Building Maintenance Management System

Requirements Determination and Use Case Analysis / System Proposal / Analysis Phase   
(Homework No.2)

Project team: Team 06

Instructor: Dr. Araz Yusubov

Submitted in partial fulfillment of the requirements of the INFT 2303: Systems Analysis and Design course project

|  |  |
| --- | --- |
| GitHub repository | https://github.com/ADA-SITE-INFT2303-2023-Spring/<name of the repository> |
| Version date | Version information |
| 22.03.23 | Initial draft |
| 01.04.2023 | Final Version |

|  |  |
| --- | --- |
| Other documents in the package | |
| File name | Brief description of the document |
| Use Case Diagram | This file shows the picture of USE CASE diagram of the system |
| <File name> | <Description of the document> |

|  |  |  |
| --- | --- | --- |
| Team member | Contribution to this homework (NOT the project) | Estimated % |
| Kazimli Ayatay | Nonfunctional Requirements, Use Case 05 and 06, Use Case Diagram | 25% |
| Latif Asgarov | Requirements Definition, Use Case 02, 09, and 10, Use Case Diagram | 25% |
| Fahmin Mahili | Functional Requirements, External Actor Descriptions, Use Case Diagram, Use Case 01, 03, and 04 | 25% |
| Nijat Shirinov | Introduction, definitions, design constrains, Use Case 07 and 08, Use Case Diagram, references | 25% |
| Arzuman Alakbarli | Nothing | 0% |

# Table of Contents

Contents

[Table of Contents 3](#_Toc131267262)

[1. Introduction 3](#_Toc131267263)

[Definitions 4](#_Toc131267264)

[Requirements Definition 4](#_Toc131267265)

[Functional Requirements 5](#_Toc131267266)

[Nonfunctional Requirements 5](#_Toc131267267)

[Design Constraints 6](#_Toc131267268)

[2. Use Case Analysis 7](#_Toc131267269)

[External Actor Descriptions 7](#_Toc131267270)

[User Story Descriptions 7](#_Toc131267271)

[Use Case Diagram 8](#_Toc131267272)

[Use Case Descriptions 9](#_Toc131267273)

[Use Case 1 9](#_Toc131267274)

[Use Case 2 10](#_Toc131267275)

[Use Case 3 11](#_Toc131267276)

[Use Case 4 12](#_Toc131267277)

[Use Case 5 13](#_Toc131267278)

[Use Case 6 14](#_Toc131267279)

[Use Case 7 15](#_Toc131267280)

[Use Case 8 16](#_Toc131267281)

[Use Case 9 17](#_Toc131267282)

[Use Case 10 19](#_Toc131267283)

[References 21](#_Toc131267284)

# Introduction

Introduction

The main purpose of this system is creating the opportunity for the apartment owners so they can manage their maintenance and expense operations from the single platform. This document was created to cover all of the requirements that must be connected to develop the system. During the next phase this document will be used for the project’s development process.

Nowadays, apartment owners pay and manage their maintenance as direct to the employee of the building they live in if they are present in their apartment at the same time. Owners can set up these processes wherever they are thanks to our application.

Because our system is online, card payments are possible.

The system will provide an easy-to-use interface for apartment owners to manage their maintenance and expenses. The platform will include features like a dashboard where users can view maintenance requests, payment history, and upcoming expenses. The platform will also allow users to view and manage their payment methods.

We recognize the importance of security in payment transactions and will take the necessary steps to ensure that all transactions are secure. Our platform will be regularly updated to ensure its security against any potential threats.

## Definitions

|  |  |
| --- | --- |
| Term | Definition |
| Use Case  Trigger  Use Case Diagram | Formal way of representing the way in which a business system interacts with its environment.  The event that causes the use case to begin  This links an actor with the use case(s) with which interacts |
| Pre-condition  Post Condition  Actor  Priority | Define what must be complete before beginning the use case.  Define what is complete when the use case ends.  Refers to a person, another, system or a hardware device interacts with the system.  This assigned to indicate the relative significance |

# Requirements Definition

* The system shall let the users create an account to access the system. The system should support different authentication methods such as email and password or social media login.
* The system shall have a dashboard so that users can see an overview of their maintenance requests, payment history, upcoming expenses, and any other relevant information.
* Users must be able to submit maintenance requests through the application, with the option to provide images or videos that further explain the problem.
* The system shall allow users to manage their payment methods, such as adding and removing payment options, analyzing payment histories, and establishing automatic payments.
* Every money transaction should be secure, and user information should be as well. To safeguard user data and hinder illegal access, the system should employ encryption and other security measures.
* Apartment owners should receive messages from the system whenever maintenance requests are filed or answered and if payments are received or become past due.
* The system shall have a mobile application which provides IOS and Android operating system. The features and functionality of the web application should be replicated in the mobile application.
* A huge number of users and transactions should be supported by the system, which should be scalable. As the user base expands, the system should be built to accommodate more traffic and data storage needs.
* The system should enable apartment owners to comment on and rate the upkeep performed in their residences. The feedback and ratings offered by the apartment owners should be saved by the system, displayed to the maintenance team, and used to raise the caliber of services offered by the system.

## Functional Requirements

**Process-oriented:**

• The system shall allow apartment owners to create a new maintenance request and assign it to the building maintenance staff.

• The system shall notify apartment owners about the status of their maintenance requests, for example, whenever the requests are assigned, active, or terminated.

• The system shall allow apartment owners to see and pay their maintenance costs with the help of the platform.

• The system shall provide building maintenance staff with a dashboard for assigning, updating, completing, and tracking maintenance requests.

• The system shall provide building managers with a dashboard to manage building maintenance, expenses, and staff.

• The system shall allow building managers to distribute maintenance requests among individual maintenance staff members in accordance with their qualifications and availabilities.

• The system shall allow building managers to approve or reject maintenance requests submitted by apartment owners.

• The system shall allow administrators to handle apartment and building information, including adding and removing them, as well as creating and deleting user accounts.

• The system shall allow administrators to manage payment information, and especially transactions, such as refunding payments or canceling bills.

**Information-oriented:**

• The system shall contain a list of all buildings and the apartments they are connected to, together with information about maintenance costs and history.

• The system shall store user information, such as apartment owner and building manager contact information and login credentials.

• The system shall store maintenance staff information, such as their skills, qualifications and availability.

• The system shall store payment information securely and provide apartment owners with the ability to manage their payment methods.

• The system must track and store all maintenance expenses for each apartment.

• The system shall provide reports for the building management and administrators, indicating maintenance costs by building, maintenance requests, and payment history by apartment.

• The system shall have a search capability for building managers and administrators to quickly find and manage apartments, maintenance requests, and user accounts.

• The system shall have real-time updates on the status of maintenance requests and payments, such as notifying building managers, administrators, maintenance staff, and apartment owners.

[[1]](#footnote-1)

## Nonfunctional Requirements

Operational Requirements:

• The system shall be accessible from any device with an internet connection.

• The system shall be available for any browser.

• The system shall have a user-friendly interface, requiring minimal knowledge for users.

• The system shall support multiple payment methods, such as PayPal and Apple Pay

• The system should provide users with the ability to customize their dashboard and user settings.

Security Requirements:

• The system shall use industry-standard encryption to protect all user’s data.

• The system shall enforce strong password requirements, including minimum length and complexity.

• The system shall have a user authorization mechanism to prevent unauthorized access.

• The system shall have an automated backup and recovery plan to ensure that data is not lost in case of a system’s failure.

• The system shall allow apartment owners to see only personal payments and information.

• The system shall have virus protection.

Performance Requirements:

• The system shall be available 24/7

• The system shall support 300 simultaneous users on weekdays without any degradation, and 150 simultaneous users on weekends.

• The system shall response within 3 seconds for any user interaction.

• The system shall have a maximum page load time of 5 seconds.

• The system shall be able to handle a maximum of 100 maintenance requests per day.

• The system shall be able to scale horizontally and vertically.

Cultural and Political Requirements:

• The system shall be designed to adapt users from diverse cultural backgrounds by including support for multiple languages.

• The system shall be compliant with all relevant data protection and privacy laws.

• The system shall be designed to minimize any potential cultural bias in the user interface.

• The system shall not collect any personally identifiable information without the permission of the user.

• The system shall not display any content that is considered discriminatory.

[[2]](#footnote-2)

## Design Constraints

The system is designed to work in conditions such as low bandwidth and different network issues.

For the owners to manage their maintenance easily, the system should have a simple user interface.

If the payment processing features are smooth and dependable, users should be able to pay their expenses with credit cards, bank cards, and bank transfers.

The system must be designed to handle a large number of users and operations while remaining scalable in terms of performance and reliability.

# Use Case Analysis

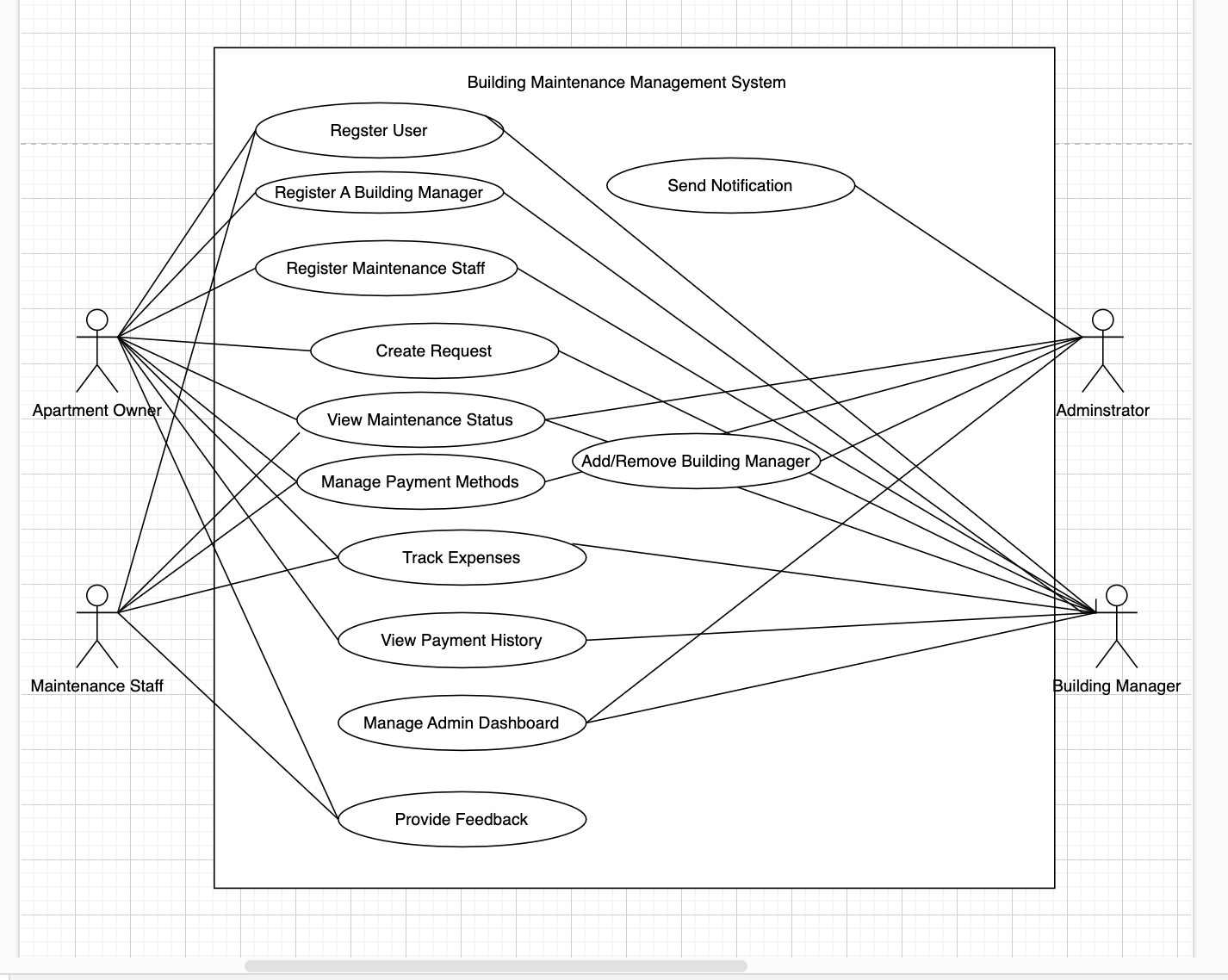
## External Actor Descriptions

1. Apartment Owner – An apartment owner is a user who owns an apartment and utilizes the system to keep track of their costs, and maintenance for an apartment.
2. Maintenance Staff - Apartment maintenance and repairs are the responsibility of the maintenance staff in response to requests from apartment owners.
3. Building Manager - The maintenance and upkeep of the entire structure of the building are the responsibility of the building manager.
4. Administrator - The administrator is in charge of handling the dashboard, managing payment transactions, adding, and removing building managers, and other aspects of the system.

## User Story Descriptions

|  |  |  |
| --- | --- | --- |
| User story name | Description | Release |
| Register user | In this use case, the procedure for creating a new user to the Building Maintenance Management System is described. Users register on the system by submitting their personal, and apartments information. | R1 |
| View Payment History  Send notification  Manage admin dashboard  Create Request  View Maintenance Status  Manage Payment Methods  Track Expenses  Add/Remove Building Manager  Give feedback and rating  Register a Building Manager  Register a Maintenance Staff | This use case allows the apartment owner to view payment history on the application, including amount, purpose of the transaction and date.  In this use case, users can receive notifications from the System regarding upcoming maintenance tasks, pending payments, and other significant occurrences.  The dashboard can be controlled by the system administrator and building manager to observe and manage all system activity, such as pending maintenance requests, payment status, and user management.  This use case describes how an apartment owner or building manager can create a maintenance request.  This use case describes how an apartment owner and building manager can view the status of their maintenance requests.  In this use case, apartment owners manage their payment methods such as adding or deleting credit, and debit cards.  In this use case, the past and upcoming expenses can be seen by the users including utility bills, maintenance charges, and other expenses.  For the purpose of managing building bills and maintenance on behalf of apartment owners, this use case enables the system administrator to add or remove building managers from the system.  The system allows apartment owners to provide feedback and rating on the repairs made to their apartments, and this will increase quality of the services which is provided by the system. Apartment owners, maintenance staff and system administrators can see the feedback and rating on control panel  The Building Manager provides personal and building information to the System upon registering and becoming a user.  Maintenance Staff Applicant shall provide personal and employment information to the System upon registering and becoming a Maintenance Staff. | R2  R1  R1  R1  R2  R1  R1  R1  R1  R1  R1 |

### Use Case Diagram



[[3]](#footnote-3)

## Use Case Descriptions

### Use Case 1

|  |  |
| --- | --- |
| **Use Case Number:** | UC-01 |
| **Use Case Name:** | Register user |
| **Actor(s):** | Apartment Owners |
| **Description:** | In this use case, the procedure for creating a new user to the Building Maintenance Management System is described. Users register on the system by submitting their personal, and apartments information. |
| **Priority (Release)** | R1 |
| **Trigger:** | External – The user registers on the system. |
| **Pre-condition(s):** | * The URL of the website is accessible to the User. * The system has been installed and configured. * The system is open for registrations. |
| **Main (Success) Flow:** | 1. The user picks the option to register for the system.  2. The system asks personal information, including First name, Last Name, Date of birth, Email address, and Phone number.  3. The user enters personal information.  4. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 2 to fill personal information again correctly.   5. The system asks for apartment details, including the building name, floor, and apartment number.  6. The user enters apartment details.  7. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 5 to fill apartment details again correctly.   8. The system asks for login information, including a username and password.  9. The user enters login information.  10. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 8 to fill login information again correctly.   11. The system displays a confirmation of registration. |
| **Alternate Flows:** | **Alternate Flow #1**  The system shall provide an option to cancel the registration after step 2, and the steps would be as follows:  1. During the Registration, the user picks the option to cancel registration.  2. The system asks confirmation to cancel registration.  3. The user confirms the intent.  4. The system returns the user to the home menu. |
| **Post Condition:** | **Main Flow:**   * The registration has been successfully completed by the user with their personal and apartment information on the system. * All information about the user has been stored successfully in the system.   **Alternate Flow #1**   * The registration was not completed by the user. * Information about the user was not saved in the system. |
| **Requirements:** | 1. The user shall register with personal and apartment information on the system.  2. The user shall have an option to cancel registration during the registration process.  3. The system shall validate all information has been entered by the user for registration |

### Use Case 2

|  |  |
| --- | --- |
| **Use Case Number:** | UC-02 |
| **Use Case Name:** | View Payment History |
| **Actor(s):** | Apartment Owner |
| **Description:** | This use case allows the apartment owner to view payment history on the application, including amount, purpose of the transaction and date. |
| **Priority (Release)** | R2 |
| **Trigger:** | Temporal- Apartment owner wants to see payments which was paid 3 month ago. |
| **Pre-condition(s):** | The system must contain and store the payment history. |
| **Main (Success) Flow:** | 1. Apartment owner logs in the system. 2. The system verifies the credentials of the unit owner. 3. The dashboard of the apartment owner is shown by the system. 4. The owner of the apartment goes to the dashboard's section for payment history. 5. The system shows the apartment owner's payment history, including each transaction's date, total, and goal. 6. Apartment owner logs out from history section when finished |
| **Alternate Flows:** | Alternate Flow#1: In Step 5, apartment owner may want to view payment history in an interval of time. Then, the following occur:   * He chooses the initial date. * He chooses final date. * Click “Show payment history in ‘01/12/22-01/12/23’ interval” |
| **Post Condition:** | * The apartment owner was able to access their platform payment history. |
| **Requirements:** | * All payments made by apartment owners should be tracked by the system. * Payment history should be accurately displayed by the system in a simple-to-understand style. * The date, sum, and reason for the transaction should all be listed in the payment history. * The platform should allow apartment owners to view their payment history. |

### Use Case 3

|  |  |
| --- | --- |
| **Use Case Number:** | UC- 03 |
| **Use Case Name:** | Send notification |
| **Actor(s):** | Apartment Owners, Building Managers, and Maintenance staffs |
| **Description:** | In this use case, users can receive notifications from the System regarding upcoming maintenance tasks, pending payments, and other significant occurrences. |
| **Priority (Release)** | R1 |
| **Trigger:** | External - The user asks to be notified |
| **Pre-condition(s):** | * The system has been installed and configured. * The user enters the system. * The user's contact information has been submitted in order to receive notifications. |
| **Main (Success) Flow:** | 1. The user picks the option to get a notification.  2. System asks for notification details, including notification type (maintenance activity, or payment or other), notification message, and notification recipient).  3. The user enters notification details.  4. The system validates required information has been entered.   * If information is invalid, System displays an error message. It returns the user to Step 2 to fill notification details again correctly.   5. The user confirms the notification settings.  6. The system sends the notification to the specified recipient.  7. The system displays a confirmation message for sending notification to the user. |
| **Alternate Flows:** | Alternate Flow #1: The user cancels notification settings.  1. The user picks the option to cancel notification settings.  2. The system asks confirmation to cancel the notification settings.  3. The user confirms intent.  4. The system returns the user to the home menu.  Alternate Flow #2: The notification is not sent because of failure in the system.  1. The system identifies a failure in the process of sending notification.  2. The system displays an error message for failing notification to the user.  3. The user decides to attempt sending the notification again.  4. The notification is resent by the system. |
| **Post Condition:** | Main Flow:   * The notification has been successfully sent to the user. * The notification information has been stored by the system.   Alternate Flow #1:   * The notification was not sent by the system. * The notification information was not stored in the system. |
| **Requirements:** | 1. The user shall ask for getting a notification.  2. The user shall enter the necessary information for sending the notification.  3. The system shall validate all information provided by the user for sending the notification.  4. The user shall be able to cancel the notification request during the process.  5. The system shall store all the notification details. |

### Use Case 4

|  |  |
| --- | --- |
| **Use Case Number:** | UC-04 |
| **Use Case Name:** | Manage admin dashboard |
| **Actor(s):** | Administrators, and Building Managers. |
| **Description:** | The dashboard can be controlled by the system administrator and building manager to observe and manage all system activity, such as pending maintenance requests, payment status, and user management. |
| **Priority (Release)** | R1 |
| **Trigger:** | External - login into the system as the building manager or system administrator. |
| **Pre-condition(s):** | * The system has been installed and configured. * The system is operational and reachable by the building manager or system administrator. * The building manager or system administrator has valid login credentials. |
| **Main (Success) Flow:** | 1. System Administrator or Building Manager enters to the system with the help of using valid login information. 2. The system verifies the login information has been entered.   * If information is invalid, System displays an error message. It returns the user to Step 1 to fill login details again correctly.   3. The System Administrator or Building Manager picks "Manage Admin Dashboard" from the main menu.  4. The dashboard with multiple widgets is displayed by the system.  5. System Administrator or Building Manager picks a widget to see information.  6. Selected widget is displayed by the system.  7. The building manager or system administrator takes the appropriate steps to manage the information.  8. The information is updated by the system. |
| **Alternate Flows:** | Alternate Flow #1: It is unable for the System Administrator and Building manager to log in.  1. The System displays an error message for System Administrator and Building Manager not to enter the system.  2. The System asks the System Administrator and Building Manager to enter valid login information.  3. The System returns the System Administrator and Building Manager to Step 1 of the Main Flow.  Alternate Flow #2: The Dashboard cannot be viewed by the System Administrator or Building Manager.  1. The error message is displayed by the system.  2. The system returns the System Administrator or Building Manager to Step 3 of the Main Flow. |
| **Post Condition:** | Main Flow:   * The system administrator or building manager has successfully entered into the system. * The system administrator or building manager has managed the dashboard and its information.   Alternate Flow #1:   * The system administrator or building manager could not enter into the system. * The dashboard was not managed by the system administrator and building manager. |
| **Requirements:** | 1. The System Administrator or Building Manager shall enter the system.  2. The System Administrator or Building Manager shall see multiple widgets on the dashboard.  3. The System Administrator or Building Manager shall pick a widget to see information.  4. The System Administrator or Building Manager shall take the appropriate steps to manage the details.  5. The System shall update the status of the information. |

### Use Case 5

|  |  |
| --- | --- |
| **Use Case Number:** | UC-05 |
| **Use Case Name:** | Create Request |
| **Actor(s):** | Apartment Owner, Building Manager |
| **Description:** | This use case describes how an apartment owner or building manager can create a maintenance request. |
| **Priority (Release)** | R1 |
| **Trigger:** | External - The trigger for this use case is when an apartment owner or building manager notices a maintenance issue in the building that requires attention from the maintenance staff. |
| **Pre-condition(s):** | * The user must be registered and logged into the system. * The apartment unit and building must be associated with the apartment owner's account if it is owner who makes request. |
| **Main (Success) Flow:** | 1. The user navigates to the "Create Request" section of the dashboard.  2. The system displays a form to fill in the details of the maintenance request, including:  a. Description of the issue.  b. Urgency level.  c. Location of the issue in the apartment or building.  3. The user fills in the form and submits the request.  4. The system confirms the request and add it to the system  5. The system notifies the maintenance staff of the new request.  6. The maintenance staff reviews the request and schedules a visit to the apartment unit.  7. The maintenance staff resolves the issue and updates the request status in the system.  8. The system notifies the apartment owner or building manager of the request status update. |
| **Alternate Flows:** | A1. If the user leaves a required field empty, the system displays an error message and require the user to fill in the information.  A2. If the maintenance staff is unable to resolve the issue during the scheduled visit, they update the request status in the system and schedule another visit.  A3. If the maintenance staff determines that the issue is not their responsibility, they update the request status in the system and notify the apartment owner. |
| **Post Condition:** | The maintenance request is successfully created and associated with the apartment owner's account, and the maintenance staff is notified about the request. |
| **Requirements:** | * The system must provide a form for the user to create a maintenance request. * The system must associate the maintenance request with the apartment owner's unit and building. * The system must notify the maintenance staff of the new maintenance request. * The maintenance staff must be able to review and update the maintenance request status in the system. * The system must notify the apartment owner or building manager of the maintenance request status updates. |

### Use Case 6

|  |  |
| --- | --- |
| **Use Case Number:** | UC-06 |
| **Use Case Name:** | View Maintenance Status |
| **Actor(s):** | Apartment Owner, Maintenance Staff, Building Manager |
| **Description:** | This use case describes how an apartment owner and building manager can view the status of their maintenance requests. |
| **Priority (Release)** | R2 |
| **Trigger:** | External - The trigger for this use case is when an apartment owner wants to check the status of a maintenance request that they have previously submitted. |
| **Pre-condition(s):** | * The apartment owner must be registered and logged into the system. * There must be at least one maintenance request associated with the apartment owner's account. |
| **Main (Success) Flow:** | 1. The apartment owner or building manager navigates to the "View Maintenance Status" section of the dashboard.  2. The system displays a list of all maintenance requests associated with the apartment owner's account, sorted by date.  3. The user selects a maintenance request from the list.  4. The system displays the details of the selected maintenance request, including:  a. Description of the issue.  b. Urgency level.  c. Date and time the request was submitted.  d. Current status of the request (e.g. scheduled, in progress, resolved).  5. The apartment owner can choose to close the maintenance request if they are satisfied with the resolution.  6. The system updates the maintenance request status and notifies the maintenance staff of the request closure. |
| **Alternate Flows:** | A1. If there are no maintenance requests associated with the apartment owner's account, the system displays a message indicating that there are no requests to view.  A2. If the apartment owner or building manager selects a closed maintenance request, the system displays the details of the request but does not allow the user to close it.  A3. If the maintenance staff updates the status of the request while the apartment owner is viewing it, the system automatically refreshes the page to display the updated status. |
| **Post Condition:** | The apartment owner can view the details and status of their maintenance requests and can choose to close a request if they are satisfied with the resolution. |
| **Requirements:** | * The system must provide a list of all maintenance requests associated with the apartment owner's account. * The system must display the details of a selected maintenance request, including the current status. * The apartment owner must be able to close a maintenance request if they are satisfied with the resolution. * The system must update the maintenance request status and notify the maintenance staff of the request closure. |

### Use Case 7

|  |  |
| --- | --- |
| **Use Case Number:** | UC-07 |
| **Use Case Name:** | Manage Payment Methods |
| **Actor(s):** | Apartment Owner |
| **Description:** | In this use case, apartment owners manage their payment methods such as adding or deleting credit, and debit cards. |
| **Priority (Release)** | R1 |
| **Trigger:** | External - The apartment owners manage want to manage payment methods. |
| **Pre-condition(s):** | * The apartment owner should have a valid account in the system. * The apartment owner should have at least one payment method added to their account. |
| **Main (Success) Flow:** | 1. The apartment owners log in to their accounts. 2. The system displays the apartment owners' accounts dashboard. 3. The apartment owners navigate to the "Payment Methods" section. 4. The system displays a list of the apartment owners' current payment methods. 5. The apartment owners select the option to add a new payment method. 6. The system displays a form for the apartment owners to input the new payment method details, such as credit card number, expiration date, and billing address. 7. The apartment owners submit the form, and the system validates the payment method details. 8. The system adds the new payment method to the apartment owners' accounts and displays it in the list of payment methods if the validation is successful. 9. The system displays an error message and prompts the apartment owners to input the correct details if the validation fails. 10. For deleting a payment method, the apartment owners select the option. 11. For confirming the deletion of the selected payment method., the system prompts the apartment owner. 12. The apartment owners confirm the deletion, and the system removes the payment method from the apartment owners' account. 13. The system displays a confirmation message that the payment method has been deleted. |
| **Alternate Flows:** | Alternate Flow #1: After Step 2, the apartment owner can update the payment method by selecting "Edit". If an error occurs, the system displays an error message.  Alternate Flow #2: In the emergency contact section, the Registrant can select an existing user by searching, choosing from a list, and setting as the emergency contact. |
| **Post Condition:** | Main Flow:   * According to the apartment owners' actions, their payment methods have been updated. * In the database, the system has updated the apartment owners' payment methods.   Alternate Flow#1:   * According to the apartment owners' actions, their payment methods have not been updated. * In the database, the system has not updated the apartment owners' payment methods. |
| **Requirements:** | 1. The system shall provide functionality for the apartment owners to add or remove payment methods as required. 2. The system shall verify the payment method details provided by the apartment owners, before adding a new payment method. 3. The system shall display a confirmation prompt to verify the apartment owner's intention to proceed with the deletion, whenever they attempt to delete a payment method. |

### Use Case 8

|  |  |
| --- | --- |
| **Use Case Number:** | UC-08 |
| **Use Case Name:** | Track Expenses |
| **Actor(s):** | User |
| **Description:** | In this use case, the past and upcoming expenses can be seen by the users including utility bills, maintenance charges, and other expenses. |
| **Priority (Release)** | R1 |
| **Trigger:** | External - the user tracks their expenses |
| **Pre-condition(s):** | * The user must have a valid account in the system. * The user must have incurred expenses related to the apartment. |
| **Main (Success) Flow:** | 1. Users log in to their accounts. 2. The system displays the users' account dashboard. 3. Navigate to the "Expenses" section. 4. The system displays a list of users' past and upcoming expenses sorted by date. 5. Select a specific expense to view the details. 6. The system displays the amount, due date, and description of the selected expense. 7. Mark the expense as paid, if applicable. 8. The system updates the status of the expense and displays a confirmation message. |
| **Alternate Flows:** | Alternate Flow #1:  To filter expenses based on a specific category, such as maintenance charges or utility bills, select the corresponding filter option.  To view expenses for a specific period, such as the last month or the last year, select the relevant date range filter.  The specific steps for applying filters may vary based on the design and functionality of the system.  Alternate Flow #2:  If users have questions or concerns about a specific expense, contact the relevant person or department for further assistance.  The system displays the contact information for the relevant party.  Users can use the provided method of communication, such as email or phone, to reach out for assistance. |
| **Post Condition:** | Main Flow:  - The users can view their expenses and details.  Alternate Flow #1:   * - The users unable to view their expenses and details. |
| **Requirements:** | 1. The system shall be able to display a list of the user's expenses. 2. The system shall provide the ability for the user to select a specific expense and view its details. 3. The system shall handle errors gracefully and provide informative error messages to the user. |

### Use Case 9

|  |  |
| --- | --- |
| **Use Case Number:** | UC-09 |
| **Use Case Name:** | Add/Remove Building Manager |
| **Actor(s):** | Administrator |
| **Description:** | For the purpose of managing building bills and maintenance on behalf of apartment owners, this use case enables the system administrator to add or remove building managers from the system. |
| **Priority (Release)** | R1 |
| **Trigger:** | External-System administrator needs to add new Building manager to the system. |
| **Pre-condition(s):** | * The ability to add or remove building managers from the system depends on the system administrator having the required rights. * With a valid username and password or another kind of authentication, the administrator must have already logged into the system. * While adding or deleting a building manager from the system, the administrator must grasp all of the tasks and responsibilities of a building management as well as the relevant policies and procedures. |
| **Main (Success) Flow:** | 1. Administrators click on add/remove button. 2. Administrator needs to mention budling manager’s personal information to the system in order to delete or add building manager. 3. Administrator finishes. 4. The system accepts information which the administrator provided about building manager. 5. If there is not an error, system create or remove building manager in system. 6. The list of building managers is updated by the system to reflect the changes made by the administrator. |
| **Alternate Flows:** | * Alternate Flow#1:The system notifies the administrator that there are no building managers to remove if there are no building managers to remove. * Alternate Flow#2: The system displays a notice telling the administrator that there are no buildings available to assign to a new building manager if there are no buildings available. * Alternate Flow#3: The system displays an error message and asks the administrator to fix any mistakes if the administrator's information contains problems. |
| **Post Condition:** | * If a new building manager was added, their data is now saved in the system, and they can access it using the login information that was supplied to them in the notification. * In the event that an existing building manager was fired, their account was terminated and access to the system was revoked. * Any modifications made by the administrator are updated in the system's list of building managers. * The system should make sure that the access privileges for the newly added or deleted building manager are set appropriately. These privileges should include access to maintenance requests, payment history, and upcoming charges. |
| **Requirements:** | 1. 1. The system shall permit administrators to add and remove building managers. 2. 2. The system shall have a form where the administrator can input the building manager's information, including name and contact data. 3. 3. The system shall reliably and securely store building manager data. 4. 4. The system must be able to notify the building manager and include instructions on how to log in. 5. 5. When a building manager is removed, the system shall give the administrator a list of other building managers to pick from. 6. 6. The system shall make sure that building managers' access privileges are established appropriately in accordance with their roles and duties, such as access to maintenance requests, payment history, and upcoming expenses. |

### Use Case 10

|  |  |
| --- | --- |
| **Use Case Number:** | UC-10 |
| **Use Case Name:** | Give feedback and rating |
| **Actor(s):** | Apartment owner |
| **Description:** | The system allows apartment owners to provide feedback and rating on the repairs made to their apartments, and this will increase quality of the services which is provided by the system. Apartment owners, maintenance staff and system administrators can see the feedback and rating on control panel |
| **Priority (Release)** | R1 |
| **Trigger:** | External-Apartment owner wants to increase the quality of maintenance work is done based on the feedbacks and rating. |
| **Pre-condition(s):** | * An account on the platform is required for the apartment owner. * The apartment's maintenance tasks must be finished. |
| **Main (Success) Flow:** | 1. The owner of the apartment logs into the system. 2. To get to the dashboard, the apartment owner navigates. 3. The owner of the apartment chooses the finished maintenance tasks. 4. The apartment's owner comments on and rates the maintenance job. 5. The feedback and rating given by the apartment owner are saved by the system. |
| **Alternate Flows:** | Alternate Flow#1: If apartment owner did not rate the maintained work, the following steps would occur: 1. The apartment owner receives a notification from the system asking them to rate the maintenance job.  2. The owner of the apartment complex offers comments and a rating for the maintenance job.  3. The system records the apartment owner's comments and ratings. |
| **Post Condition:** | Alternate Flow#1: If the apartment user did not rate and gave feedback to the system, send feedback and rating button will give a remainder message. Apartment owners have to back home page if did not write anything in feedback and did not rate the system. |
| **Requirements:** | 1.Apartment owners should be able to rate and comment on the maintenance work using the system.  2.The system ought to record the comments and ratings made by the apartment managers  3.The system needs to show the maintenance staff's feedback and ratings from apartment owners.  4.Apartment owners' comments and ratings should be used by the system to enhance the standard of services it offers. |

Use Case 11

|  |  |
| --- | --- |
| **Use Case Number:** | UC-11 |
| **Use Case Name:** | Register a Building Manager |
| **Actor(s):** | Building Manager |
| **Description:** | The Building Manager provides personal and building information to the System upon registering and becoming a user. |
| **Priority (Release)** | R1 |
| **Trigger:** | External –The Building Manager registering as a user. |
| **Pre-condition(s):** | * The URL of the website is accessible to the User. * The system has been installed and configured. * The system is open for registrations. |
| **Main (Success) Flow:** | 1. Building Manager selects an option to register as a Building Manager  2. System requests personal information, including First name, Last Name, Date of birth, Email address, and Phone number.  3. Building Manager provides personal information.  4. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 2 to fill personal information again correctly.   5. System requests building information, including building name, address, number of units, and contact information  6. Building Manager provides building’s information  7. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 5.   8. The system asks for login information, including a username and password.  9. The user enters login information.  10. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 8 to fill login information again correctly.   11. The system displays a confirmation of registration. |
| **Alternate Flows:** | **Alternate Flow #1**  The system shall provide an option to cancel the registration after step 2, and the steps would be as follows:  1. During the Registration, the Building Manager picks the option to cancel registration.  2. The system asks confirmation to cancel registration.  3. The user confirms the intent.  4. The system returns the Building Manager to the home menu. |
| **Post Condition:** | **Main Flow:**   * The registration has been successfully completed by the user with their personal and building information on the system. * All information about the Building Manager and building has been stored successfully in the system.   **Alternate Flow #1**   * The registration was not completed by the user. * Information about the user was not saved in the system. |
| **Requirements:** | 1. The user shall register with personal and building information on the system.  2. The user shall have an option to cancel registration during the registration process.  3. The system shall validate all information has been entered by the user for registration |

Use Case 12

|  |  |
| --- | --- |
| **Use Case Number:** | UC-12 |
| **Use Case Name:** | Register a Maintenance Staff |
| **Actor(s):** | Maintenance Staff |
| **Description:** | Maintenance Staff Applicant shall provide personal and employment information to the System upon registering and becoming a Maintenance Staff. |
| **Priority (Release)** | R1 |
| **Trigger:** | External – Maintenance Staff Applicant registering as a Maintenance Staff. |
| **Pre-condition(s):** | * The URL of the website is accessible to the User. * The system has been installed and configured. * The system is open for registrations. |
| **Main (Success) Flow:** | 1. Maintenance Staff selects the option to register as a Maintenance Staff.  2. The system asks personal information, including First name, Last Name, Date of birth, Email address, and Phone number.  3. The Maintenance Staff enters personal information.  4. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 2 to fill personal information again correctly.   5. System requests employment information i.e. job title, years of experience, certifications, and work history.  6. The Maintenance Staff enters employment information.  7. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 5 to fill again correctly.   8. The system asks for login information, including a username and password.  9. The Maintenance Staff enters login information.  10. The system verifies the required information has been entered.   * If the information is invalid, the system displays a warning message. It Returns the user to Step 8 to fill login information again correctly.   11. The system displays a confirmation of registration. |
| **Alternate Flows:** | **Alternate Flow #1**  The system shall provide an option to cancel the registration after step 2, and the steps would be as follows:  1. During the Registration, the Maintenance Staff picks the option to cancel registration.  2. The system asks confirmation to cancel registration.  3. The Maintenance Staff confirms the intent.  4. The system returns the Maintenance Staff to the home menu. |
| **Post Condition:** | **Main Flow:**   * The registration has been successfully completed by the user with their personal and employment information on the system. * All information about the user has been stored successfully in the system.   **Alternate Flow #1**   * The registration was not completed by the user. * Information about the user was not saved in the system. |
| **Requirements:** | 1. The user shall register with personal and employment information on the system.  2. The user shall have an option to cancel registration during the registration process.  3. The system shall validate all information has been entered by the user for registration |

# References

* 1. <https://drawio-app.com/>
  2. <https://enkonix.com/blog/functional-requirements-vs-non-functional/>

1. <https://enkonix.com/blog/functional-requirements-vs-non-functional/> [↑](#footnote-ref-1)
2. <https://enkonix.com/blog/functional-requirements-vs-non-functional/> [↑](#footnote-ref-2)
3. <https://drawio-app.com/> [↑](#footnote-ref-3)