



UNIVERSITY *of* TASMANIA

KIT519 – Software Engineering and HCI

Individual Component – Assessment Task 3 : Software Design And Development

Group Number :36

Name : Ravikumar Kishorbhai Savani

Student ID : 685167

Table of Contents

Introduction.....	3
Extended Use Cases with High-Level Interaction Diagrams	4
Component 2: Bin Monitoring and Dashboard	4
Use Case 2.1: View Dashboard	4
Use Case 2.2: View Bin Statuses on Map	6
Component 4: Report Generation and Feedback	8
Use Case 4.1: Generate Report	8
Use Case 4.2: Submit Feedback.....	10
System Sequence Diagrams for Each Use Case	12
Sequence Diagram for Use Case 2.1: View Dashboard	12
Sequence Diagram for Use Case 2.2: View Bin Statuses on Map	13
Sequence Diagram for Use Case 4.1: Generate Report	14
Sequence Diagram for Use Case 4.2: Submit Feedback	15
Sequence Diagram for Use Case 5.1: View Feedback.....	16
Sequence Diagram for Use Case 5.2: Respond to Feedback.....	17
Sequence Diagram for Use Case 4.3: View Reports	18
Sequence Diagram for Use Case 2.3: View Notifications.....	19
Black-Box Test Design for Each Use Case	20
OO Code Representing the Design Report Artefacts in C#	21

Introduction

In this assignment, Ravikumar Savani was to build some key aspects of bin monitoring, reporting and feedback management within the Smart Bin Monitoring System – a solution designed to improve waste collection and sustainability on the University of Tasmania campus. Ravikumar Savani participated in developing the extended use cases, high-level interaction diagrams and the system sequence diagrams for viewing bin statuses, generating reports, and submitting feedback.

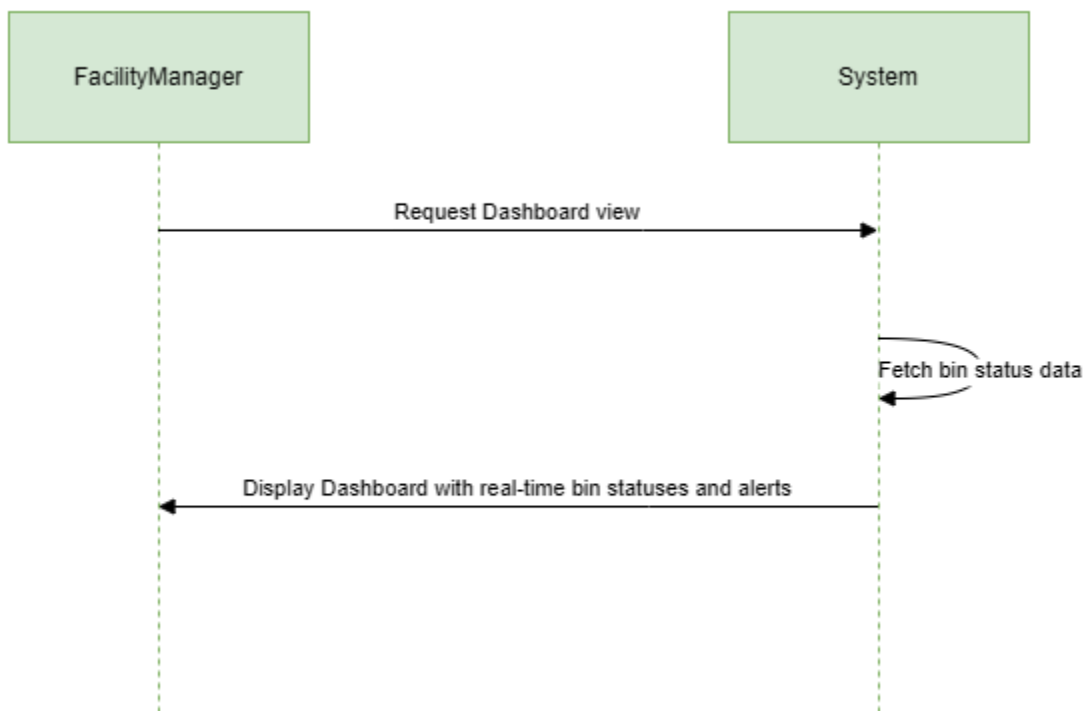
The experience helped Ravikumar Savani gain better insight into designing data interfaces that would meet functional and usability specifications when it comes to real-time bin status updates and report generation features. Black-box testing was also used to check whether the used system behaves properly in different conditions and increase reliability. It presents a brief description of the roles that Ravikumar Savani has played in these core features, evidencing a good understanding of the best and recommended practices of software design for the purpose of the IoT-based monitoring of the waste management systems for enhanced and efficient responses.

Extended Use Cases with High-Level Interaction Diagrams

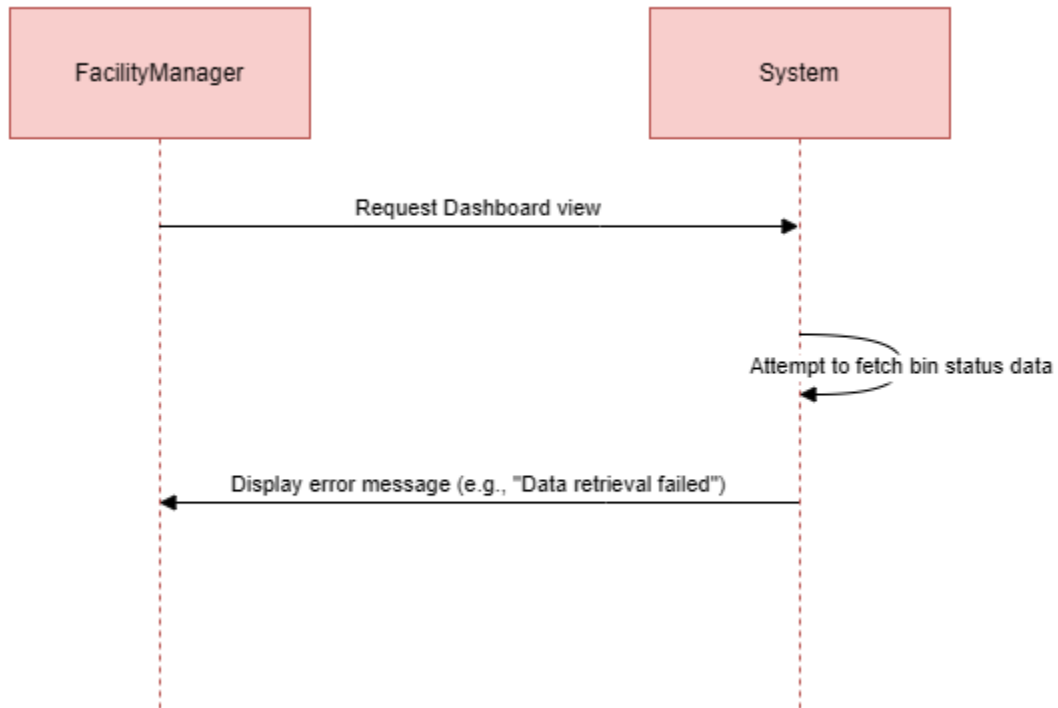
Component 2: Bin Monitoring and Dashboard

Use Case 2.1: View Dashboard

Sunny-Day Interaction Diagram

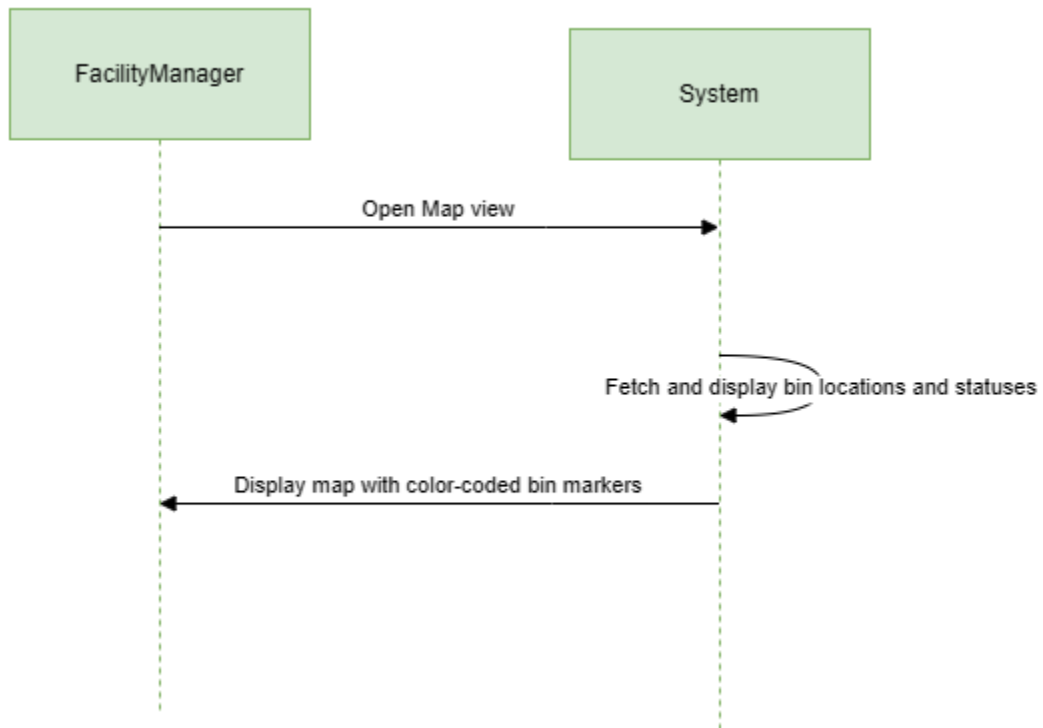


Rainy-Day Interaction Diagram

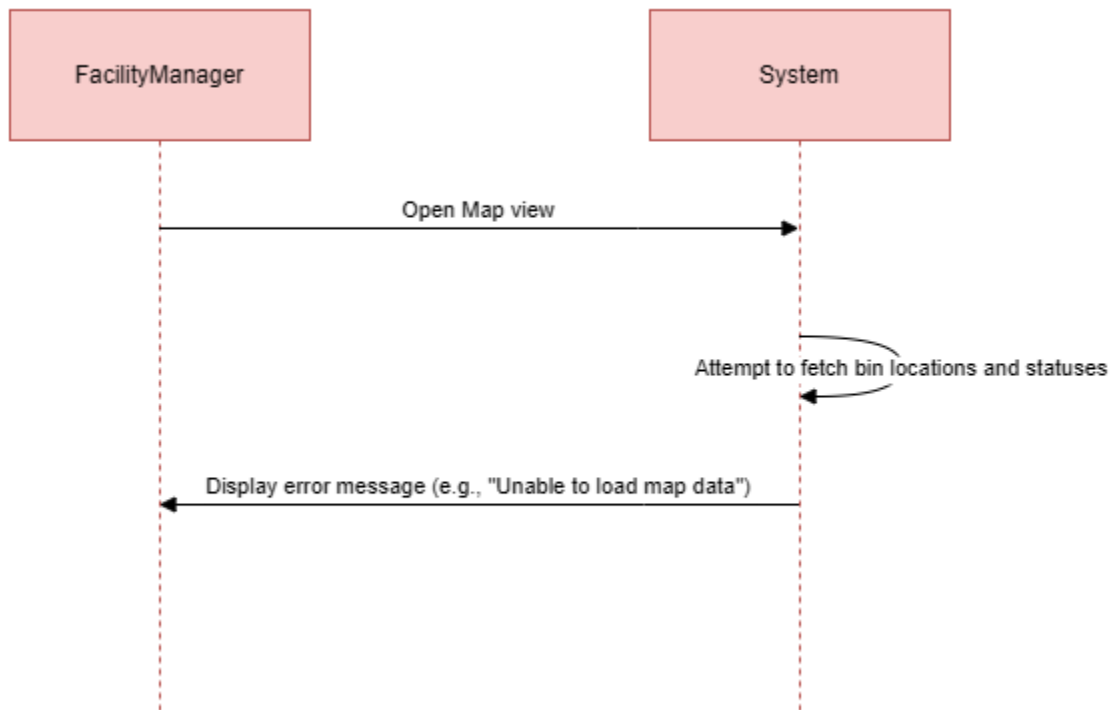


Use Case 2.2: View Bin Statuses on Map

Sunny-Day Interaction Diagram



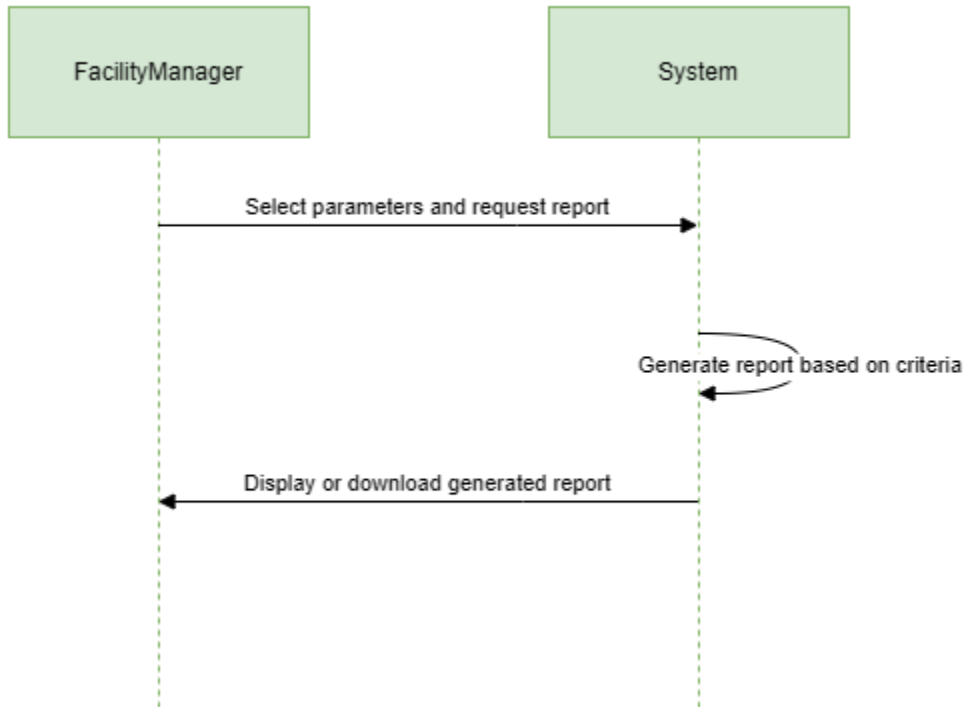
Rainy-Day Interaction Diagram



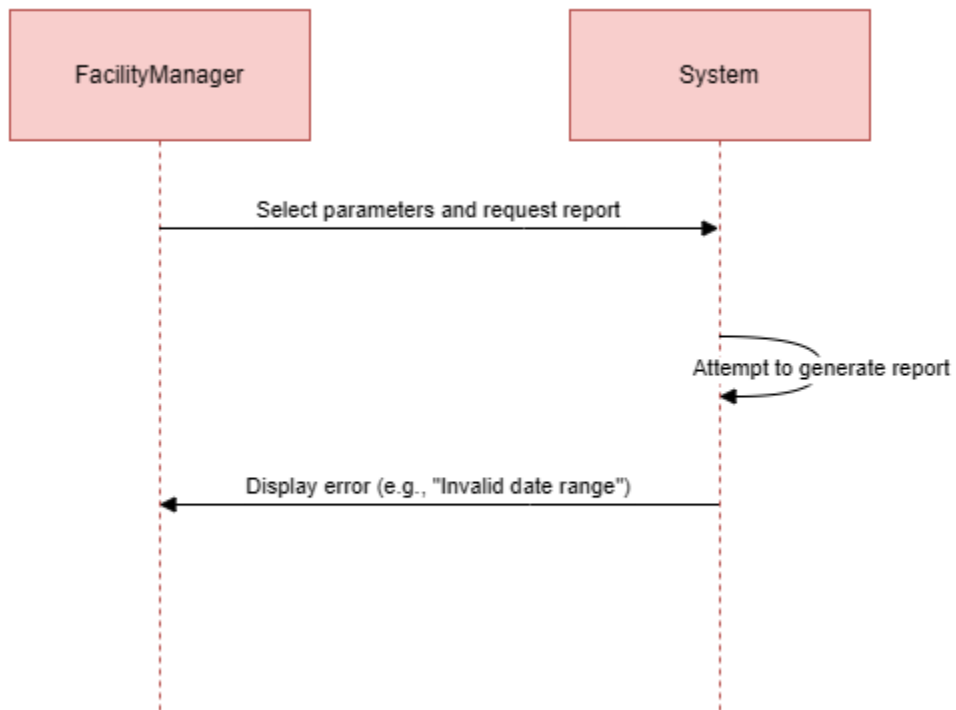
Component 4: Report Generation and Feedback

Use Case 4.1: Generate Report

Sunny-Day Interaction Diagram

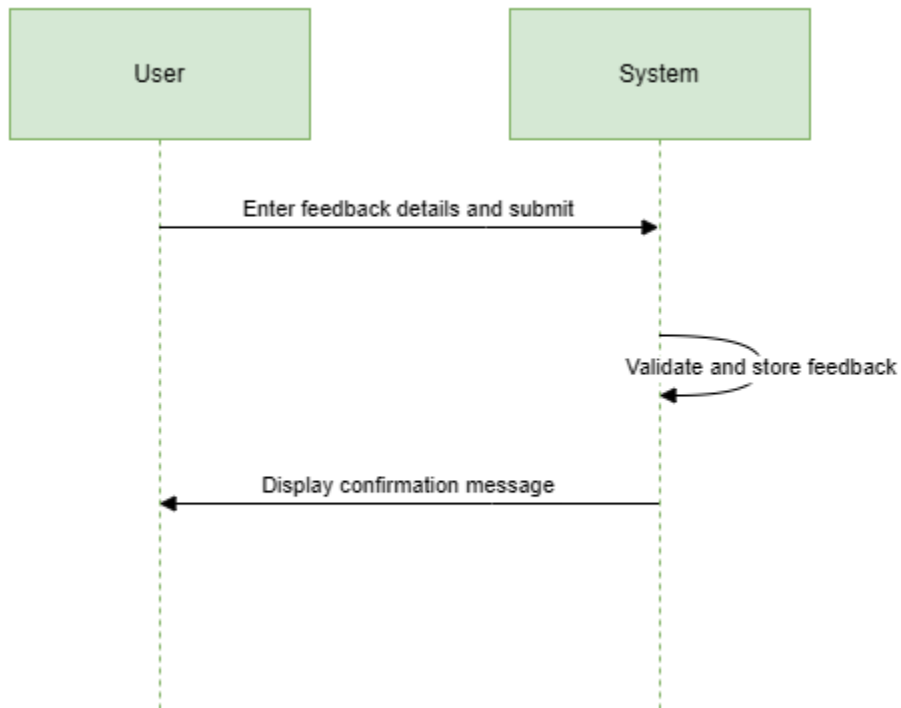


Rainy-Day Interaction Diagram

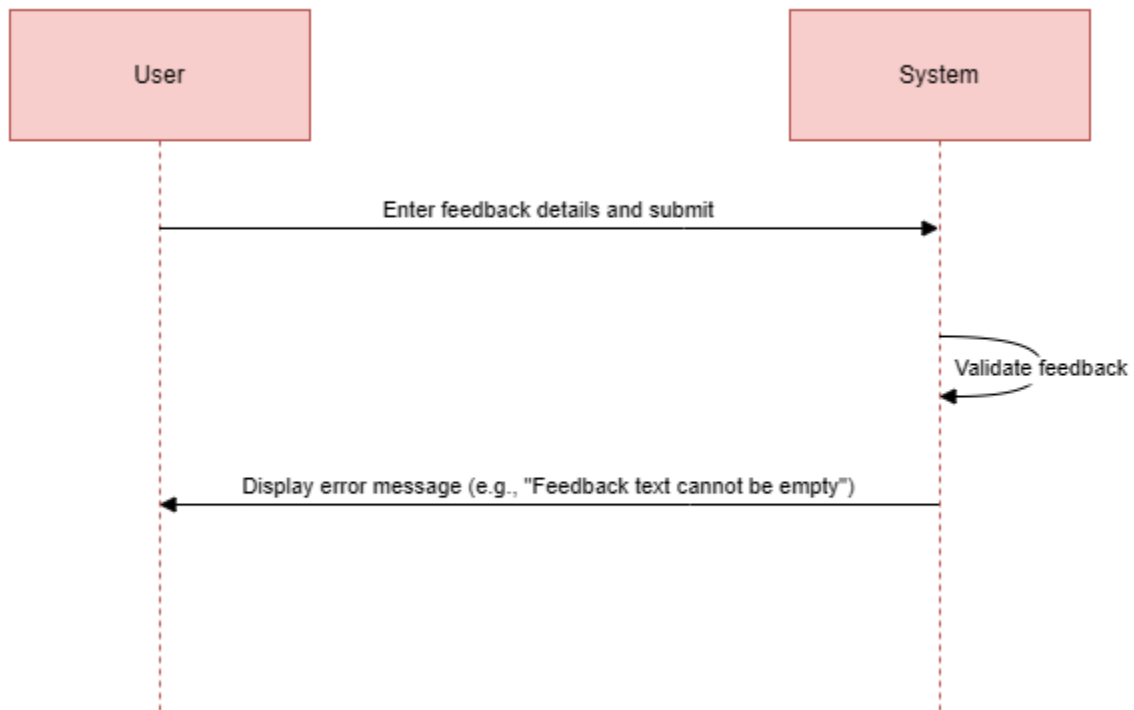


Use Case 4.2: Submit Feedback

Sunny-Day Interaction Diagram

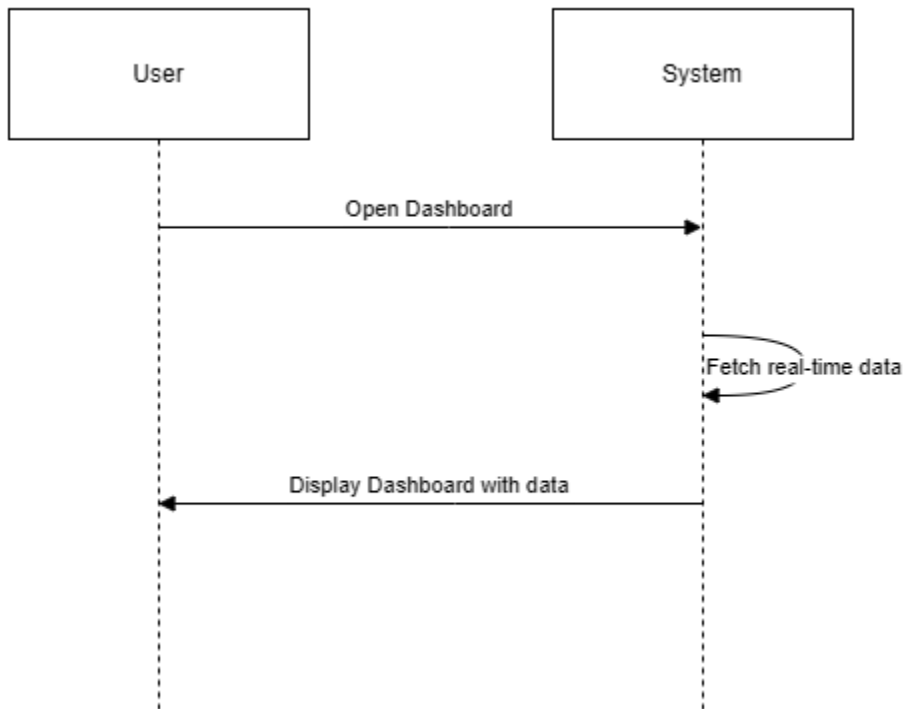


Rainy-Day Interaction Diagram

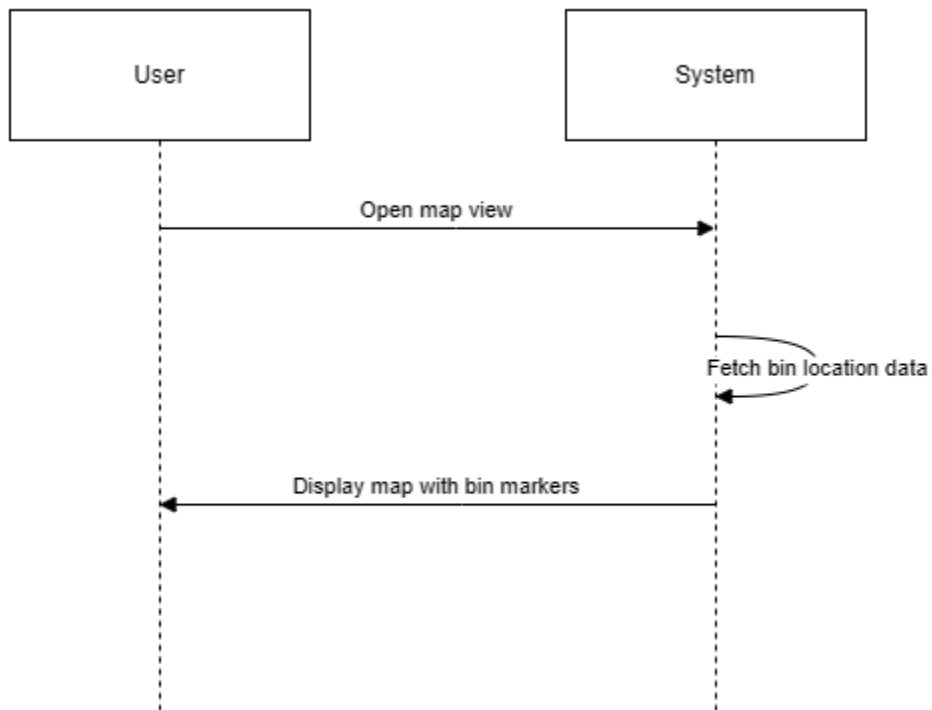


System Sequence Diagrams for Each Use Case

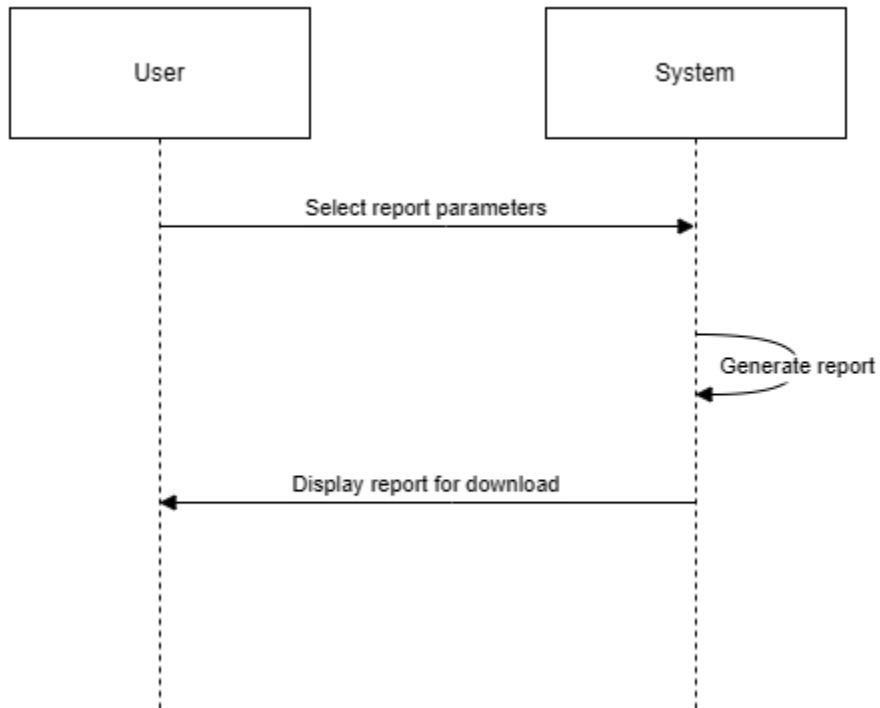
Sequence Diagram for Use Case 2.1: View Dashboard



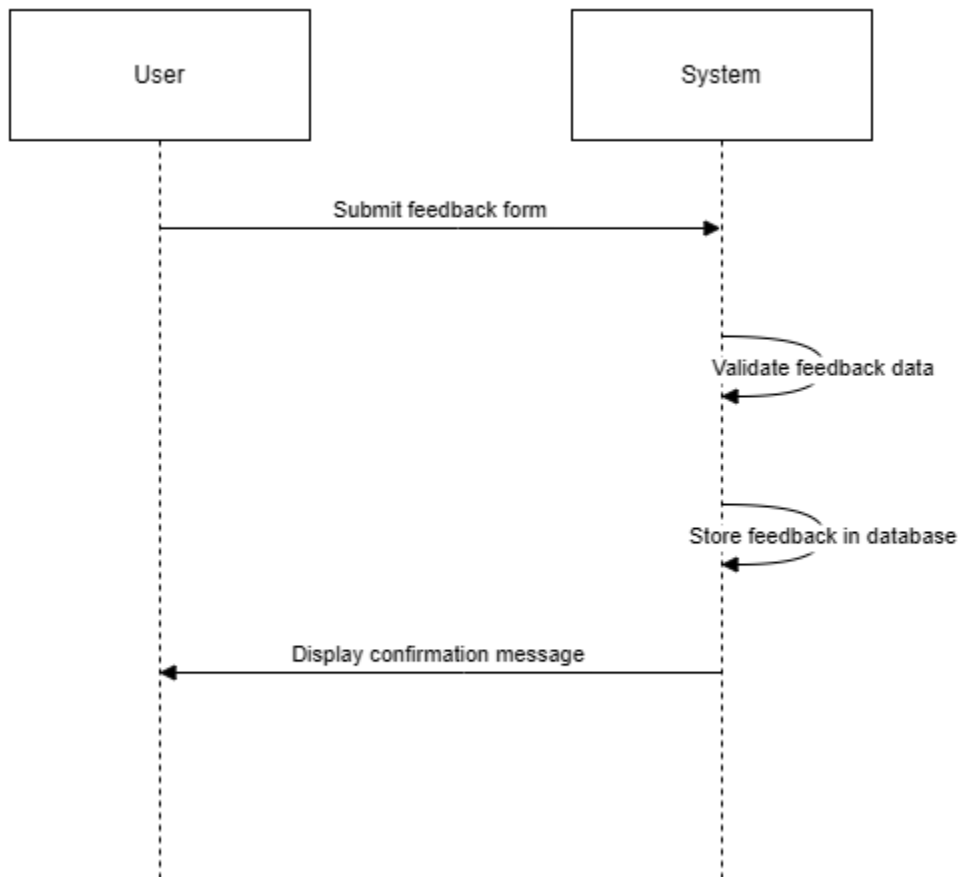
Sequence Diagram for Use Case 2.2: View Bin Statuses on Map



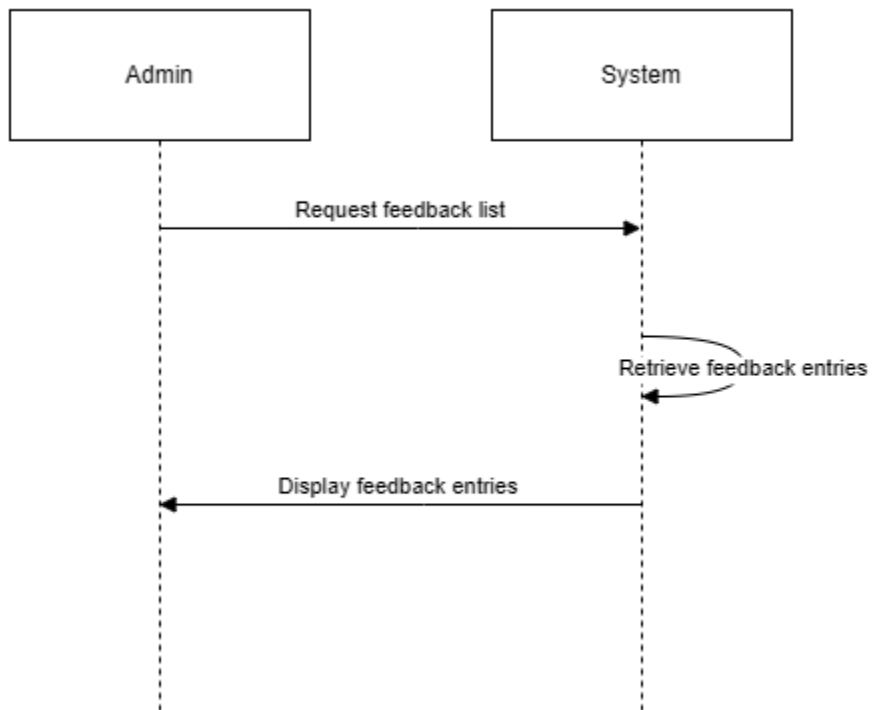
Sequence Diagram for Use Case 4.1: Generate Report



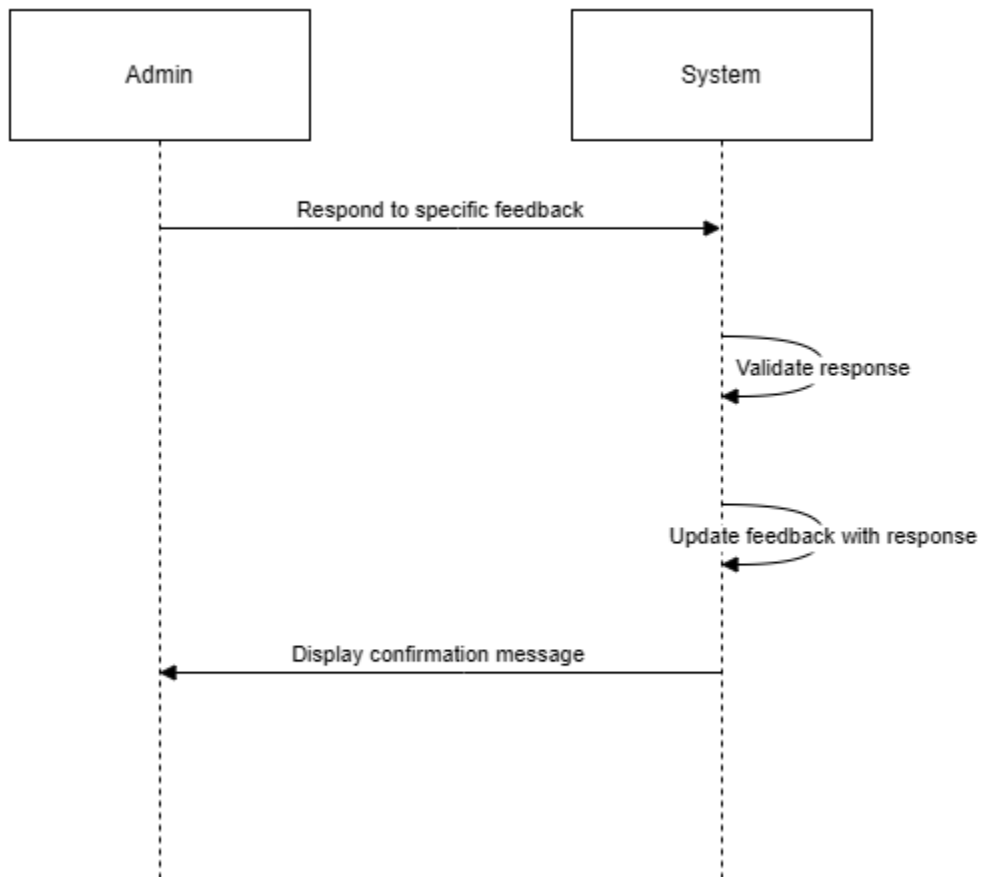
Sequence Diagram for Use Case 4.2: Submit Feedback



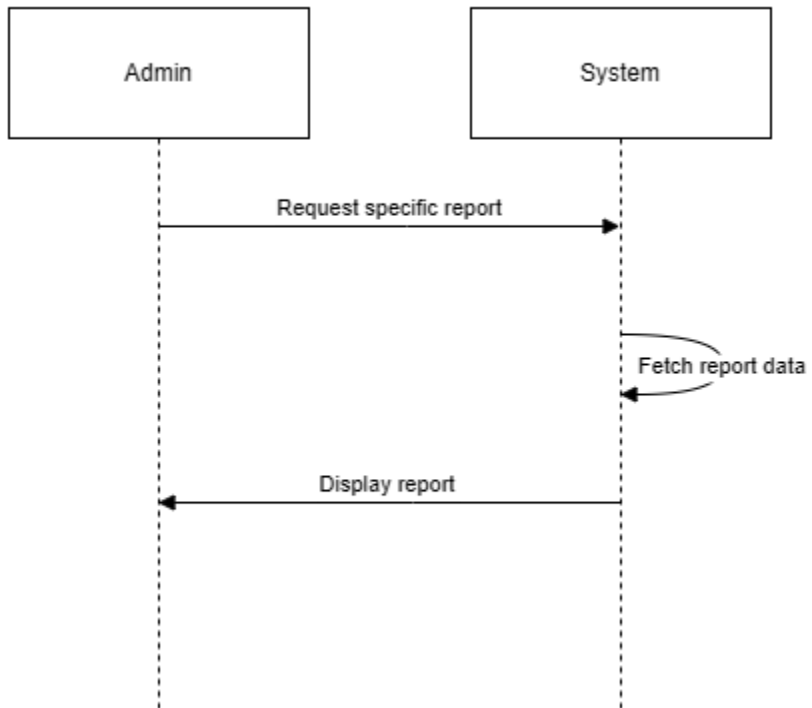
Sequence Diagram for Use Case 5.1: View Feedback



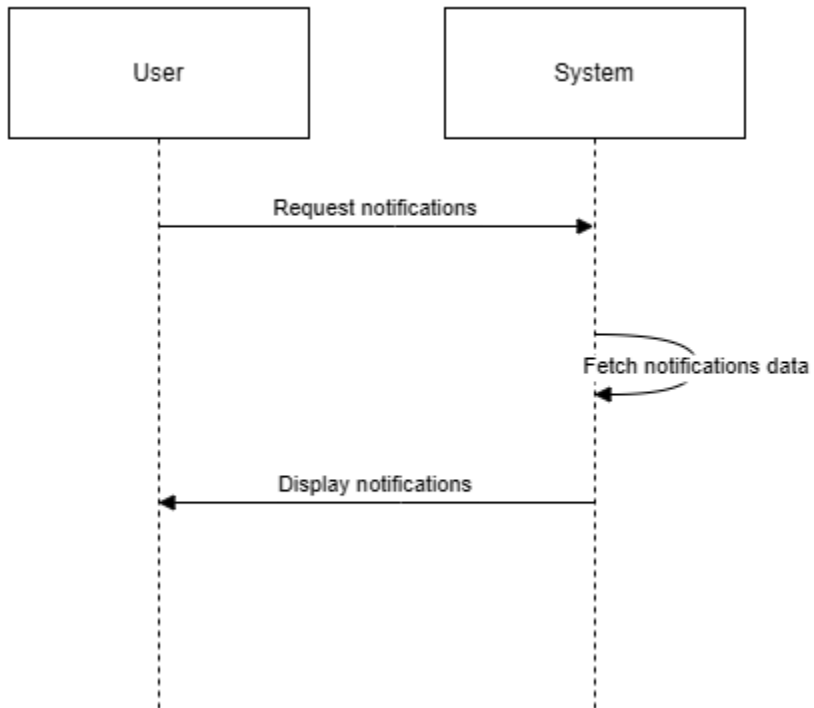
Sequence Diagram for Use Case 5.2: Respond to Feedback



Sequence Diagram for Use Case 4.3: View Reports



Sequence Diagram for Use Case 2.3: View Notifications



Black-Box Test Design for Each Use Case

Use Case	Test Case	Input	Expected Output	Result
View Bin Statuses	View bin statuses	Request to view bin statuses	Bin statuses displayed with color-coded indicators	Pass/Fail
Generate Reports	Generate report with valid parameters	Location, date range, bin status filter	Report generated and displayed/downloaded	Pass/Fail
	Generate report with invalid date range	Invalid date range	Display error message ("Invalid date range")	Pass/Fail
View Dashboard	Access Dashboard	Request to access Dashboard	Dashboard loaded with real-time bin data and alerts	Pass/Fail
View Map of Bins	Access map view	Request to view map	Map displayed with color-coded bin markers	Pass/Fail
View Reports	View specific report	Report type, date range	Selected report displayed with accurate data	Pass/Fail
Manage Notifications	Add notification with valid data	Notification message and priority	Notification saved and displayed	Pass/Fail
	Add notification with missing data	Empty notification message	Display error message ("Message cannot be empty")	Pass/Fail
View Feedback	View list of feedback	Request feedback list	Feedback entries displayed in list format	Pass/Fail
Respond to Feedback	Respond to specific feedback	Feedback ID, response text	Response saved and displayed in feedback section	Pass/Fail
	Respond with empty response text	Feedback ID, empty response	Display error message ("Response cannot be empty")	Pass/Fail

OO Code Representing the Design Report Artefacts in C#

```
using System.ComponentModel.DataAnnotations;
```

```
namespace SmartBin.Models
```

```
{  
    public class Report  
    {  
        [Key]  
        public int Id { get; set; }  
  
        [Required]  
        public string Name { get; set; }  
  
        [Required]  
        public string HouseNumber { get; set; }  
  
        [Required]  
        public string Address { get; set; }  
  
        [Required]  
        public int BinId { get; set; }  
  
        [Required]  
        public string BinStatus { get; set; } // Options: "Filled", "Empty", "Partial"  
  
        [Required]  
        public string CollectBinUpdate { get; set; } // Options: "Collected" or "Not Collected"  
  
        [Required]  
        public string ContaminationAlert { get; set; } // Options: "Good", "Bad", "Normal"  
    }  
}
```

```
using System.ComponentModel.DataAnnotations;
```

```
namespace SmartBin.Models
```

```
{  
    public class Bin  
    {  
        [Key]  
        public int BinId { get; set; }  
  
        [Required]  
        public string FillStatus { get; set; } // Options: "Filled", "Empty", "Partial"  
  
        [Required]
```

```
public bool IsContaminated { get; set; } // True if contaminated, otherwise false
```

```
[Required]
```

```
public double Latitude { get; set; }
```

```
[Required]
```

```
public double Longitude { get; set; }
```

```
    }  
}
```