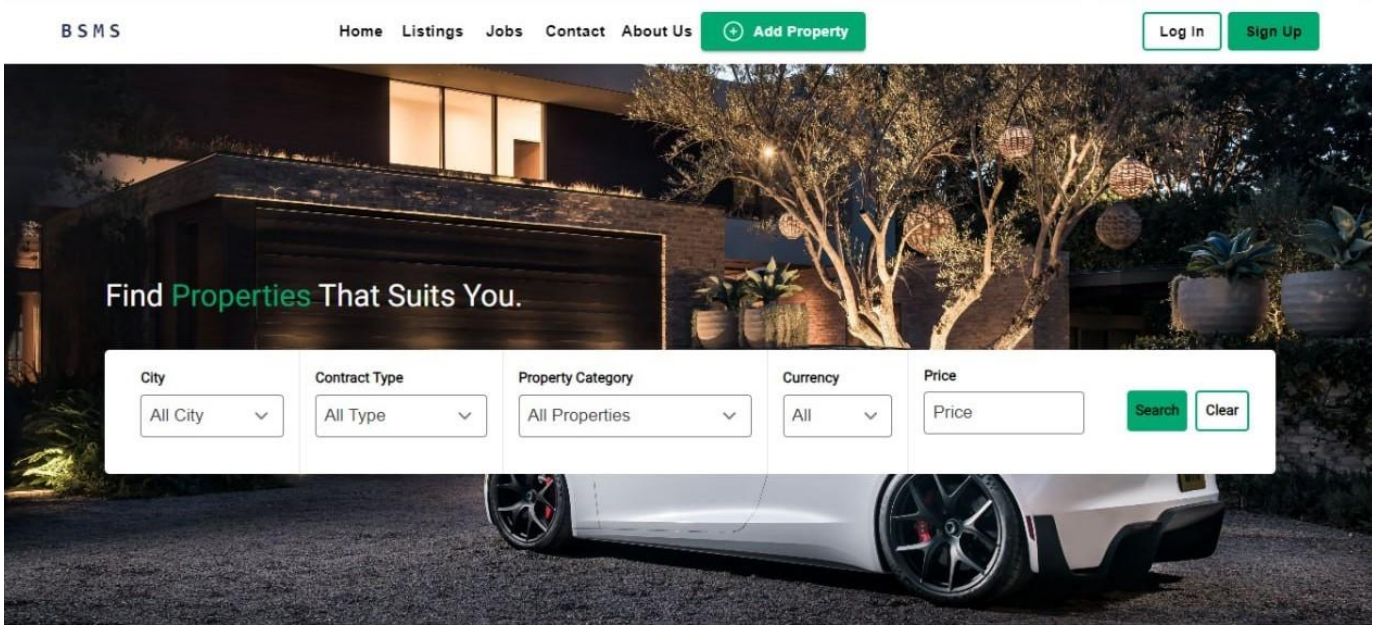


Figure 5 7: BSMS Landing Page Interface

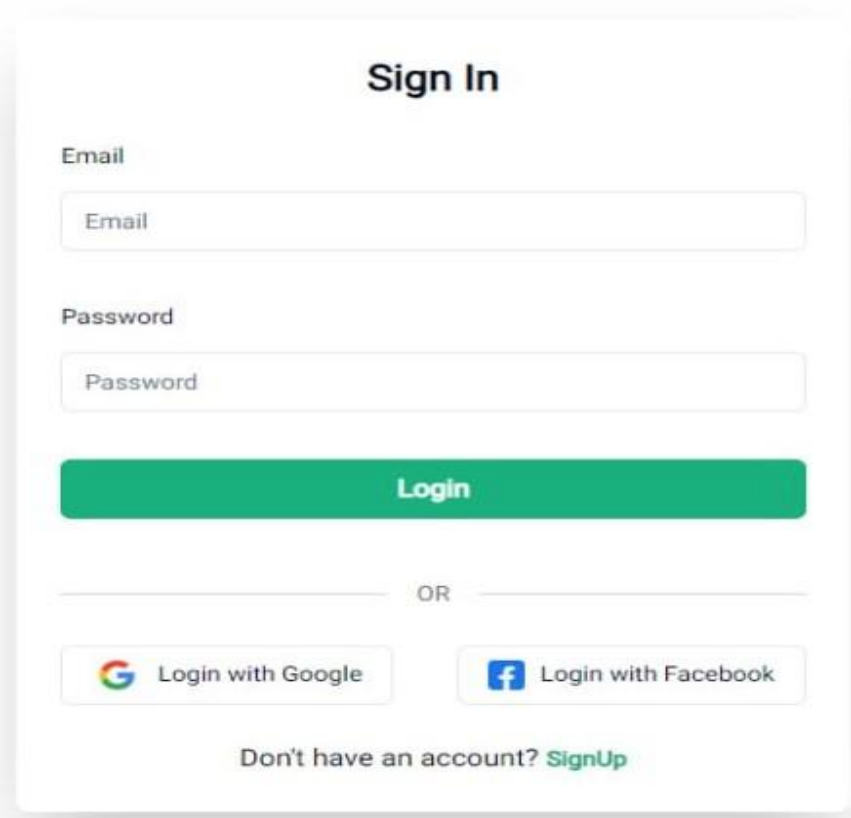


Figure 5 8: BSMS Login Page Interface

CHAPTER SIX

6. IMPLEMENTATION AND TESTING

In developing the Broker Service Management System (BSMS), MongoDB, Next.js, and Node.js are integral components of the tech stack. MongoDB serves as the database, efficiently storing data like property listings and user information. Node.js powers the backend, handling data processing and communication with MongoDB through RESTful APIs. Next.js drives the frontend, providing a seamless user interface and optimizing performance with features like server-side rendering. Throughout development, rigorous testing ensured the reliability and functionality of the BSMS, covering unit tests for individual components, integration tests for frontend-backend interactions, and end-to-end tests to validate the system's behaviour. This cohesive approach to implementation and testing ensures a robust and user-friendly platform for brokerage management.

6.1. IMPLEMENTATION OF DATABASE

For the Broker Service Management System (BSMS), MongoDB was selected as the database management system (DBMS) due to its flexibility and scalability. we structured collections to accommodate the document-oriented nature of MongoDB. Each document includes a unique _id field, serving as its primary key. Instead of foreign keys, we utilized references. Indexes were implemented for efficient query performance, and database backups were scheduled for data protection. role-based access control was configured to safeguard user ensuring confidentiality and compliance. We implemented separate database structures for users, properties including houses, land, and cars.

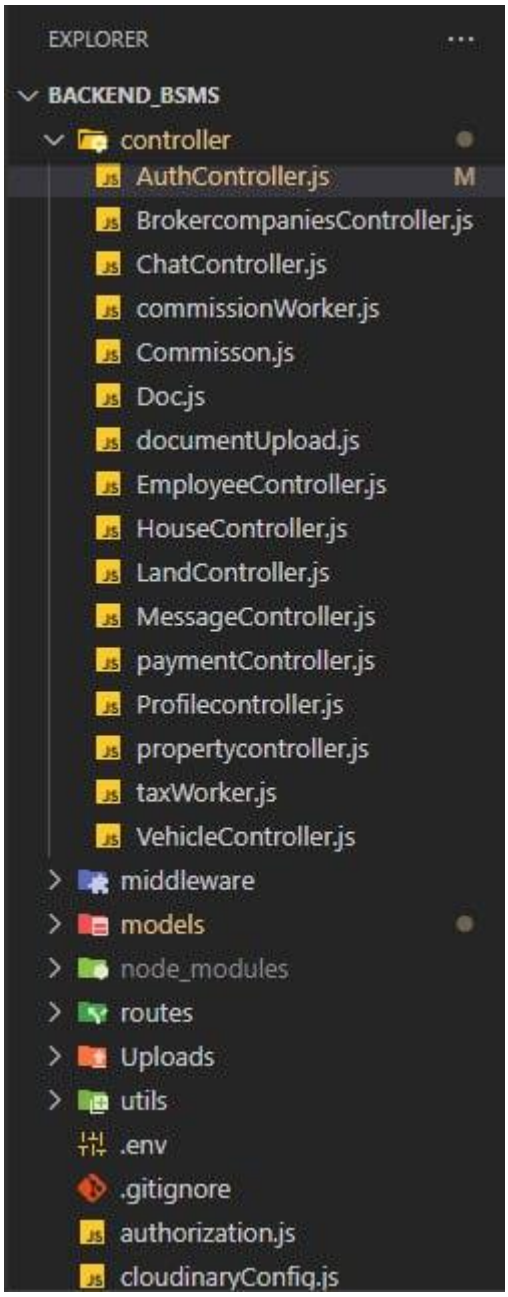
The mongodump and mongorestore utilities are typically located in the bin directory of the MongoDB installation folder. We executed the mongodump command to create a backup of to our database.

mongodump --db users --out C:/Users/Robot/Desktop

mongorestore –db users –dir C:/Users/Robot/Desktop/users

6.2. IMPLEMENTATION OF THE CLASS DIAGRAM

For the implementation of the class diagram, we follow the MVC (Model-View-Controller) architectural pattern to organize our code and ensure separation of concerns.



Sample codes for Property Listing

```
import React from "react"; // Import React library
import Card from "@/components/propertyList/Card"; // Import the Card
component
import { Button } from "@mui/material"; // Import Button component from
Material-UI
import Link from "next/link"; // Import Link component from Next.js

const Listings = () => {
  // Define Listings component
  const [latestProperties, setLatestProperties] = React.useState([]); // State
for latest properties
```

```
  const [loading, setLoading] = React.useState(true); // State for loading
indicator

  React.useEffect(() => {
    // Effect hook to fetch data when component mounts
    const fetchListings = async () => {
      try {
        // Try fetching data from API endpoint
        const response = await fetch(
          `http://localhost:3001/api/Allproperty/all`, // API endpoint URL
          {
            method: "GET", // HTTP GET request
            headers: {
              "Content-Type": "application/json", // JSON content type
            },
          }
        );
        if (!response.ok) {
          // If response is not OK, throw an error
          throw new Error("Network response was not ok");
        }

        // Parse response data as JSON
        const data = await response.json();
        // Set latestProperties state with fetched data
        setLatestProperties(data.data);
      } catch (error) {
        // If an error occurs during fetch, log it and set loading to false
        console.error("Error:", error);
        setLoading(false);
      }
    };
    fetchListings(); // Call fetchListings function
  }, []); // Empty dependency array means this effect runs only once on mount

  return (
    <div className="text-center">
      {/* Render title */}
      <h4 className="text-black text-3xl mt-10 font-light">Latest
Listings</h4>
      {/* Render property cards */}
      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 ml-12">
        {latestProperties.map((property, index) => (
          <Card property={property} /> // Render Card component for each
property
        ))}
      </div>
      {/* Render Load More button with Link to /listings page */}
```

```
      <Link href="/listings">
        <Button
          variant="outlined"
          className="mb-7 px-8 text-black capitalize hover:font-bold border-
green hover:border-green"
        >
          Load More
        </Button>
      </Link>
    </div>
  );
};

export default Listings; // Export Listings component
```

Samples codes for House Controller

```
const cloudinary = require("cloudinary").v2; // Import Cloudinary SDK
const House = require("../models/House"); // Import House model
const User = require("../models/Users"); // Import User model
const streamifier = require("streamifier"); // Import streamifier library for
stream handling
const jwt = require("jsonwebtoken"); // Import JWT for token handling
const NotificationService = require("../utils/notificationservice"); // Import
NotificationService

// Configure Cloudinary
cloudinary.config({
  cloud_name: "ds3wsc8as",
  api_key: "714722695687768",
  api_secret: "iTi78ih5itaEnbiFF8oc7raVbvw",
});

// Function to upload images
const uploadImages = async (req, res) => {
  try {
    // Parse location data from request body
    const locationString = req.body.Location;
    const locationObject = JSON.parse(locationString);
    console.log("Latitude:", locationObject.lat);
    console.log("Longitude:", locationObject.lng);

    // Array to store image URLs and document URLs
    const imageUrls = [];
    const documentUrls = [];
```

```
    // Check if files were uploaded
    if (!req.files || req.files.length === 0) {
      return res.status(400).json({ success: false, error: "No files
uploaded." });
    }

    // Check if number of files exceeds limit
    if (req.files.length > 4) {
      return res.status(400).json({ success: false, error: "Maximum of 3 files
allowed." });
    }

    // Upload images to Cloudinary
    const uploadPromises = req.files
      .filter((file) => {
        const allowedTypes = ["image/jpeg", "image/jpg", "image/png"];
        return allowedTypes.includes(file.mimetype);
      })
      .map((file) => {
        return new Promise((resolve, reject) => {
          if (file.size > 10485760) {
            // Reject if file size exceeds limit
            reject({
              success: false,
              error: `File ${file.originalname} is too large. Maximum size is
10 MB.`,
            });
          } else {
            // Upload image to Cloudinary
            const folder = "Houses";
            const stream = cloudinary.uploader.upload_stream({ resource_type:
"auto", folder: folder }, (error, result) => {
              if (error) {
                // Reject if upload fails
                console.error("Error uploading to Cloudinary:", error);
                reject({ success: false, error: "Error uploading to
Cloudinary" });
              }
              // Store secure URL of uploaded image
              imageUrls.push(result.secure_url);
              resolve();
            });
            // Pipe file stream to Cloudinary upload stream
            streamifier.createReadStream(file.buffer).pipe(stream);
          }
        });
      });
```

```
    // Await all image upload promises
    await Promise.all(uploadPromises);

    // Extract user email from JWT token
    const token = req.headers.authorization;
    if (!token) {
      return res.status(401).json({ success: false, error: "Token is missing."
});
    }
    const decodedToken = jwt.verify(token.split(" ")[1], "AZQ,PI)0(");
    if (!decodedToken) {
      return res.status(401).json({ success: false, error: "Invalid token."
});
    }
    const userEmail = decodedToken.Email;

    // Create new House object with uploaded data
    const newHouse = new House({
      Title: req.body.title,
      Location: locationObject,
      ContractType: req.body.ContractType,
      PropertyType: req.body.PropertyType,
      PropertyCategory: req.body.PropertyCategory,
      PriceCategory: req.body.PriceCategory,
      Currency: req.body.Currency,
      Bedroom: req.body.Bedrooms,
      Bathroom: req.body.Bathrooms,
      Area: req.body.Area,
      City: req.body.City,
      Description: req.body.description,
      Price: req.body.Price,
      UploadedBy: userEmail,
      imageUrls: imageUrls,
      documentUrls: documentUrls,
    });

    // Save new house to database
    const savedHouse = await newHouse.save();

    // Return success response with saved house data
    res.json({ success: true, message: "House Information added Successfully",
data: { newHouse: savedHouse } });
  } catch (error) {
    // Handle errors
    console.error("Server error:", error);
    if (!res.headersSent) {
      res.status(500).json({ success: false, error: "Internal Server Error"
});
```

```
    }
  }
};

// Function to retrieve all houses
const showHouse = (req, res, next) => {
  House.find()
    .then((response) => {
      res.json({ response });
    })
    .catch((error) => {
      res.json({ message: "An error Occurred!" });
    });
};

// Function to delete a property
const deleteProperty = async (req, res) => {
  try {
    const { propertyId } = req.body;
    const deletedProperty = await House.findByIdAndDelete(propertyId);
    if (!deletedProperty) {
      return res.status(404).json({ success: false, error: "Property not
found" });
    }
    res.json({ success: true, message: "Property deleted successfully",
deletedProperty });
  } catch (error) {
    console.error("Server error:", error);
    if (!res.headersSent) {
      res.status(500).json({ success: false, error: "Internal Server Error"
});
    }
  }
};

// Function to update a property
const updateProperty = async (req, res) => {
  try {
    const { propertyId } = req.params;
    const updatedData = {
      ContractType: req.body.ContractType,
      HouseType: req.body.HouseType,
      Bedroom: req.body.Bedroom,
      Bathroom: req.body.Bathroom,
      Area: req.body.Area,
      Location: req.body.Location,
      City: req.body.City,
      Description: req.body.Description,
```

```
      Price: req.body.Price,
    };
    if (req.files && req.files.length > 0) {
      // Upload new images if provided
      const newImageUrls = [];
      const uploadPromises = req.files.map((file) => {
        return new Promise((resolve, reject) => {
          if (file.size > 10485760) {
            reject({
              success: false,
              error: `File ${file.originalname} is too large. Maximum size is
10 MB.`,
            });
          }
          const folder = "Houses";
          const stream = cloudinary.uploader.upload_stream({ resource_type:
"auto", folder: folder }, (error, result) => {
            if (error) {
              console.error("Error uploading to Cloudinary:", error);
              reject({ success: false, error: "Error uploading to Cloudinary",
file: file.originalname });
            }
            newImageUrls.push(result.secure_url);
            resolve();
          });
          streamifier.createReadStream(file.buffer).pipe(stream);
        });
      });
      await Promise.all(uploadPromises);
      updatedData.imageUrls = newImageUrls;
    }
    // Update property in database
    const updatedProperty = await House.findByIdAndUpdate(propertyId,
updatedData, { new: true });
    if (!updatedProperty) {
      return res.status(404).json({ success: false, error: "Property not
found." });
    }
    res.json({ success: true, message: "Property updated successfully", data:
updatedProperty });
  } catch (error) {
    console.error("Server error:", error);
    if (!res.headersSent) {
      res.status(500).json({ success: false, error: "Internal Server Error",
data: null });
    }
  }
};
```

6.3. CONFIGURATION OF APPLICATION SERVER

To configure the application server, we execute "npm run build" for the front end and "npm start" for the back end.where all the necessary files and assets for the front end are compiled, optimized, and bundled together. This streamlined process ensures efficient handling of user requests and seamless interaction within the Broker Service.

```
const express = require("express");
var bodyParser = require("body-parser");
const dotenv = require("dotenv");
const mongoose = require("mongoose");
const url = "mongodb://0.0.0.0:27017/User";
const app = express();
dotenv.config();

mongoose
  .connect(url, {})
  .then((_result) => console.log("database connected"))
  .catch((err) => console.log(err));


const AuthRoute = require("./routes/auth");
const docRoute = require("./routes/Doc");
const HouseRoutes = require("./routes/HouseRoutes");
const brokerCompaniesRoutes = require("./routes/brokerCompaniesRoutes");
const vehicleRoutes = require("./routes/vehicleRoutes");
const landRoutes = require("./routes/landRoutes");
const employee = require("./routes/employeeRoutes");
const Message = require("./routes/MessageRoute");
const Chat = require("./routes/ChatRoute");
const paymentRoute = require("./routes/paymentRoute");
const employeerelative = require("./routes/employeeRoutes");
const Allproperty = require("./routes/Allproperty");
const commissionroutes = require("./routes/commissionroutes");
const cors = require("cors");

const corsOptions = {
  origin: "*",
  credentials: true,
  optionSuccessStatus: 200,
};

app.use(cors(corsOptions));
app.use(express.json({ limit: "50mb" }));
app.use("/api/docs", docRoute);
app.use("/api/User", AuthRoute);
app.use("/api/payment", paymentRoute);
```

```
app.use("/api/House", HouseRoutes);
app.use("/api/Vehicle", vehicleRoutes);
app.use("/api/Land", landRoutes);
app.use("/api/Employee", employee);
app.use("/api/Message", Message);
app.use("/api/Chat", Chat);
app.use("/api/brokerCompanies", brokerCompaniesRoutes);
app.use("/api/Allproperty", Allproperty);
app.use("/api/commission", commissionroutes);

app.listen(3001, () => {
  console.log("Node API App is running on port 3001 ");
});
```

6.4. CONFIGURATION OF APPLICATION SECURITY

During development, we integrated validation techniques like Formik for ensuring data integrity and accuracy in user inputs. Additionally, we implemented encryption and decryption functionalities using libraries like bcrypt to secure sensitive information such as user passwords. These measures are crucial for maintaining the confidentiality and integrity of user data within the system. Additionally, we incorporated essential security features like RBC (Role-Based Access Control) and JSON Web Token (JWT) authentication. RBC ensures that access to system resources and functionalities is based on predefined roles, such as admin, broker, or user. With RBC, we enforce granular permissions to restrict unauthorized access and maintain data confidentiality and integrity.

Additionally, JWT authentication provides a secure method for transmitting authentication credentials between the frontend and backend. When a user logs in, the server generates a JWT token containing user information and signs it with a secret key.

6.5. TESTING

Testing is a crucial aspect of software development that plays a pivotal role in verifying the correctness, reliability, and functionality of a system. In the context of the Broker Service Management System (BSMS), thorough testing is essential to ensure that the system meets the requirements and expectations of its users.

6.5.1. TEST CASE

Test Case 1:

- ✓ Test Case Name: Verify User Registration
- ✓ Description: Test the user registration functionality to ensure users can successfully register with valid credentials.
- ✓ Inputs: User details (name, email, password)
- ✓ Expected Output: User account created successfully
- ✓ Steps to Reproduce:

    1. Navigate to the registration page.
    2. Enter valid user details in the registration form.
    3. Click the "Register" button.
    4. Verify that the user account is created successfully and a confirmation message is displayed.

Test case 2:

- ✓ Test Case Name: Property Registration Functionality
- ✓ Description: Verify that the property registration process functions correctly, allowing users to register new properties in the Broker Service Management System (BSMS).
- ✓ Inputs: Property Details (Property title, Location (latitude and longitude), Contract type and so on)
- ✓

Steps:

1. Navigate to Property Registration.
2. Input Property Details.
3. Submit Registration.
4. Verify Success.

Expected:

Successful registration confirmation.

Property visible in dashboard.

Pass Criteria:

All steps executed without errors.

Fail Criteria:

- ✓ Error in any step.
- ✓ No registration confirmation.
- ✓ Property not visible in dashboard.

6.5.2. TESTING TOOLS AND ENVIRONMENT

Testing Tools:

- ✓ Jasmine: Used for unit testing to validate individual components and functions.
- ✓ Postman: Utilized for integration testing to test API endpoints and ensure proper communication between different modules.

Testing Environment:

- ✓ Operating System: Windows 10

6.5.3. UNIT TESTING

Unit testing in the Broker Service Management System involves testing individual components and modules to ensure they function correctly in isolation using jasmine frameworks. making our system stronger and more reliable. This method helps us find and fix any problems early on, making sure our system runs smoothly.

```
Login Test Case
  ✓ return a valid authentication token for valid credentials
  ✓ return an error for invalid credentials

2 specs, 0 failures
```

```
User Registration API
  ✓ Successfully register a new user with valid details
  ✓ return an error for registering with an existing email address
  ✓ return an error for registering with invalid email format
  ✓ return an error for registering with a weak password
```

```
4 specs, 0 failures
```

6.5.4. INTEGRATION TESTING

Integration testing in the Broker Service Management System involves assessing the interaction between individual modules to ensure they function together seamlessly. By using tools Postman, API endpoints are tested to verify data flow and communication between different parts of the system.

6.5.5. ACCEPTANCE TESTING

For acceptance testing in the Broker Service Management System, we employ real users to interact with the system and assess its functionality, usability, and compliance with requirements. This ensures that the BSMS meets user expectations and is ready for deployment in real-world scenarios.

CHAPTER SEVEN

7. CONCLUSION AND RECOMMENDATION

7.1. CONCLUSION

In conclusion, the Broker Service Management System represents a significant advancement in the brokerage industry, offering a streamlined and efficient platform for connecting buyers, sellers, lessors, lessees, and brokers. Rooted in the objective of enhancing communication and interaction, the BSMS resolves existing system issues by offering a user-friendly web-based application encompassing functions such as information management, communication tools, payment processing, and property management.

Built with MongoDB, React, Next.js, and Node.js, the BSMS ensures data integrity, scalability, and security while offering a responsive and intuitive user experience. Rigorous testing using tools like Jasmine for unit testing and post man for integration testing. The BSMS sets itself apart with its precision and effectiveness in meeting the unique needs of brokers and clients, emphasizing simplicity and efficiency.

The BSMS revolutionizes communication and interaction between brokers or agents and their clients, emphasizing simplicity and efficiency in its design. Unlike other projects, it stands out for its precision and effectiveness in meeting the specific needs of brokers and clients alike. Looking ahead, opportunities for future project works include ongoing refinement and adaptation to evolving industry demands, ensuring the BSMS remains a leader in brokerage innovation.

7.2. RECOMMENDATION

To further enhance the effectiveness of the Broker Service Management System, the following recommendations are suggested:

- ✓ Implement a process for collecting feedback from users regularly to identify areas for improvement and incorporate new features or enhancements to meet evolving needs.
- ✓ We highly recommend comprehensive training and ongoing support to brokers and users to ensure they can fully utilize the system's features and maximize its benefits.
- ✓ We are highly recommended to explore the integration of artificial intelligence algorithms to provide users with predictive insights and recommendations based on their usage patterns and transaction history.

- ✓ We are highly recommended opportunities for strategic partnerships with other industry stakeholders, such as real estate agencies or financial institutions, to expand the reach and capabilities of the BSMS ecosystem.
- ✓ We recommended ensure accessibility for people with disabilities is essential for creating an inclusive and equitable user experience in the Broker Service Management System (BSMS)

REFERENCES

Sommerville, I. (2011). *Software Engineering: A Practitioner's Approach.* Addison-Wesley.

Doe, J. B. (2020). *Programming Paradigms: Exploring the World of Code.* TechBooks Publishing.

Martin, R. C. (2002). Agile Software Development: Principles, Patterns, and Practices. Pearson Education.

Educba. (n.d.). Class Diagram. EDUCBA. https://www.educba.com/class-diagram/

Miro. (n.d.). Package Diagram. Miro. https://miro.com/diagramming/what-is-a-uml-package-diagram/

Lucidchart. (n.d).UML package Diagram.Lucidchart. https://www.lucidchart.com/pages/uml-package-diagram

Kung-Kiu Lau, Jeff Kramer, & Jeff Magee. (1998). UML Component Diagrams and Software Architecture - Experiences from the Wren Project. https://www.researchgate.net/publication/2378060_UML_Component_Diagrams_and_Software_Architecture_-_Experiences_from_the_Wren_Project

Baeldung. (n.d.). Requirements: Functional vs. Non-functional. Baeldung. https://www.baeldung.com/cs/requirements-functional-vs-non-functional

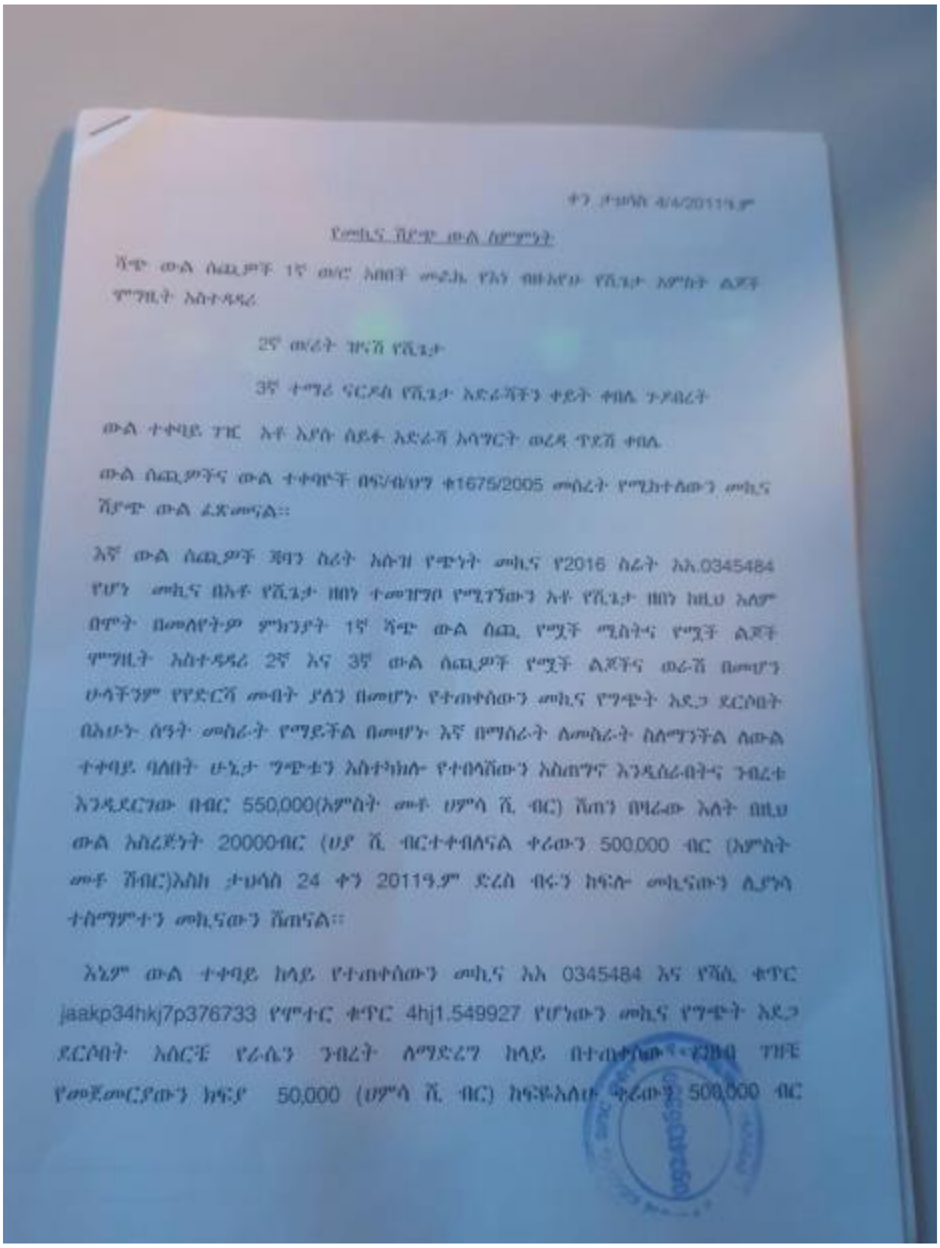
Rudderstack (n.d.). Data Security.Rudderstack https://www.rudderstack.com/learn/data-security/what-is-persistent-data/

APPENDIX

Appendix I: Interview Questions

1. What is your name and role in the company?
2. What kind of properties do you sell or rent?
3. How do you find properties that you sell or rent?
4. How do you find customers that can buy or rent?
5. How do you conduct the commission?
6. How do you communicate with customers?
7. What are specific property details that you look for?
8. What are legal issues that need to be considered in the current system?
9. How do you handle job related issues?
10. What is the main problem that you are facing in the current system?
11. How does the payment mechanism look like?
12. How do you conduct inspection?
13. How do you feel the current manual system is changed to computerized system?
14. What kind of functionalities do you want to be included into this new system?

Appendix II: Existing System Forms and Reports



ቀን ታህሳስ 4/4/2011ዓ.ም

የመኪና ሽያጭ ውል ስምምነት

ሻጭ ውል ሰጪዎች 1ኛ ወ/ሮ አበበች መረሳ የእነ ብዙአየሁ የሺጌታ አምስት ልጆች ሞግዚት አስተዳዳሪ

2ኛ ወ/ሪት ዝናሽ የሺጌታ

3ኛ ተማሪ ናርዶስ የሺጌታ አድራሻችን ቀድት ቀበሌ ጉዶበረት

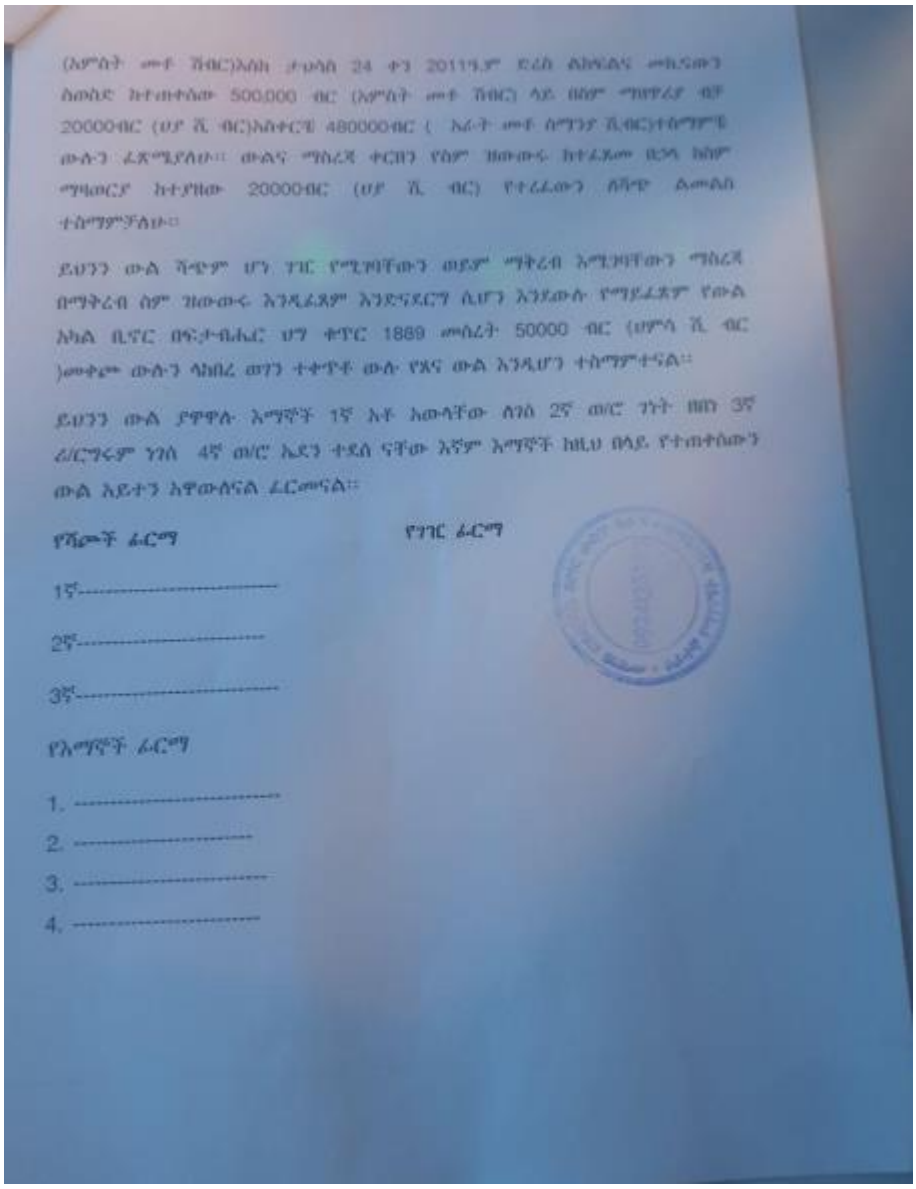
ውል ተቀባይ ገዢ አቶ እያሱ ሰይፉ አድራሻ አባግርት ወረዳ ጥደሽ ቀበሌ

ውል ሰጪዎችና ውል ተቀባዮች በፍ/ብ/ህግ ቁ1675/2005 መሰረት የሚከተለውን መኪና ሽያጭ ውል ፈጽመናል።

እኛ ውል ሰጪዎች ጃባን ስሪት አሱዝ የጭነት መኪና የ2016 ስሪት አአ.0345484 የሆነ መኪና በአቶ የሺጌታ ዘበን ተመዝግቦ የሚገኘውን አቶ የሺጌታ ዘበን ከዚህ አለም በሞት በመለየትም ምክንያት 1ኛ ሻጭ ውል ሰጪ የሟች ሚስትና የሟች ልጆች ሞግዚት አስተዳዳሪ 2ኛ እና 3ኛ ውል ሰጪዎች የሟች ልጆችና ወራሽ በመሆን ሁላችንም የየድርሻ መብት ያለን በመሆኑ የተጠቀሰውን መኪና የግጭት አደጋ ደርሶበት በአሁኑ ሰዓት መስራት የማይችል በመሆኑ እኛ በማሰራት ለመስራት ስለማንችል ለውል ተቀባይ ባለበት ሁኔታ ግጭቱን አስተካክሎ የተበላሸውን አስጠግኖ እንዲሰራበትና ንብረቱ እንዲያደርገው በብር 550,000(አምስት መቶ ሀምሳ ሺ ብር) ሸጠን በዛሬው እለት በዚህ ውል አስረጅነት 20000ብር (ሀያ ሺ ብር)ተቀብለናል ቀሪውን 500,000 ብር (አምስት መቶ ሺብር)እስከ ታህሳስ 24 ቀን 2011ዓ.ም ድረስ ብሩን ከፍሎ መኪናውን ሊያነሳ ተስማምተን መኪናውን ሸጠናል።

እኔም ውል ተቀባይ ከላይ የተጠቀሰውን መኪና አአ 0345484 እና የሻሲ ቁጥር jaakp34hkj7p376733 የሞተር ቁጥር 4hj1.549927 የሆነውን መኪና የግጭት አደጋ ደርሶበት እስርቼ የራሴን ንብረት ለማድረግ ከላይ በተጠቀሰው [illegible] ገዝቼ የመጀመርያውን ክፍያ 50,000 (ሀምሳ ሺ ብር) ከፍዬአለሁ ቀሪውን 500,000 ብር

Figure 1:Form1



(አምስት ሙቶ ሽህር)አስከ ታህሳስ 24 ቀን 2011ዓ.ም ድረስ ልከፍልና መክፍሉን
ስወስድ ከተወሰነው 500.000 ብር (አምስት ሙቶ ሽህር) ላይ በሰም ማህደረ ብቻ
20000ብር (ሀያ ሺ ብር)አስቀርቼ 480000ብር (አራት ሙቶ ሰማንያ ሺብር)ተሰማምቼ
ውሉን ፈጽሚያለሁ። ውልና ማስረጃ ቀርቦን የሰም ዝውውሩ ከተፈጸመ ዉኃላ ከሰም
ማሳወርያ ከተያዘው 20000ብር (ሀያ ሺ ብር) የተረፈውን ለሻጭ ልመልስ
ተስማምቻለሁ።

ይህንን ውል ሻጭም ሆነ ገዢ የሚከተለውን ወይም ማቅረብ አሚከተለውን ማስረጃ
በማቅረብ ስም ዝውውሩ እንዲፈጸም እንደናደርግ ሲሆን እንደውሉ የማይፈጽም የውል
አካል ቢኖር በዓ/ቡሔር ህግ ቁጥር 1889 መሰረት 50000 ብር (ሀምሳ ሺ ብር
)መቀጮ ውሉን ላከበረ ወገን ተቀጥቶ ውሉ የጸና ውል እንዲሆን ተስማምተናል።

ይህንን ውል ያዋዋሉ እማኞች 1ኛ አቶ አውላቸው ስንሰ 2ኛ ወ/ሮ ነኑት በዘ 3ኛ
ሪ/ርግሩም ነገሰ 4ኛ ወ/ሮ ኤደን ተደሰ ናቸው እኛም እማኞች ከዚህ በላይ የተጠቀሰውን
ውል አይተን አዋውሰናል ፈርመናል።

የሻጮች ፊርማ

የገዢ ፊርማ

1ኛ----------

2ኛ----------

3ኛ----------

የእማኞች ፊርማ

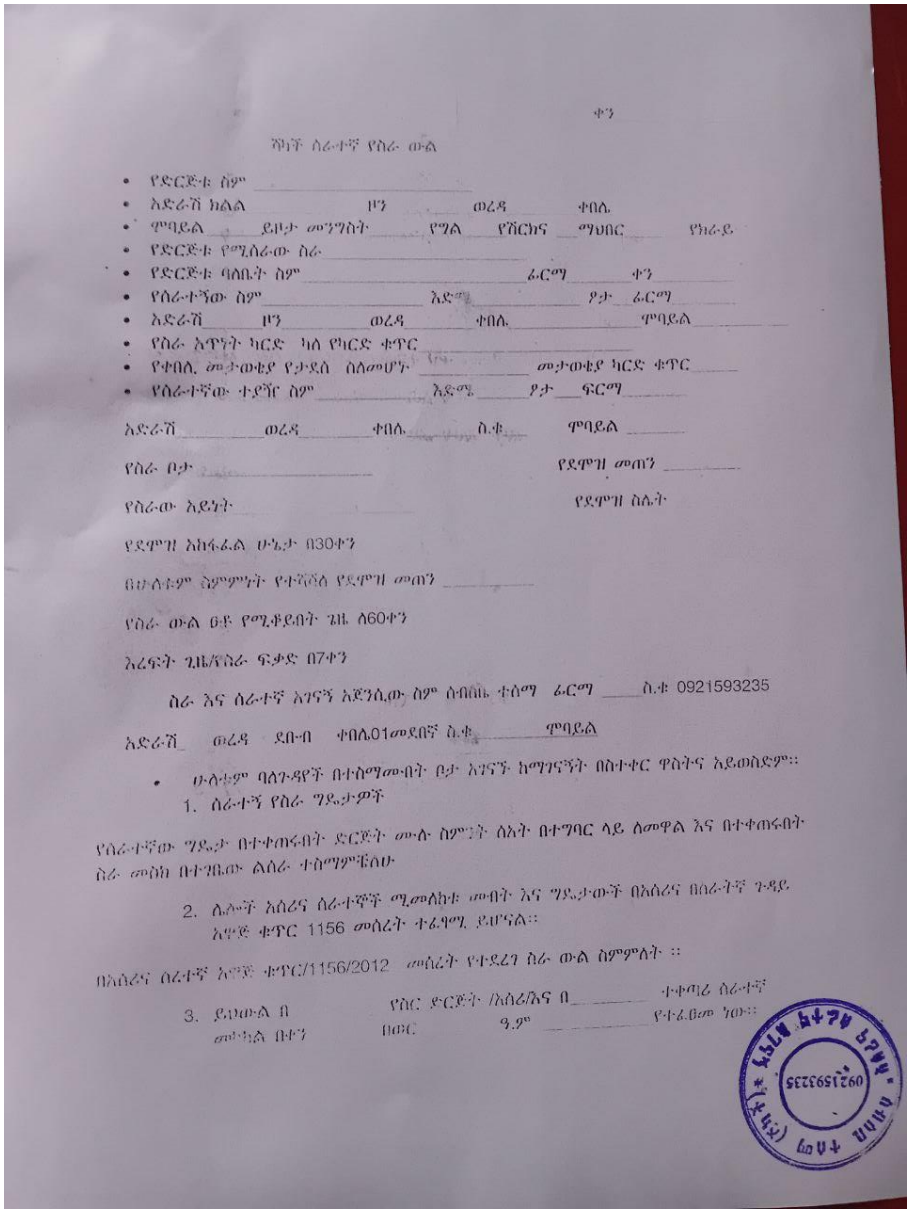
1. ----------

2. ----------

3. ----------

4. ----------

Figure 2:Form 2



ቀን

ግል ሰራተኛ የስራ ውል

- የድርጅቱ ስም
- አድራሻ ክልል ዞን ወረዳ ቀበሌ
- ሞባይል ይዞታ መንግስት የግል የሽርክና ማህበር የክራይ
- የድርጅቱ የሚሰራው ስራ
- የድርጅቱ ባለቤት ስም ፆታ ዜግነት
- የሰራተኛው ስም እድሜ ፆታ ዜግነት
- አድራሻ ዞን ወረዳ ቀበሌ ሞባይል
- የስራ ልምድ ካርድ ካለ የካርድ ቁጥር
- የቀበሌ መታወቂያ የታደሰ ስለመሆኑ መታወቂያ ካርድ ቁጥር
- የሰራተኛው ተያዥ ስም እድሜ ፆታ ፍርማ

አድራሻ ወረዳ ቀበሌ ስ.ቁ ሞባይል

የስራ ቦታ የደሞዝ መጠን

የስራው አይነት የደሞዝ ስሌት

የደሞዝ አከፋፈል ሁኔታ በ30ቀን

በሁለቱም ስምምነት የተሻሻለ የደሞዝ መጠን

የስራ ውል ፀድ የሚቆይበት ጊዜ ለ60ቀን

አረፍት ጊዜ/ፍቃድ ፍቃድ በ7ቀን

ስራ እና ሰራተኛ አገናኝ አጀንሲው ስም ስብስቤ ተሰማ ፊርማ ስ.ቁ 0921593235

አድራሻ ወረዳ ደቡብ ቀበሌ01መደበኛ ስ.ቁ ሞባይል

- ሁለቱም ባለጉዳዮች በተስማሙበት ቦታ አገናኙ ከማገናኝት በስተቀር ዋስትና አይወስድም።

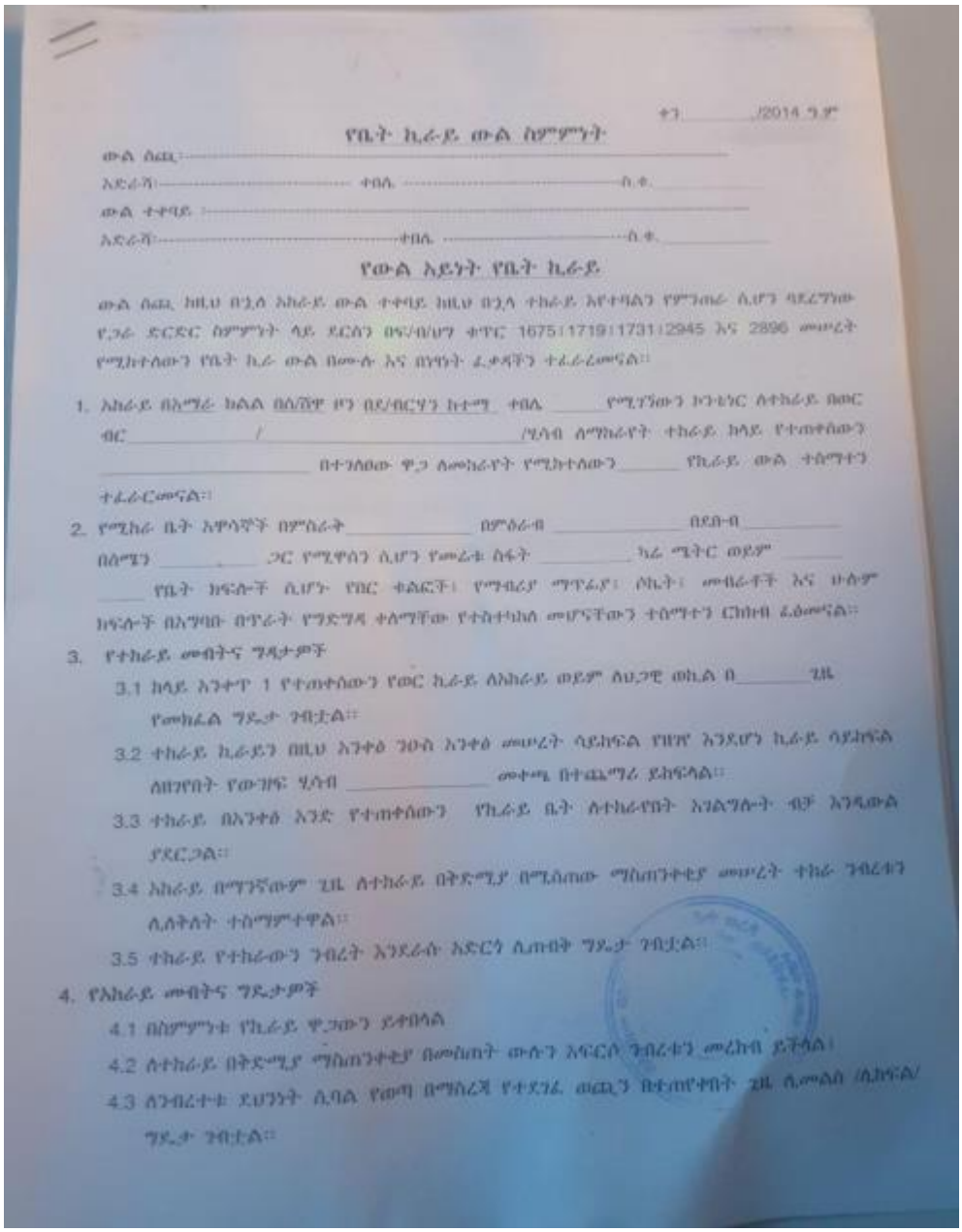
1. ሰራተኛ የስራ ግዴታዎች

የሰራተኛው ግዴታ በተቀጠሩበት ድርጅት ሙሉ ስምንት ሰአት በተግባር ላይ ለመዋል እና በተቀጠሩበት ስራ መስክ በተገቢው ልሰራ ተስማምቼበሁ

2. ሌሎች አሰሪና ሰራተኞች ሚመለከቱ መብት እና ግዴታውች በአሰሪና በሰራተኛ ጉዳይ አዋጅ ቁጥር 1156 መሰረት ተፈፃሚ ይሆናል።

በአሰሪና ሰራተኛ አዋጅ ቁጥር/1156/2012 መሰረት የተደረገ ስራ ውል ስምምነት ።

3. ይህውል በ መሰረት በቀን የስር ድርጅት /አሰሪ/ኤና በ ተቀጣሪ ሰራተኛ በወር ዓ.ም የተፈፀመ ነው።



ቀን _____/2014 ዓ.ም

የቤት ኪራይ ውል ስምምነት

ውል ሰጪ፡-

አድራሻ፡- ተበሉ ስ.ቁ.

ውል ተቀባይ ፡-

አድራሻ፡- ተበሉ ስ.ቁ.

የውል አይነት የቤት ኪራይ

ውል ሰጪ ከዚህ በኋላ አከራይ ውል ተቀባይ ከዚህ በኋላ ተከራይ እየተባልን የምንጠራ ሲሆን ባደረግነው የጋራ ድርድር ስምምነት ላይ ደርሰን በፍ/ብ/ህግ ቁጥር 1675፣1719፣1731፣2945 እና 2896 መሠረት የሚከተለውን የቤት ኪራ ውል በሙሉ እና በነፃት ፈቃዳችን ተፈራረመናል።

1. አከራይ በአማራ ክልል በሰሜን ሾን በደ/ብርሃን ከተማ ቀበሌ _____ የሚገኘውን ኮንቴነር ለተከራይ በወር ብር _____/_____/ሂሳብ ለማከራየት ተከራይ ከላይ የተጠቀሰውን _____ በተገለፀው ዋጋ ለመከራየት የሚስተሰውን _____ የኪራይ ውል ተስማተን ተፈራርመናል።
2. የሚከራ ቤት አዋሳኞች በምስራቅ _____ በምዕራብ _____ በደቡብ _____ በሰሜን _____ ጋር የሚዋሰን ሲሆን የመሬቱ ስፋት _____ ካሬ ሜትር ወይም _____ የቤት ክፍሎች ሲሆኑ የበር ቁልፎች፣ የማብሪያ ማጥፊያ፣ ሶኬት፣ መብራቶች እና ሁሉም ክፍሎች በአግባቡ በጥራት የማድማጸ ቀለማቸው የተስተካከለ መሆናቸውን ተስማተን ርክክብ ፈፅመናል።
3. የተከራይ መብትና ግዴታዎች
   3.1 ከላይ አንቀጥ 1 የተጠቀሰውን የወር ኪራይ ለአከራይ ወይም ለህጋዊ ወኪል በ_____ ጊዜ የመክፈል ግዴታ ንብረታል።
   3.2 ተከራይ ኪራይን በዚህ አንቀፅ ንዑስ አንቀፅ መሠረት ሳይከፍል የዘገየ እንደሆነ ኪራይ ሳያከፍል ለዘገየበት የውዝፍ ሂሳብ _____ መቀጫ በተጨማሪ ይከፍላል።
   3.3 ተከራይ በአንቀፅ አንድ የተጠቀሰውን የኪራይ ቤት ለተከራየበት አገልግሎት ብቻ እንዲውል ያደርጋል።
   3.4 አከራይ በማንኛውም ጊዜ ለተከራይ በቅድሚያ በሚሰጠው ማስጠንቀቂያ መሠረት ተከራ ንብረቱን ሊለቅለት ተስማምተዋል።
   3.5 ተከራይ የተከራውን ንብረት እንደራሱ አድርጎ ሊጠብቅ ግዴታ ንቢታል።
4. የአከራይ መብትና ግዴታዎች
   4.1 በስምምነቱ የኪራይ ዋጋውን ይቀበላል
   4.2 ለተከራይ በቅድሚያ ማስጠንቀቂያ በመስጠት ውሉን አፍርሶ ንብረቱን መረከብ ይችላል፤
   4.3 ለንብረተቱ ደህንነት ሲባል የወጣ በማስረጃ የተደገፈ ወጪን በተጠየቀበት ጊዜ ሊመልስ /ሊከፍል/ ግዴታ ንብቷል።

Figure 4: Form 4