One-Stop Community Service App

-CS3342 Software Design

Group 30

Group Member

Wang Ying	100%
Wan Zimeng	100%
Chen Yihuan	100%
Gao Nanjie	100%
Wang Fan	100%
Liu Hengche	100%

Table of contents

01 Background & Storyline

02 Objective & Function

03 User Case

04 Prototype

05 Class Diagram

06 Sequence diagram

07 Team Work

08 Overall Reflection

09 Conclusion

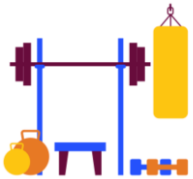
01

Background and Storyline

Current Situation

Manager

Community Service

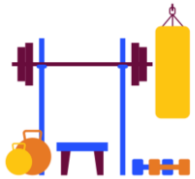


Resident

Background

Manager

Community Service



Community App

Resident

02

Objectives and Functions

Objectives

- **Facilitate Convenience of Residents :**
  To provide residents with a seamless and efficient way to access various community services through a single platform.

- **Improve Management Efficiency :**
  To streamline the management of community services, reducing the time and effort required for the manager.

- **Improve Engagement of Community:**
  To enhance community engagement by facilitating easy communication and interaction among residents.

- **Ensure Transparency:**
  To ensure transparency in service provision and maintenance, building trust in the community.

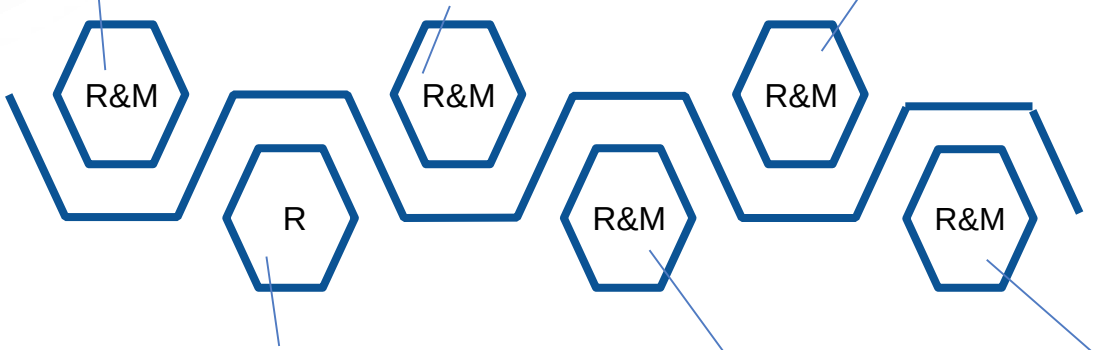
Functions

R:Resident

Login and Authentication

Maintenance Requests

Make Complaints



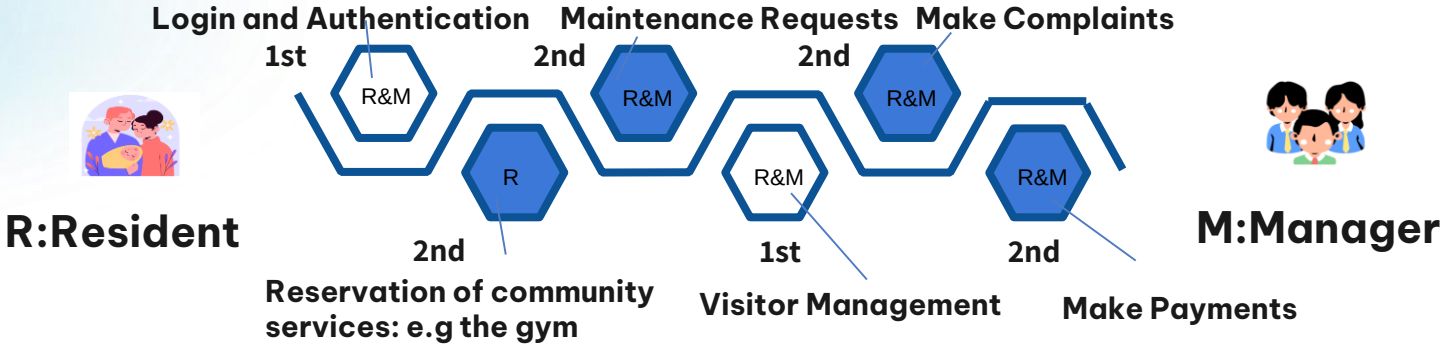
M:Manager

Reservation of community services: e.g the gym

Visitor Management

Make Payments

Incremental Model

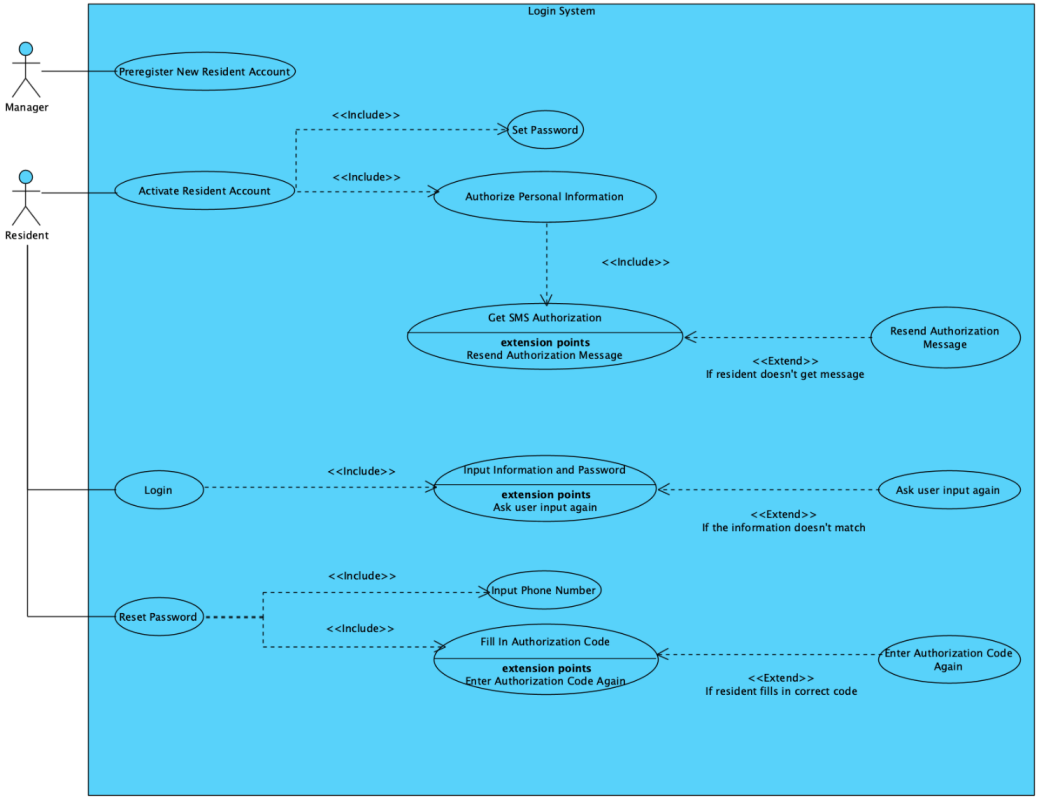


- Like facebook design, this app design can be broken into **increments**
- We firstly built the **CORE** functionalities

03

Use Case Diagram

Login System



Functionalities:

- ❑ **Manager** Creates **New Resident Account**
- ❑ Residence **Activate Account**
- ❑ **Resident Login**
- ❑ **Resident Reset Password**

Use Case Name:	Activate Resident Account	
Actor(s):	Resident	
Description:	This use case describes the process by which a resident activates their preregistered account.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: The manager successfully preregister a new resident Account, and the resident initiates the account activation process.	
		Step 2: System invokes the use case "Authorize Personal Information".
		Step 3: System invokes the use case " Get SMS Authorization".
	Step 4: Resident enters the required personal information into the system	
		Step 5: System invokes the use case "Set Password".
	Step 6: Resident inputs a new password for the account.	
		Step 7: System activates the account
Alternative Courses:	Step 3a: [Extension point: if resident doesn't get message, the system will resend the authorization message again.]	
Pre-condition:	The resident has a preregistered account.	
Post-condition:	The resident account is active.	

Table 1: Activate Resident Account

Use Case Name:	Preregister New Resident Account	
Actor(s):	Manager	
Description:	This use case outlines the steps for a manager to preregister a new resident account.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Step 1: Resident processes to Preregister New Resident Account.	
		Step 2: System successfully preregisters a new resident account.
Pre-condition:	There is a new resident.	
Post-condition:	Manger creates a non-activated new account for newly arrived resident.	

Table 2: Preregister New Resident Account

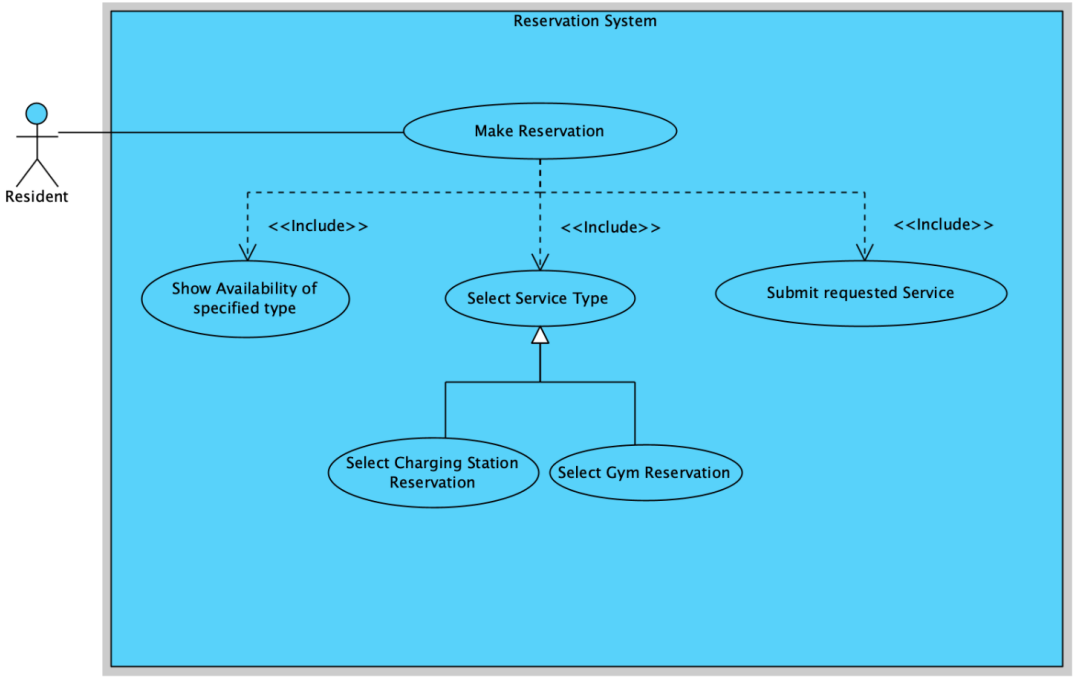
Use Case Name:	Login	
Actor(s):	Resident	
Description:	This use case describes the process by which a resident logs into their account.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Resident processes to login.	
		Step 2: System invokes the use case "Input Information and Password".
	Step 3: Resident enters the required personal information and password into the system.	
		Step 4: System authorizes the
Alternative Courses:	Step 3a: [Extension point: if the password and personal information don't match, the system will invoke use case "Ask user input again".]	
Pre-condition:	The resident has an activated account.	
Post-condition:	The resident is logged into the system.	

Table 3: Login

Use Case Name:	Reset Password	
Actor(s):	Resident	
Description:	This use case outlines the steps for a resident to reset their forgotten password.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Resident processes to Reset Password.	
		Step 2: System invokes the use case "Input phone number".
		Step 3: System invokes the use case " Fill in Authorization Code".
	Step 4: Resident enters the phone number and authorization code.	
		Step 5: System reset the password.
Alternative Courses:	Step 3a: [Extension point: if the authorization code is not correct, the system will invoke use case "enter authorization code again".]	
Pre-condition:	The resident has an activated account.	
Post-condition:	The resident has a new password.	

Table 4: Reset Password

Reservation System

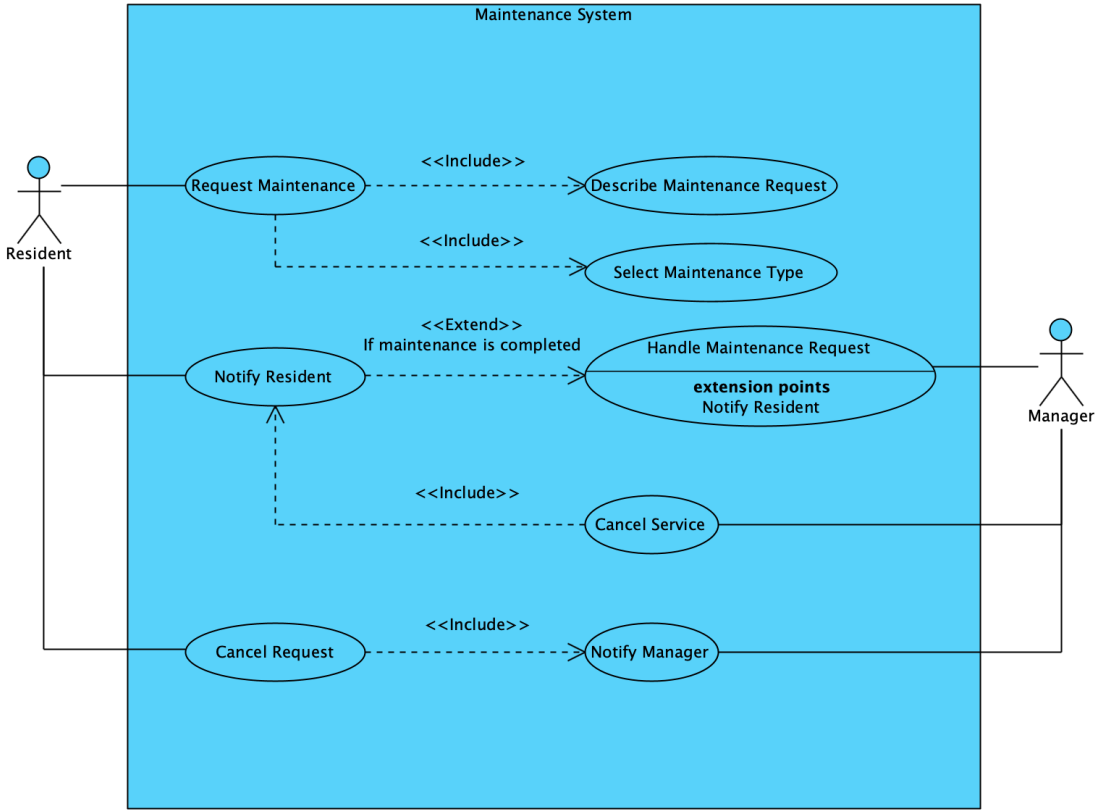


Functionalities:

- **Resident Makes Reservations** for Facilities

Use Case Name:	Make Reservation	
Actor(s):	Resident	
Description:	This use case outlines the process for a resident to make a reservation for services provided by the facility, such as a gym session or a charging station.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Resident accesses the Reservation System and selects the "Make Reservation" option.	
		Step 2: The system prompts Resident4 to select the type of service they wish to reserve (e.g., Gym or Charging Station).
	Step 3: Resident selects the desired service type.	
		Step 4: The system displays the availability of the selected service type for various time slots.
	Step 5: Resident chooses a suitable time slot based on the displayed availability.	
		Step 6: The system prompts Resident4 to confirm the details and proceed with the reservation.
	Step 7: Resident confirms the reservation details.	
Alternative Courses:	Step 4a: [Extension point: If Resident decides to change the service type after viewing availability, the system allows Resident to go back to the service selection step.]	
Preconditions:	Resident is registered and logged into the system. The reservation system is operational and accessible.	
Postconditions:	A reservation is made for the selected service. Resident is informed of the reservation result.	

Maintenance System



Functionalities:

- ❑ **Resident** Request Maintenance
- ❑ **Manager** Update Maintenance State
- ❑ **Resident** Cancel Request

Use Case Name:	Request Maintenance	
Actor(s):	Resident	
Description:	This use case outlines the steps for a resident to conduct a maintenance request	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Resident processes to Request Maintenance.	
		Step 2: System invokes the use case "Describe Maintenance Request".
	Step 3: Resident enters maintenance description.	
		Step 4: System invokes the use case "Select Maintenance Type ".
	Step 5: Resident select maintenance type.	
		Step 6: System submit the maintenance request.
Precondition:	Resident has an activated account in the system.	
Postcondition:	Maintenance request is logged in the system.	

Table 6: Request Maintenance

Use Case Name:	Handle Maintenance Request	
Actor(s):	Manager, Resident	
Description:	This use case outlines the steps for a manager to handle maintenance request.	
Typical Course of Events:	Actor Action	System Response
	Step 1: Manager processes to Handle Maintenance Request.	
Alternative Flows:	Step 1a: [Extension point: if maintenance is completed, the system will invoke use case " Notify Resident".]	
Precondition:	There are maintenance requests in the system.	
Postcondition:	Maintenance request is handled.	

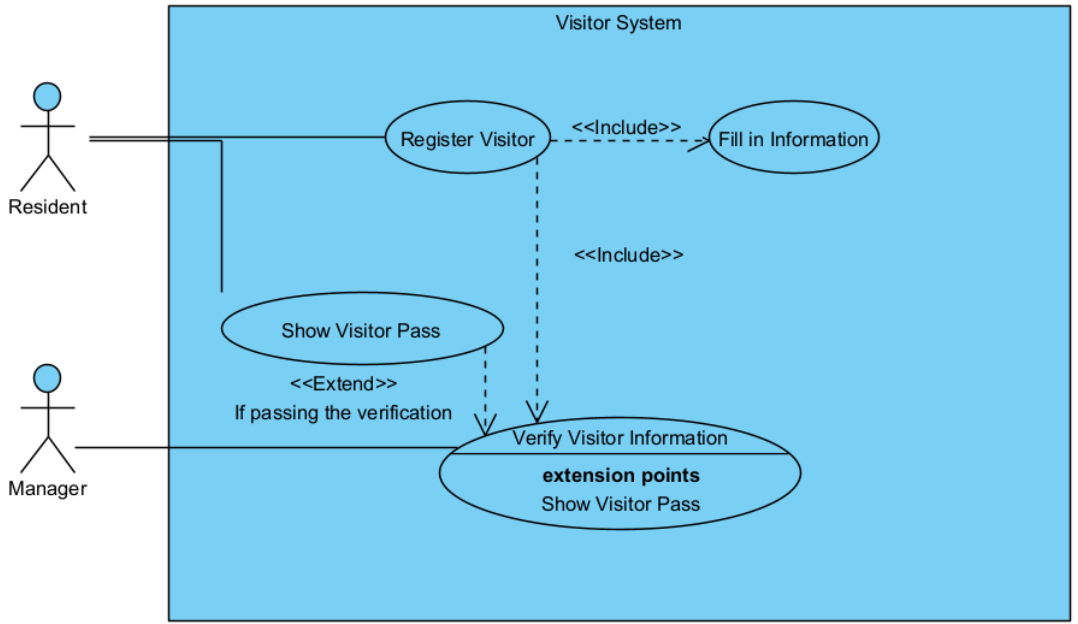
Table 7: Handle Maintenance Request

Use Case Name:	Cancel Service	
Actor(s):	Manager, Resident	
Description:	This use case outlines the steps for a manager to cancel a service.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Manager processes to Cancel Service.	Step 2: System invoke the use case Notify Resident.
Precondition:	There are maintenance requests in the system.	
Postcondition:	Maintenance request is canceled.	

Table 8: Cancel Service

Use Case Name:	Cancel Request	
Actor(s):	Manager, Resident	
Description:	This use case outlines the steps for a resident to cancel a request.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Resident processes to Cancel Request.	Step 2: System invokes the use case Notify Manager.
Precondition:	The resident sent a request in the system	
Postcondition:	Maintenance request is canceled.	

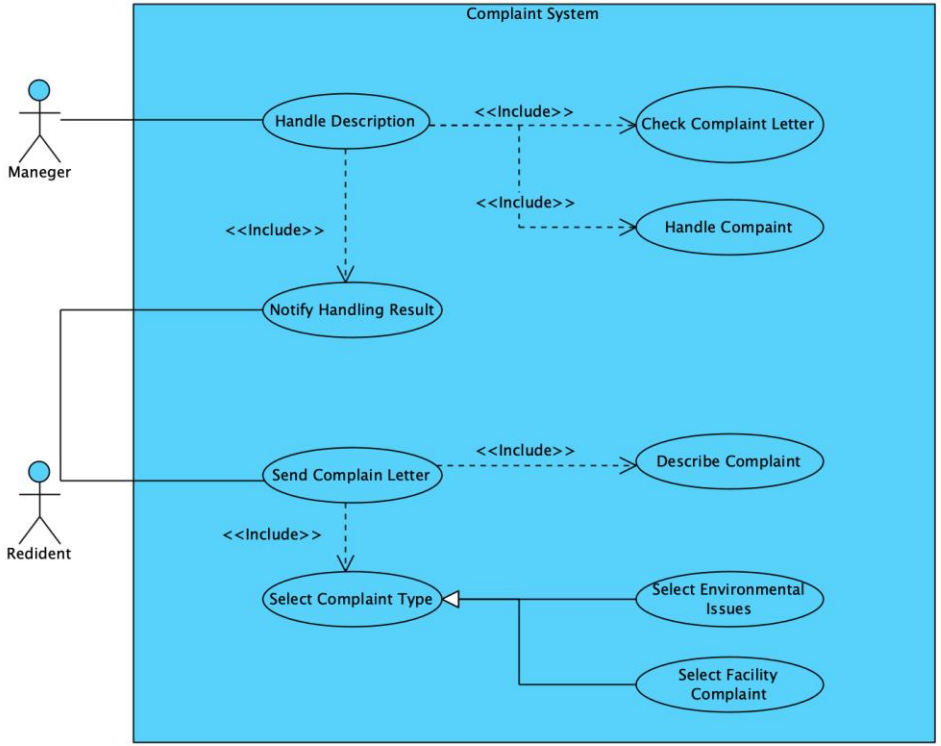
Visitor System



Functionalities:

- ❏ **Resident Register Visitor**
- ❏ **Manger Verify the Visitor Information**
- ❏ **Resident** Receive Visitor Pass

Complaint System



Functionalities:

- **Resident Send Complaint Letter** and Provide Issue Description.
- **Manager** assesses the letter and **handle complaint.**

Use Case Name:	Handle Description	
Actor(s):	Manager	
Description:	This use case outlines the process for a manager to handle the description of a complaint received from a resident.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: The manager selects the "Handle Description" option to review the details of the complaint.	
		Step 2: The system presents the complaint details, including any descriptions provided by the resident.
	Step 3: The manager assesses the information and decides on an appropriate course of action to address the complaint. Step 4: The manager may use the "Check Complaint Letter" feature if additional review of the complaint's documentation is necessary. Step 5: After reviewing, the manager proceeds to "Handle Complaint" and inputs the actions taken or to be taken into the system. Step 6: The manager notifies the handling result and gives feedback to the resident.	
Alternative Courses:	Step 3a: [Extension point: The manager provides different communication channels for the resident to provide additional information or clarification.]	
Precondition:	The complaint has been logged into the system by the resident.	
Postcondition:	The manager has reviewed the complaint's description and taken necessary action.	

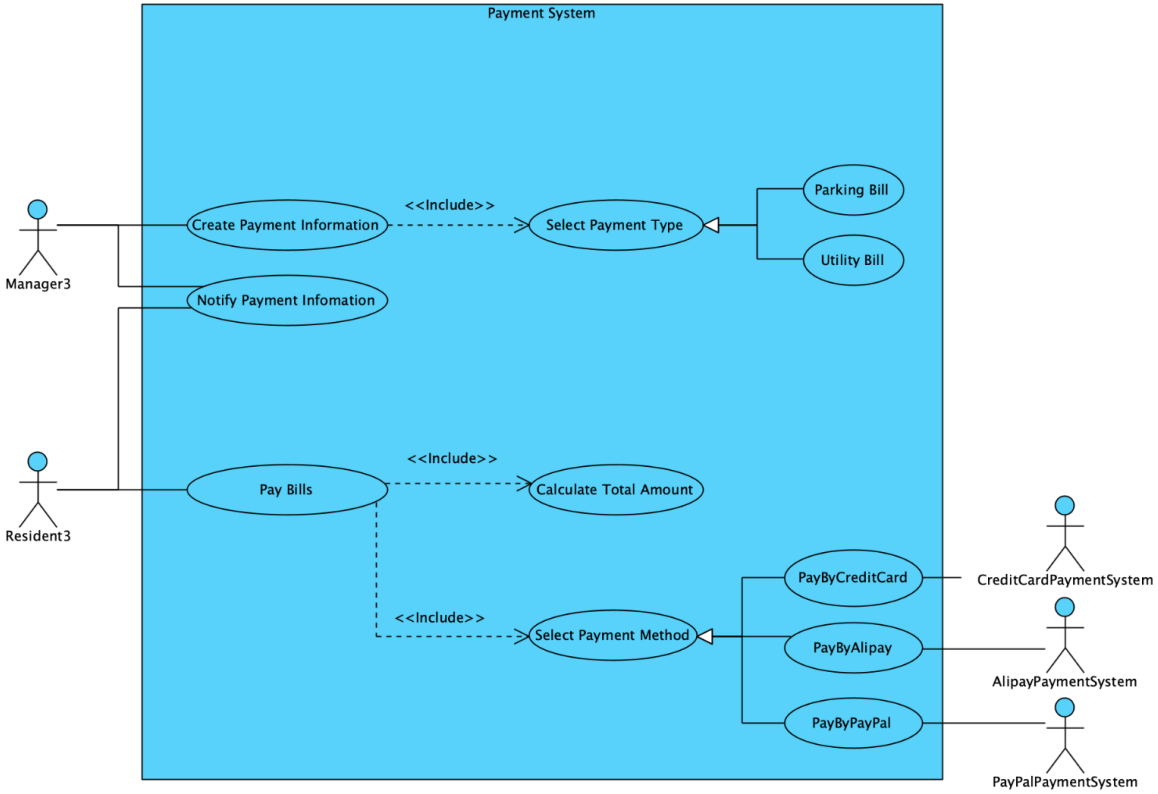
Table 10: Handle Description

Use Case Name:	Send Complaint Letter	
Actor(s):	Resident	
Description:	This use case describes the process by which a resident can send a complaint letter through the system.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Resident selects the "Send Complaint Letter" option.	
		Step 2: The system prompts the resident to select the type of complaint they are submitting ("Select Complaint Type").
	Step 3: Resident chooses the appropriate complaint category, such as "Environmental Issues" or "Facility Complaint."	
		Step 4: The system requests a detailed description of the complaint from the resident ("Describe Complaint").
	Step5: Resident provides a detailed account of the issue, including all pertinent information.	
		Step6: The system generates a confirmation message for the resident indicating successful submission.

Alternative Courses:	Step 4a: [Extension point: If the resident needs to attach additional documentation, the system provides an option to upload files.]
Precondition:	The resident is registered and can access the Complaint System.
Postcondition:	The complaint letter is sent and logged into the system.

Table 11: Send Complaint Letter

Payment System



Functionalities:

- **Manager Creates Payment Information.**
- **Resident** is Prompted with the total amount and **Pays Bills** with a method.

Use Case Name:	Create Payment Information	
Actor(s):	Manager	
Description:	This use case outlines the process for creating payment information for various bills in the payment system.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Step 1: Manager selects the option to create new payment information.	
		Step 2: Manager is prompted to select the type of bill for which the payment information is being created (Parking Bill or Utility Bill).
	Step 3: Manager selects the bill type	
		Step 4: The system validates and saves the payment information.
Precondition:	Manager has access rights to the payment system.	
Postcondition:	Payment information is created and stored in the system.	

Table 12: Create Payment Information

Use Case Name:	Notify Payment Information	
Actor(s):	Manager, Resident, Notification System	
Description:	This use case describes how the Payment System interacts with the Notification System to inform residents about new payment information.	
Typical Course of Events:	**Actor Action**	**System Response**
		Step 1: Upon the creation of payment information by Manager3, the Payment System triggers the notification process. Step 2: The Notification System generates a notification message including the bill type, amount, and due date. Step 3: The Notification System sends the notification to the Resident.
	Step 4: Resident receives the notification and becomes aware of the new payment information.	
Precondition:	New payment information has been created in the Payment System.	
Postcondition:	Resident is notified of the payment information.	

Table 13: Notify Payment Information

Use Case Name:	Pay Bills	
Actor(s):	Resident, CreditCardPaymentSystem, AlipayPaymentSystem, PayPalPaymentSystem	
Description:	This use case outlines the process for a resident to pay bills using the payment system.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Resident selects the "Pay Bills" option from their dashboard, which displays outstanding bills.	
		Step 2:The system calculates the total amount due for the selected bill(s).
	Step 3: Resident is prompted to select a payment method (Credit Card, Alipay, PayPal).	
		Step 4: The system processes the payment through the chosen payment gateway (CreditCardPaymentSystem, AlipayPaymentSystem, or PayPalPaymentSystem).
Precondition:	The Resident has received notification of the bill. The resident has sufficient funds available for the payment.	
Postcondition:	The bill is paid.	

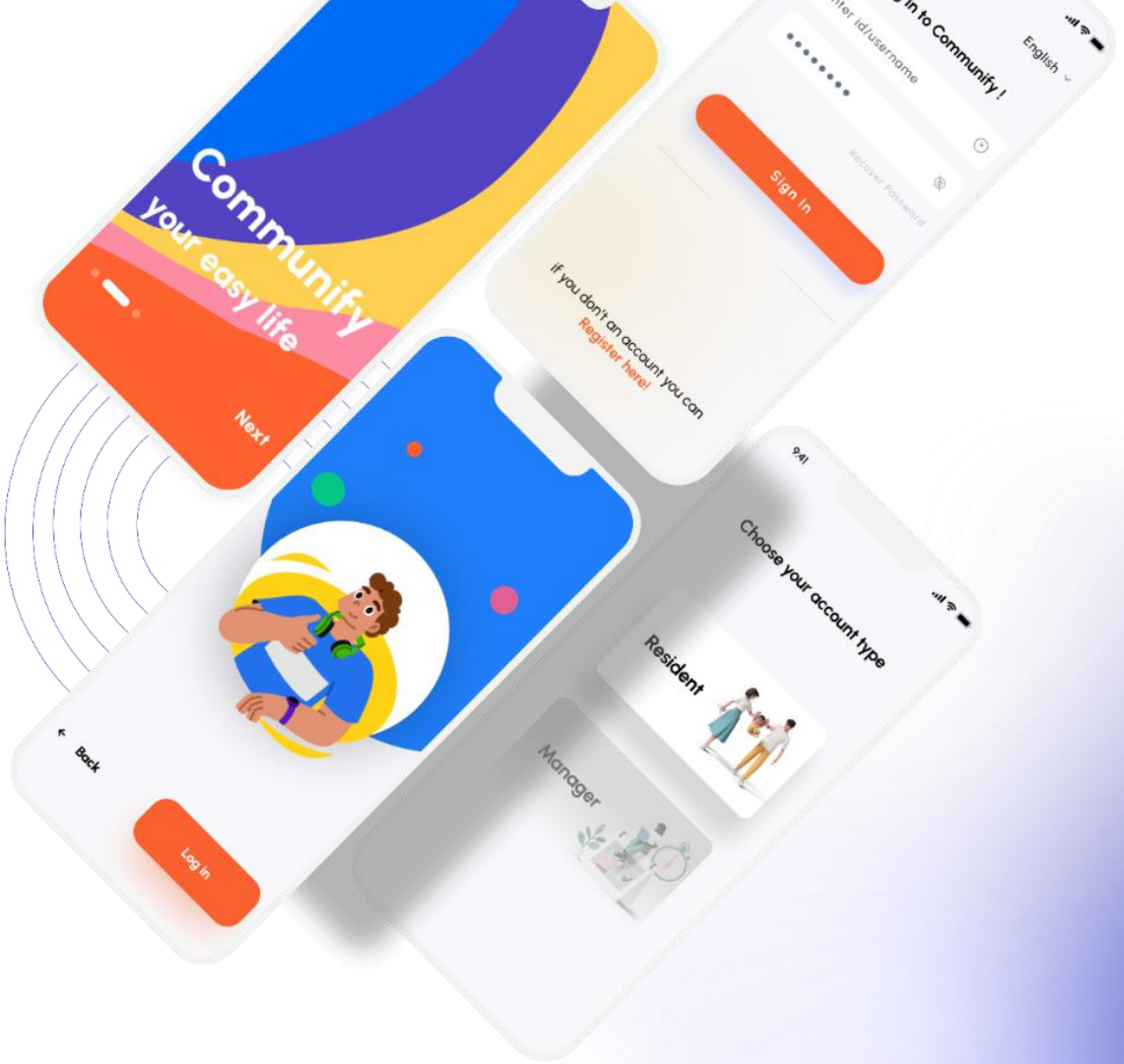
Table 14: Pay Bills

04

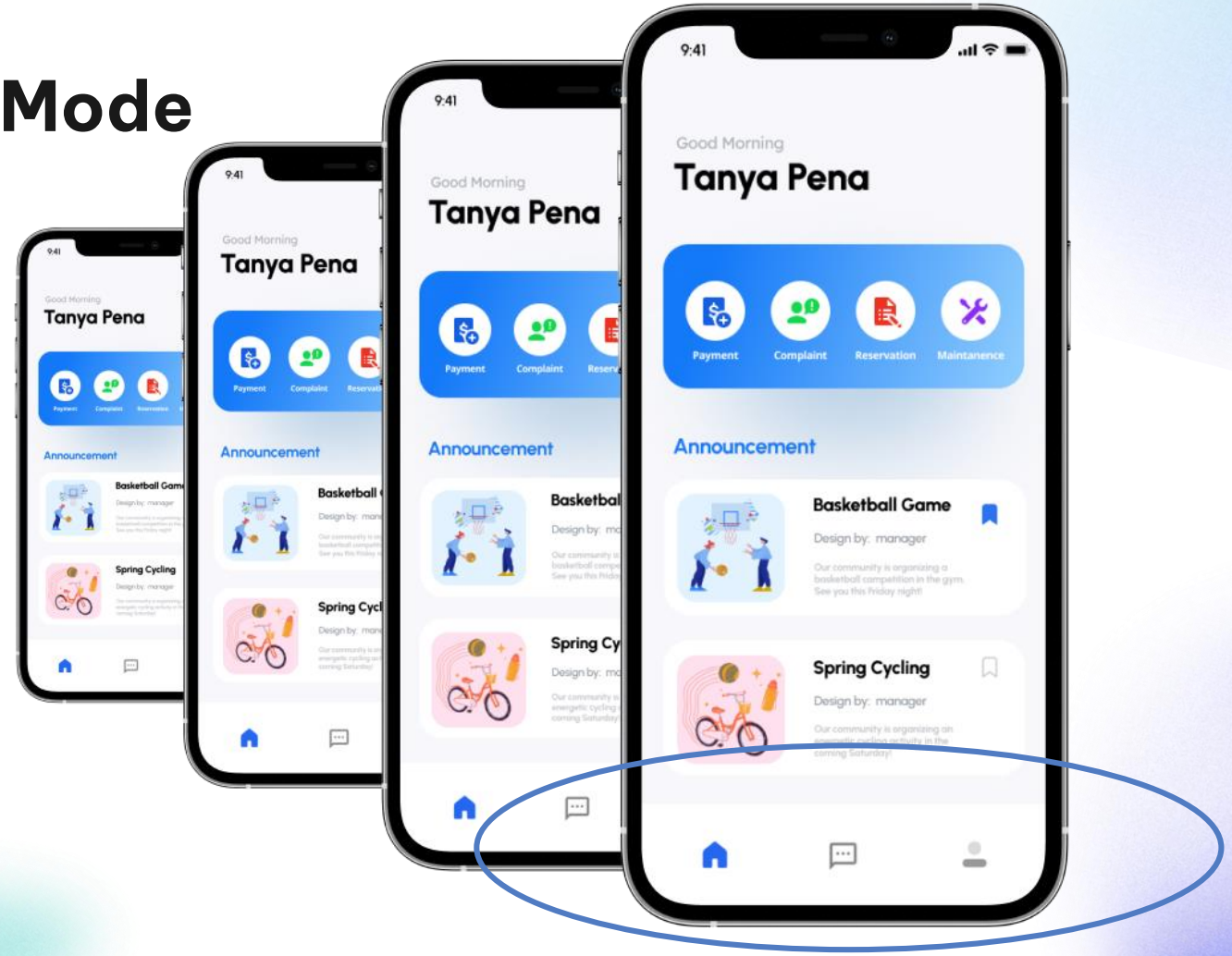
The prototype

Log-in

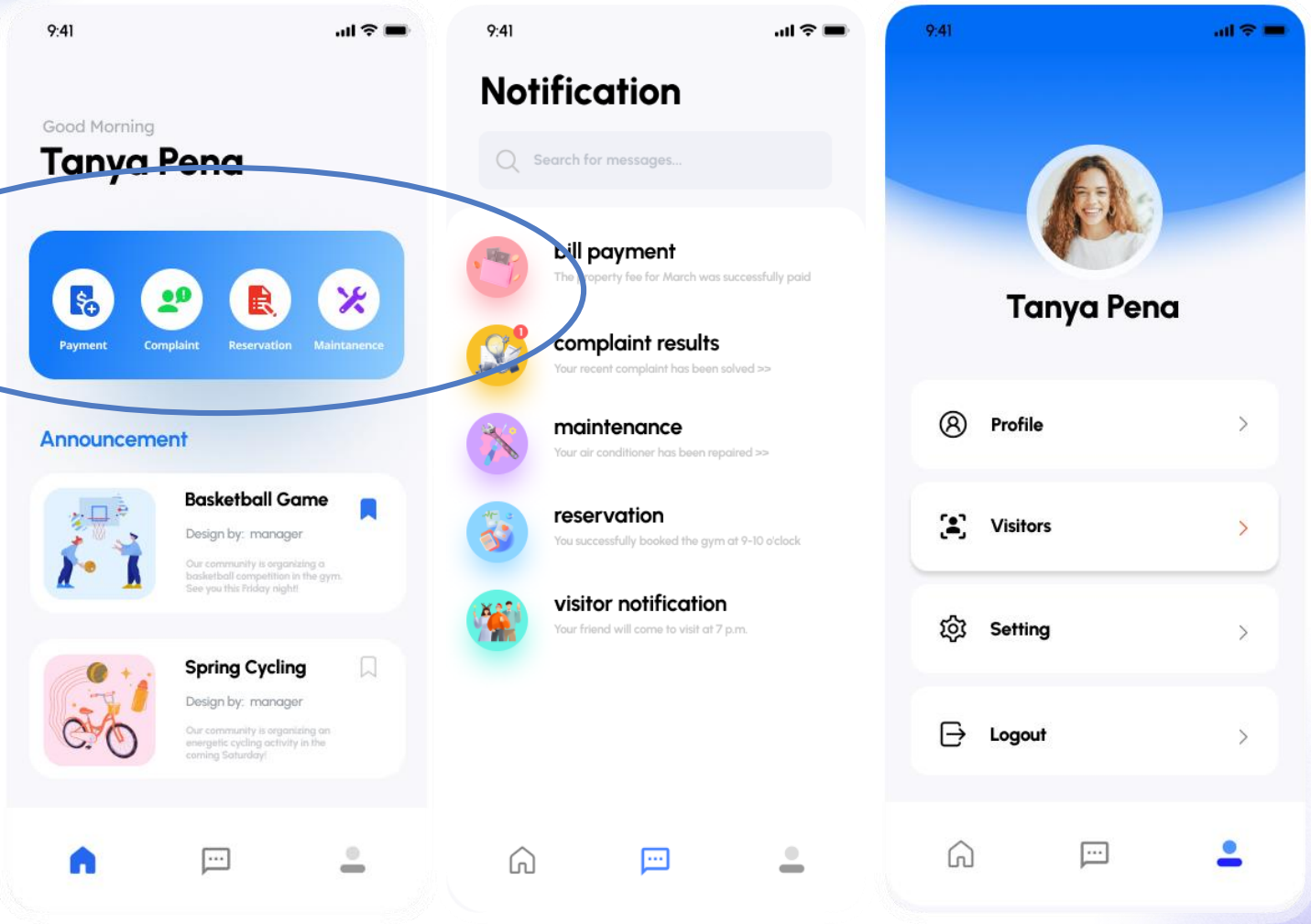
Choose Account Type !



Resident Mode

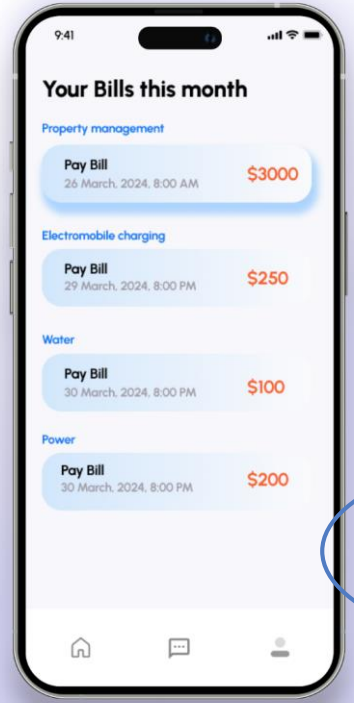


Three Main Page

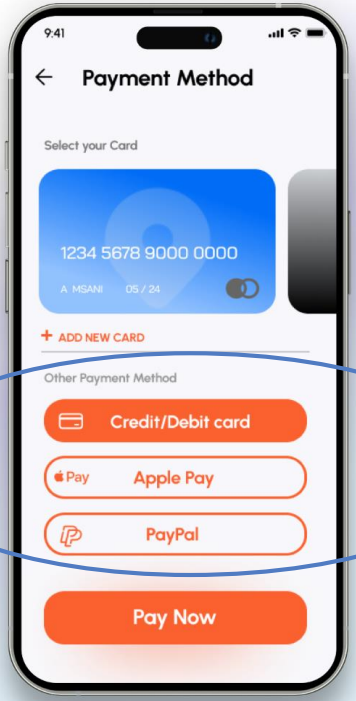


Payment

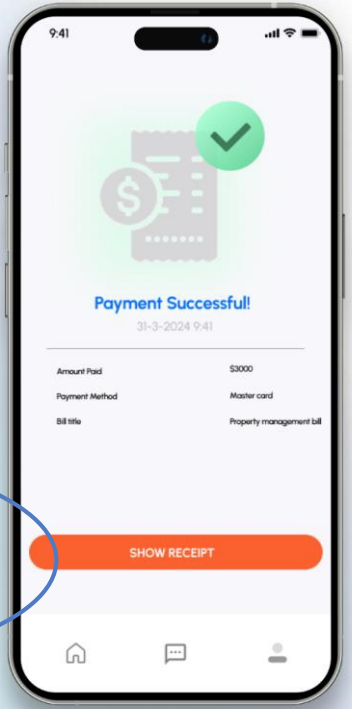
Check Your Bills



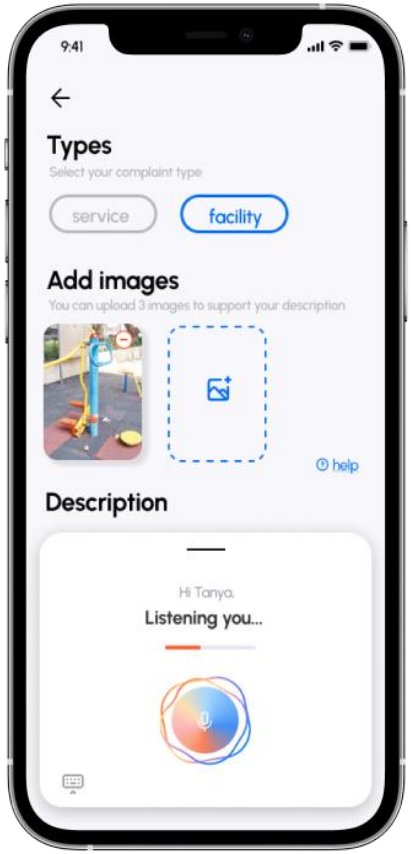
Choose your preferred Payment Method

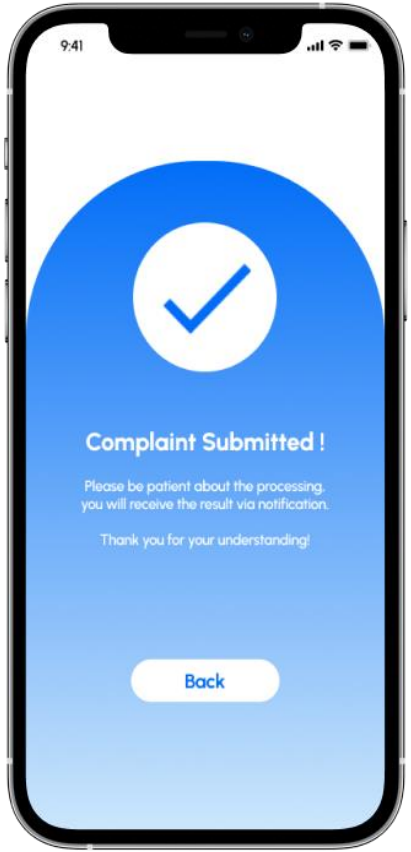


Paid Successfully

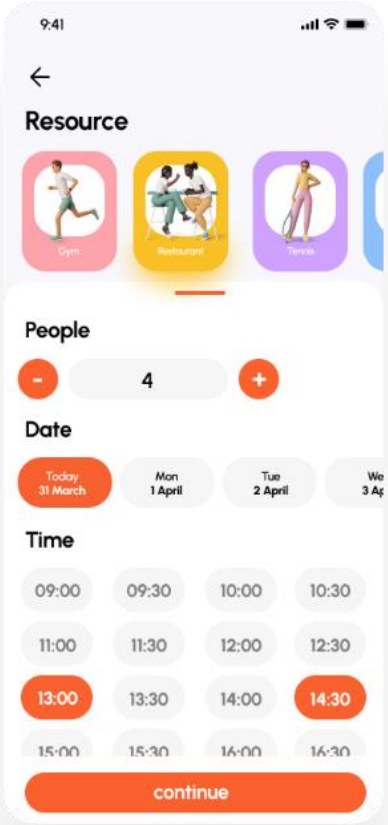


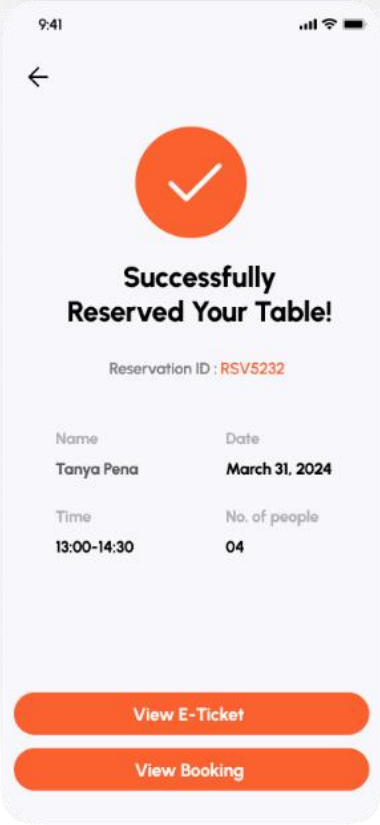
Complaint


9:41
Types
Select your complaint type
service
facility
Add images
You can upload 3 images to support your description
help
Description
Hi Tanya,
Listening you...

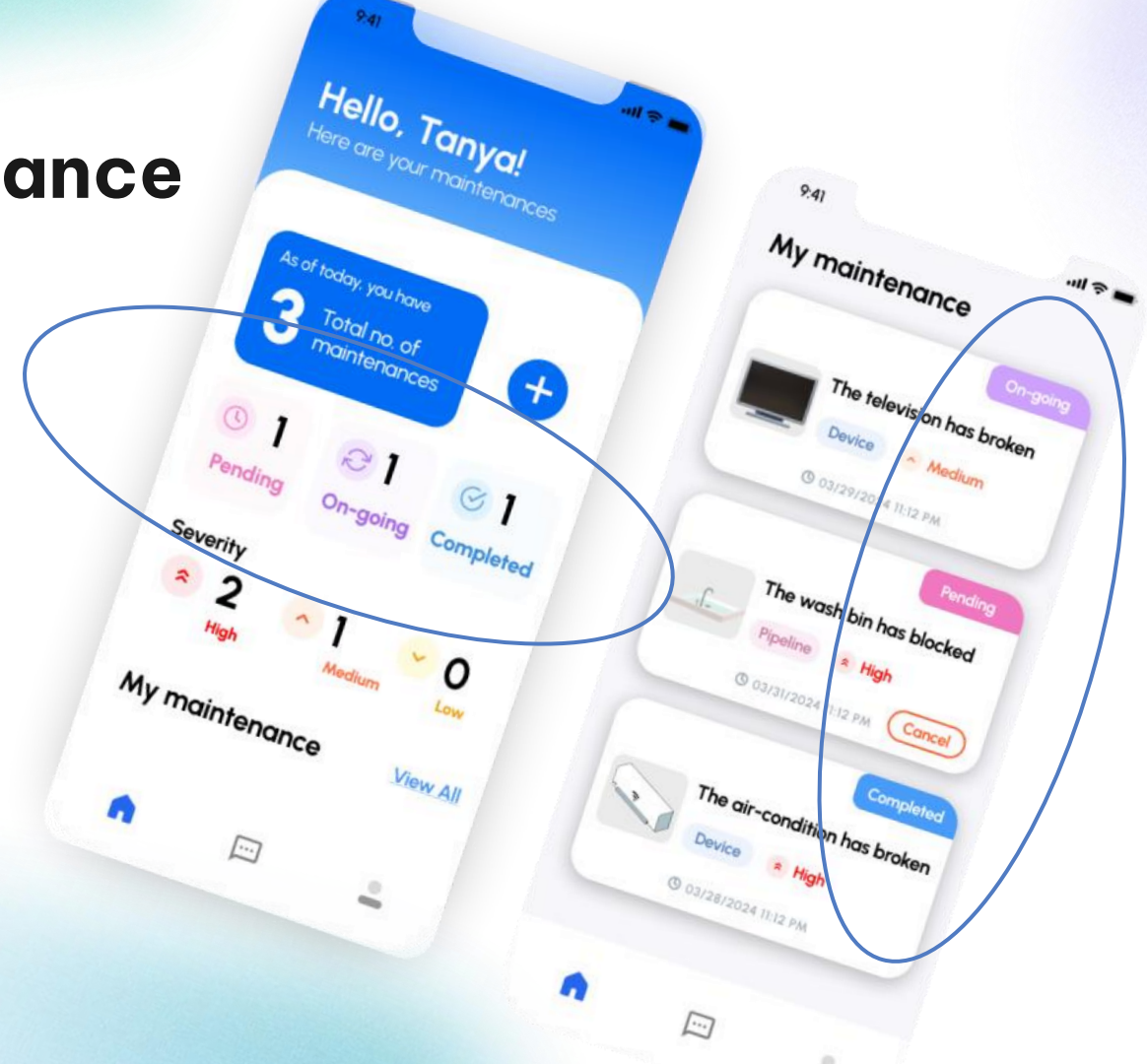

9:41
Complaint Submitted !
Please be patient about the processing.
you will receive the result via notification.
Thank you for your understanding!
Back

Reservation

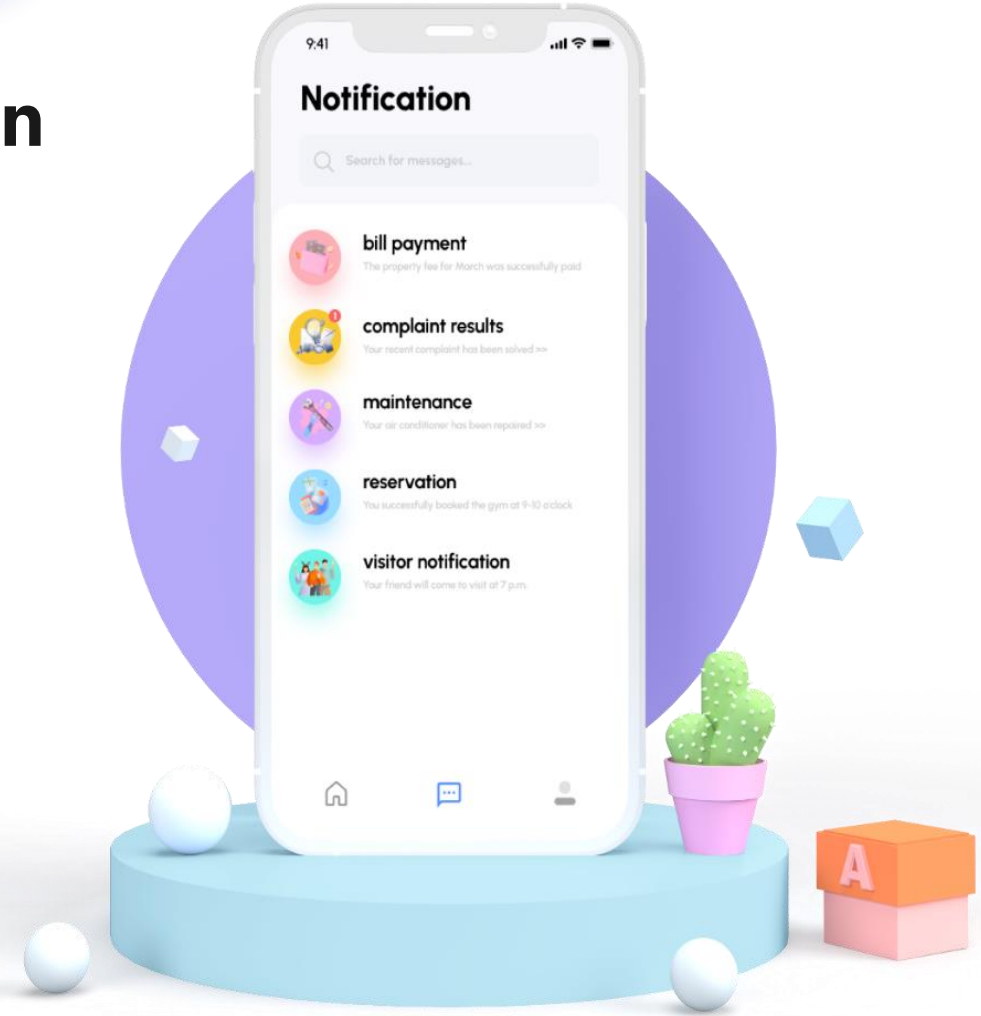




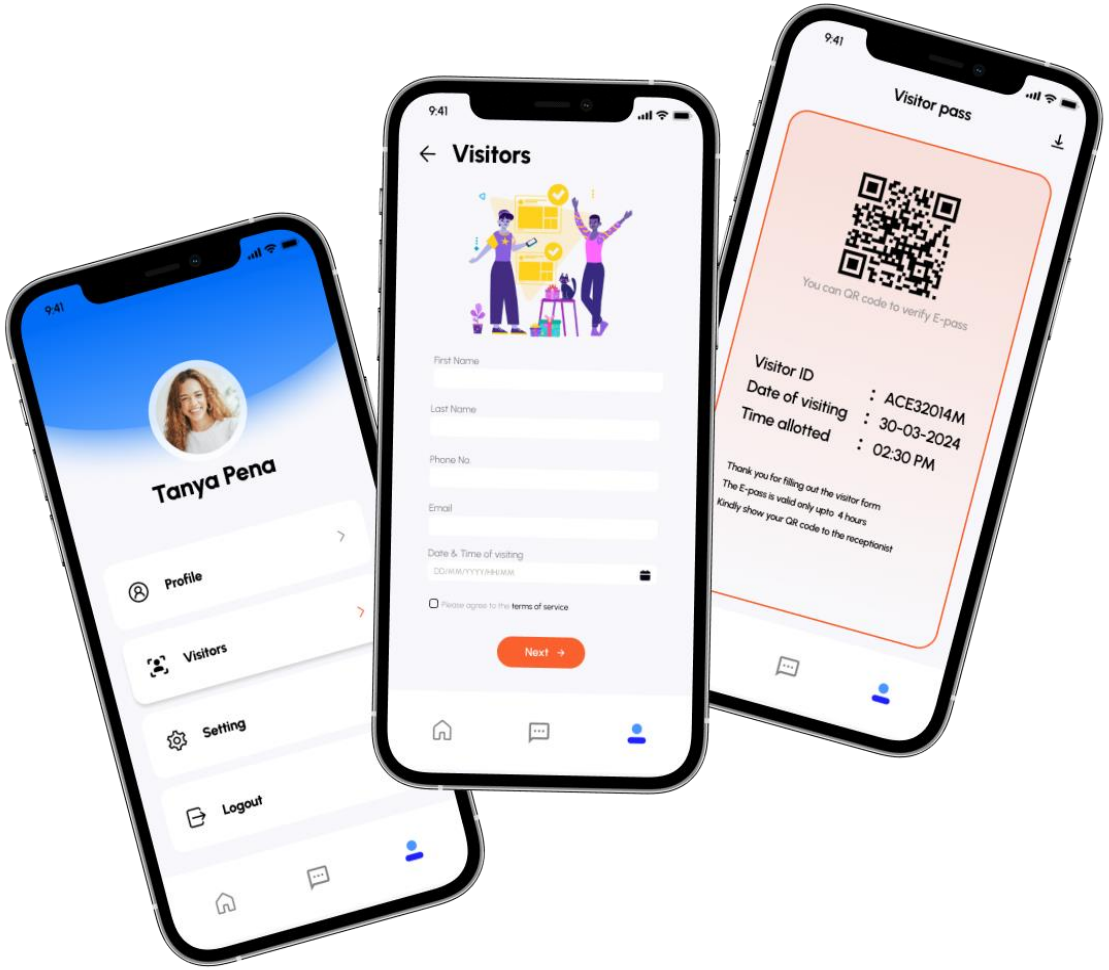
Maintenance



Notification

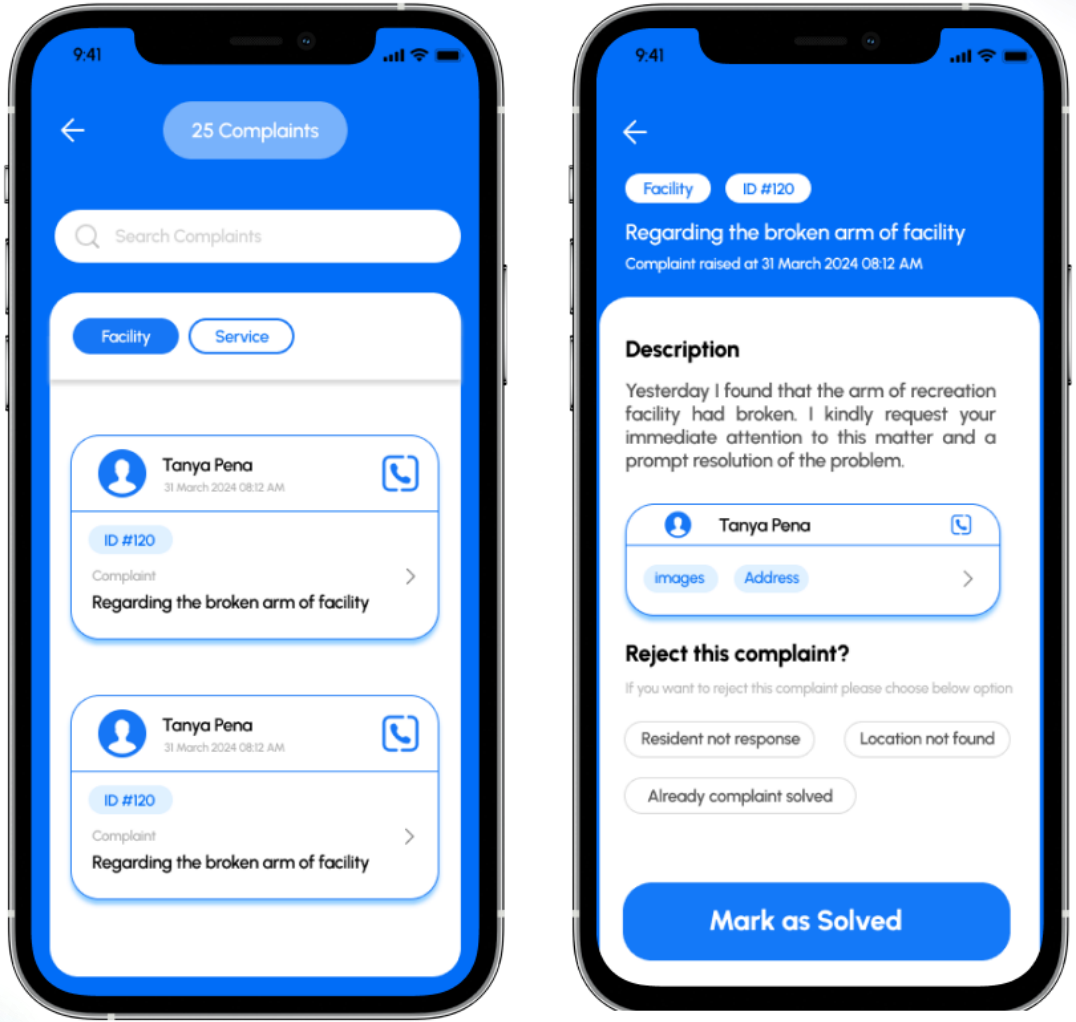


Visitor


9:41
Tanya Pena
Profile
Visitors
Setting
Logout
9:41
Visitors
First Name
Last Name
Phone No.
Email
Date & Time of visiting
Please agree to the terms of service
Next
9:41
Visitor pass
You can QR code to verify E-pass
Visitor ID : ACE32014M
Date of visiting : 30-03-2024
Time allotted : 02:30 PM
Thank you for filling out the visitor form
The E-pass is valid only upto 4 hours
Kindly show your QR code to the receptionist

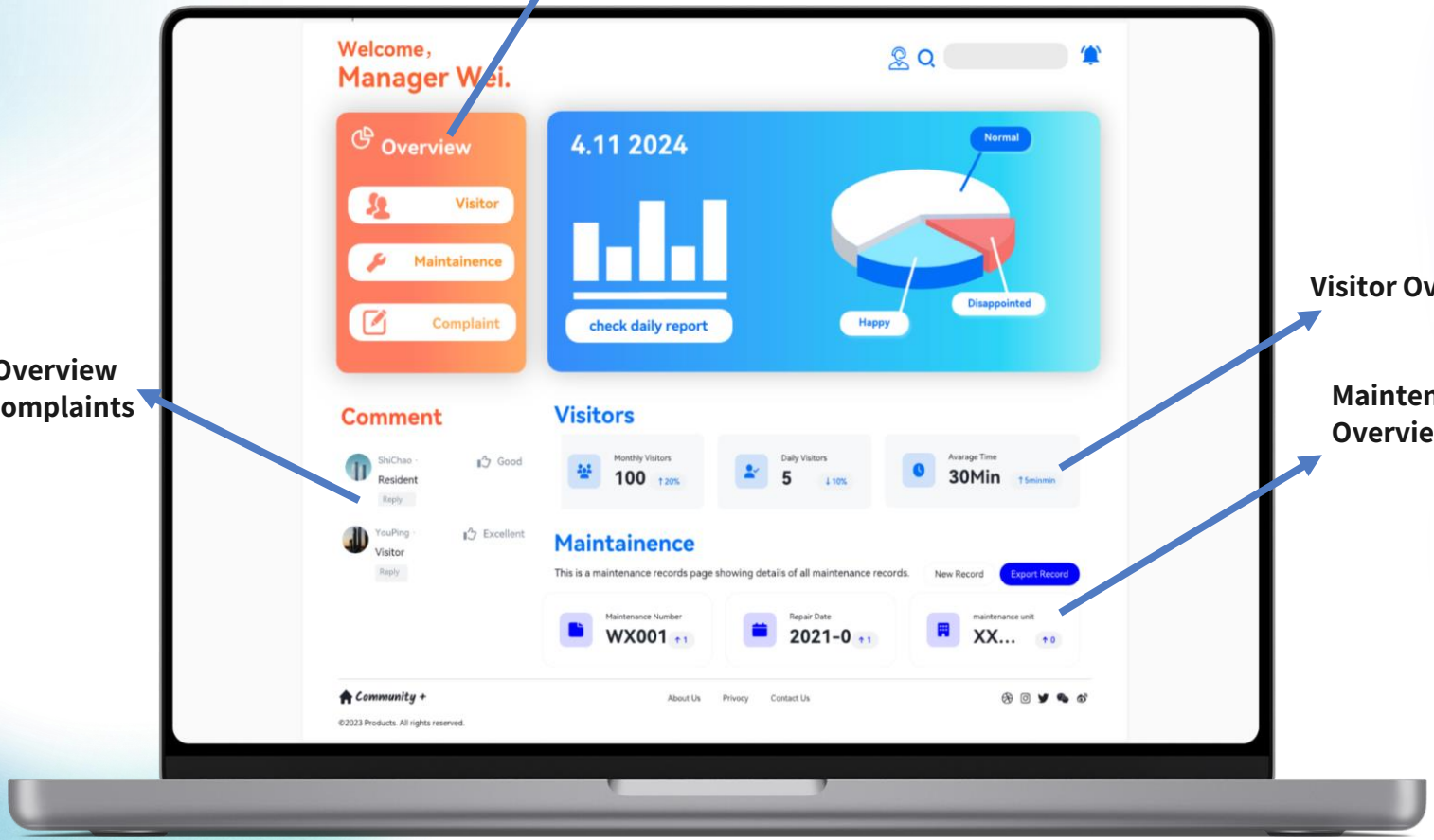
Manager Mode

Example: Complaint



Manager Dashboard

Navigation block



Visitor Overview

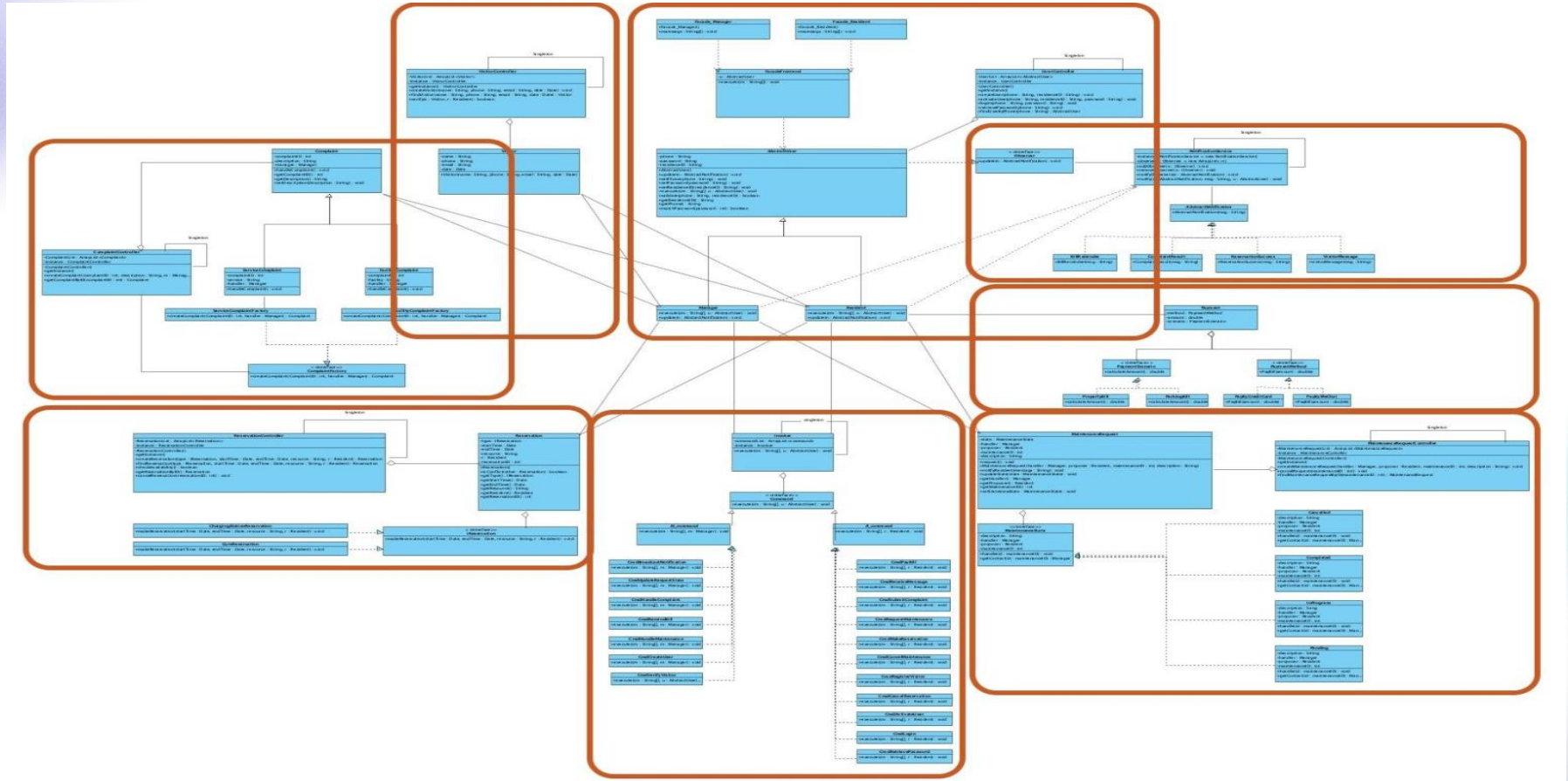
Maintenance Overview

Comment Overview Including complaints

05

Class Diagram

Class Diagram



Design Pattern

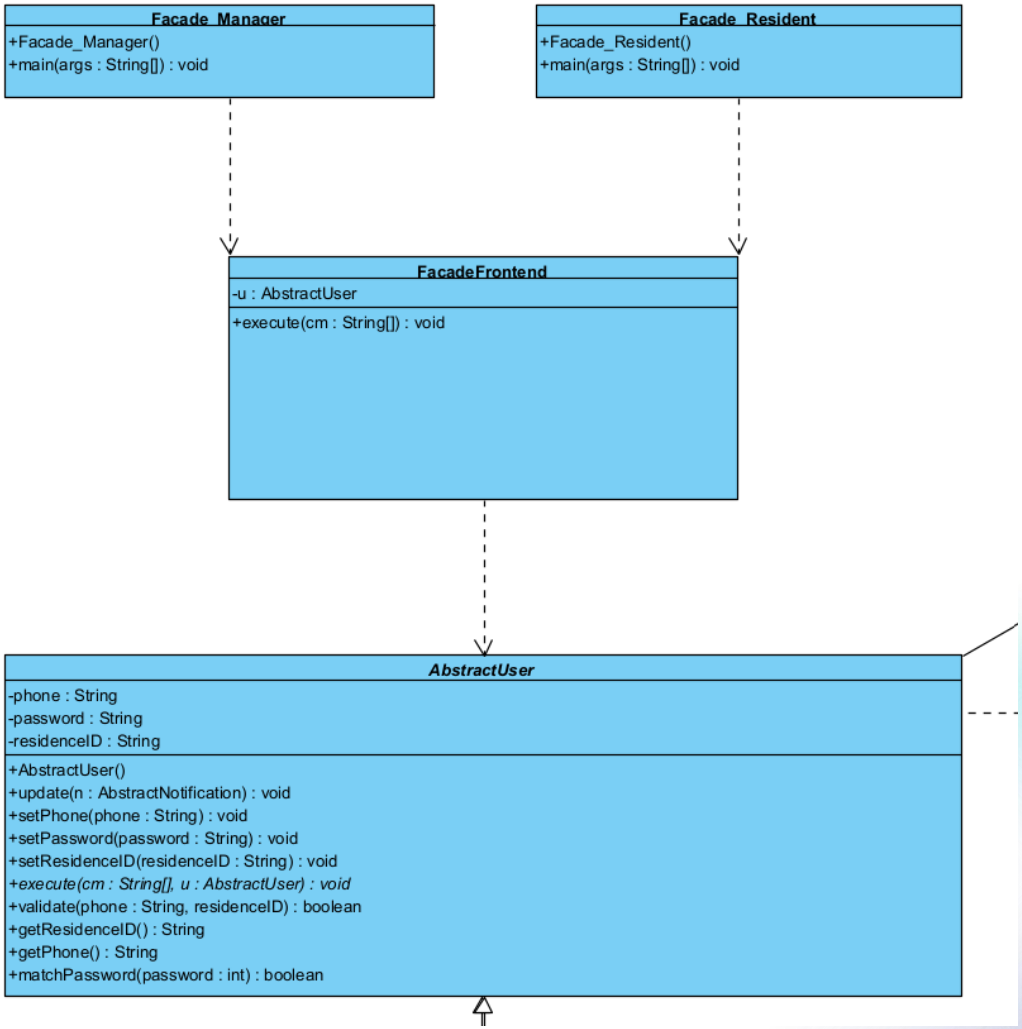
- ✓ Strategy Pattern (e.g. ***Payment*** Page 49)
- ✓ State Pattern (e.g. ***Maintenance*** Page 50)
- ✓ Factory Pattern (e.g. ***Complaint*** Page 52)
- ✓ Command Pattern (e.g. ***Command*** Page 47)
- ✓ Facade Pattern (e.g. ***Account Type*** Page 45)
- ✓ Observer Pattern (e.g. ***Notification*** Page 48)
- ✓ Singleton Pattern (e.g. ***Reservation*** Page 51)

Design Principle

- ✓ Open-closed principle (OCP) (e.g. ***Payment*** Page 49)
- ✓ Liskov substitution principle (LSP) (e.g. ***Account Type*** Page 45)
- ✓ Dependency inversion principle (DIP) (e.g. ***Complaint*** Page 52)
- ✓ Single responsibility principle (SRP) (e.g. ***Command*** Page 47)
- ✓ Interface segregation principle (ISP) (e.g. ***Reservation*** Page 51)
- ✓ Law of Demeter(LoD) (e.g. ***Notification*** Page 48)

Account Type

- Design Pattern
  - ✓ **Facade Pattern**
  - **Singleton Pattern**
- Design Principle
  - **SRP**
  - **OCP**
  - ✓ **LSP**
  - **ISP**
  - **DIP**



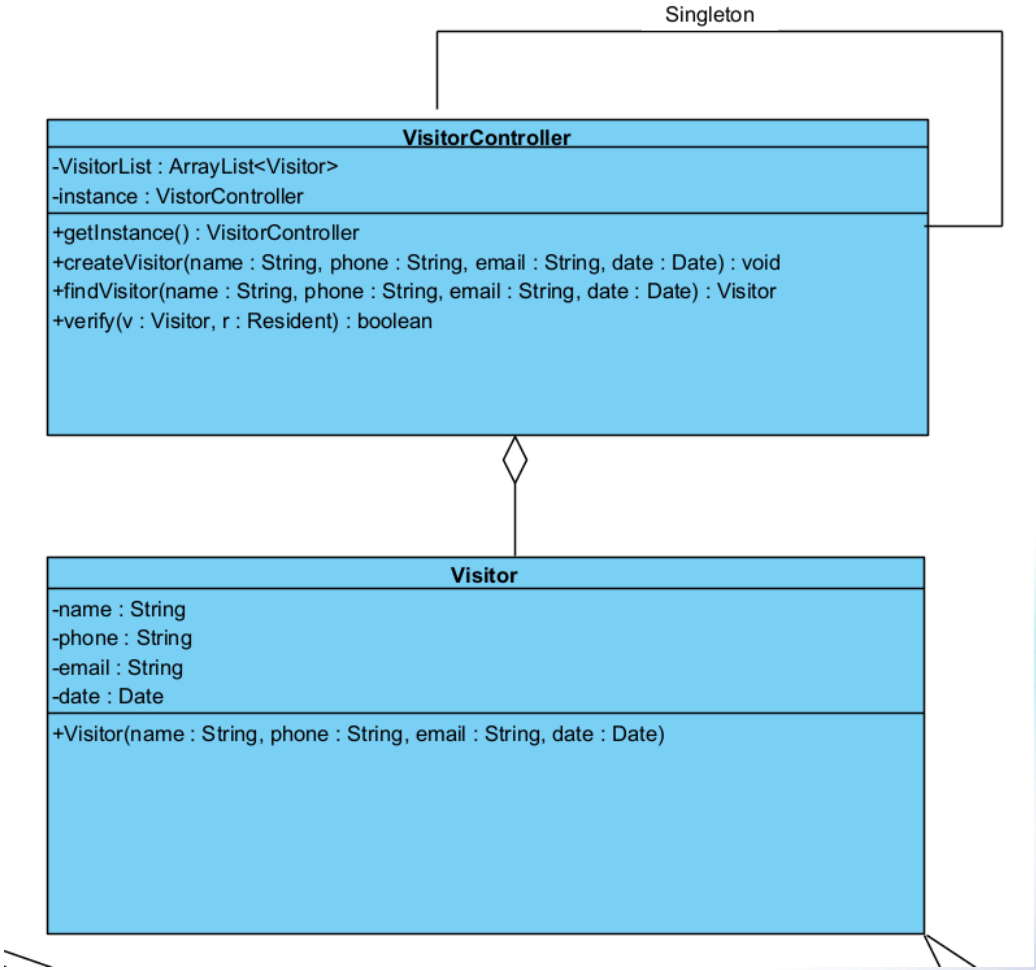
Visitors

- Design Pattern

  Singleton Pattern

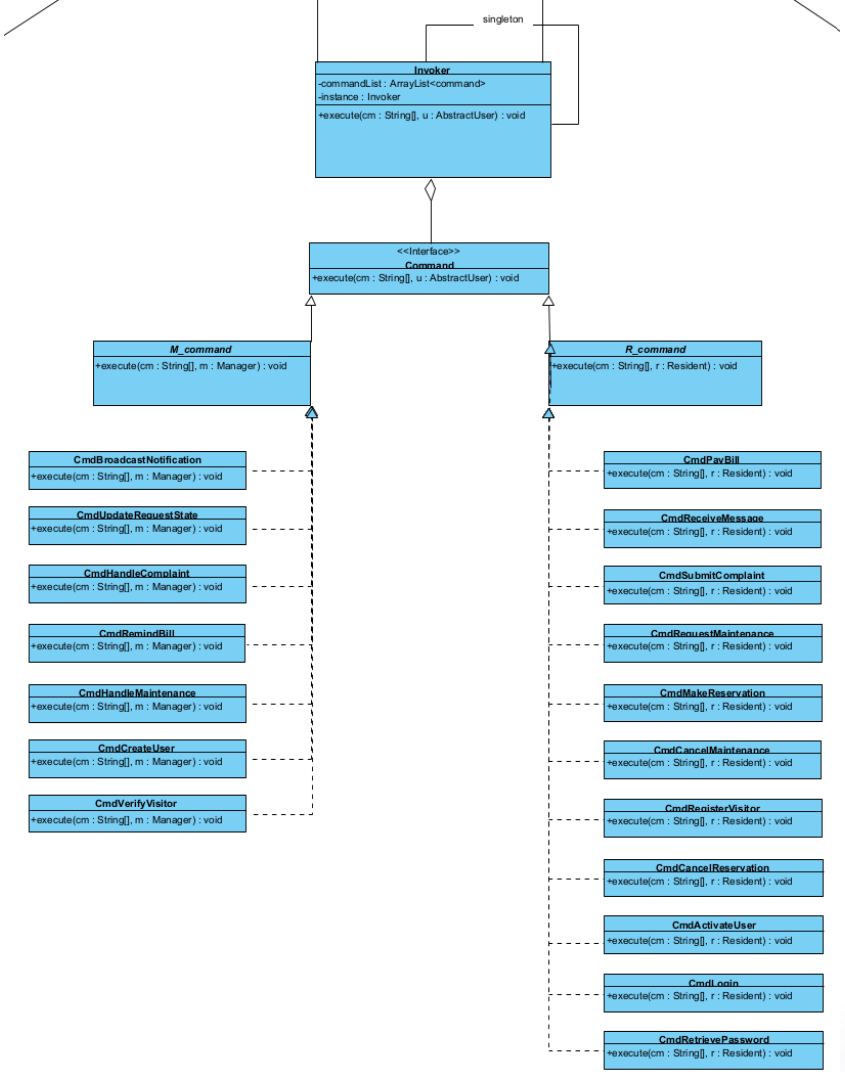
- Design Principle

  SRP
  OCP



Command

- Design Pattern
  - ✓ **Command Pattern**
  - **Singleton Pattern**
- Design Principle
  - ✓ **SRP**
  - **OCP**
  - **ISP**
  - **DIP**



Notification

Design Pattern

✓ **Observer Pattern**

Singleton Pattern

Design Principle

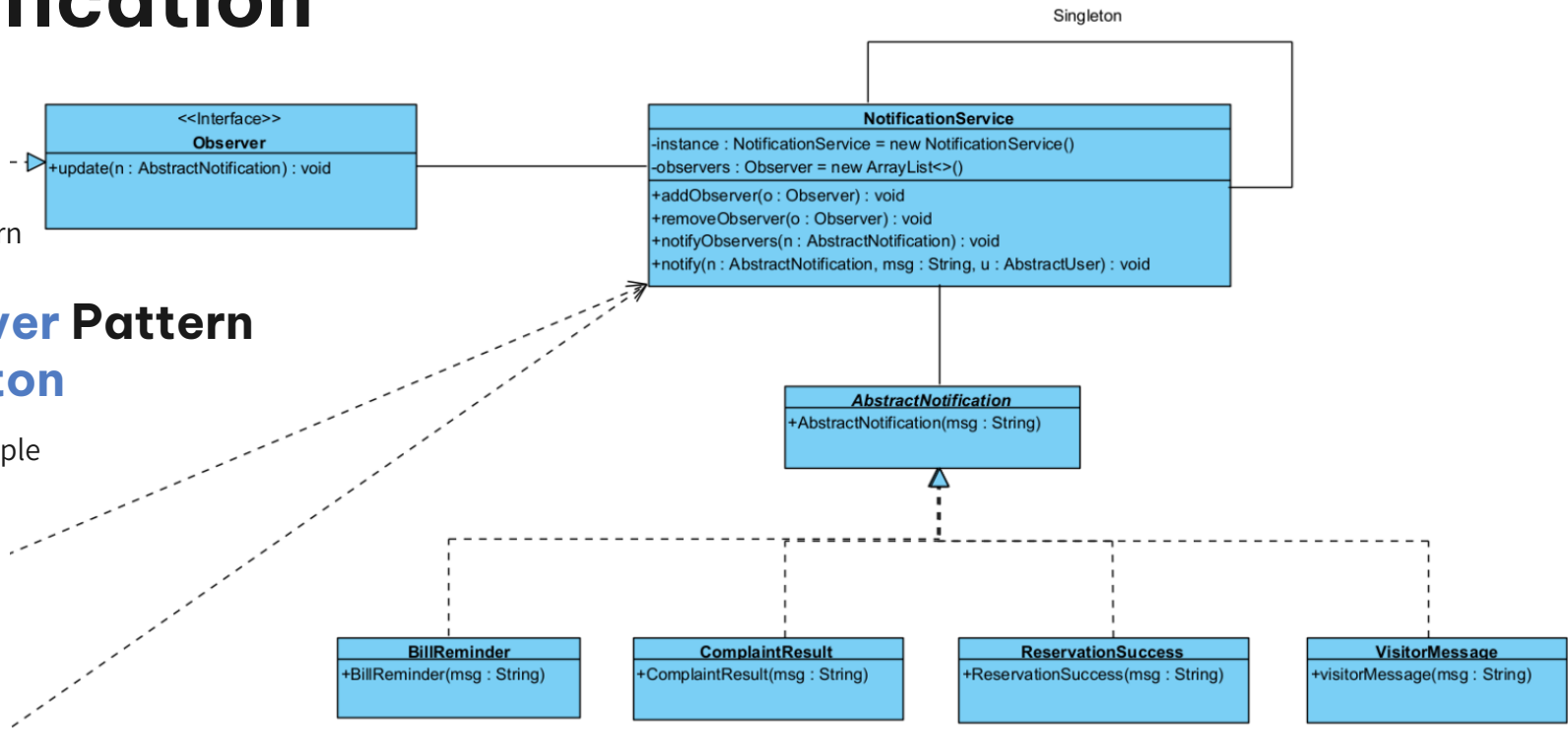
SRP

OCP

✓ **LoD**

ISP

DIP



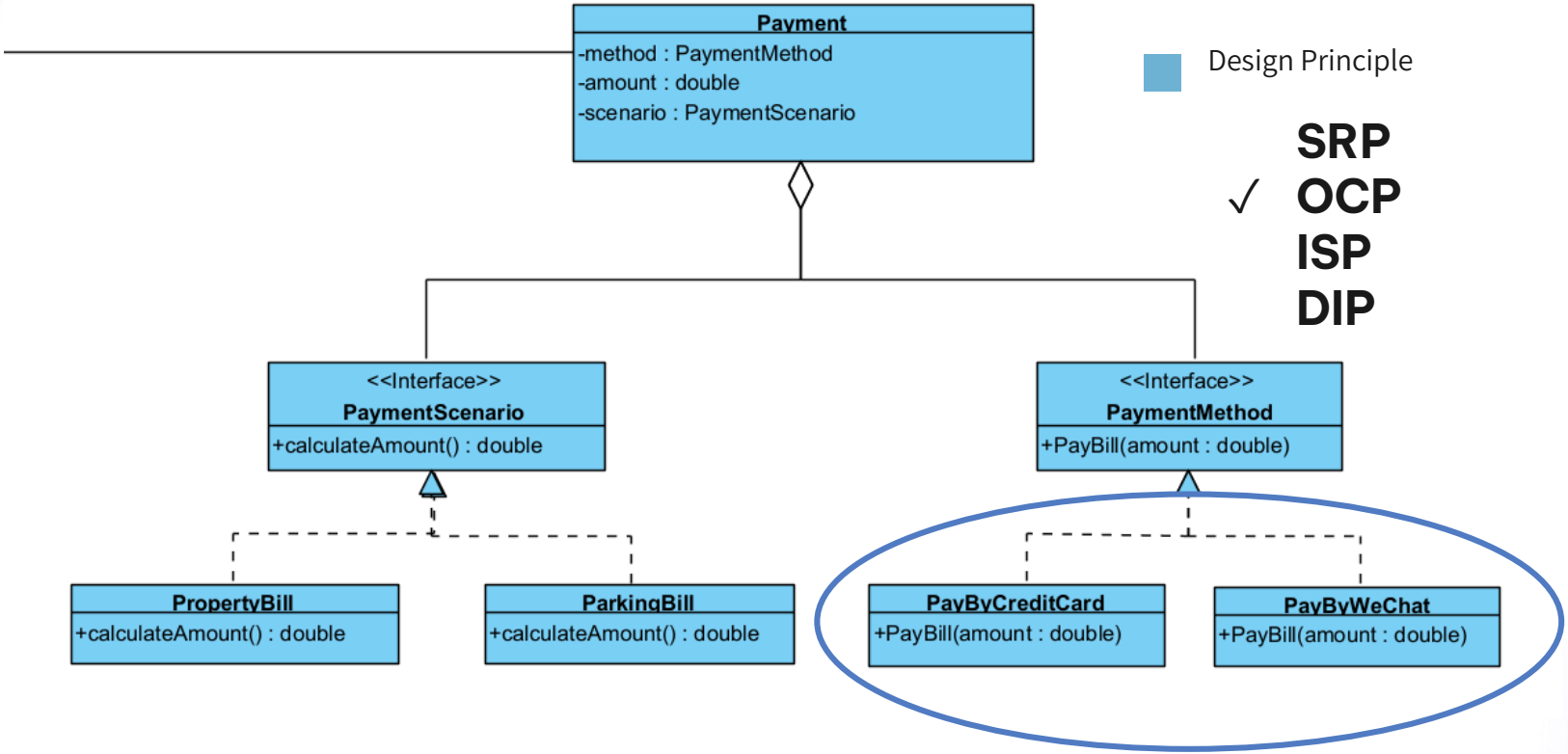
Payment

Design Pattern

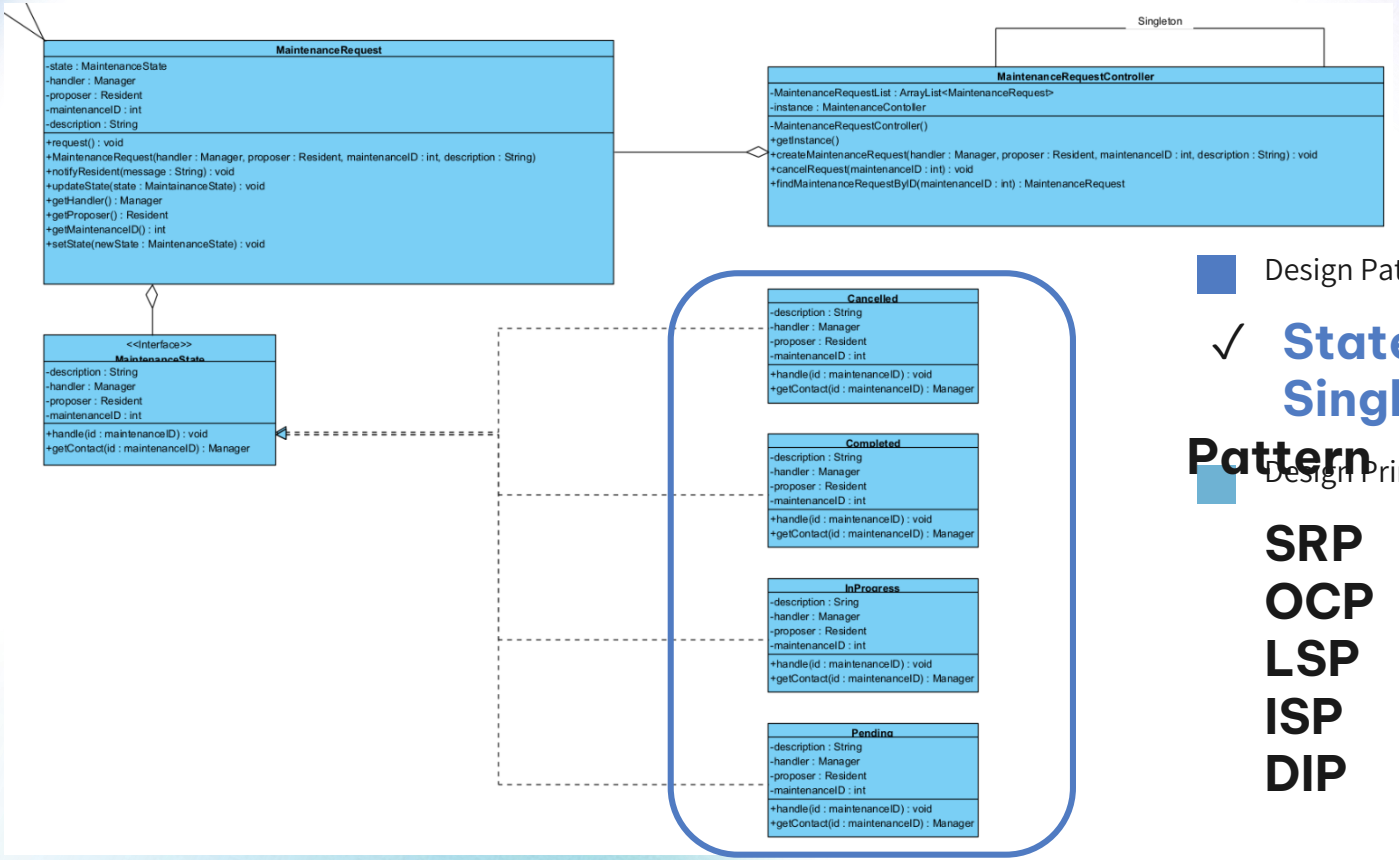
✓ **Strategy Pattern**

Design Principle

SRP
✓ **OCP**
ISP
DIP



Maintenance System

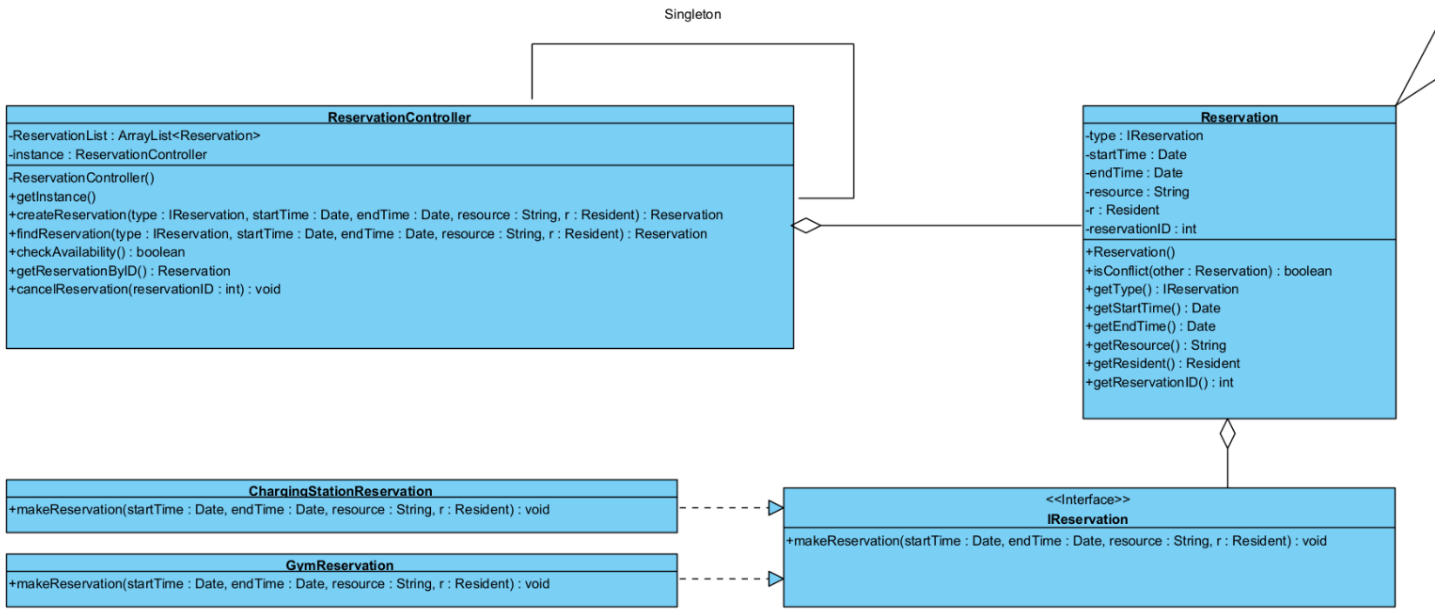


- Design Pattern
  - ✓ **State Pattern**
  - **Singleton Pattern**
- Design Principle
  - **SRP**
  - **OCP**
  - **LSP**
  - **ISP**
  - **DIP**

Reservation

- Design Pattern
- ✓ **Singleton Pattern**
- Design Principle

SRP
OCP
LSP
ISP
DIP



Complaint

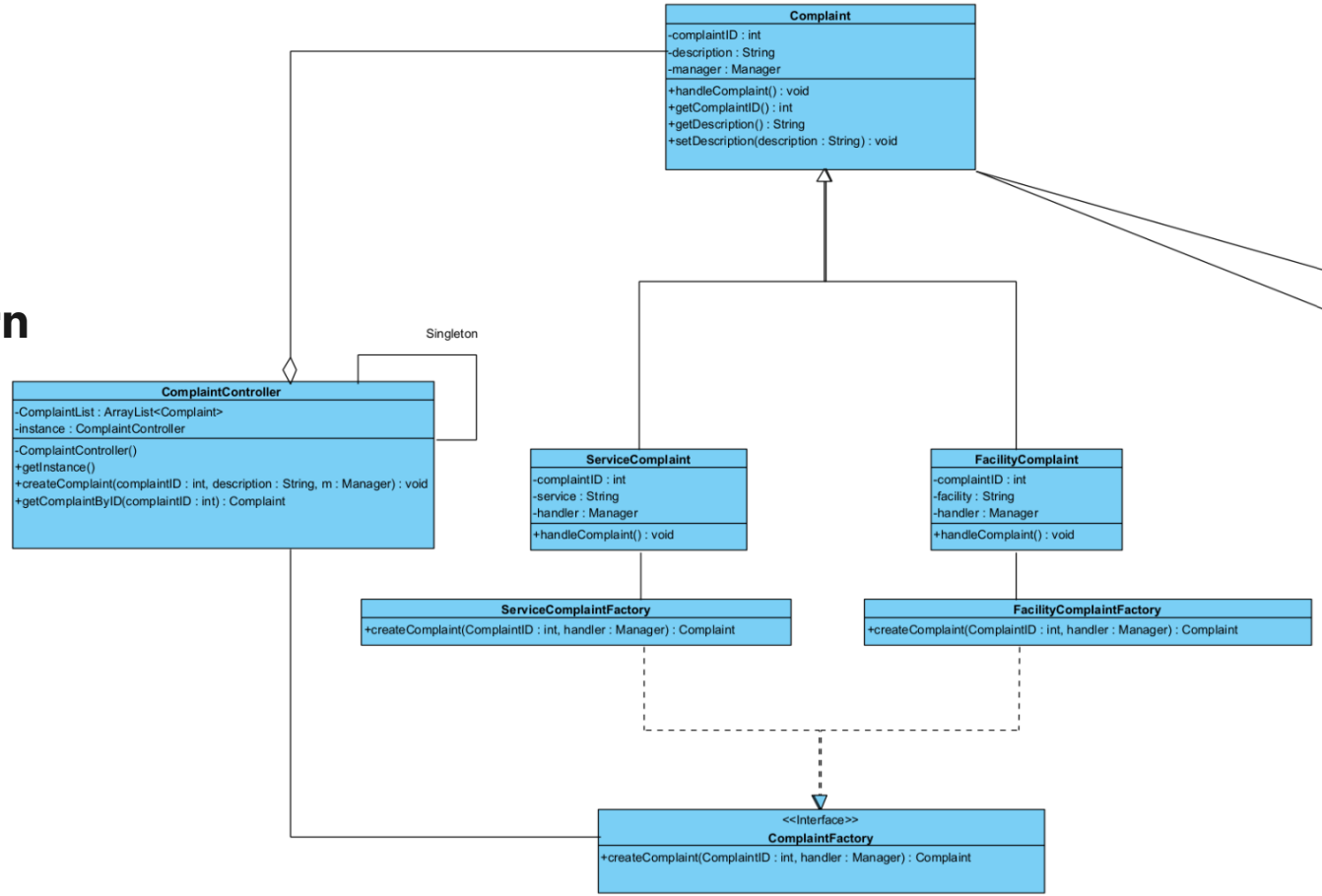
Design Pattern

Singleton Pattern

✓ **Factory Pattern**

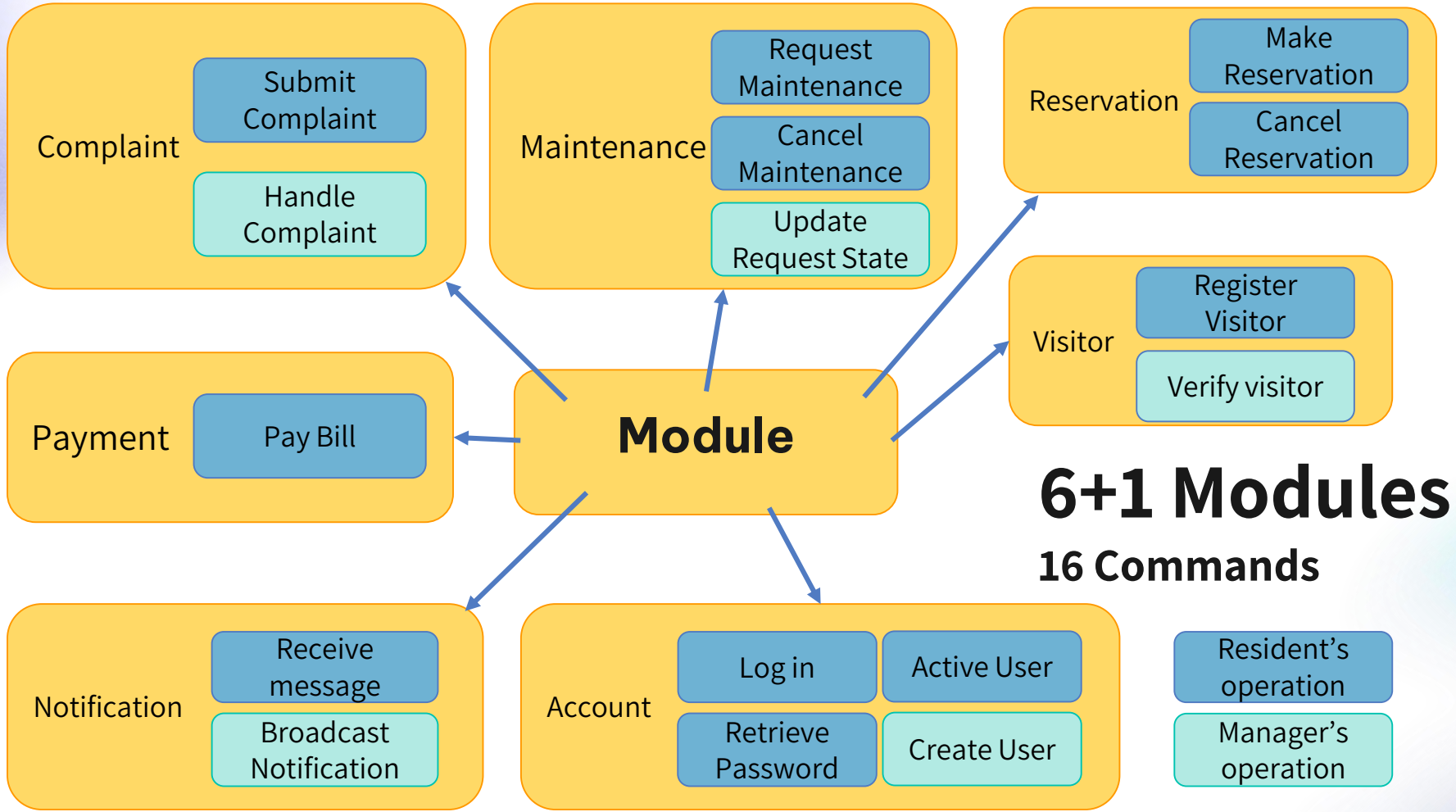
Design Principle

SRP
OCP
LSP
ISP
DIP



06

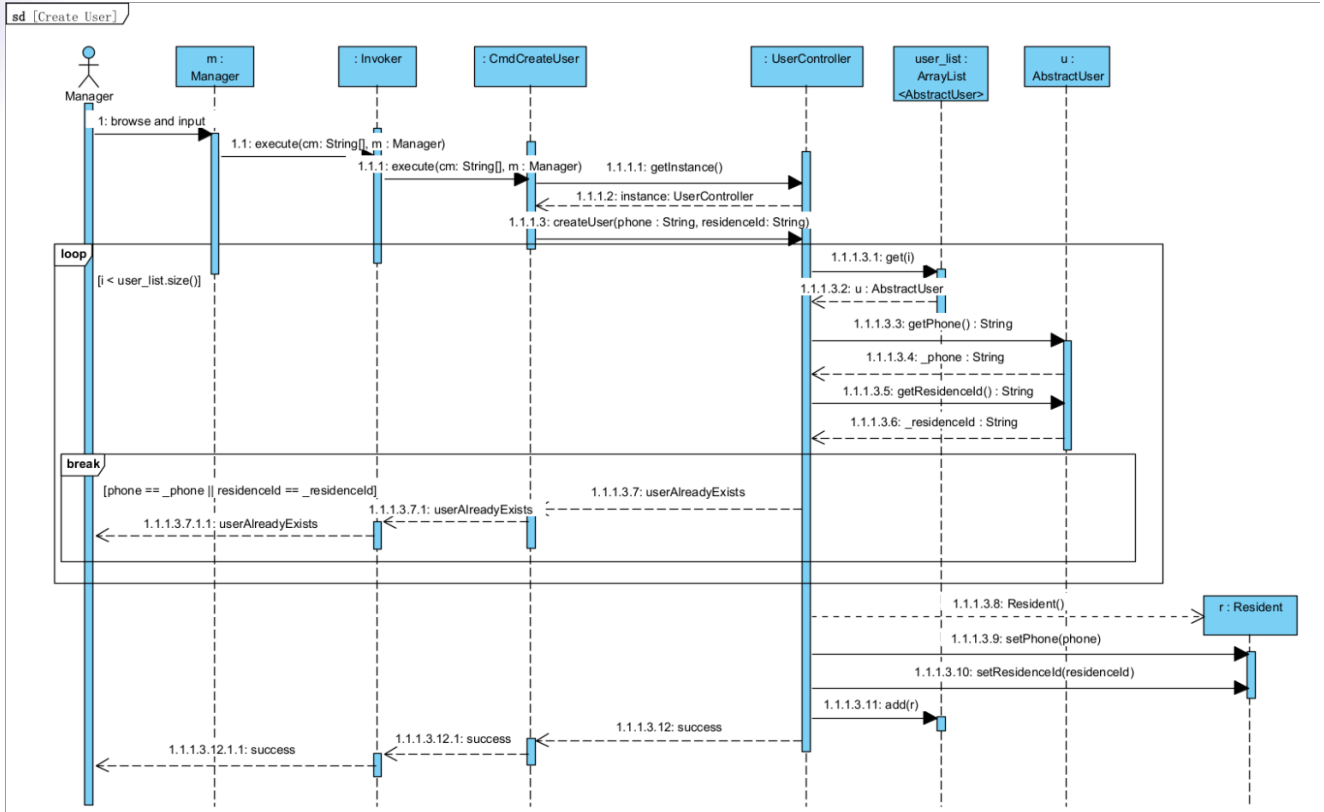
Sequence Diagram



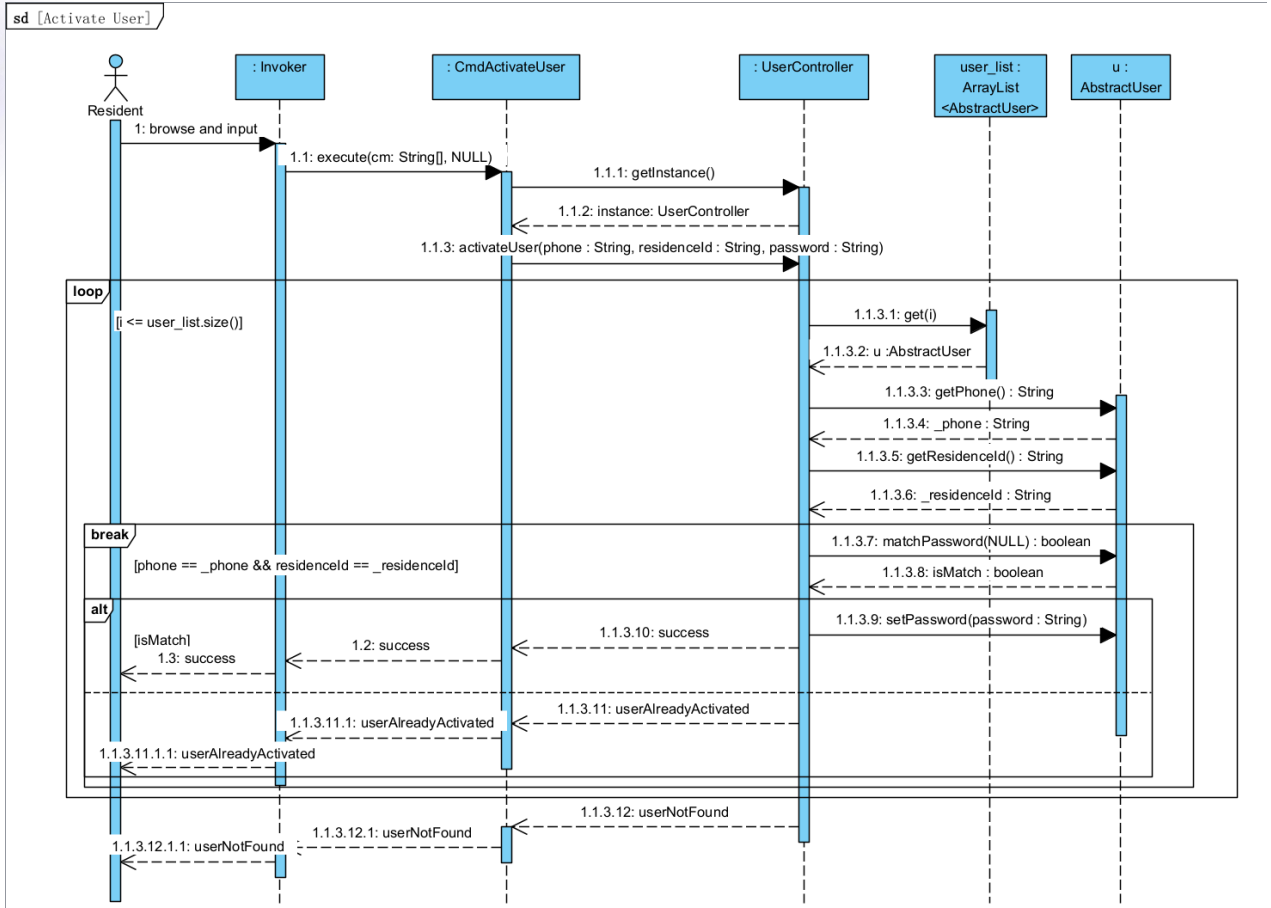
6+1 Modules

16 Commands

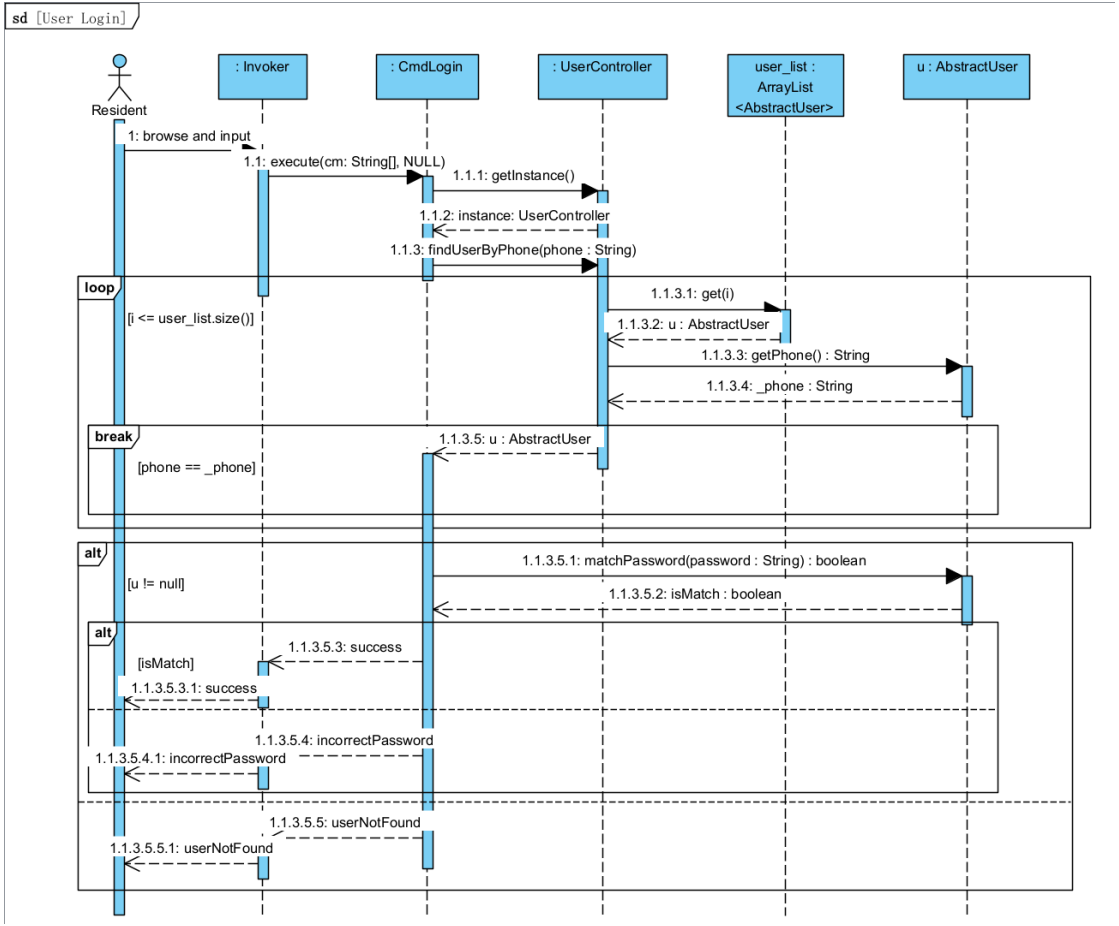
Account Module - Create User



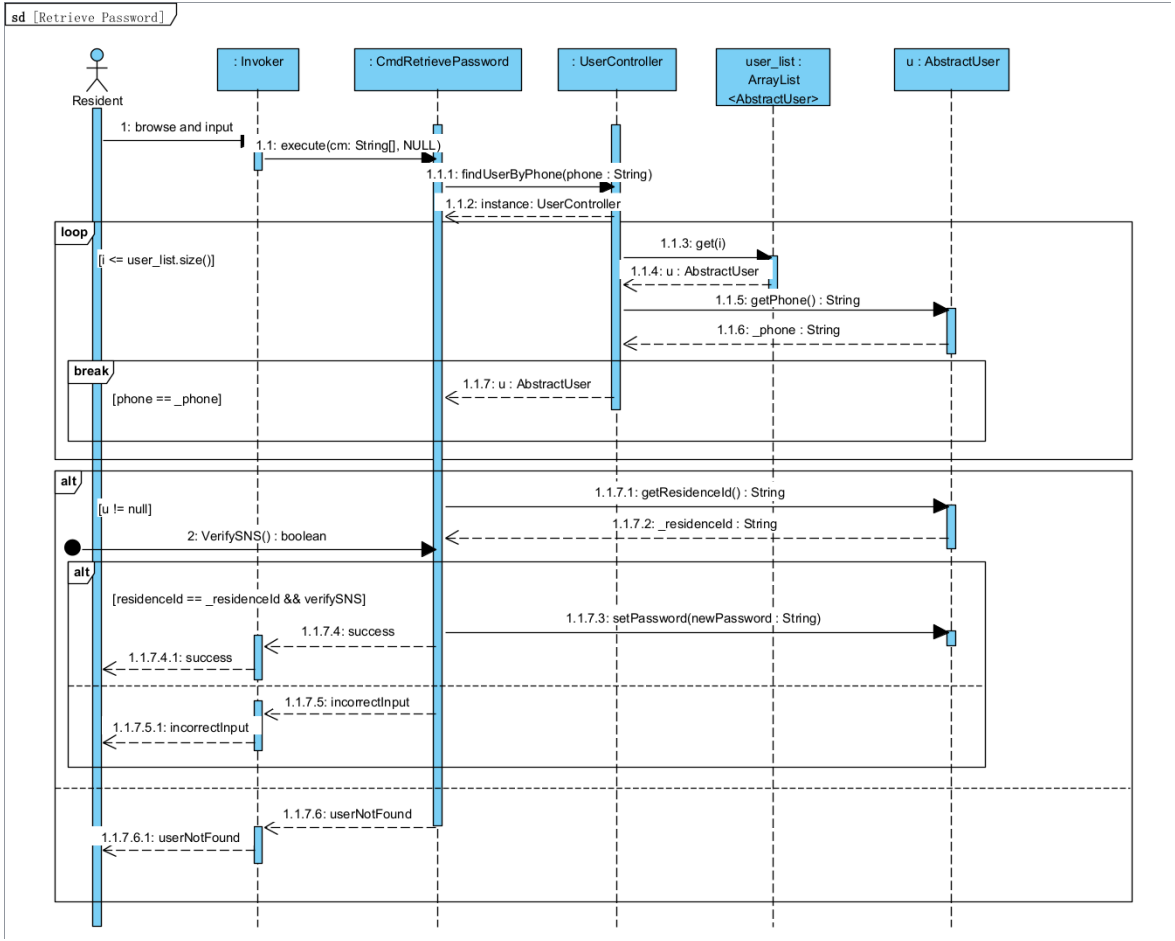
Account Module - Activate User



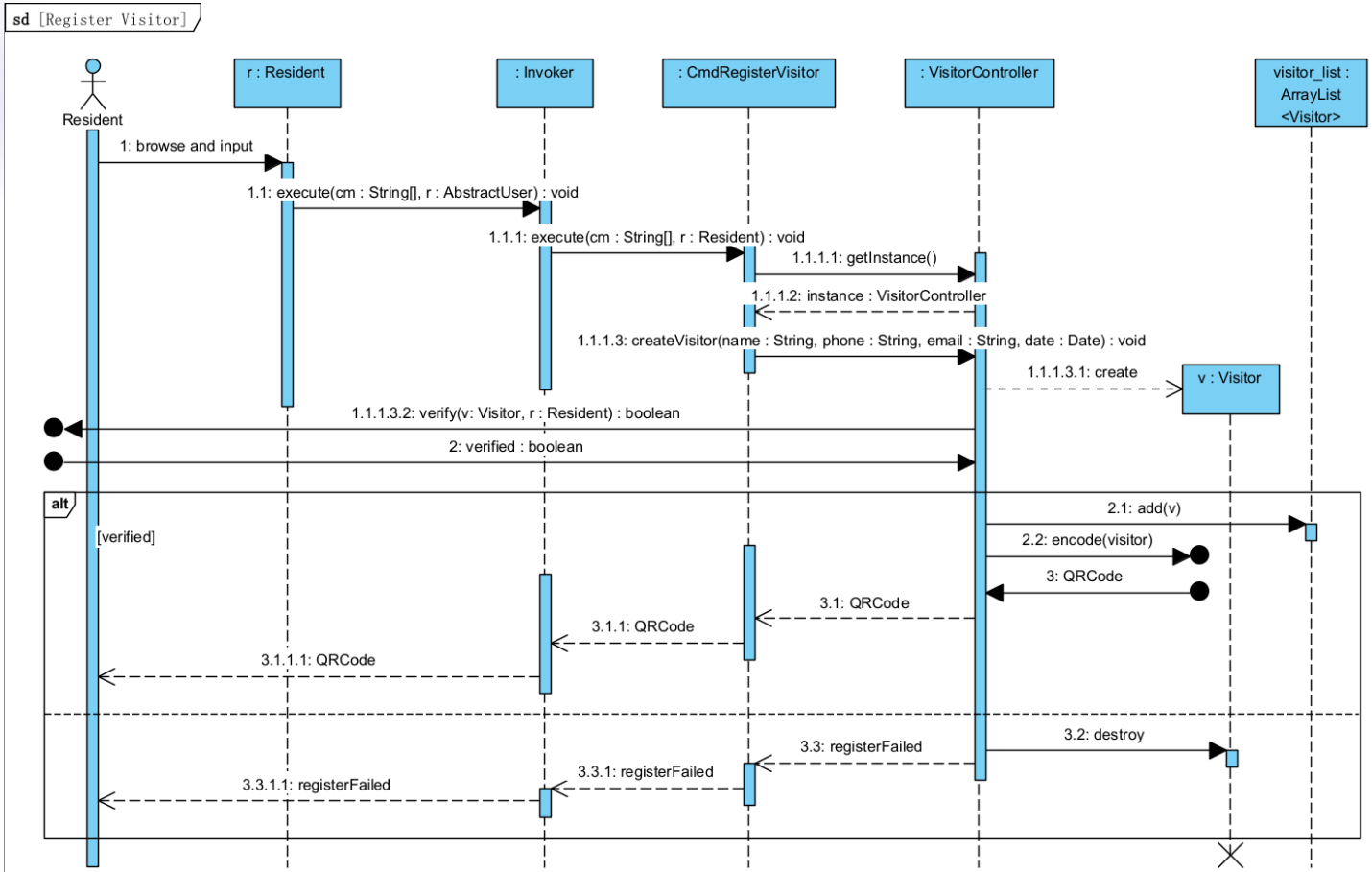
Account Module - User Login



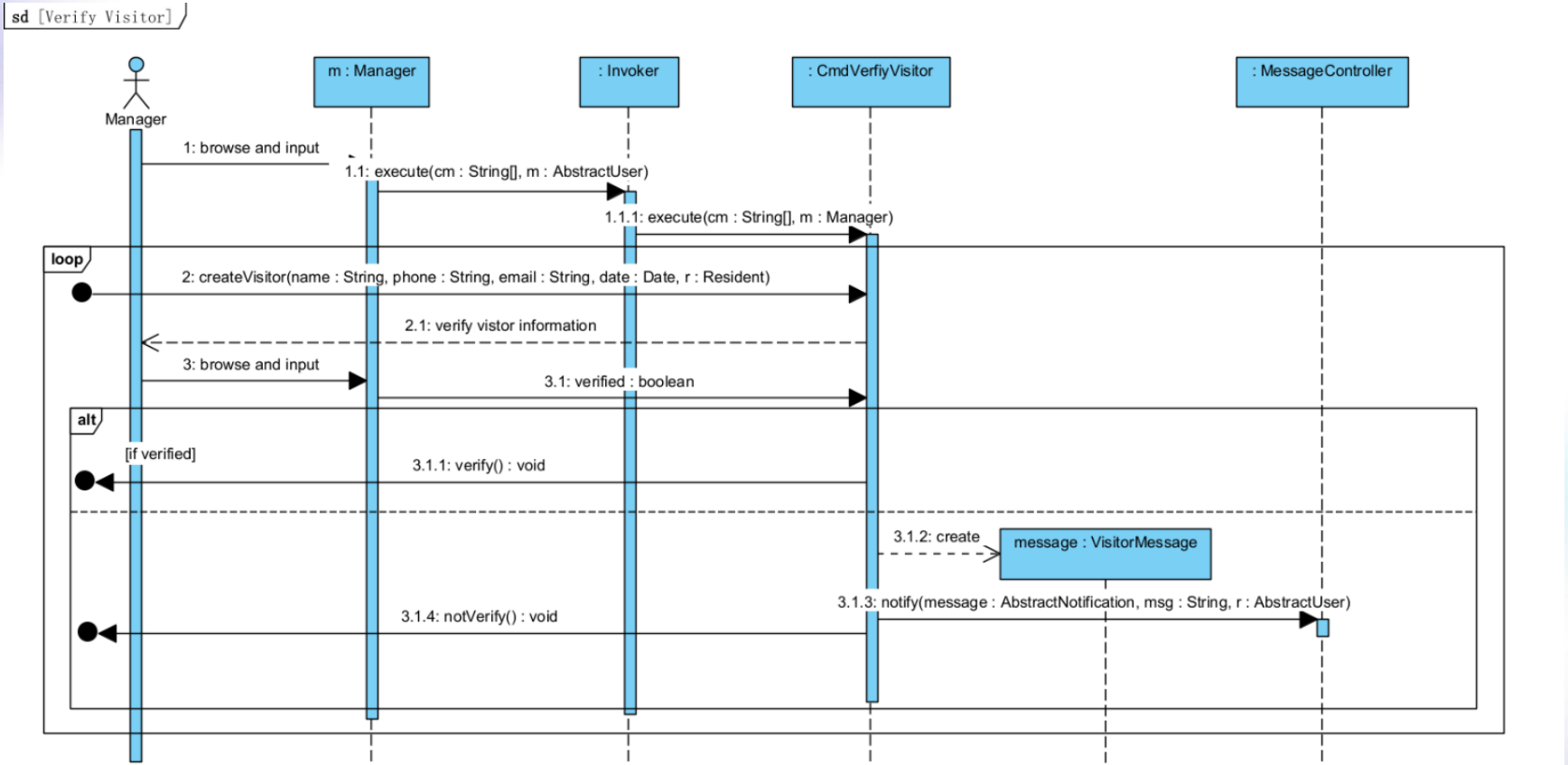
Account Module - Retrieve Password



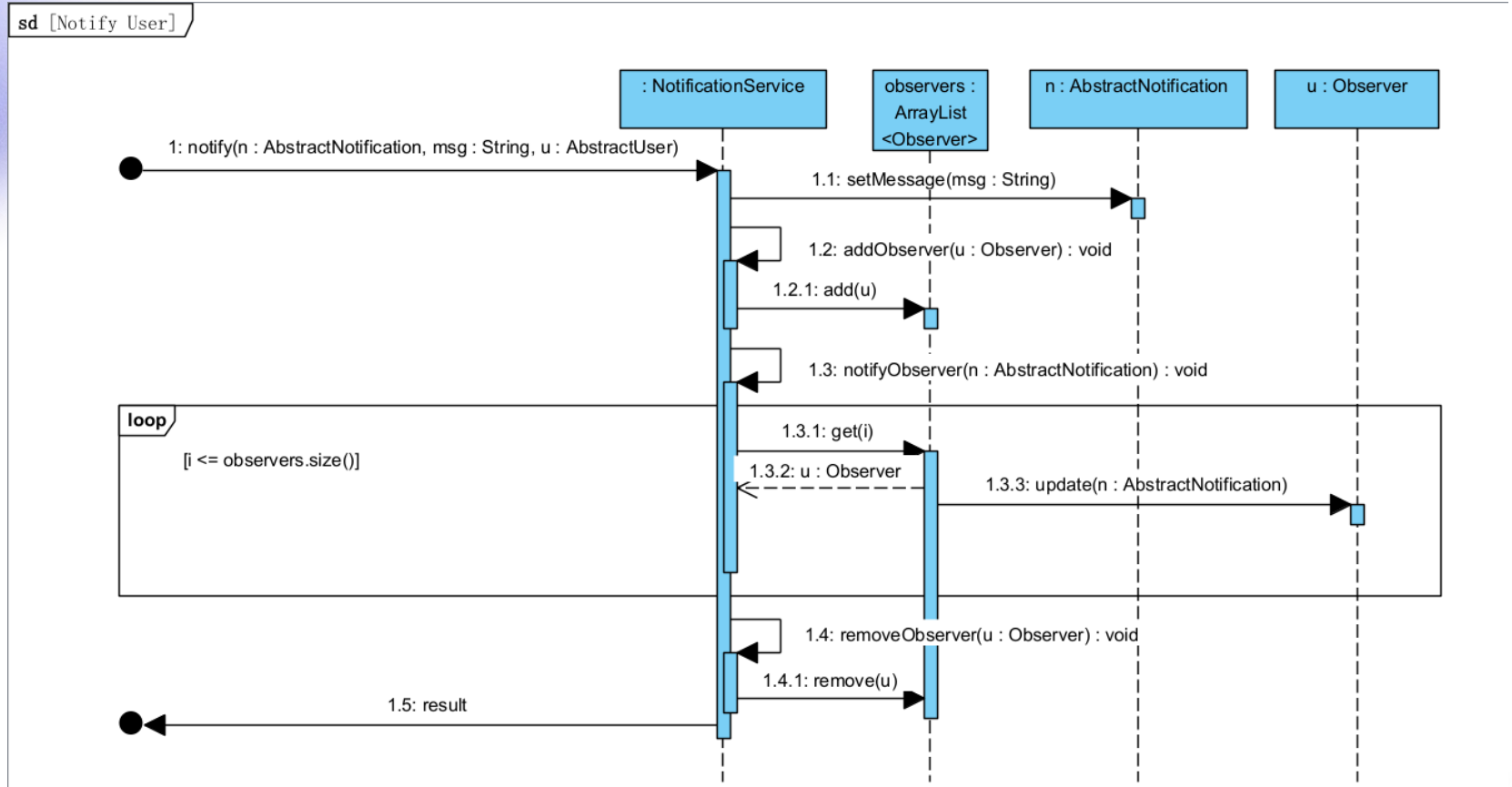
Visitor Module - Register Visitor



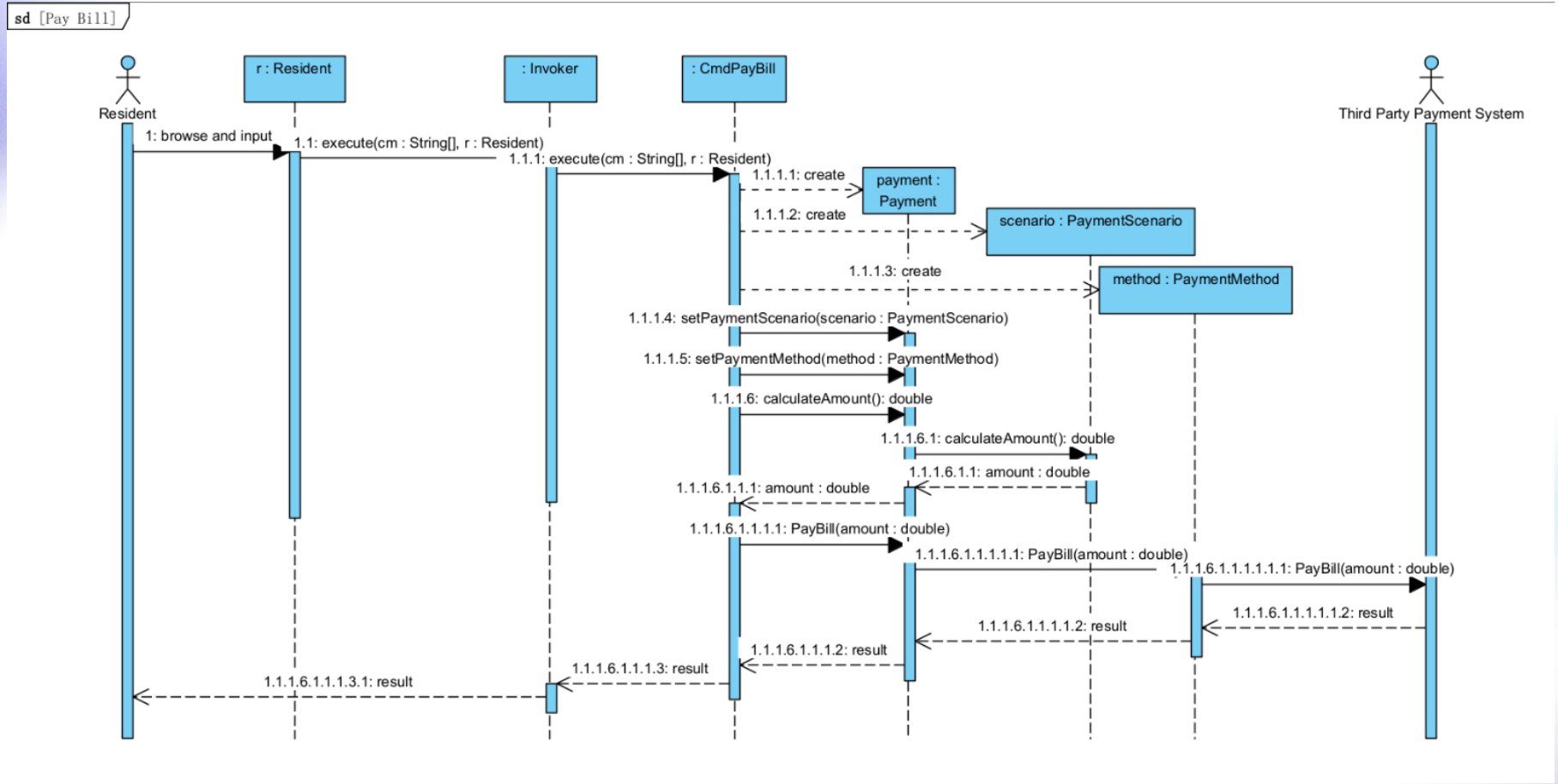
Visitor Module - Verify Visitor



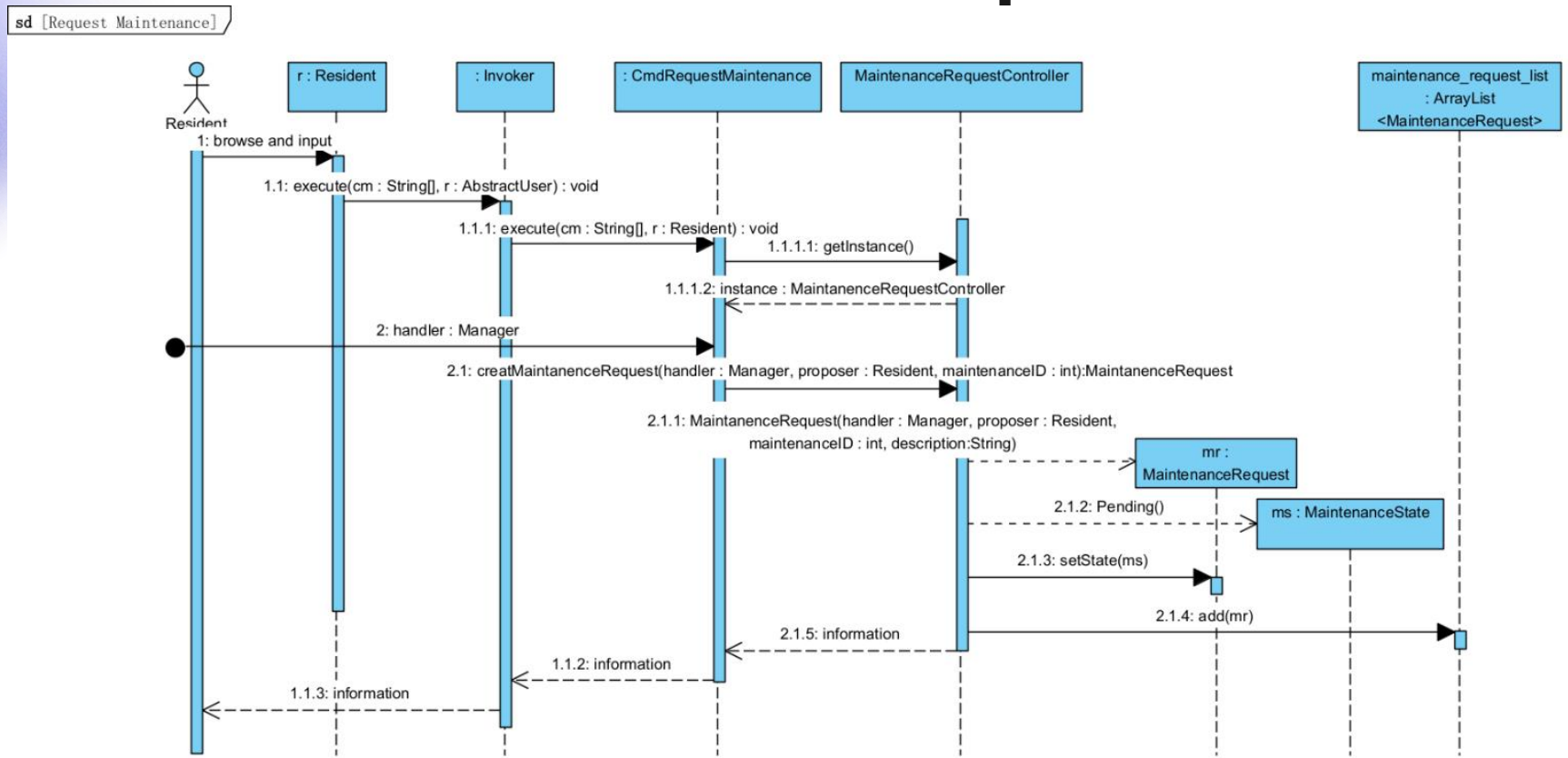
Notification Module



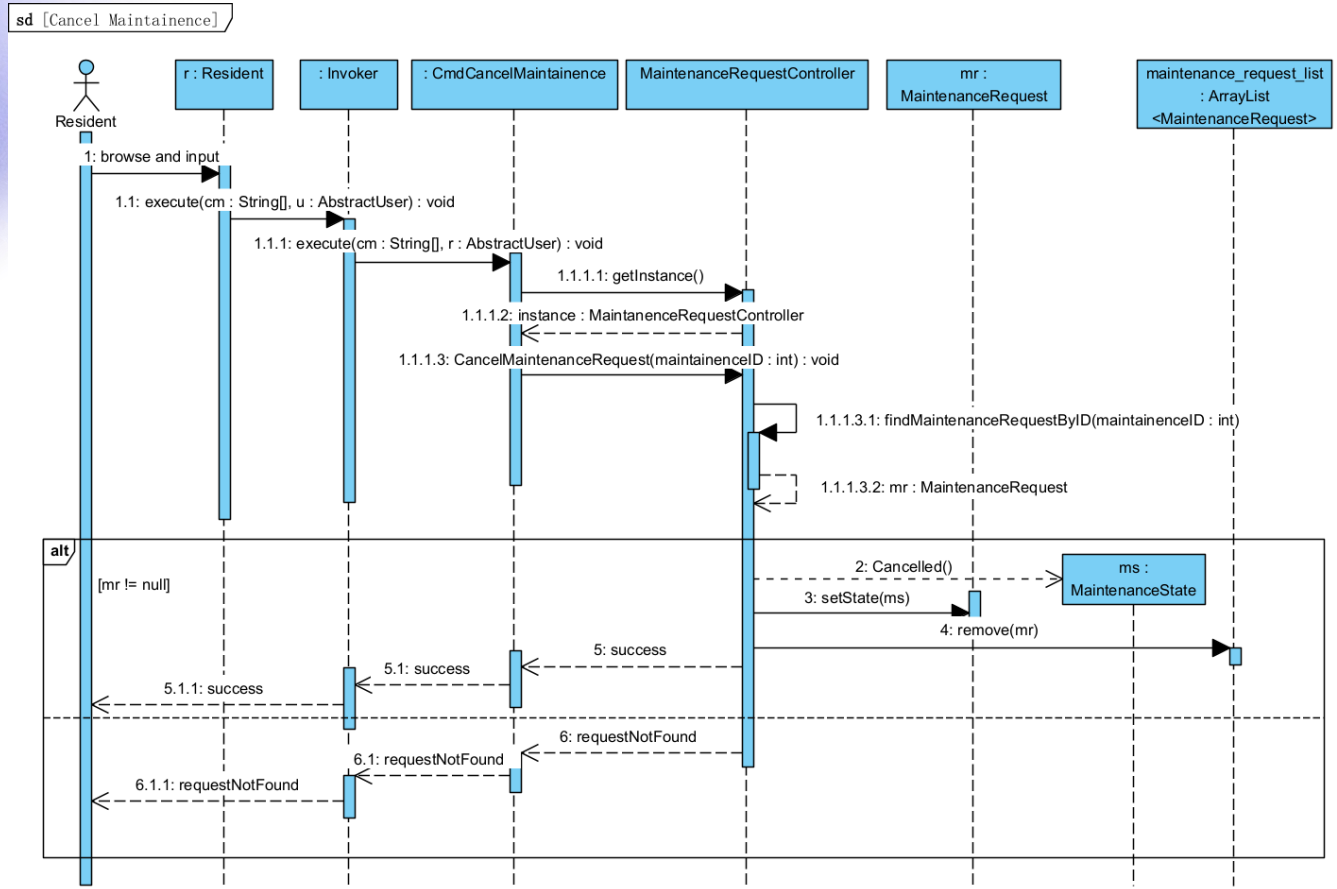
Payment Module - Pay Bill



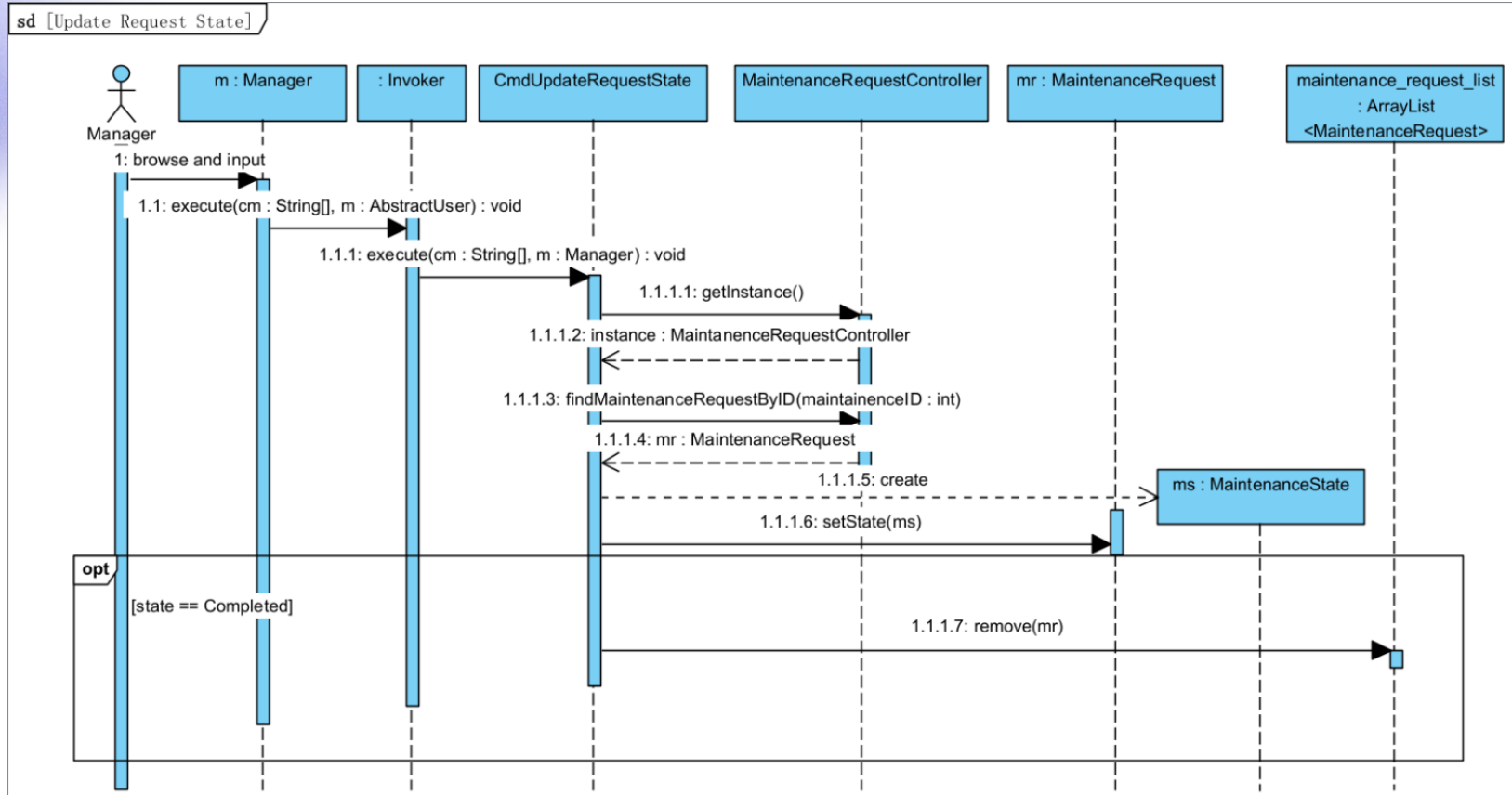
Maintenance Module - Request Maintenance



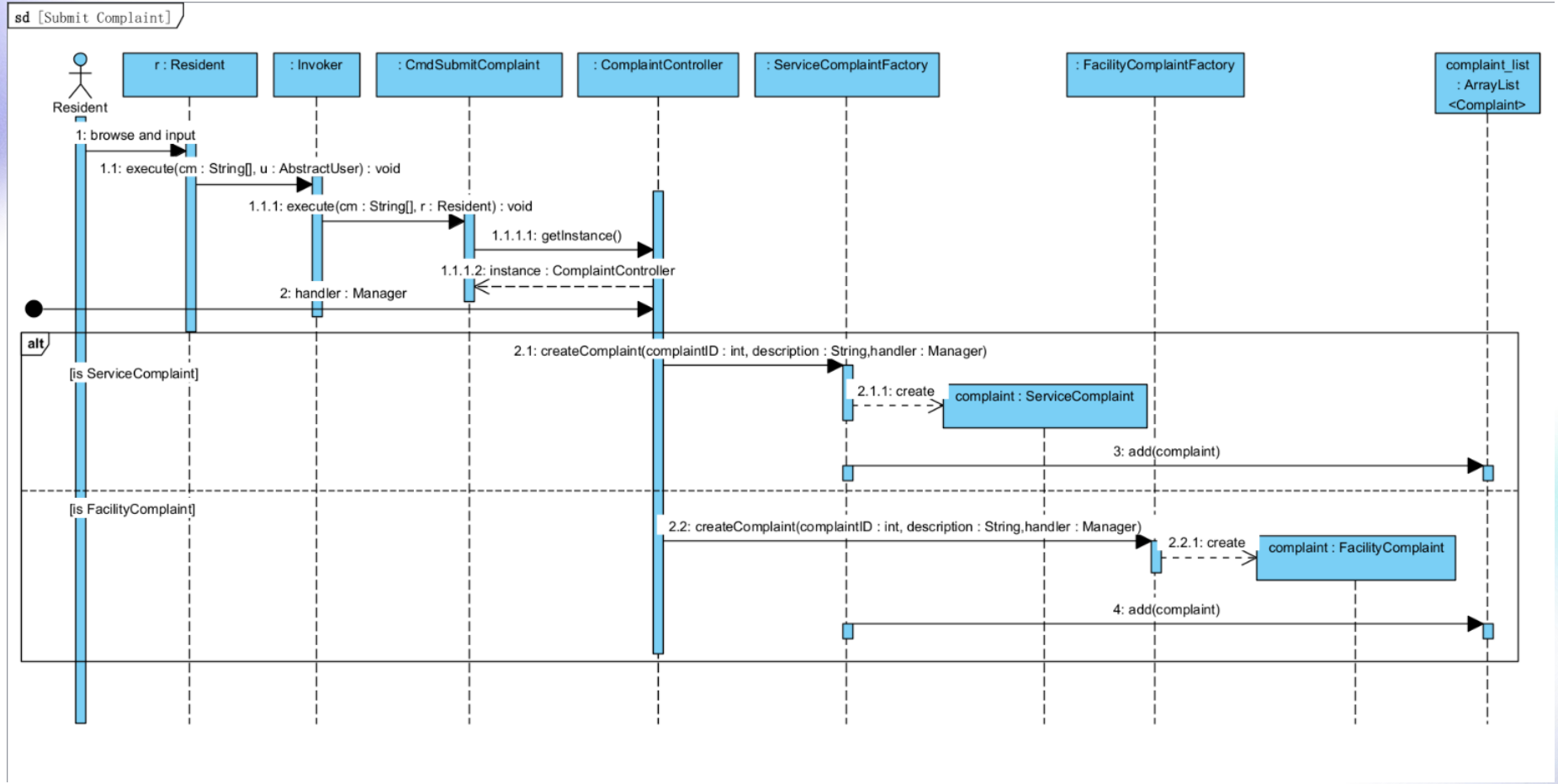
Maintenance Module - Cancel Maintenance



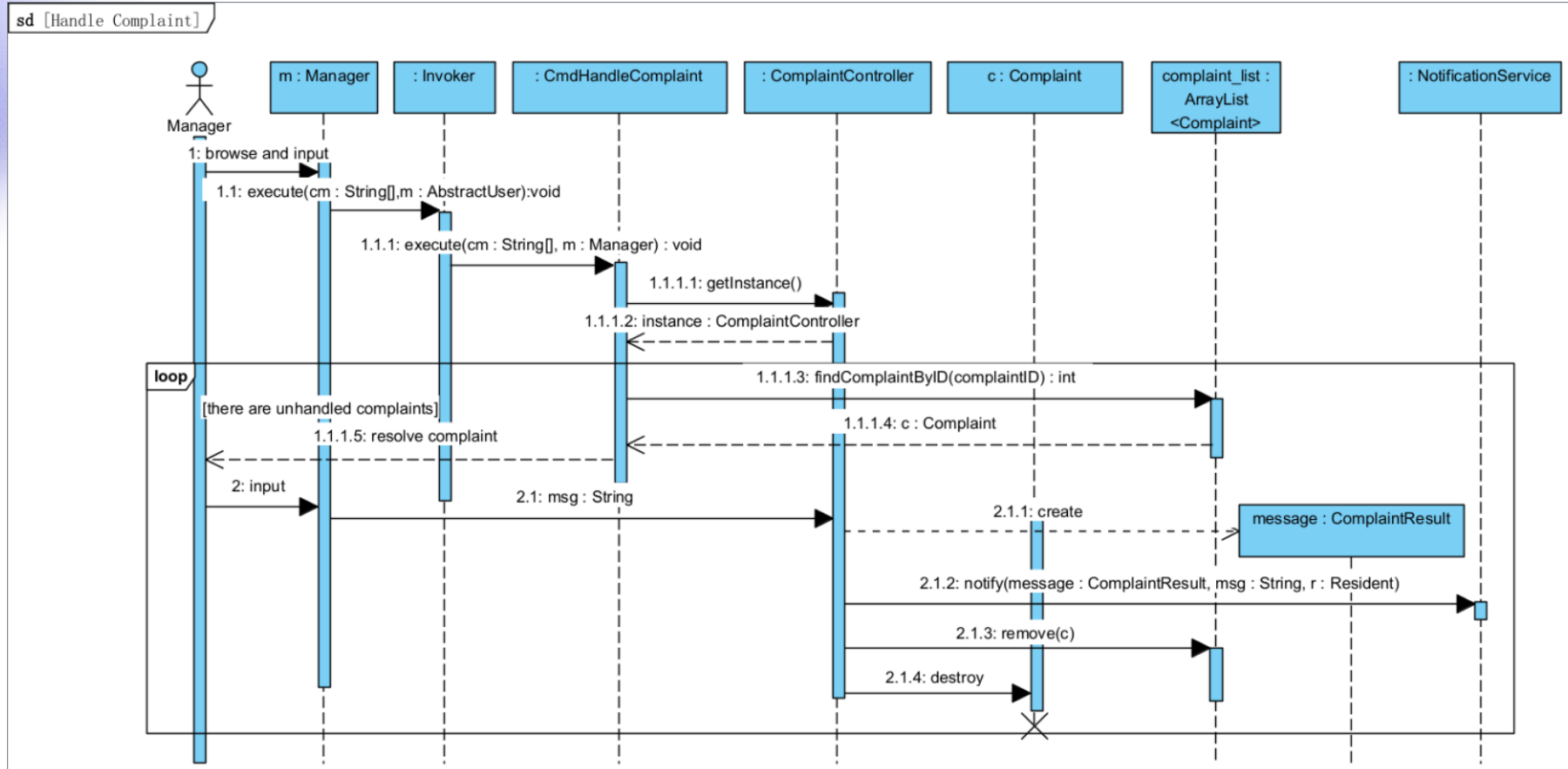
Maintenance Module - Update request state



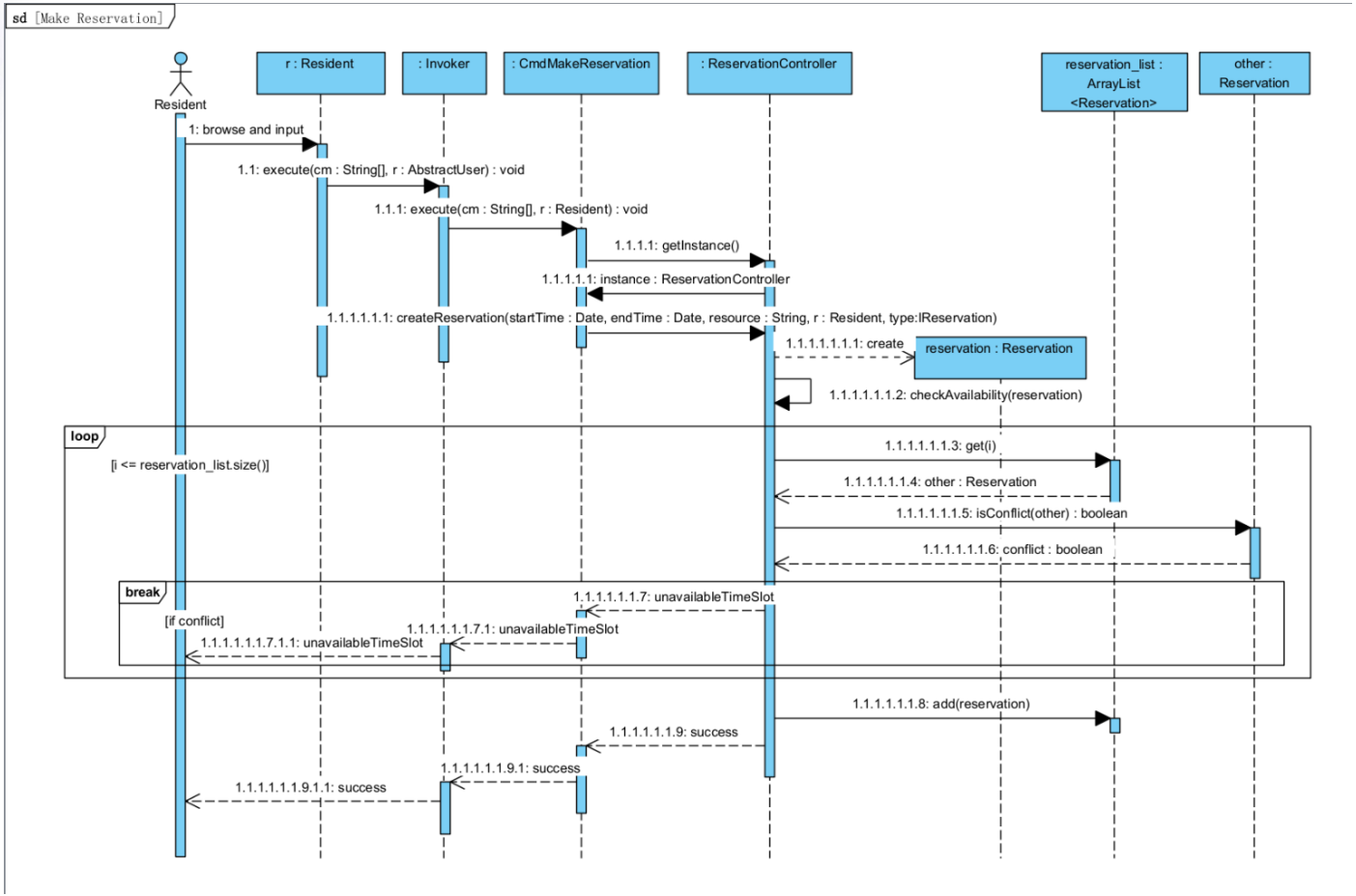
Complaint Module - Submit Complaint



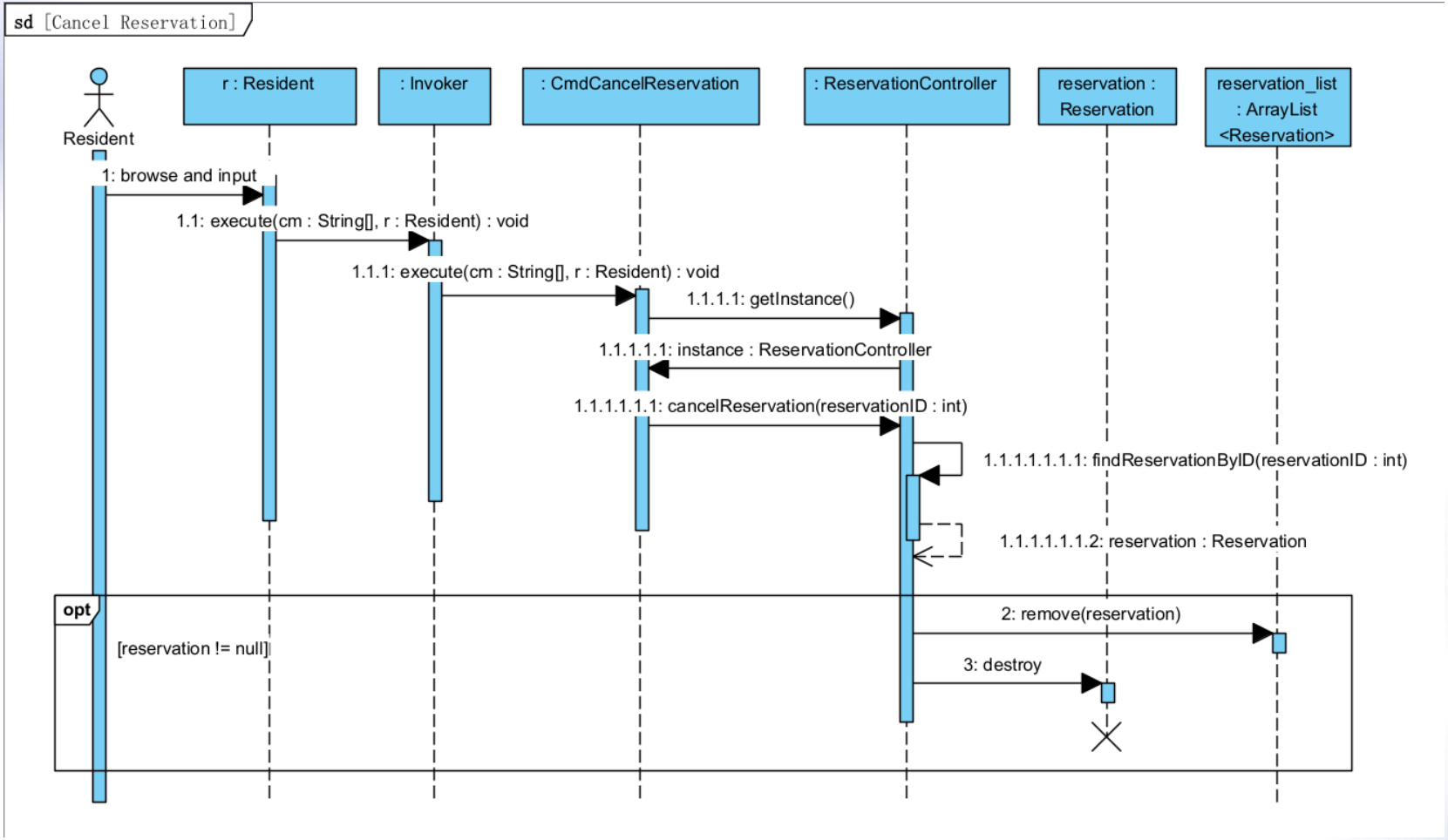
Complaint Module - Handle Complaint



Reservation Module - Make reservation



Reservation Module - Cancel reservation



Team Work

Challenges:

- ➔ Recognizing the market requirements for our application, as there are a lot of competitors
- ➔ Editing files without notice since everyone has new ideas popping up

Solutions:

- ➔ Lots and lots of discussion and sharing of our opinions
- ➔ We figure out version control on visual paradigm to ensure everyone’s on the same page.

Team Member	Post	Responsibilities
GAO Nanjie	Project Manager	● Requirement Analysis ● Sequence Diagram
WANG Ying	Assistant Project Manager	● Requirement Assembly ● Use Case Diagram
WAN Zimeng	Designer	● Background Research ● Use Case Diagram
WANG Fan	Designer	● Requirement Analysis ● Sequence Diagram
CHEN Yihuan	Designer	● Prototype Visualization ● Class Diagram
LIU Hengche	Designer	● Requirement Analysis ● Class Diagram

Contribution & Weekly

Name (Last, First)	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Total
LIU HENGCHE	1	0	2	1	3	1	5	2	2	1	2	3	2	25
WAN ZIMENG	1	2	1	1	3	5	1	0	2	1	2	2	4	25
WANG FAN	1	2	2	3	1	1	2	3	2	0	3	1	4	25
CHEN YIHUAN	1	2	1	2	0	1	3	3	2	1	2	3	4	25
GAO NANJIE	1	0	1	1	1	5	3	3	1	2	1	3	3	25
WANG YING	1	1	2	1	4	2	5	1	1	2	2	2	1	25
Total Effort	6	7	9	9	12	15	19	12	10	7	12	14	18	150

Student	Relative	Contribution	e.g. Group Mark	Final Adjusted
LIU HENGCHE	1.00	100.00%	80.00	80.00
WAN ZIMENG	1.00	100.00%	80.00	80.00
WANG FAN	1.00	100.00%	80.00	80.00
CHEN YIHUAN	1.00	100.00%	80.00	80.00
GAO NANJIE	1.00	100.00%	80.00	80.00
WANG YING	1.00	100.00%	80.00	80.00
Max	1.00		Average	80.00

Week	Weekly Activity Log	Completed By
Week 1	Recruit Project Memebers to Form a Team.	All
Week 2	Select the topic	All
Week 3	Confirm the desirable core functions	All
Week 4	Confirm other functions to enlarge the function scale (add increments)	All
Week 5	Use case diagram + discussion	Wan Zimeng, Wang Ying + All
Week 6	Use case description	Wang Ying, Wan Zimeng
Week 7	Class diagram design + discussion	LIU Hengche,Chen Yihuan + All
Week 8	Sequence diagram design + discussion	Wang Fan, Gao Nanjie + All
Week 9	Midterm preparation	/
Week 10	User Interface making (mobile+dashboard)	Chen Yihuan, Gao Nanjie
Week 11	Writing Report	All
Week 12	Powerpoint making	All
Week 13	Formatting, preparing for the presentation	All

Overall reflection

Things that we learnt…

- Teamwork and communication to achieve our mutual goal
- Design Principles and Design Patterns to enhance the scalability of our application
- The workflow of designing software

Things to be done…

- Adequate time management and proper workload for everyone
- Effective communication

Conclusion

- Our application is aimed at facilitating managers to **better control their community services** and for residents to better **engage and participate in community activities**.
- We have demonstrated our design in the form of **showing prototype**, use case diagrams, sequence diagrams and class diagrams
- While we acknowledge that our application is not without its flaws, and there are potential risks and hazards that may only surface upon market release, we are confident in our foundation and ready to embrace the challenges ahead.

Thanks!

Q & A