APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 362-381
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Resident Management System for Lestari Mansions with Role-Based Access Control

Chin Hui Yee[1], Cik Feresa Mohd Foozy[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: feresa@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.020

Article Info

Received: 29 July 2024
Accepted: 18 June 2025
Available online: 30 June 2025

Keywords

Resident Management System, Role-Based Access Control (RBAC), Agile, Data Security

Abstract

This resident management system is developed for Lestari Mansions, Seri Kembangan to manage the residents efficiently. The existing resident management at Lestari Mansions does not provide an access control mechanism and paper-based information can be accessed easily by all the residents, committee members, and security guards. Therefore, this study aimed to develop a system that allows only the authorized users to login to the system and perform specific tasks by implementing Role-Based Access Control (RBAC). Agile methodology is used in the development of this system using the Java programming language. The developed resident management system has improved the efficiency of the existing management process while ensuring data security with modules such as registration, login, manage users, manage feedback, manage invoice, and phone book. The system protects the confidentiality and integrity of the data by restricting system access based on the assigned roles. The functional testing result shows the system performs key tasks as intended. The user acceptance testing result shows that the resident management system is well-accepted by the end-users.

1. Introduction

Lestari Mansions is a gated and guarded housing area in Seri Kembangan, Selangor with a total of 154 housing units. A gated and guarded community are a guarded housing area which surrounded by fences, has restricted access, and is managed by the residents with their own management [1]. The committees of Lestari Mansions are currently managing the residents manually as they do not own any computerized system to store the residents' records. The only communication channel which connects the committees with the residents is WhatsApp where it is used to make announcements updates, complaints, and payment collections. Payments are collected by the security guards and the committee will compile the payment received and record them in a spreadsheet. The residents need to contact the committee personally if they want to update their contact information.

The problem that occurred because of managing the residents manually included time-consuming and inefficient payment processing. The committee can face problems to trace back transaction when payment amount is collected wrongly, or the handwritten receipts is wrongly issued by the security guards. Besides, the problem that is faced is having out-of-date and inaccurate data. Paper-based data systems are said to have the risk of introducing errors during data collection and transforming into digital form [2]. Data inconsistency could occur if the committee failed to update the data in the guardhouse phone book. In terms of the security issues, as the paper-based records are currently being stored in the office room of the club house, the records are susceptible to physical damage, theft and loss which can threaten data confidentiality, integrity, and availability. The

involvement of security guards in manual payment processing can threaten data security as well since the handwritten receipt can be easily altered without consent.

Therefore, the objectives of this project are to design a resident management system for Lestari Mansions with role-based access control using object-oriented approach, to develop resident management system using web-based approach, and to test the functionality of the developed system.

The developed resident management system implemented security techniques such as role-based access control and cryptography. Role-based access control is used to restrict system access to authorized users based on their roles [3]. While cryptography ensures authentication and integrity through hashing of password [4]. The system is designed and developed using Apache NetBeans Integrated Development Environment (IDE) version 19, which supports development of Java web application. The target users of this system are Lestari Mansions residents, committee members and security guards which all having different responsibilities. The main modules are registration, login, profile, dashboard, feedback, payment, phonebook, and report.

At the end of this project, a resident management system for Lestari Mansions with role-based access control is expected to be developed which will be able to implement online security fee payment processing. The system also should be able to implement automation concept by computerizing the current manual management system and be able to protect the confidentiality, integrity, and availability of the resident's data through role-based access control.

The project is significant in promoting the automation of resident management system in a residential community which could be then proposed to be used by residential communities which are operating their management of residents manually currently. The project also promotes the awareness of information security by implementing security practices in the system which included role-based access control and password hashing.

The rest of the paper is organized as follows. Section 2 discusses the related work of the resident management system. Section 3 describes the methodology used in the development of the system. Section 4 discusses the GUI of the system and section 5 concludes the project.

2. Related Work

Relevant research and studies are conducted on the theories, techniques and algorithms related to the developed system. Comparative studies were conducted on three existing systems that are related to the developed system.

2.1 Theories, Techniques and Algorithms

Role-based access control is the security model where access permissions are based on the roles individuals have in an organization. The users are assigned with specific roles that align with their responsibilities in the organization. This model is useful in multi-user computing as the access permissions are tied to roles rather than individual users, simplifying the access control management [5]. RBAC is implemented with 3 assigned roles which are residents, committee members, and security guards.

User authentication is an important verification process to validate the identity of an active user. Multi-factor authentication (MFA) involves using two or more authentication methods to verify the identity of a user [6]. MFA is implemented by combining features of knowledge factor with features of possession factor in the developed system.

Secure Hash Algorithms (SHA) is the cryptographic hash function developed by National Institute of Standards and Technology (NIST). There are two hash functions in SHA-2 group which are SHA-256 and SHA-512 algorithms. SHA is chosen as the hashing algorithm to be used in the developed system as SHA is a more suitable choice when prioritizing security requirements [7].

2.2 Existing Systems

Comparative studies were conducted on three existing systems that are related to the developed system which are Resident Easy Decision System (MYREDS) [8], M4U Home Resident System [9], and i-Neighbour Resident App [10].

MYREDS is a residence management system developed by AR Mechatronics PLT. MYREDS has a web panel for admin and mobile applications for residents and guards. It allows residents to submit events, fill reports, and pay monthly maintenance fees. Every transaction is securely logged in to the admin web panel and application.

M4U Home Resident System is the resident management system owned by Manage 4 U Sdn. Bhd. The features offered for the admins are receiving resident feedback, generating invoices to residents, managing residents' unit profile, passing live messages to residents, managing users' details, managing parking lots, and checking for facilities booking. While the main features that are offered to residents include booking facilities, making payment, submitting feedback, and managing visitors.

I-Neighbor Resident App is the resident community system developed by TimeTec Group. New users are required to activate their account using an activation code that is sent to their registered email. The application allows residents to make payments on maintenance fees and book facilities within the community. While the

admin can manage e-billing on the portal such as importing invoices and generating collection reports. Table 1 shows the comparison between the existing systems and the developed system.

Table 1 *System's Comparison*

Features	MYREDS (Resident Easy Decision System)	M4U Home Resident System	i-Neighbour Resident App	Resident Management System for Lestari Mansions
Login	√	√	√	√
Logout	√	√	√	√
Role-based access control	√	√	√	√
Forgot password	√	√	√	√
CAPTCHA	√	√	X	√
Two-factor authentication	X	X	X	√
Strong password policy	X	X	√	√
Account lockout policy	X	X	X	√

3. Methodology

This section explains the use of Agile model in this project and the activities that had been carried out in each phase. Agile software development is an iterative and incremental approach which continue to deliver small and functional pieces of software. It involves concurrent and collaborative development and testing throughout the entire development life cycle [11]. Fig. 1 shows the shows the 5 phases in Agile model which are planning, analysis, design, implementation, and testing.

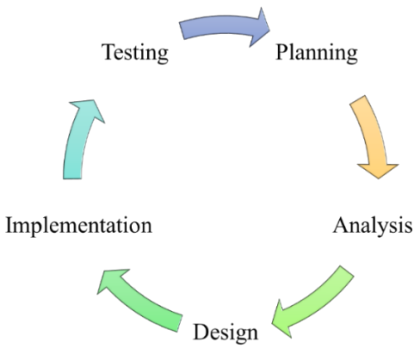


Fig. 1 *Agile Software Development Cycle (SDLC)*

3.1 Planning Phase

The planning phase is the fundamental process of identifying the goal of developing the system and determine how to structure the project. The problem faced in the existing system is first identified through observation and interview with the resident and committee members of Lestari Mansions residents' association. A solution is then proposed which is to develop a resident management system. After the objective of the project is determined, the project scope and significance of project is identified to define the major deliverables of the project.

3.2 Analysis Phase

The analysis phase is to understand the existing system, identify improvements and define requirements for the new system. Literature review was conducted to gain insights from existing research and security techniques.

The functional requirements, non-functional requirements, and user requirements are then determined by conducting an interview with the resident and committee member of Lestari Mansions residents' association. The software requirements and hardware requirements are also determined to ensure software compatibility during system design. Table 2 shows the functional requirements and Table 3 shows the non-functional requirements.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381

Table 2 *Functional requirements of the system*

Modules	Functionalities
Registration	• The system should allow committee member to register admin accounts using email and password. • The system should allow committee member to register inactive resident account using resident's email and house number. • The system should send resident account activation link to the registered email upon successful registration. • The system should allow resident to activate their accounts by setting up a password and filling up their personal information. • The system should allow committee member to register security guard accounts using email and password. • The system should validate the inputs such as duplicate house number and confirm password. • The system should display invalid error messages for invalid input. • The system should display a message on successful account registration. • The system should display a message on successful account activation.
Login	• The system should allow all users to login to the system using email and password. • The system should authenticate the users using a One-Time Passcode which will be sent to user's email. • The system should allow resident to reset password through a link sent to their email.
Profile	• The system should allow users to view their own personal information. • The system should allow users to update their own personal information.
Dashboard	• The system should allow committee member to create new post. • The system should allow residents to view the created post.
Users	• The system should allow committee member to view list of users of the system. • The system should allow committee member to update information of the users. • The system should allow committee member to remove users.
Feedback	• The system should allow residents to submit feedback. • The system should allow residents to view feedback status. • The system should allow committee member to update feedback status.
Invoice	• The system should allow committee member to issue invoice to the residents. • The system should display the unpaid amount on resident's account. • The system should allow residents to make payment for security fees. • The system should update the invoice status after receipt is uploaded. • The system should allow committee member to verify and update the invoice status.
Phone book	• The system should allow security guard to view the resident's contact details.
Report	• The system should allow committee member to generate invoice report.

Table 3 *Non-functional requirements of the system*

Characteristics	Requirements
Operational	• The system should be able to run on computing devices. • The system should be able to work on any Web browser.
Performance	• The response time of system to users should not exceed 2 seconds. • The system should be available for use 24 hours a day, 365 days a year. • The system should be able to support 160 simultaneous users at all times.
Security	• The system should only accept passwords with more than eight characters including at least one uppercase letter, lower case letter, number, and symbol.

Table 3 *(cont.)*

Characteristics	Requirements
	• Passwords in the system database should be hashed. • The system should validate the OTP before allowing the user to log in. • The system should restrict access of users based on their roles.
Cultural and political	• The system should support English language. • The system should support transactions and display financial information in Malaysian ringgit (RM).

Modelling of the gathered requirements is then accomplished using the Unified Modelling Language (UML). Three types of UML diagrams are used to illustrate the process and behaviour of Resident Management System for Lestari Mansions which are the use case diagrams, sequence diagrams, and activity diagrams. Use case diagrams capture the business requirements of the system by illustrating its main functionalities and the different users who will interact with it. Fig. 2 shows the use case diagram for committee member. The committee members can manage users, manage dashboard announcements, manage payment, and manage feedback after successfully log in to the system.

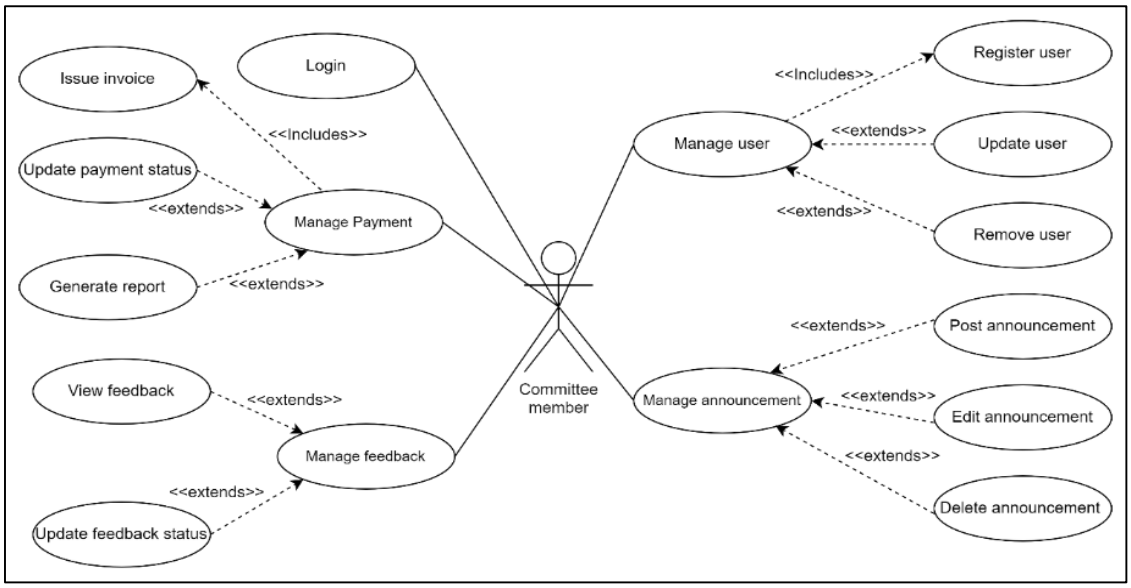


Fig. 2 *Use case diagram for committee member*

Fig. 3 shows the use case diagram for resident. The resident has to activate their account before able to log in to the system. After successful login, residents can manage their profile, view announcement made by committee members, upload receipt, and submit feedback.

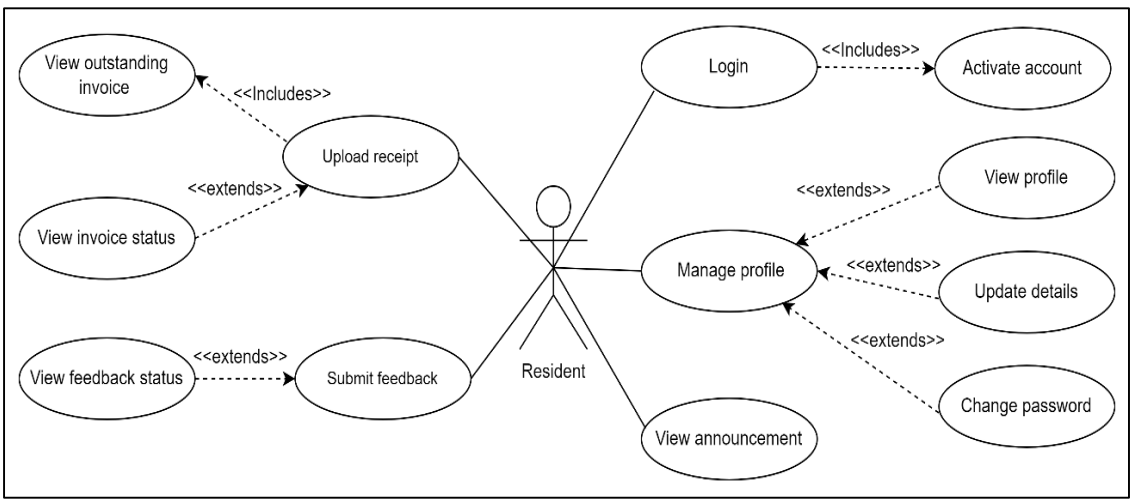


Fig. 3 *Use case diagram for resident*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381

Fig. 4 shows the use case diagram for security guards. The security guard can manage their own profile and view the resident's contact details after successfully login to the system.

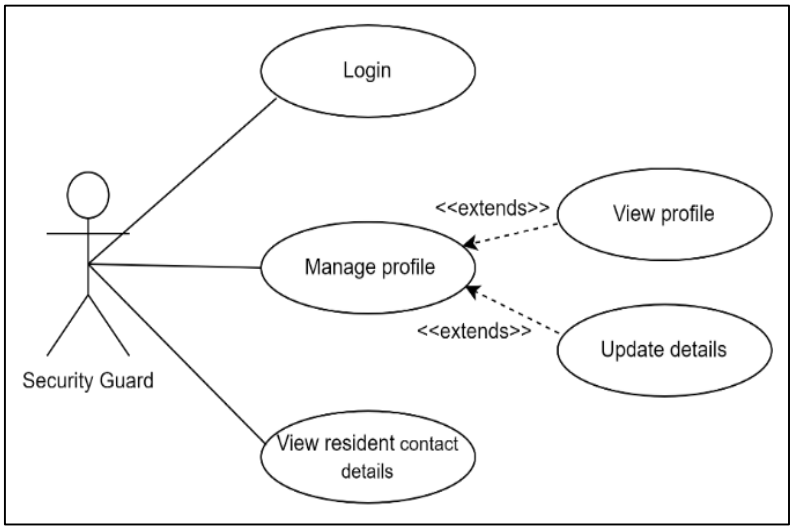


Fig. 4 *Use case diagram for security guard*

Sequence diagrams which illustrate the objects that involves in a use case and the messages that pass between them over time are shown in Appendix A.1, A.2 and A.3. It shows the dynamic view of the evolving system by illustrating the detailed sequence of messages passed between objects to help in understanding real-time specifications. Sequence diagrams of the system are created for three actors which are the committee member, resident, and security guard. Registration of users have to be done before they can login to the system. The user details will be stored in the database and the system will return a registration completion status message. After successful login, the committee member can perform function such as managing dashboard, profile, users, feedback, invoice, and generate report. The residents are required to activate their account before able to login to the system. After successful login, residents can view the post posted on dashboard, manage their own profile, submit feedback, and upload receipt for unpaid invoice. While the guard can update their own information and view the resident contact information.

Activity diagrams which illustrate the flow from one activity to another activity in the system are shown in Appendix B.1, B.2 and B.3. All users are required to login before accessing the system. If the entered credentials are invalid, an error message will be displayed, while if the entered credentials are valid, the committee member will be redirect to the committee dashboard, which they can perform various actions including manage dashboard post, update profile, manage users, manage payment, manage feedback, and generate report. The resident will be redirect to the resident dashboard where they can view the dashboard, update profile, view outstanding amount, upload receipt, submit feedback, and check for feedback status. The guard will be redirect to the guard panel where they can update profile and view the resident contact. Session will be terminated when user logout of the system.

3.3 Design Phase

In design phase, the requirements are transformed into detailed and complete system design specifications which help visualize the whole system before proceeding with coding of the system The overall system architecture is presented in Fig. 5. The users which included the committee members, residents, and security guards can access the system with a web browser. The web server will be receiving request from the users and send the response back to users after retrieving it from the database.

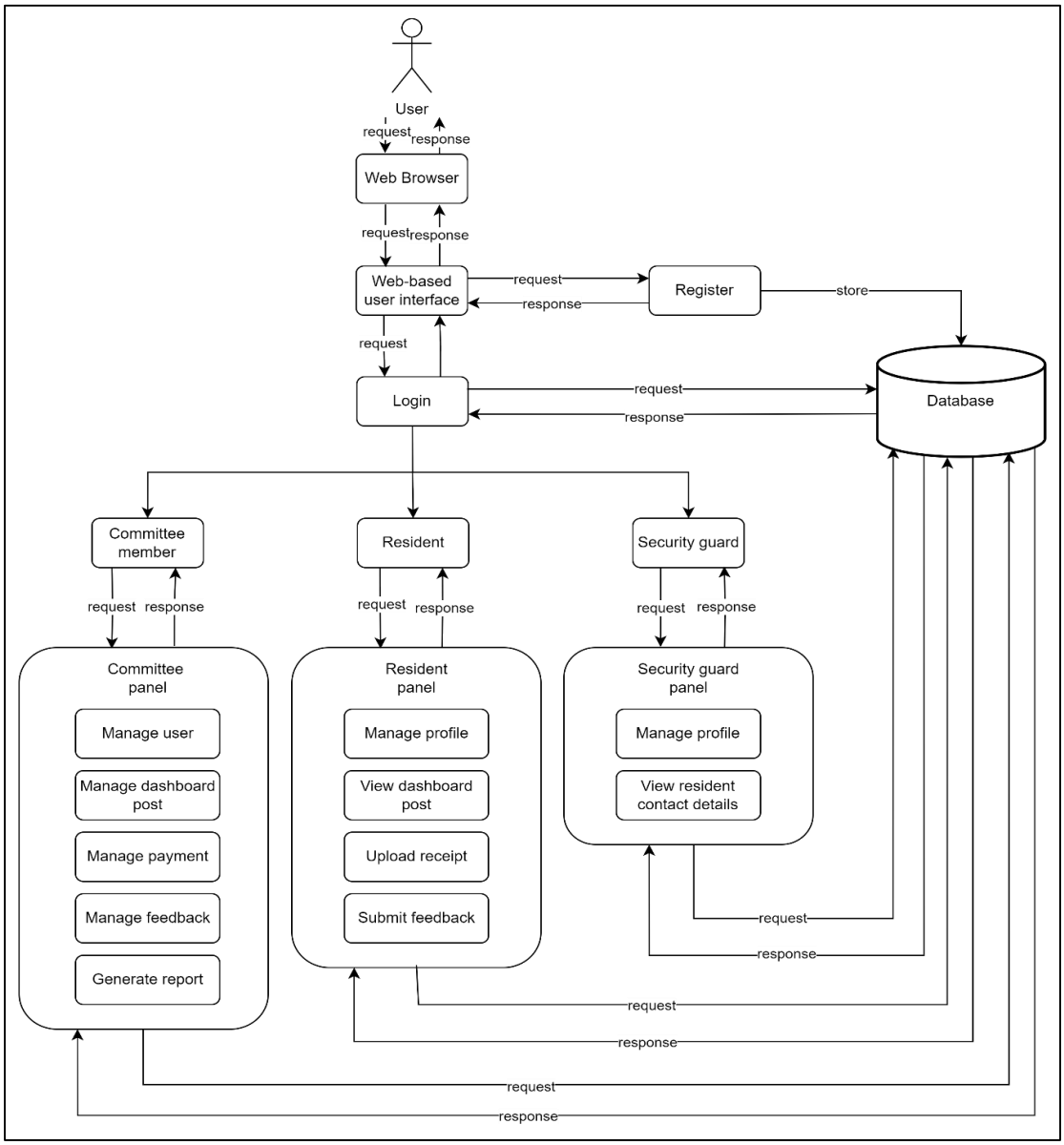


Fig. 5 *Overall system architecture of resident management system*

Access control matrix defines the access permissions between specific subjects and objects in a table. Table 4 shows the matrix that has specific access permissions defined by the user roles and the actions they can perform. The letters that are used to abbreviate the access permissions are 'C' for create access, 'R' for read access, 'U' for update access and 'D' for delete access.

Table 4 *Access control matrix table*

Role	Committee member	Resident	Guard
Manage user	C, R, U, D	-	-
Manage profile	R, U	R, U	R, U
Manage dashboard	C, R, U, D	R	-
Manage house	C, R, U, D	R	-
Manage feedback	R, U	C, R, D	-
Manage payment	C, R, U	R, U	-
Manage contact details	C, R, U, D	R, U, D	R
Generate report	C, R	-	-

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381

Fig. 6 shows the class diagram of the Resident Management System for Lestari Mansions. The resident class, committee member class, and security guard class are inherited from the user class. Each of the user sub-classes are related to other classes which included the dashboard, invoice, payment, feedback, and phone book. Two enum classes which define the status are created for invoice and feedback class.

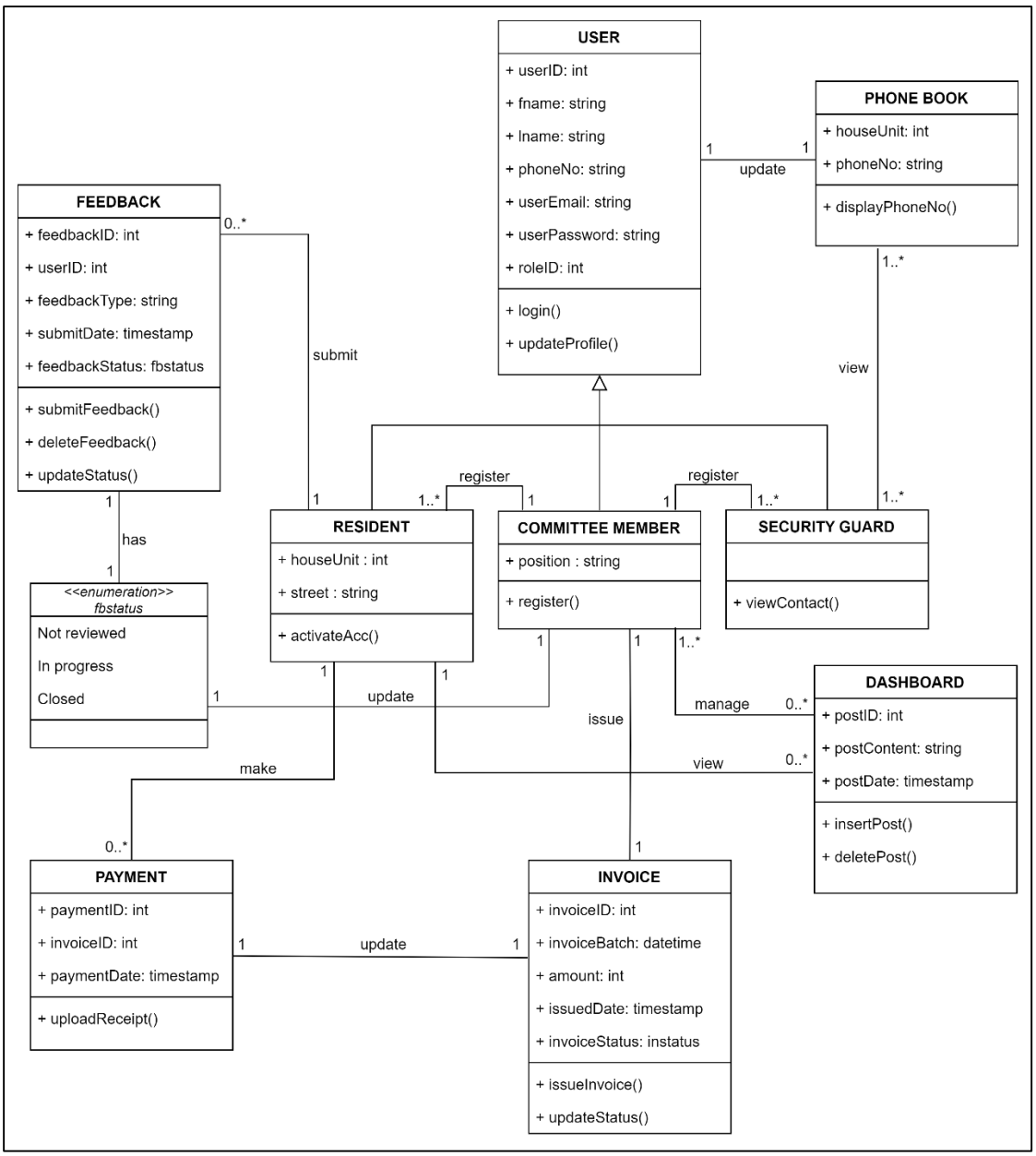


Fig. 6 *Class diagram*

3.4 Implementation Phase

The implementation phase is where the tasks associated with building the system are focused on. The system design specifications are converted into codes and testing for each module's usability are carried out with an iterative approach. Bugs are fixed immediately to ensure the smooth functioning of each module.

3.5 Testing Phase

The testing phase takes place after coding of the system is completed. Testing the system before deployment is important to ensure the system meets the specified requirements. The system is tested against the aspects of system functionality, usability, and security. Two sets of test plan which are the functional test plan and security test plan are designed to test if the actual result of each module meets the expected results. User acceptance test is also carried out to ensure the user requirements are fulfilled.

4. Result and Discussion

This section discusses the result of the implementation phase of the developed system.

4.1 Security Features

The developed system implemented several security features which aimed to prevent errors, maintain data consistency, and enhance overall security of the developed system. Fig. 7 shows the code for CAPTCHA. The users are required to complete the reCAPTCHA on the login page. It uses a secret key and the user's response token to check if the reCAPTCHA is successful.

```
private boolean verifyRecaptcha(String recaptchaResponse) throws IOException {
    String secret = "6LflR0opAAAAAICE0dDqR_gxc0r8ltmwbvWElg30";
    String url = "https://www.google.com/recaptcha/api/siteverify";
    String params = "secret=" + URLEncoder.encode(s: secret, enc: "UTF-8") +
            "&response=" + URLEncoder.encode(s: recaptchaResponse, enc: "UTF-8");

    HttpURLConnection conn = (HttpURLConnection) new URL(spec: url).openConnection();
    conn.setDoOutput(dooutput: true);
    conn.setRequestMethod(method: "POST");
    conn.setRequestProperty(key: "Content-Type", value: "application/x-www-form-urlencoded");

    try (OutputStream os = conn.getOutputStream()) {
        os.write(b: params.getBytes());
        os.flush();
    }

    try (BufferedReader br = new BufferedReader(new InputStreamReader(in: conn.getInputStream()))) {
        StringBuilder responseBuilder = new StringBuilder();
        String line;
        while ((line = br.readLine()) != null) {
            responseBuilder.append(str: line);
        }
        String response = responseBuilder.toString();
        return response.contains(s: "\"success\": true");
    }
}
```

Fig. 7 *Code for CAPTCHA*

Fig. 8 shows the code for account lockout mechanism. If users attempt to login to the account with wrong password 3 times, the user account will be locked temporarily for 15 minutes.

```
public void incrementFailedAttempts(String userEmail) {
    try (Connection con = dbConneot.createConnection()) {
        String sql = "UPDATE credentials SET failed_attempt = failed_attempt + 1 WHERE userID = (SELECT userID FROM users WHERE userEmail = ?)";
        try (PreparedStatement ps = con.prepareStatement(string:sql)) {
            ps.setString(i: 1, string:userEmail);
            ps.executeUpdate();
        }
        sql = "SELECT failed_attempt FROM credentials WHERE userID = (SELECT userID FROM users WHERE userEmail = ?)";
        try (PreparedStatement ps = con.prepareStatement(string:sql)) {
            ps.setString(i: 1, string:userEmail);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    int failedAttempts = rs.getInt(string:"failed_attempt");
                    if (failedAttempts >= MAX_FAILED_ATTEMPTS) {
                        lockAccount(userEmail);
                    }
                }
            }
        }
    }
```

Fig. 8 *Code for Account Lockout mechanism*

Fig. 9 shows the code for Two-Factor Authentication (2FA) using a 6-digit One-Time Password (OTP). The OTP will be sent to user's email and redirect the users to the 2FA page.

```
public String generateVerificationCode() {
    byte[] randomBytes = new byte[10];
    new SecureRandom().nextBytes(bytes: randomBytes);
    String verificationCode = Base64.getUrlEncoder().withoutPadding().encodeToString(src:randomBytes);
    return verificationCode.substring(beginIndex: 0, endIndex: 10);
}

String generateOtp() {
    StringBuilder otp = new StringBuilder();
    otp.append(1 + random.nextInt(bound: 9));
    for (int i = 1; i < OTP_LENGTH; i++) {
        otp.append(i: random.nextInt(bound: 10));
    }
    return otp.toString();
}
```

Fig. 9 *Code for Two-Factor Authentication (2FA)*

Fig. 10 shows the code for generating salt and hashing password. The system uses SHA-256 hashing algorithm to compute a hash of the password with the salt generated and stored it in the database.

```
private String generateSalt() {
    byte[] salt = new byte[16];
    random.nextBytes(bytes: salt);
    return Base64.getEncoder().encodeToString(src:salt);
}

private String hashPassword(String password, String salt) {
    try {
        MessageDigest md = MessageDigest.getInstance(algorithm:"SHA-256");
        md.update(input: Base64.getDecoder().decode(src:salt));
        byte[] hashedBytes = md.digest(input: password.getBytes());
        return bytesToHex(bytes: hashedBytes);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return null;
    }
}
```

Fig. 10 *Code for Password Hashing*

Fig. 11 shows the code for implementing strong password policy. The password must contains at least 8 characters, including at least one uppercase letter, one lowercase letter, one number, and one special character.

```
function validatePassword() {
    var password = document.getElementById("password").value;
    var confirmedPassword = document.getElementById("confirmed_password").value;

    if (password !== confirmedPassword) {
        alert("Password and confirmed password do not match. Please check and try again.");
        return false;
    }
    var complexityRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/;
    if (!complexityRegex.test(password)) {
        alert("Password must be at least 8 characters long and include at least one uppercase \n\
            letter, one lowercase letter, one number, and one special character.");
        return false;
    }
    return true;
}
```

Fig. 11 *Code for Strong Password Policy*

Fig. 12 shows the code for authentication filtering. Access to the page is allowed only if the user is authenticated. While the user is redirected to the login page if the user is not logged in.

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {

    HttpServletRequest httpRequest = (HttpServletRequest) request;
    HttpServletResponse httpResponse = (HttpServletResponse) response;

    HttpSession session = httpRequest.getSession(bln: false);
    String loginURI = httpRequest.getContextPath() + "/Login";

    boolean loggedIn = session != null && session.getAttribute(string:"userID") != null;
    boolean loginRequest = httpRequest.getRequestURI().equals(anObject: loginURI);

    if (loggedIn || loginRequest) {
        chain.doFilter(sr: request, sr1:response);
    } else {
        httpResponse.sendRedirect(string:loginURI);
    }
}
```

Fig. 12 *Code for Authentication Filtering*

Fig. 13 shows the code implementing Role-Based Access Control (RBAC). The filter servlet checks if the user has a specific role and allows access to certain pages. Three filter servlets are created for three different roles. The system will redirect users to an "Access Denied" page if they attempt to access pages beyond their role permissions.

```
if (loggedIn) {
    Integer roleID = (Integer) session.getAttribute(string:"roleID");
    if (roleID != null && roleID == 1) {
        chain.doFilter(sr: request, sr1:response);
    } else {
        httpResponse.sendRedirect(string:"/accessDenied.jsp");
    }
} else {
    httpResponse.sendRedirect(httpRequest.getContextPath() + "/Login");
}
```

Fig. 13 *Code for Role-Based Access Control*

Fig. 14. shows the "Access Denied" page when a user with the assigned role of “Resident” attempts to access a page which is restricted to users with the assigned role of “Committee member”. By restricting access to pages based on roles, the system maintains the confidentiality and integrity of the stored data.

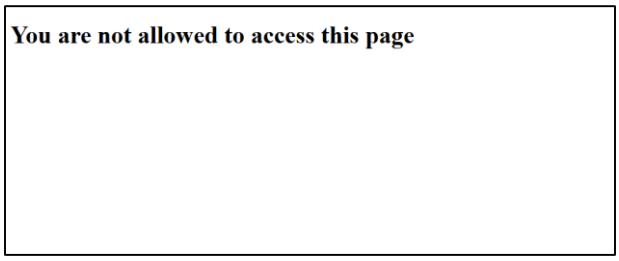


Fig. 14 *Access Denied Page*

4.2 Functional Module

The implementation of the system module within the developed system involved the creation and integration of various components to fulfill the specific functionalities and features. Fig. 15 shows the registration page for committee. The registration modules are only accessible by the committees. After inserting the details such as first name, last name, phone number, email, password, and confirm password, the user accounts will be successfully created and be able to login.

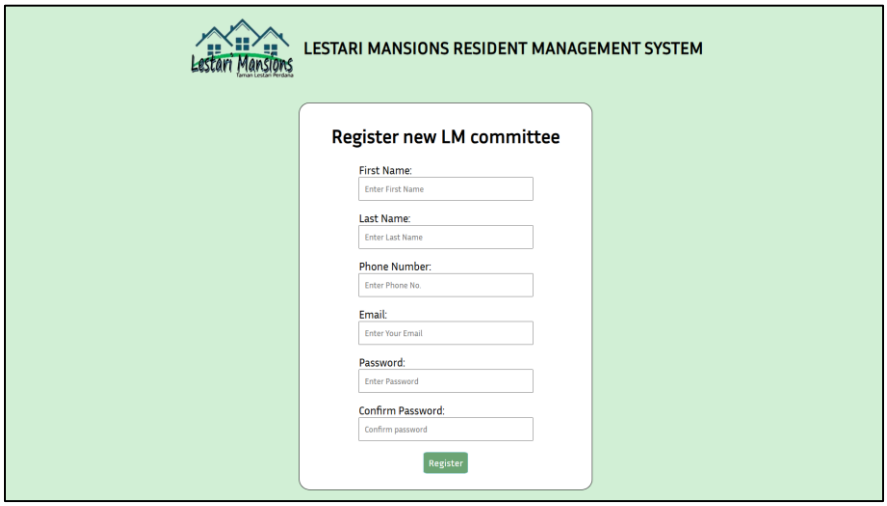


Fig. 15 *Registration page for Committee*

Fig. 16 shows the registration page for residents. The committee can register new residents by inserting the resident’s email with house unit and house street. After successfully registration, an email with activation link will be sent to the registered email address. The account activation page will require residents to input their first name, last name, phone number, password, and confirm password.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381

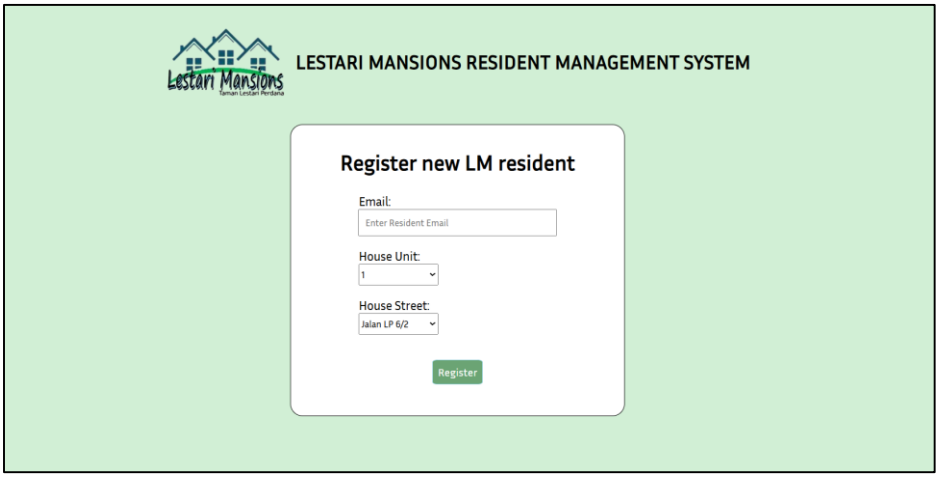


Fig. 16 *Registration page for Resident*

Fig. 17 shows the login page for all users. Users are required to enter their login credentials such as email address and password to be able to login to their account. Users are also required to complete the reCAPTCHA. Only activated users will be able to login. The users are also required to enter the one-time password (OTP) that is sent to the user's email address before accessing their user account. The system incorporates RBAC to ensure that successful logged in users are redirected based on their assigned roles, providing access to relevant pages according to their permissions.

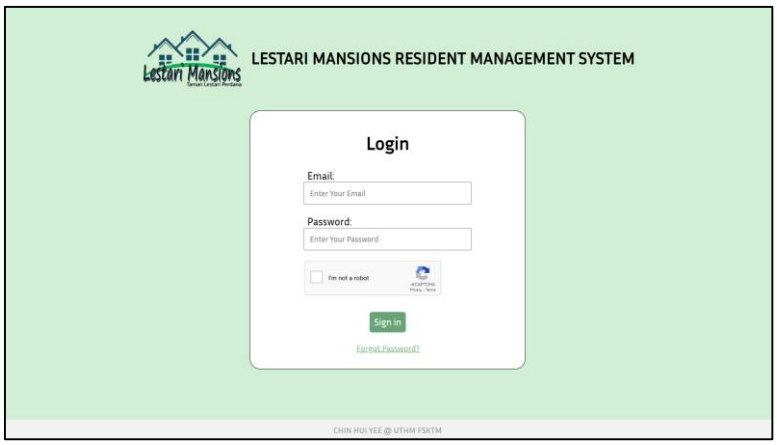


Fig. 17 *Login page*

Fig. 18 shows the change password page. The users are required to enter their old password, new password, and confirm new password. The system will first validate their old password before changing it to a new password. The system will also check the new password complexity to ensure it follows the strong password policy.

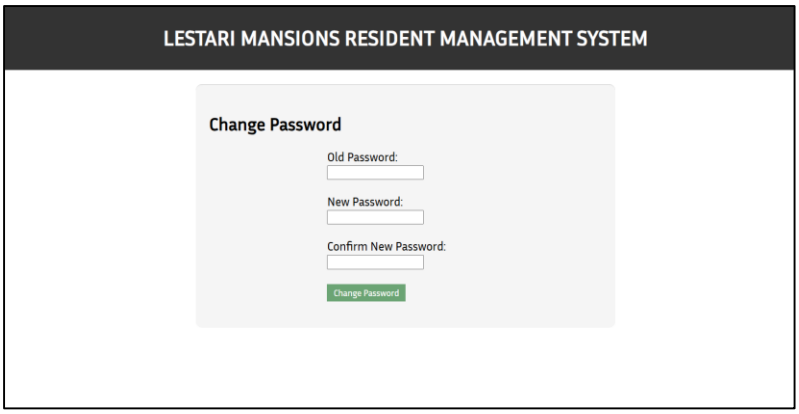


Fig. 18 *Change Password page*

Fig. 19 shows the dashboard page for committees. The committee can add new post to the dashboard and also edit the content or delete the post. Residents can view the posts added by the committee as announcements in the dashboard upon logging in to their accounts.

Fig. 19 *Committee dashboard*

Fig. 20 shows the manage users page where the list of users is displayed. The committees can navigate through different lists categorized by user roles by clicking on the tabs at the top of the page. The committee can add users accounts or delete committee and guard accounts. They can also edit the user's phone number.

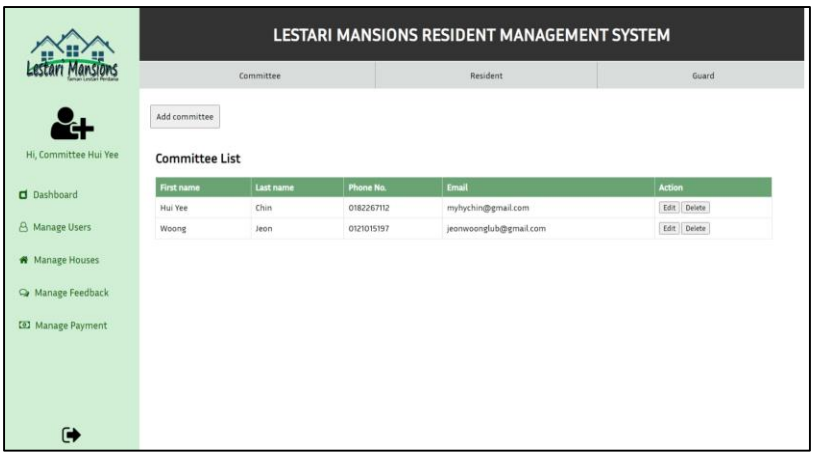


Fig. 20 *Manage Users page*

Fig. 21 shows the manage houses page where the list of occupied houses is displayed. The committee can remove houses if they are no longer occupied. If there are no other houses occupied by the users, the users will be deactivated and not be able to login to the system. The committee can also add houses for existing residents.

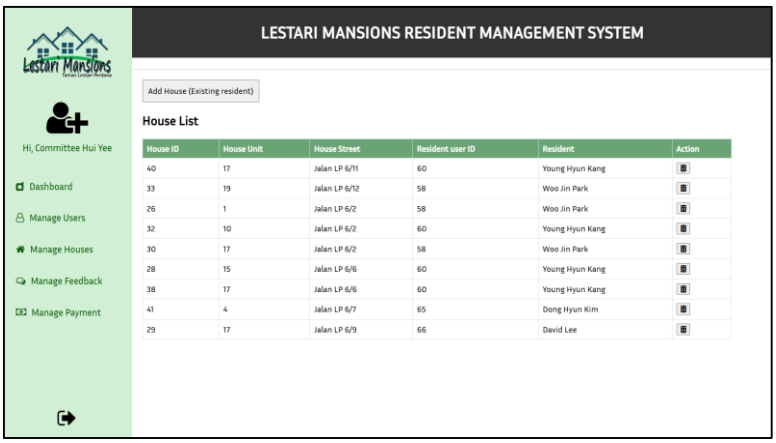


Fig. 21 *Manage Houses page*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381

Fig. 22 shows the manage houses page where the list of occupied houses is displayed. Committees can filter the feedback based on the type and status. The details of each feedback can be viewed, and the status can be updated with added remarks.

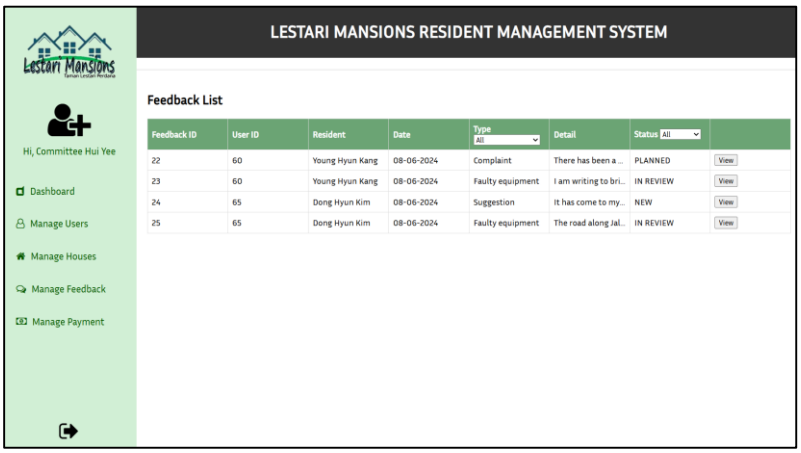


Fig. 22 *Manage Feedback page*

Fig. 23 shows the manage payment page where new invoices can be issued to residents. Committees can select the invoice batch, house, and insert the amount for the invoice. The system will check whether the house has been issued an invoice of the selected batch to avoid the creation of duplicate invoice. The committee can view the receipt uploaded by the resident and update the status of the invoice.

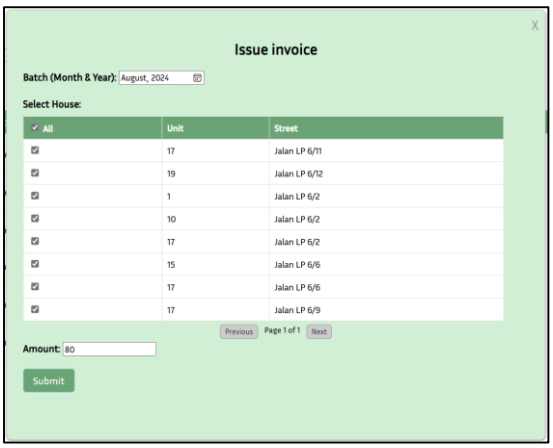
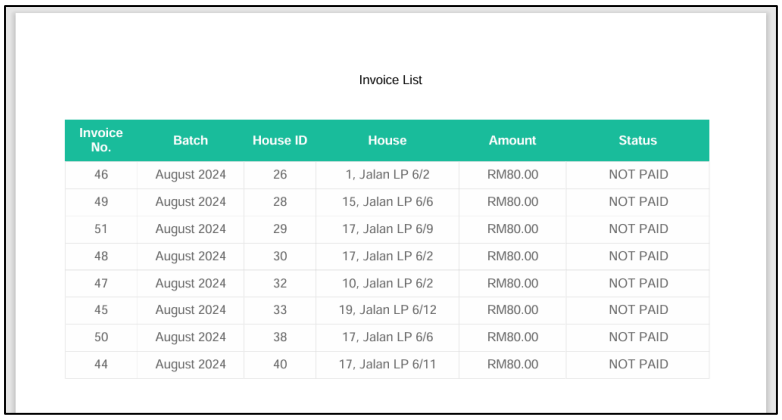


Fig. 23 *Issue Invoice*

Fig. 24 shows the downloaded invoice report as Portable Document Format (PDF). The committee can download the invoice report according to their batch.



Invoice List

Invoice No.	Batch	House ID	House	Amount	Status
46	August 2024	26	1, Jalan LP 6/2	RM80.00	NOT PAID
49	August 2024	28	15, Jalan LP 6/6	RM80.00	NOT PAID
51	August 2024	29	17, Jalan LP 6/9	RM80.00	NOT PAID
48	August 2024	30	17, Jalan LP 6/2	RM80.00	NOT PAID
47	August 2024	32	10, Jalan LP 6/2	RM80.00	NOT PAID
45	August 2024	33	19, Jalan LP 6/12	RM80.00	NOT PAID
50	August 2024	38	17, Jalan LP 6/6	RM80.00	NOT PAID
44	August 2024	40	17, Jalan LP 6/11	RM80.00	NOT PAID

Fig. 24 *Invoice Report*

Fig. 25 shows the feedback page for residents. The feedback that the resident submitted will is listed on this page. The residents can add new feedback and view the feedback details to see any updates from the committee. The feedback can also be revoked by clicking on the bin icon if the feedback status is "NEW".

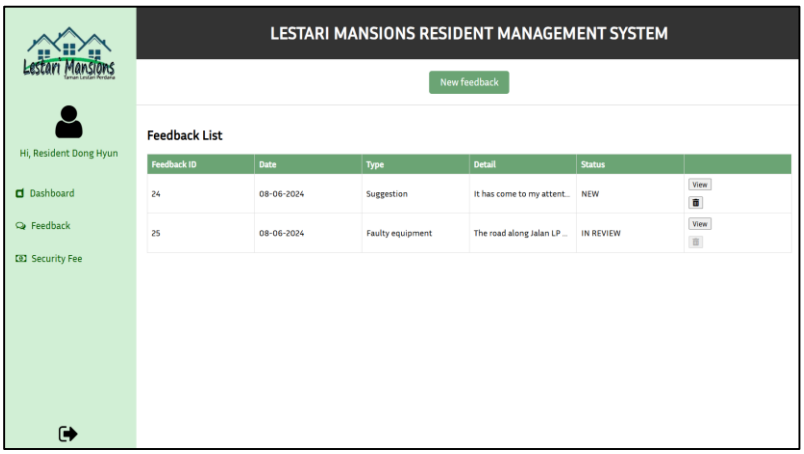


Fig. 25 *Feedback page*

Fig. 26 shows the security fee page. The list of invoices issued by the committee is listed with their details. The total unpaid amount is shown in the red box at the top right. The residents can view the details of the invoice and can upload a receipt to be reviewed by the committee. Once submitted the receipt, the status of the invoice will be changed to "PENDING CONFIRMATION".

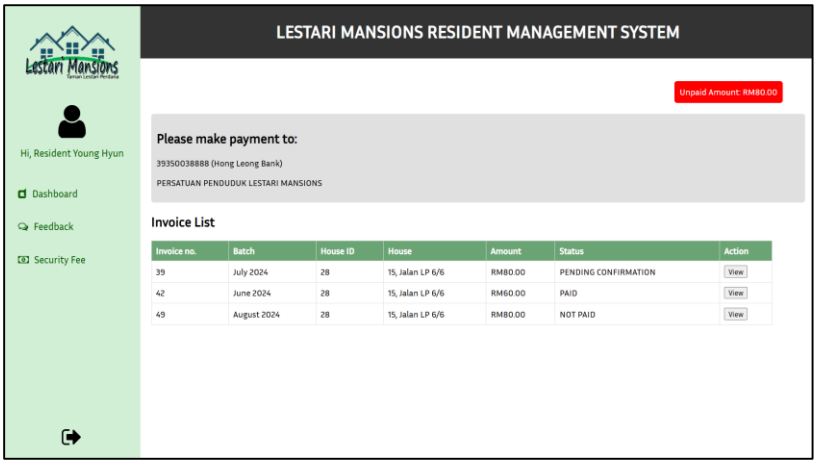


Fig. 26 *Security Fee page*

Fig. 27 shows the phone book page. This module is only available for the guards where the residents with their houses and phone number is shown. The guard can search for the record using the search bar.

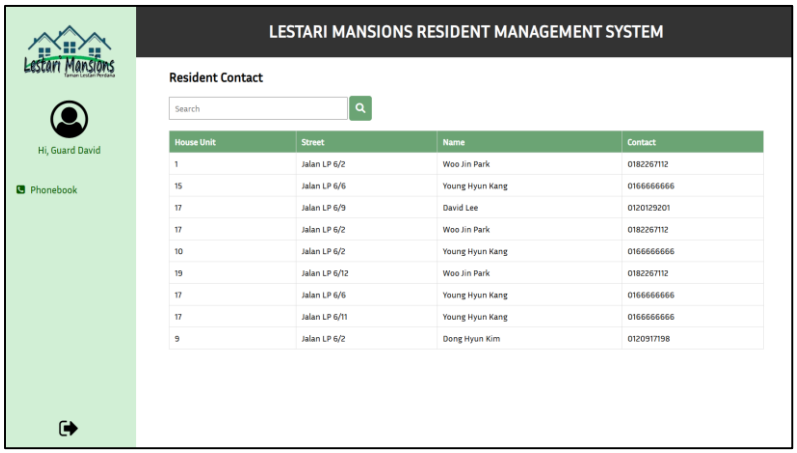


Fig. 27 *Phonebook page*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381

4.3 Security Checklist Result

Security testing is conducted to ensure that the security procedures are implemented to protect the system information. Table 5 shows the security checklist result of the developed system. The developed system managed to pass all the checklist.

Table 5 *Security Checklist Result*

No.	Check List	Actual Result
1	Ensure error message does not directly indicate the incorrect authentication data.	Pass
2	Enforce strong password policy which requires a minimum of eight characters with at least one uppercase letter, at least one lowercase letter, at least one number, and at least one special character.	Pass
3	Password should be masked in the text box.	Pass
4	Ensure users can only perform actions based on their assigned role.	Pass
5	Enforce two-factor authentication in order to add an additional layer of security to authentication process.	Pass
6	Ensure user is not allowed to login using an expired OTP.	Pass
7	Ensure user is not allowed to reset password using an expired link or used link.	Pass
8	Ensure user account is lock for 15 minutes after 3 failed login attempts.	Pass
9	Password in database is hashed with SHA256 algorithm.	Pass

4.4 User Acceptance Result

User acceptance testing is conducted by allowing the users to test the developed system. Two committee members and three residents are involved in the testing. A google form to rate satisfaction of the functional module is sent to the users after testing session.

Fig. 29 shows the user acceptance testing result. All five (5) users are very satisfied with the function of users are able to logout, users are able to view and edit their own personal details, users are able to change password, committee members are able to post announcement on dashboard, and security guards are able to view residents phone number. While four (4) users are very satisfied and one (1) user is satisfied with the function of committee members are able to register users based on the users' role, users are able to login using correct username and password, and committee members are able to generate payment status report. Three (3) users are very satisfied, one (1) user is satisfied, and one (1) user is ok with the function of residents are able to upload payment receipt using the system and committee members are able to generate payment status report.

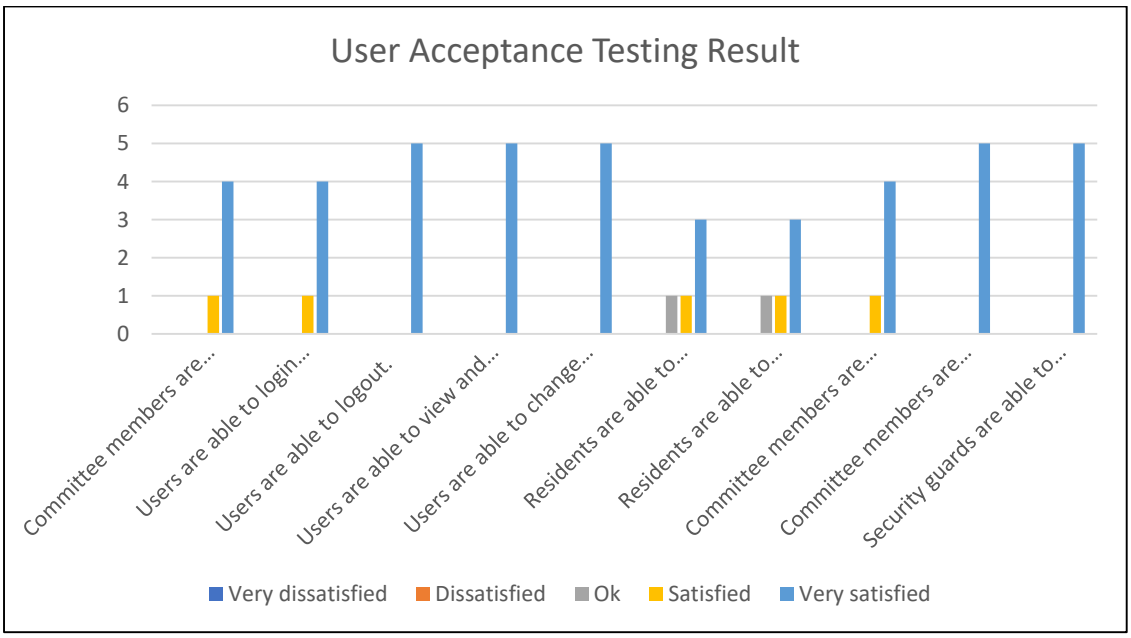


Fig. 29 *User Acceptance Testing Result*

5. Conclusion

The development of the Resident Management System for Lestari Mansions with Role-Based Access Control has been completed. The implementation of Role-Based Control (RBAC) in the developed system enhances data security and operational efficiency. For future implementation, the system can add a notification function to notify the residents when new invoices are issued to them and when their submitted feedback has any changes in status. The system can also add engagement features to the announcement module such as allowing the residents to react to post and write comments under the post. The system should offer alternative options for users to receive OTPs and password reset links such as via SMS in case email is not available or accessible to the user.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design****: H. Y. Chin, C.F Mohd Foozy;* ***data collection****: H. Y. Chin, C.F Mohd Foozy;* ***analysis and interpretation of results****: H. Y. Chin, C.F Mohd Foozy;* ***draft manuscript preparation****: H. Y. Chin, C.F Mohd Foozy. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] N. Abd Razak, U. Ujang, S. Mohd Salleh, S. Azri, and T. L. Choon, "DEVELOPMENT OF MOBILE APPLICATION FOR GATED AND GUARDED COMMUNITY MANAGEMENT," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLII-4/W16, pp. 17–22, 2019, doi: 10.5194/isprs-archives-XLII-4-W16-17-2019.

[2] B. Ley *et al.*, "Analysis of erroneous data entries in paper based and electronic data collection," *BMC Res. Notes*, vol. 12, no. 1, p. 537, 2019, doi: 10.1186/s13104-019-4574-8.

[3] Y. Lee and K. M. Lee, "Blockchain-Based RBAC for User Authentication with Anonymity," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, in RACS '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 289–294. doi: 10.1145/3338840.3355673.

[4] P. Kamal, "Security of Password Hashing in Cloud," *J. Inf. Secur.*, vol. 10, pp. 45–68, 2019, doi: 10.4236/jis.2019.102003.

[5] G. Nyame and Z. Qin, "Precursors of Role-Based Access Control Design in KMS: A Conceptual Framework," *Information*, vol. 11, no. 6, 2020, doi: 10.3390/info11060334.

[6] M. Yıldırım and I. Mackie, "Encouraging users to improve password security and memorability," *Int. J. Inf. Secur.*, vol. 18, no. 6, pp. 741–759, 2019, doi: 10.1007/s10207-019-00429-y.

[7] A. Mohammed and N. Varol, "A Review Paper on Cryptography," 2019, pp. 1–6. doi: 10.1109/ISDFS.2019.8757514.

[8] "RESIDENT EASY DECISION SYSTEM (MYREDS)." Accessed: Nov. 25, 2023. [Online]. Available: https://www.myreds.com.my/

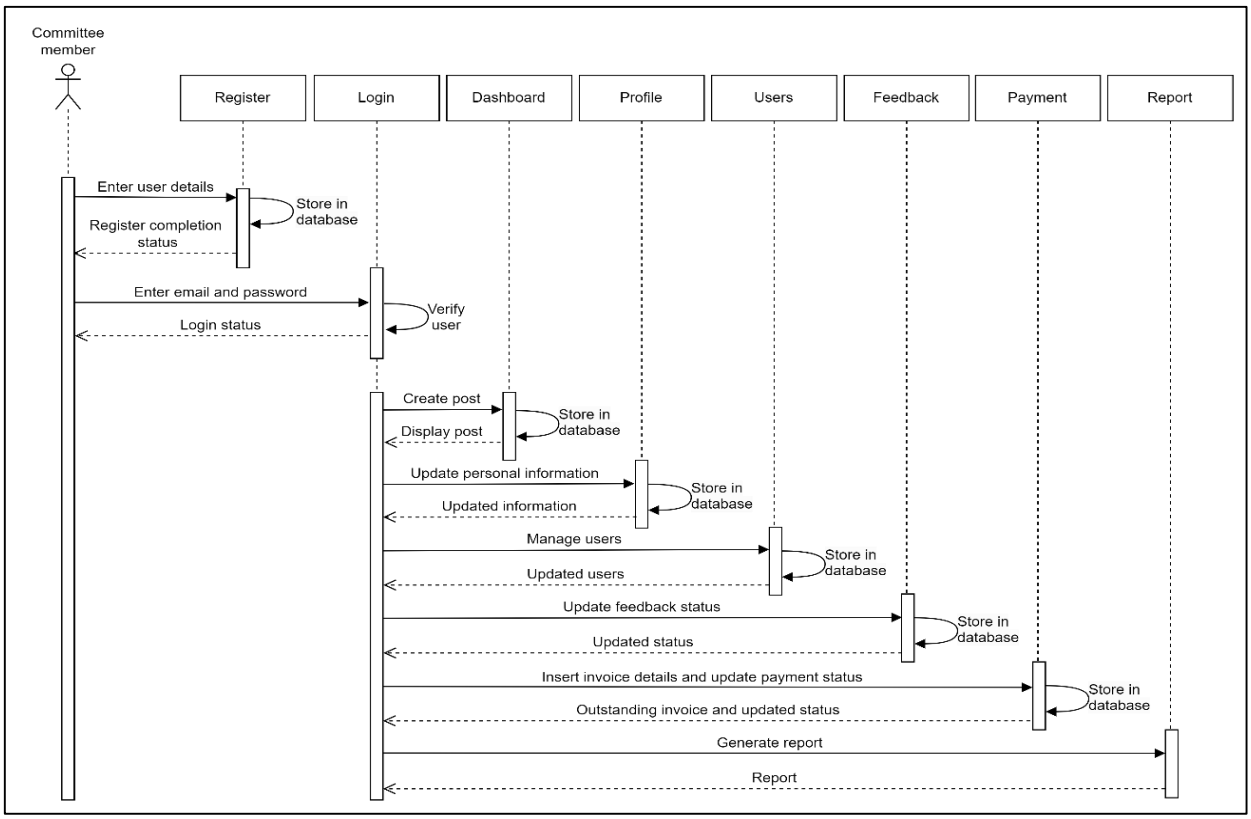
[9] "M4U Home Resident System." Accessed: Nov. 25, 2023. [Online]. Available: https://www.manage4u.com.my/residential-management-system

[10] "INEIGHBOUR RESIDENT APP Smart Residential Community." Accessed: Nov. 25, 2023. [Online]. Available: https://www.i-neighbour.com/resident-app/

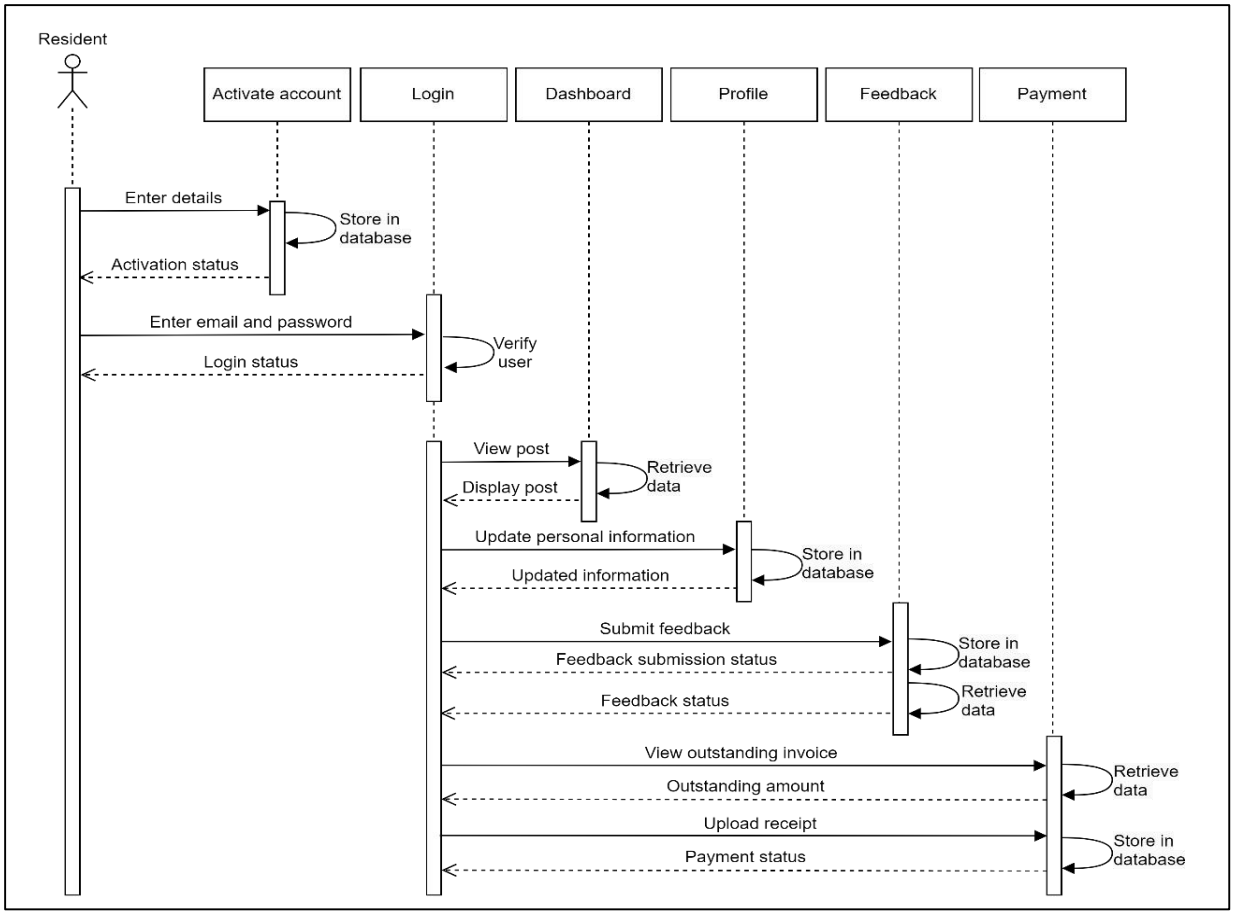
[11] S. Alsaqqa, S. Sawalha, and H. Abdel-Nabi, "Agile Software Development: Methodologies and Trends," *Int. J. Interact. Mob. Technol.*, vol. 14, no. 11, pp. 246–270, Jul. 2020.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381

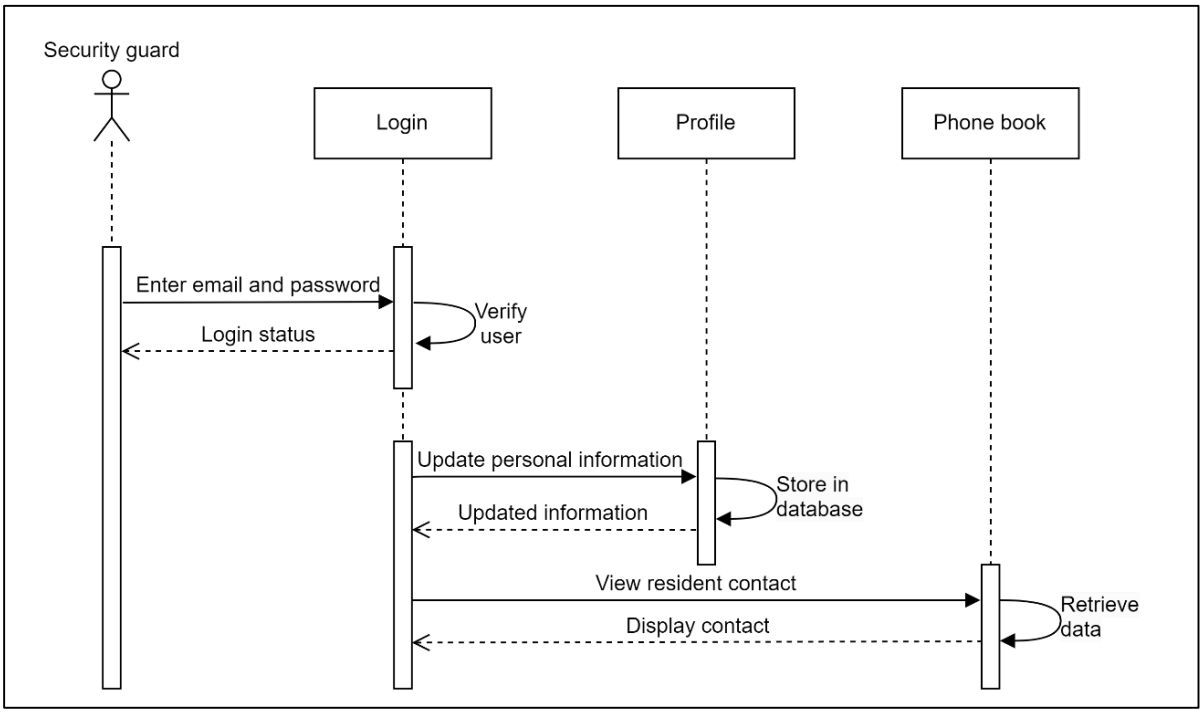
Appendix A: Sequence Diagram



Appendix A.1: Sequence diagram of committee member

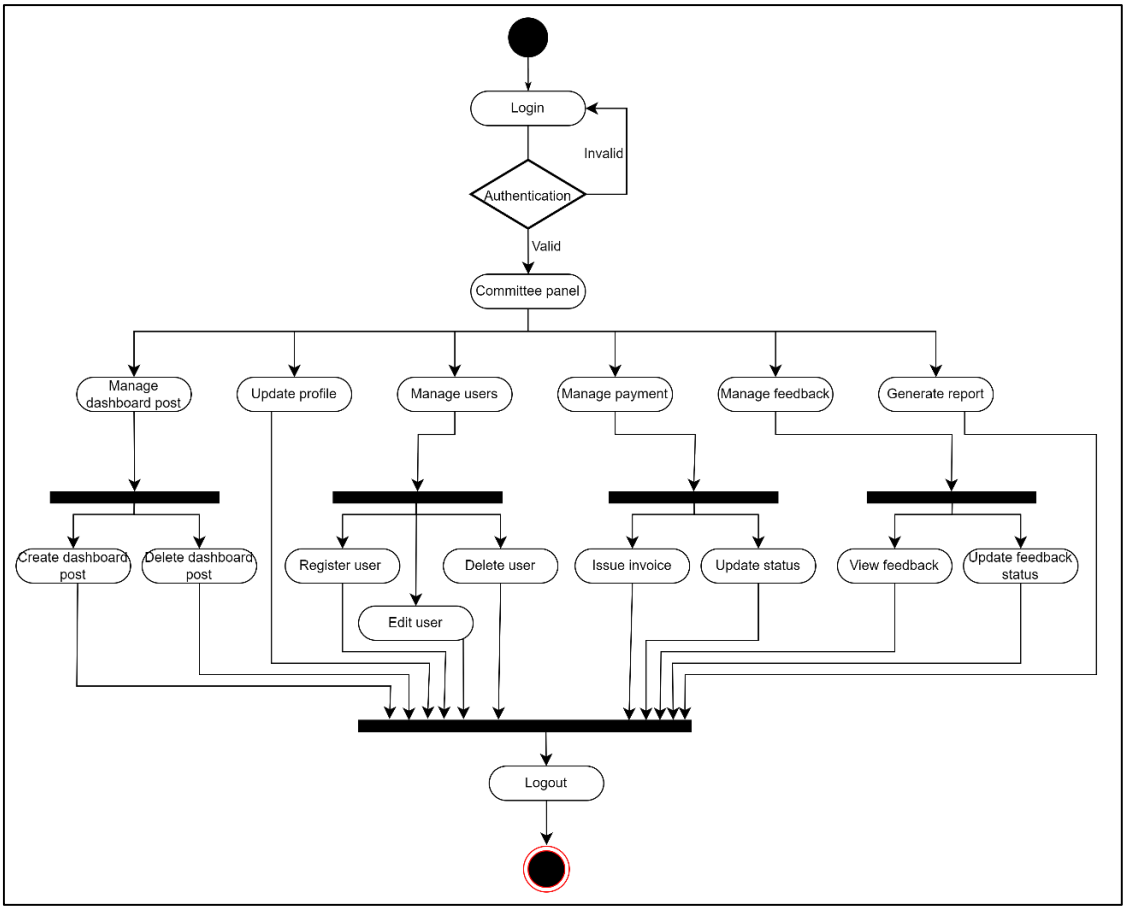


Appendix A.2: Sequence diagram of resident



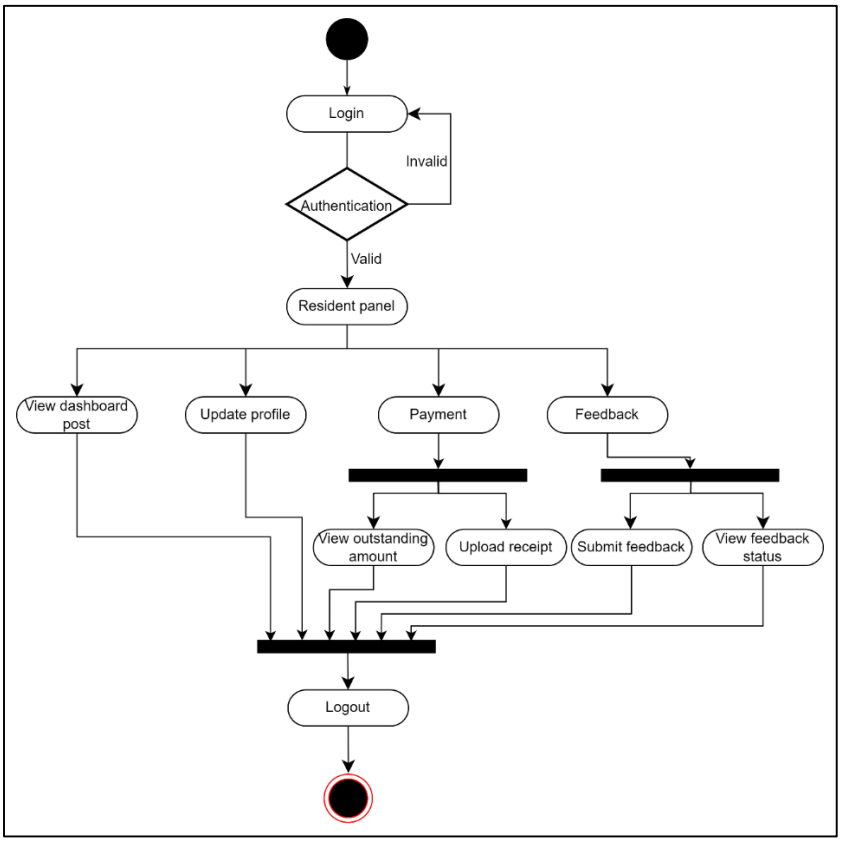
Appendix A.3: Sequence diagram of security guard

Appendix B: Activity Diagram

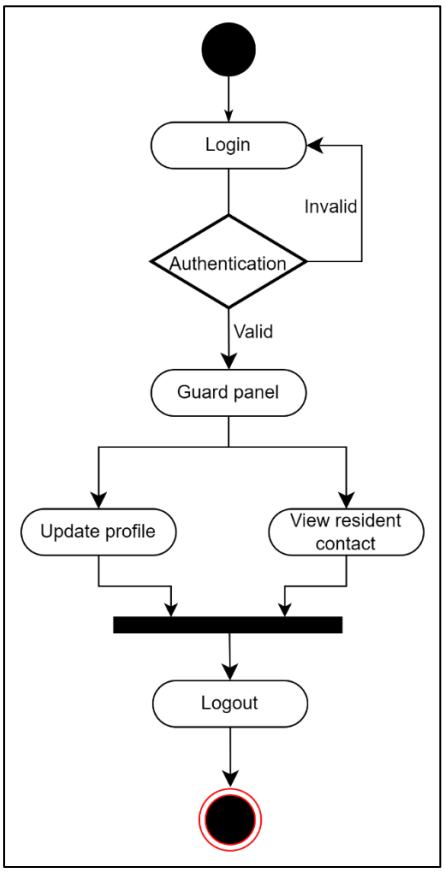


Appendix B.1: Activity diagram of committee member

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 362-381



Appendix B.2: Activity diagram of resident



Appendix B.3: Activity diagram of security guard