PROJECT REPORT

ON

Account Management System

Submitted by:

Tajanlangit, Elieser N.

Angco, Claidee Mae

Ycong, Rovielyn

Sumalinog, Joseph(inserted)

Submitted to:

Mr. Danny Obidas

DECLARATION

I solemnly affirm and declare that this submission stands as the product of my unwavering dedication and intellectual endeavor. With absolute certainty and utmost conviction, I attest that every iota of this work is an original creation, bearing no trace of material previously published or penned by another individual. Furthermore, I assert that this composition, to the best of my discernment and belief, has not been incorporated into the pursuit of any other esteemed degree or diploma from this university or any other institute of higher learning. Exceptionally, where external influence has contributed, full and sincere acknowledgment has been meticulously woven into the fabric of this text. This declaration underscores my profound commitment to the highest standards of academic integrity and authenticity.

ABSTRACT

This abstract outlines the development of a Multi-User Management System designed to enhance the efficiency of barangay administration, utilizing the MERN stack. The system caters to three primary user roles: Admin, Barangay, and Resident. Admin users possess comprehensive account management capabilities, enabling them to create, read, update, and delete accounts. Additionally, Admins can assign roles to users, defining their specific access and responsibilities within the system. The Barangay user role is centered around the meticulous recording and management of resident information, with an added feature of registering households and assigning unique identification numbers. Notably, the Resident user role is confined to inputting and managing information solely for their own household members. The MERN stack, incorporating MongoDB, Express.js, React.js, and Node.js, ensures a robust and scalable foundation for the system. This technology combination facilitates seamless interaction and responsiveness, accommodating real-time updates. Overall, the abstract highlights the system's commitment to providing a secure, user-friendly, and technologically advanced solution tailored to the diverse needs of barangay administration.

CHAPTER 1

INTRODUCTION

1.1 Overview

The Account Management System consists of three primary user modules, each with distinct roles and responsibilities:

Administrator:

Role: Holds ultimate control and authorization within the system.

Responsibilities:

- Authorizes members to assume roles such as admin, barangay, or resident.
- Manages and oversees the entire system, including user functionalities.

Barangay:

Role: Granted administrative privileges by the admin.

Responsibilities:

- Full control over resident accounts.
- Creation, deletion, editing, and viewing of resident accounts.
- Registration and management of resident households.
- Announcement and event creation for the community.

Resident:

Role: Limited administrative access.

Responsibilities:

- Manages household members (if their household is registered with the barangay).
- Submits complaints or feedback through the system.

Roles and Permissions:

Administrator Role:

Access: Complete control over all accounts (barangay members, residents).

Permissions:

- Authorization to assign admin privileges to barangay members.
- Manages accounts and oversees the entire system.

Barangay Role:

Access: Full control of the records of the residents information and their household numbers.

Permissions:

- Creation, deletion, editing, and viewing of resident informations.
- Registration and management of resident households.
- Announcement and event creation for the community.

Resident Role:

Access: Limited administrative control.

Permissions:

- Manages household members (if registered with the barangay).
- Submits complaints or feedback.

Functionalities:

Administrator:

- Authorizes faculty members to become admin-faculty.
- Manages and oversees the entire system.

Barangay:

- Manages resident information with full administrative control.
- Creates, deletes, edits, and views resident information.
- Registers and manages resident households.
- Announces community events.

Resident:

- Manages household members (if registered with the barangay).
- Submits complaints or feedback.

1.2 Objective:

The objectives of the system are concise and centered on the specific needs and goals of the project. Here's a detailed breakdown of these objectives:

- Resident Data Tracking: The project aims to efficiently and accurately keep track of the records, primarily focusing on the residents in Barangay Ibabao. This involves creating a comprehensive database of resident information, including names, addresses, contact details, birthdates, and family data.
- Resident Update and Record: The primary goal of BIUMS is to provide a streamlined and user- friendly system for updating and recording residents in the community. This ensures that the resident data is kept current and accurate, facilitating effective governance and service delivery.
- Community Awareness: While not explicitly stated in the objectives, the system also serves the purpose of maintaining community awareness. By having up-to-date resident records, local authorities can keep residents informed about important community announcements and events, fostering a sense of unity and ensuring that residents are aware of what's happening in their barangay.

CHAPTER 2

PROBLEM STATEMENT

Barangays, much like numerous others, currently grapple with the challenges posed by paper-based household records— a system that is prone to becoming outdated, challenging to access, and susceptible to damage or loss. This antiquated approach significantly impedes our capacity to respond promptly to emergencies, deliver targeted services, and effectively plan for the future. Recognizing the limitations of this outdated system, it is imperative that we embrace a modernized approach to managing household records, one that leverages technology to enhance accessibility, accuracy, and efficiency. This transition marks a crucial step forward in ensuring the agility and effectiveness of our barangay administration in meeting the evolving needs of our community.

2.1 Proposed system

1. **Secure and Intuitive Database Infrastructure:**

- Establish a robust and user-friendly digital database system to manage and update household records securely.
- Design an accessible interface for barangay personnel, ensuring ease of data entry, modification, and retrieval.
- Implement stringent data security measures and access controls to safeguard privacy.

2. **Periodic Data Refresh:**

- Develop a systematic process to routinely update household records, capturing changes in resident information.
- Engage in regular community-led data verification campaigns to enhance record accuracy and involve residents in the upkeep of their data.

3. **Synergy with Essential Services:**

- Integrate the digital records system with key barangay services, including healthcare, education, disaster preparedness, and social welfare.
- Enhance response times and service quality by streamlining access to accurate resident information, enabling targeted assistance.

4. **Community Awareness and Participation:**

- Conduct awareness campaigns to educate residents on the advantages of the new system and the importance of keeping their records current.
- Foster community engagement by encouraging residents to actively participate in data verification and update initiatives.

CHAPTER 3

SYSTEM ANALYSIS

Objective:

Create an innovative Multi-User System for Barangay Administration using the MERN stack, aiming to optimize administrative processes for Admin, Barangay, and Resident user roles.

Components:

User Roles:

Administrator (comprehensive account management)

Barangay (responsible for resident information and household registration)

Resident (limited to managing their household details)

Role Definition:

Each user role embodies distinct access levels and responsibilities.

A tiered approach to role hierarchy establishes a structured distribution of permissions.

Access Parameters:

Granular controls governing user actions (creation, viewing, modification, deletion) on accounts and system settings.

Inherited access parameters to expedite role assignment processes.

System Characteristics:

Robust identity verification and access confirmation mechanisms to safeguard against unauthorized entry.

User-friendly interfaces for efficient handling of account-related activities and permission configurations.

Integrated tracking and audit capabilities to monitor and analyze user interactions.

Evaluation:

Security Measures:

Structured role hierarchy acts as a safeguard against unwarranted access.

Granular access controls provide meticulous oversight over sensitive data.

Authentication and authorization mechanisms function as gatekeepers, ensuring only authorized users navigate the system.

Logging and auditing capabilities enable a thorough examination of user activities, reinforcing accountability.

Operational Efficiency:

Well-defined user roles simplify administrative tasks.

Inherited permissions reduce manual effort in assigning access.

Intuitive user interfaces streamline access management.

Adaptability and Scalability:

Adaptable role and access structure accommodates potential system expansion.

The system is equipped to seamlessly integrate new roles and access levels.

Encountered Challenges:

Balancing role comprehensiveness with access restriction, especially for community enablers.

Determining the optimum level of detail for access controls without introducing unnecessary complexity.

Streamlining the user management process, ensuring efficiency in role assignment and access management.

Addressing security concerns related to unauthorized access, data breaches, and misuse of privileges.

Future Steps:

Construct a comprehensive role and access hierarchy.

Design user interfaces prioritizing user-friendliness in account and access management.

Implement robust mechanisms for user authentication and authorization.

Develop a comprehensive system for tracking and auditing.

Subject the system to rigorous testing, assessing both security and usability aspects.

CHAPTER 4

4.1 SYSTEM REQUIREMENTS

Hardware:

- **Laptop:** HP
- **Processor**: AMD A10, a powerful processor from Advanced Micro Devices (AMD), providing robust processing power for efficient compilation and execution of server-side and development tasks.
- **Storage**: 128GB SSD (Solid State Drive), ensuring fast read and write speeds for data-intensive operations, enhancing overall system responsiveness.
- **RAM**: 10GB, comprising high-quality memory modules for ample memory, facilitating concurrent processes, smooth multitasking, and efficient handling of large datasets during development and runtime.

Software:

Development:

- **Visual Studio Code (VSCode)**: serves as the primary IDE, providing an intuitive and feature-rich environment. Leveraging VSCode enhances code editing, debugging, and version control, contributing to an efficient and streamlined development process.

Server-Side:

- **MongoDB**: Serving as our database solution, MongoDB offers a flexible and scalable NoSQL architecture. Its document-oriented approach allows for efficient data storage and retrieval, making it well-suited for handling diverse and evolving data structures in our server-side application.
- **Postman**: For comprehensive testing and validation of our server-side functionalities, we have integrated Postman. This versatile API development and testing tool enable us to send requests, analyze responses, and ensure the seamless communication between our server and client components, contributing to the overall reliability of our system.

Client-Side:

- **ReactJS**: The front-end of our application is powered by ReactJS, a declarative and efficient JavaScript library for building user interfaces. React's component-based architecture enables modular development, fostering a dynamic and responsive user experience.
- **NodeJS**: NodeJS is employed on the client side to facilitate server-side rendering and manage asynchronous tasks. Its event-driven architecture and non-blocking I/O contribute to the scalability and performance of our application.

- **Redux**: To manage the state in a predictable and centralized manner, we utilize Redux. This state container ensures efficient data flow and simplifies the management of complex application states, enhancing the overall stability and maintainability of our client-side codebase.

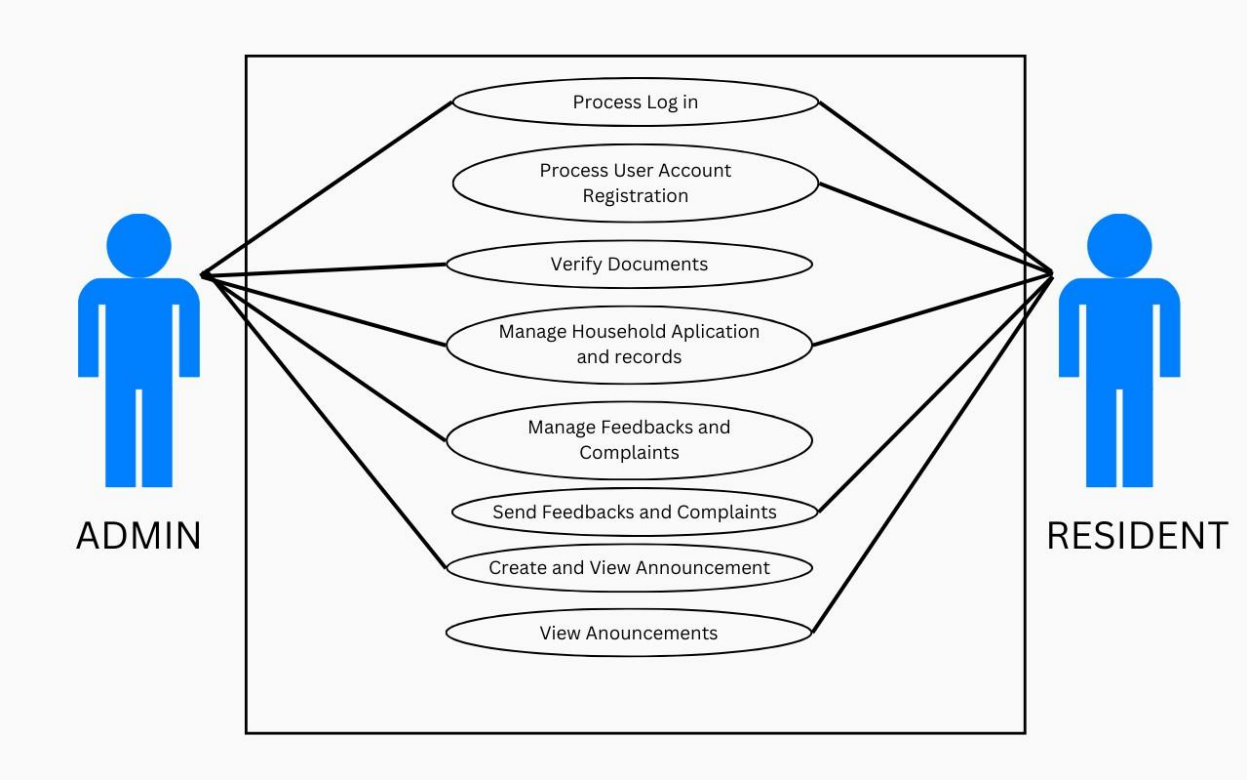
Tools:

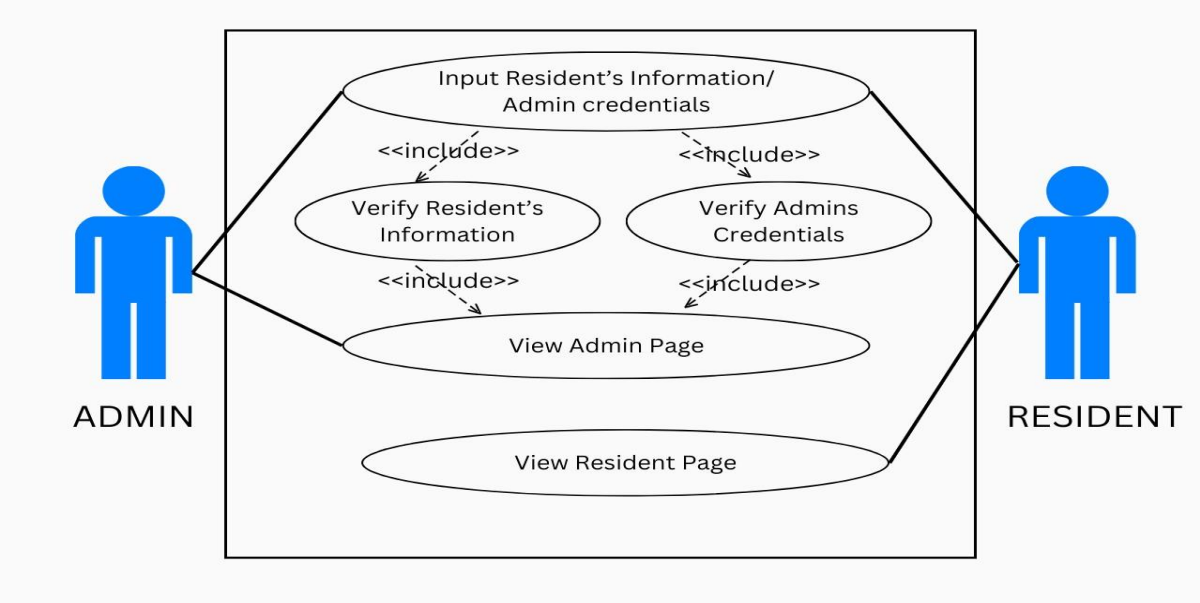
- **Postman Integration**: To facilitate comprehensive testing of the backend functionalities, Postman has been installed as an additional tool on the system. This integration empowers users to execute API requests, inspect responses, and validate the overall functionality of the developed Multi-User Management System. Postman's user-friendly interface and robust features contribute to a seamless testing experience, ensuring the reliability and effectiveness of the backend processes.

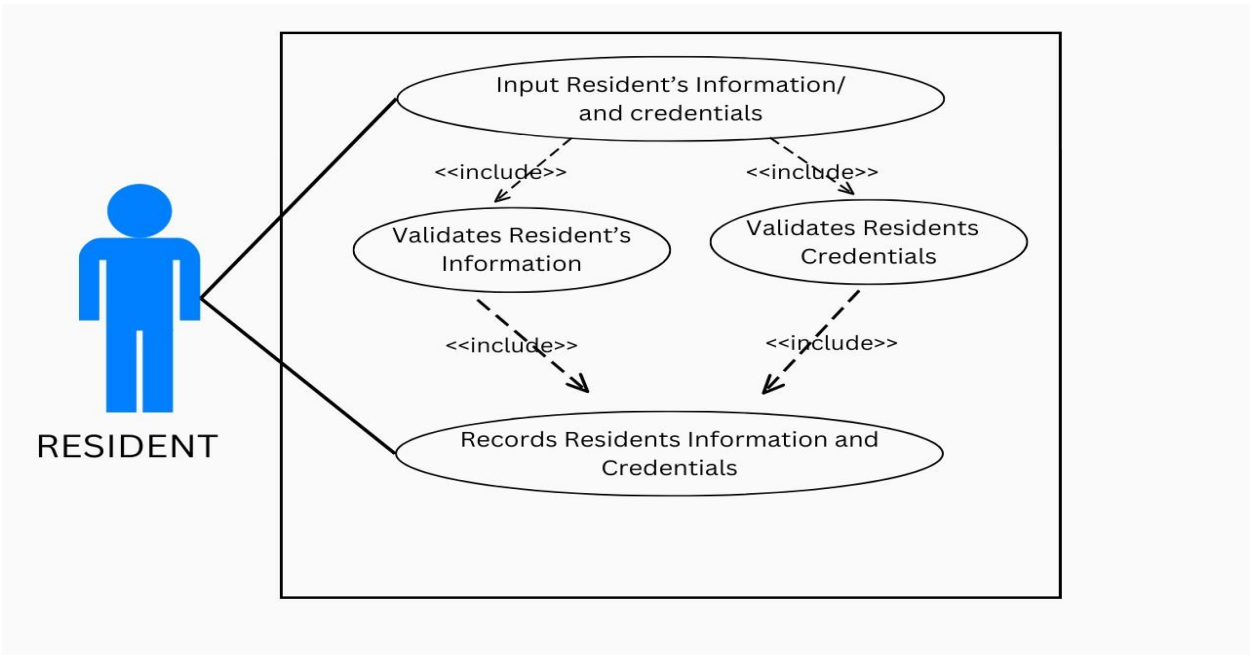
- **React DevTools Integration**: The implementation includes seamless integration with React DevTools, a browser extension that enhances the development and debugging experience for React applications. With React DevTools installed, developers gain valuable insights into the component hierarchy, state, and props of their React application directly within the browser's developer tools. This integration aids in real-time monitoring, debugging, and optimization of the user interface, offering a more efficient and streamlined development process.

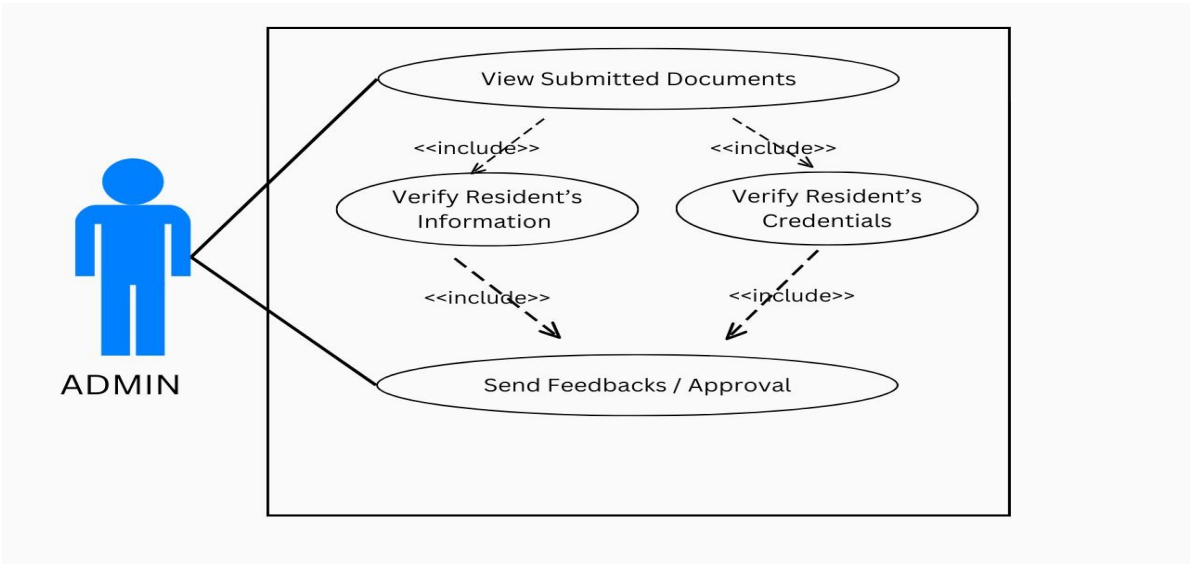
- **Redux DevTools Integration**: The development environment is enriched with the integration of Redux DevTools, a powerful extension for browser developer tools that enhances the debugging and analysis of Redux state management in the application. By leveraging Redux DevTools, developers can efficiently track state changes, inspect actions, and travel through the application's state history. This integration provides invaluable insights into the flow of data within the Redux store, facilitating a more thorough understanding of the application's state management and aiding in the identification and resolution of potential issues during development.

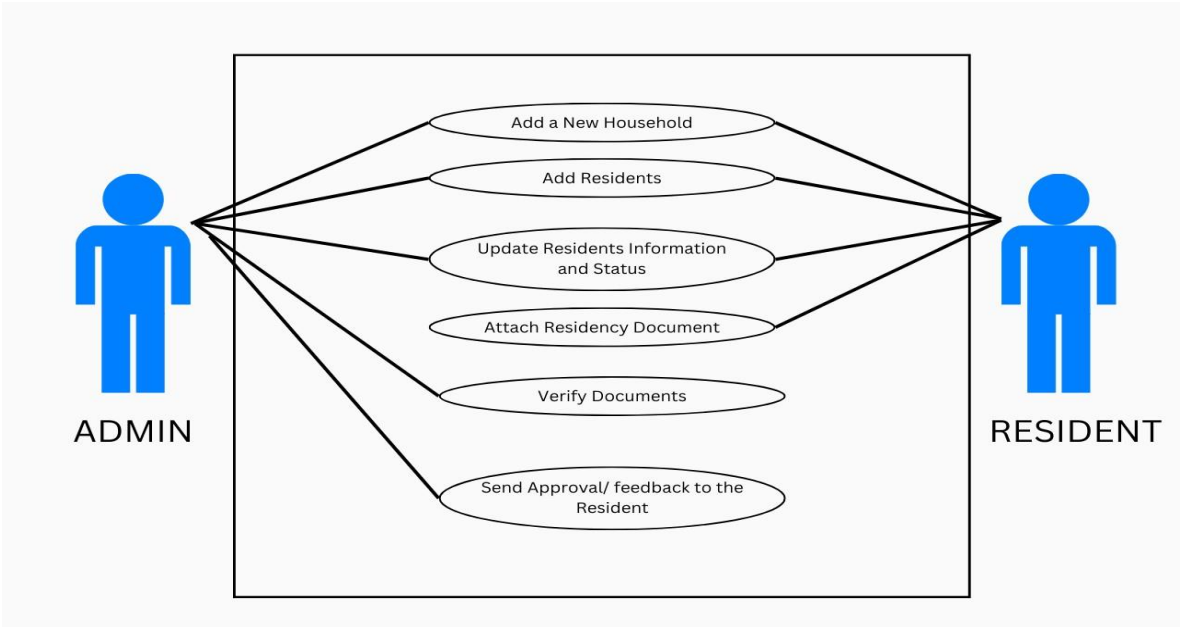
4.2 USE CASE DIAGRAM

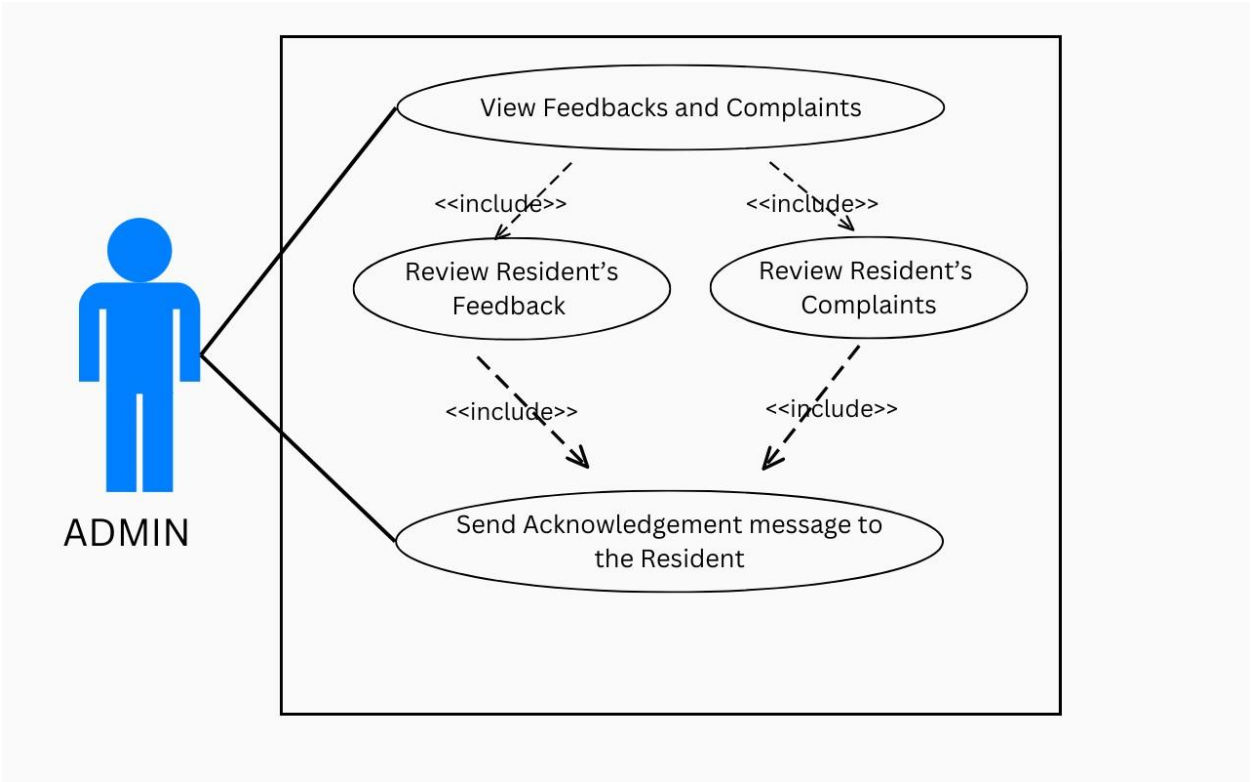


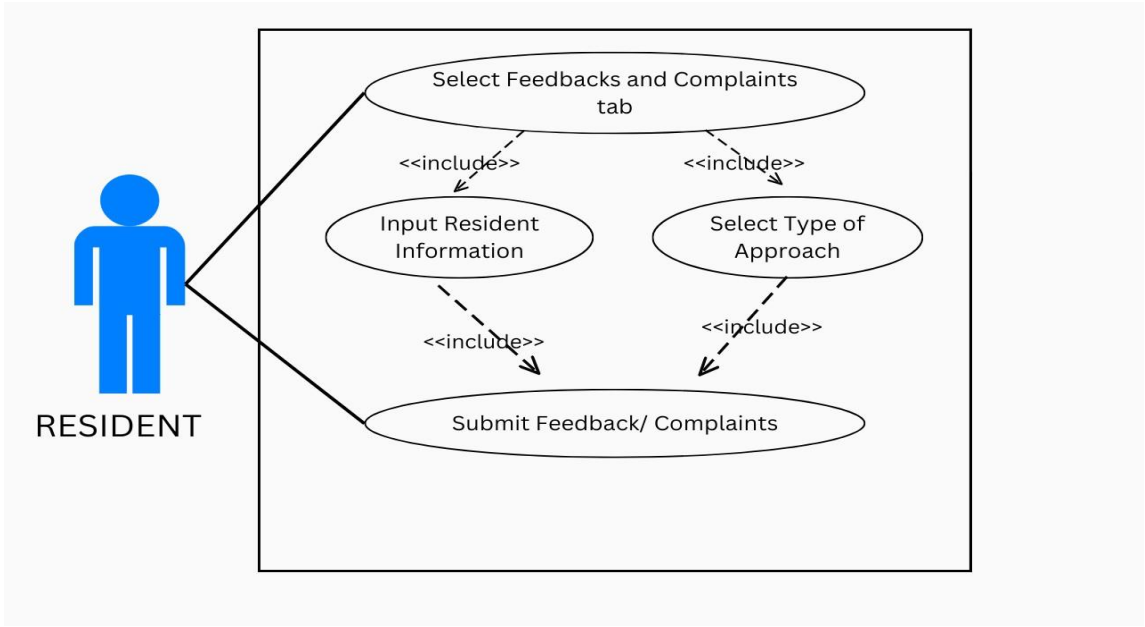


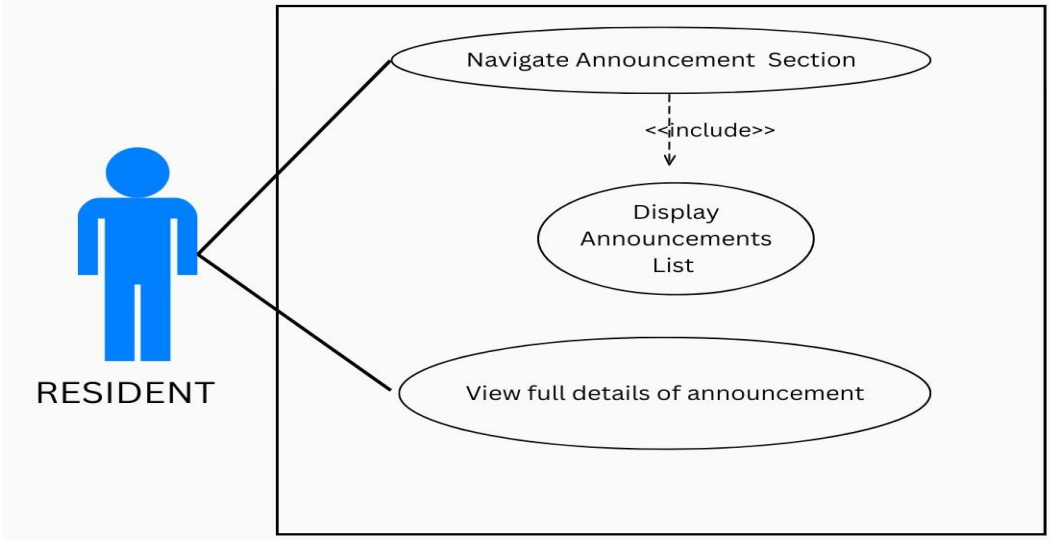

Input Resident's Information/
and credentials
<<include>>
<<include>>
Validates Resident's
Information
Validates Residents
Credentials
<<include>>
<<include>>
Records Residents Information and
Credentials
RESIDENT


View Submitted Documents
<<include>>
<<include>>
Verify Resident's
Information
Verify Resident's
Credentials
<<include>>
<<include>>
Send Feedbacks / Approval
ADMIN


Add a New Household
Add Residents
Update Residents Information and Status
Attach Residency Document
Verify Documents
Send Approval/ feedback to the Resident
ADMIN
RESIDENT


View Feedbacks and Complaints
<<include>>
<<include>>
Review Resident's Feedback
Review Resident's Complaints
<<include>>
<<include>>
Send Acknowledgement message to the Resident
ADMIN


Select Feedbacks and Complaints tab
<<include>>
<<include>>
Input Resident Information
Select Type of Approach
<<include>>
<<include>>
Submit Feedback/ Complaints
RESIDENT


Navigate Announcement Section
<<include>>
Display Announcements List
View full details of announcement
RESIDENT

4.3 ENTITY RELATIONSHIP DIAGRAM(ERD)