LaSalle College
Montréal

420-A13-AS C2_DESIGN PATTERNS section 07728

Assignment 5 - Artifacts - Final Version

Professor:	**Salar Nasr Azadani**	
Students:	Yashrith Chittoor Hari Krishna Felipe Andrés Cordero Osorio	2330655 2333047

Table of Content

Description....................3
Use Case Diagrams....................6
Use Case Diagram: Parking a car....................6
Use Case Diagram: Register User....................6
Domain Model....................7
System Sequence Diagrams....................8
System Sequence Diagram: Parking a car....................8
System Sequence Diagram: Register User....................9
System Operations....................10
Operation Contracts:....................10
Contract CO1: for read_plate()....................10
Contract CO2: create_profile()....................11
Contract CO3: add_payment_method()....................11
Sequence Diagrams:....................12
read_plate()....................12
create_profile()....................13
add_payment_(visitor)....................14
finish_process()....................15
Class Diagram:....................16

Description

The proposed parking management system is designed to efficiently manage parking spaces in different parking lots, ensure smooth entry and exit for drivers (end users), and differentiate between visitors and registered monthly clients.

The system will be composed of several elements, including a camera-based license plate recognition (LPR) system, a parking management software, and a user-friendly interface for both drivers and parking administrators. The LPR system will be installed at every entrance of the parking facility, because the total number of entrances and exits could be more than one at each parking lot.

At every entrance there will also be a barrier system and touchscreen system, which will allow or deny the access to the drivers. As a vehicle approaches the entrance, the LPR system will capture an image of the license plate and use optical character recognition (OCR) algorithms or an equivalent approach to extract the alphanumeric characters. The LPR system will then transmit this information to the parking management software for further processing.

The parking management software will be the core of the system, responsible for managing parking records, processing LPR data, and controlling the entry and exit barriers. The software will be designed with scalability and reliability in mind, ensuring that it can handle a large number of parking records and LPR requests without any significant performance problem.

The parking management software will maintain a database of parking records, including the license plate number, entry and exit times, and customer or driver type (visitor or registered monthly client). The software will also have a real-time watchlist of license plates that are currently inside the parking facility and also will have a record of each parking place and the car that is using each. The system will also track, in real-time, the availability at each parking site, and will provide real-time guidance to the drivers to reach the designated space in the parking lot. This information to the drivers will be provided by another online smartphone application served by the system.

When the LPR system transmits the recorded information to the parking management software, the software will first check the watchlist to see if the license plate is already inside the parking facility. If the license plate is found in the watchlist, it means that the LPR system has made a mistake, and the software will deny the entry request. In these cases, the system will double check the license plate, to ensure it was interpreted correctly.

If the license plate is not found in the watchlist, the software will check the parking records database to see if the license plate belongs to a registered monthly client. If a match is found, the software will add the license plate to the watchlist, mark it as a registered monthly client in the parking record, and allow the vehicle to enter the parking facility.

If the license plate is not found in the parking records database, it means that the vehicle belongs to a visitor. The software will then create a new parking record for the visitor, add the license plate to the watchlist, and allow the vehicle to enter the parking facility. The visitor will be required to pay the parking fee before exiting the parking facility. The system will provide different options to the driver, permitting to pay for minutes, hours, full day, and also for a week, using a temporarily created account.

As it was stated before, the parking management system will have a user-friendly interface for both drivers and parking administrators. Drivers will be able to view the availability of parking spaces, pay the parking fee (if applicable), and request assistance from parking attendants.

Parking attendants will be able to view the current status of the parking facility, manage parking records, and control the entry and exit barriers with manual or electronic operations if the system has any failure or bad behavior.

The drivers mobile application also notifies drivers with real-time notifications, in case the parking time is about to expire. In this case, the drivers will be allowed to make an additional payment using their smartphones.

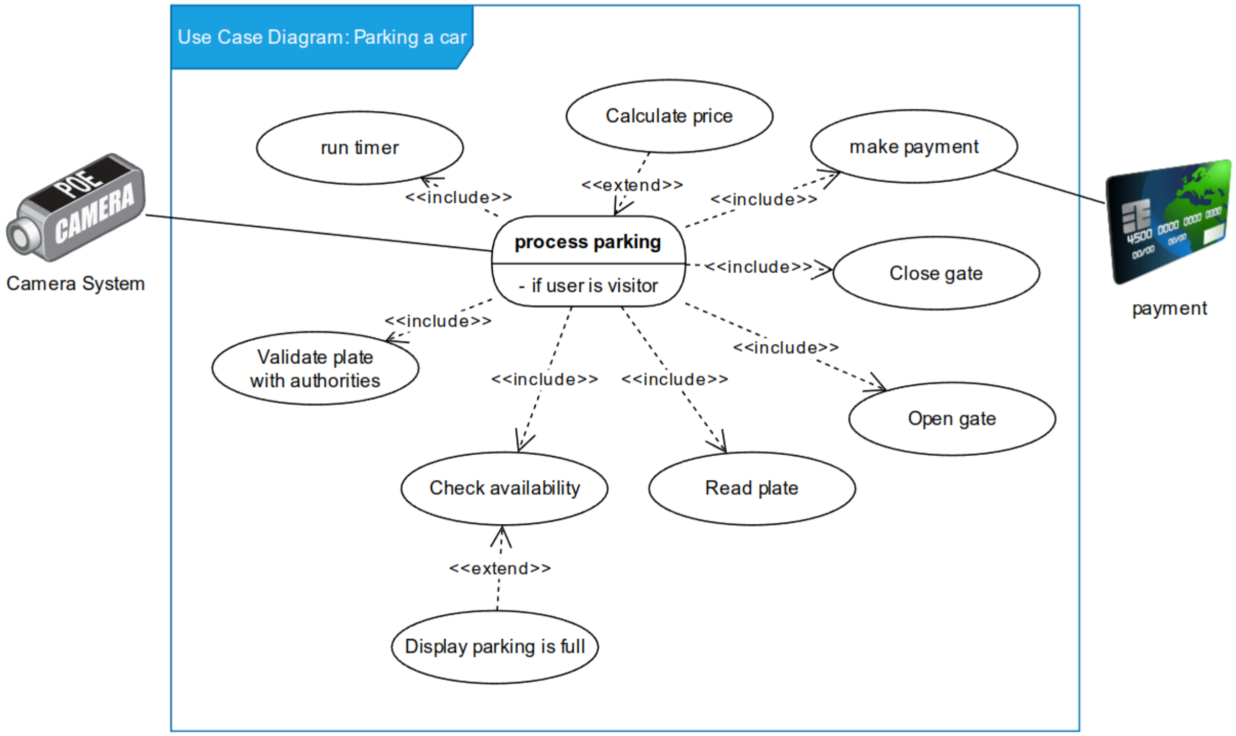
The parking management software will be also capable of operating in case there is no internet connection available. In such cases, the system will operate with an internal and local database, allowing drivers to use parking lots also. When the system gets online again, the internal database will be committed and synchronized against the online one, keeping everything up to date.

The parking management system will allow drivers to book spaces in advance via the mobile app. These enhancements will optimize revenue and ensure parking availability, improving the overall end user experience.

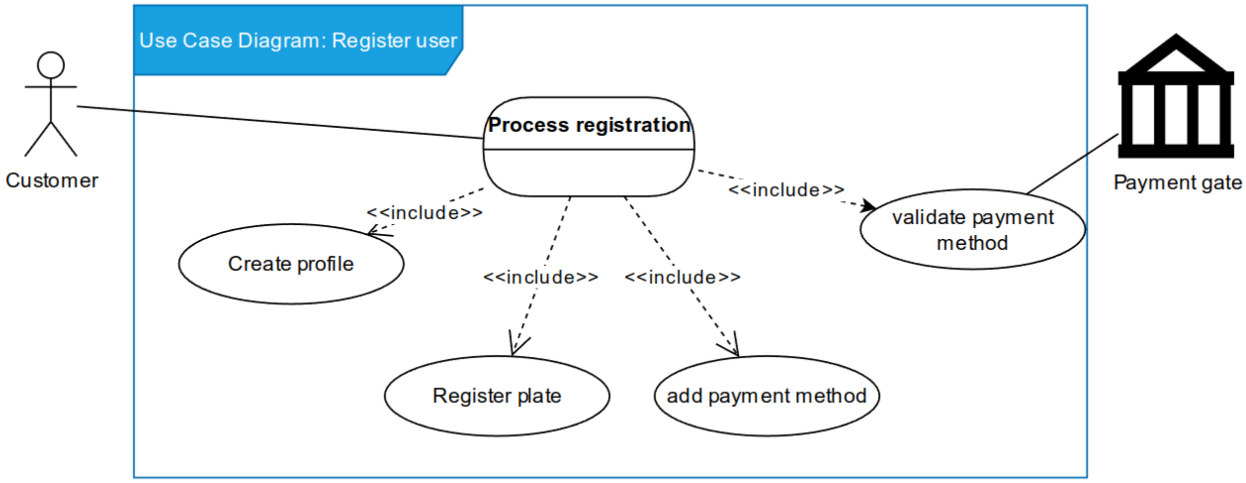
In conclusion, the proposed parking management system will provide an efficient and reliable solution for managing parking lots, differentiating between visitors and registered monthly clients, and ensuring a smooth and hassle-free parking experience for end users and stakeholders.

Use Case Diagrams

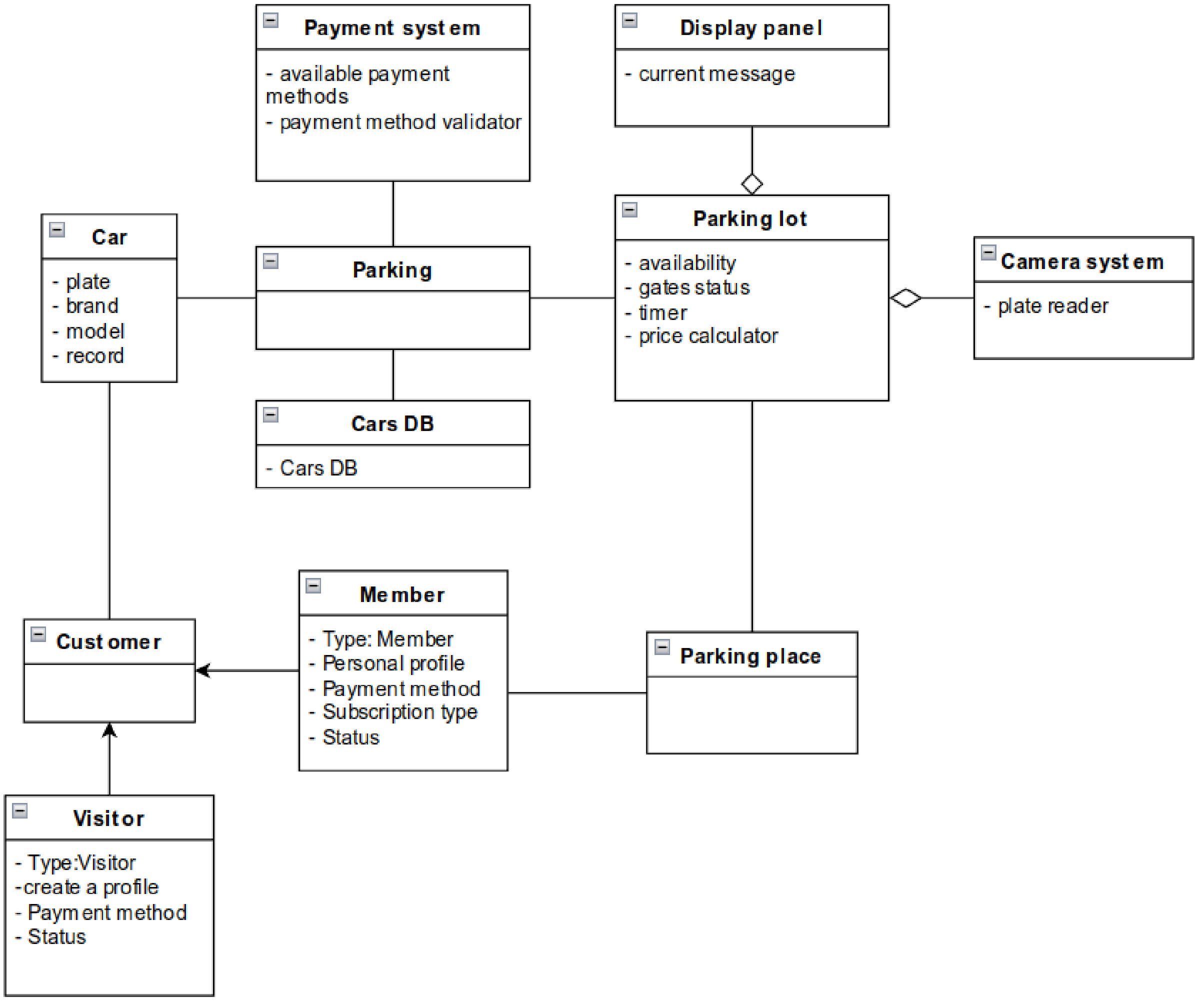
Use Case Diagram: Parking a car



Use Case Diagram: Register User

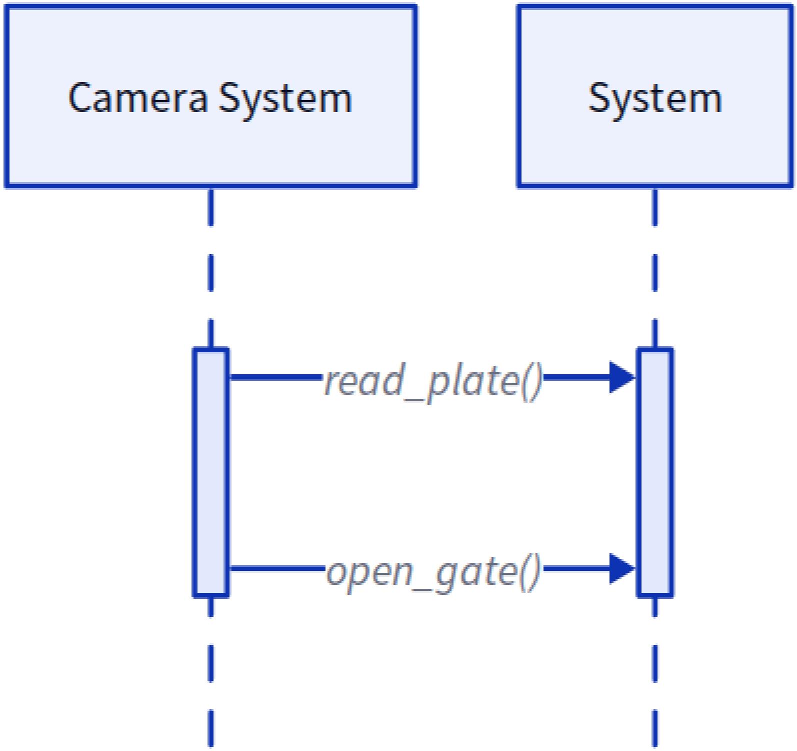


Domain Model

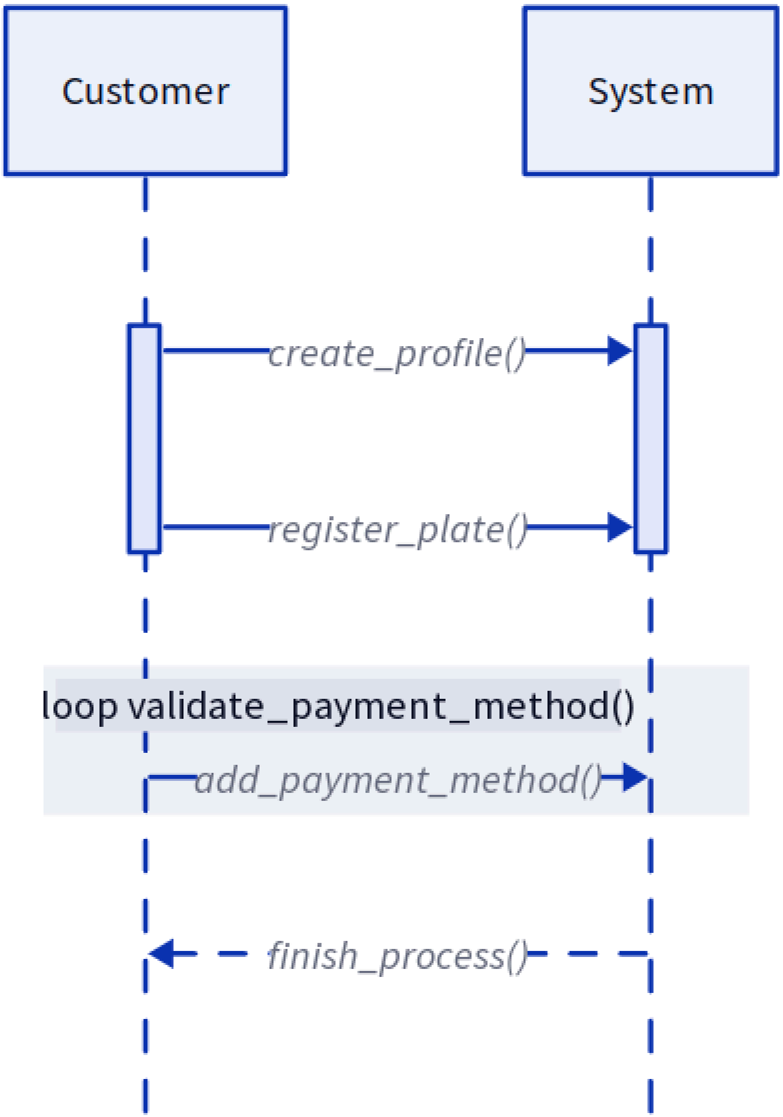


System Sequence Diagrams

System Sequence Diagram: Parking a car



System Sequence Diagram: Register User



System Operations

System

read_plate()
open_gate()
create_profile()
add_payment_method()
validate_payment_method()
finish_process()

Operation Contracts:

Contract CO1: for read_plate()

Operation:	read_plate()
Preconditions:	There is a car to be parked
Postconditions:	● Instances Creation: ○ A Car instance *car* ○ A User instance user ○ A Transaction instance trans

Contract CO2: create_profile()

Operation	create_profile():
Precondition	- The user accesses the registration interface. - All required fields (e.g., name, email, license plate number) are filled out by the user. - The parking management software is ready to process new user data
Postcondition	- A User instance (user) is created with the provided personal information. - The User instance attributes such as name, email, license plate number, etc., are set to the values provided by the user. - The User instance is stored in the user database. - The new user profile is associated with any additional entities as necessary (e.g., default payment method).

Contract CO3: add_payment_method()

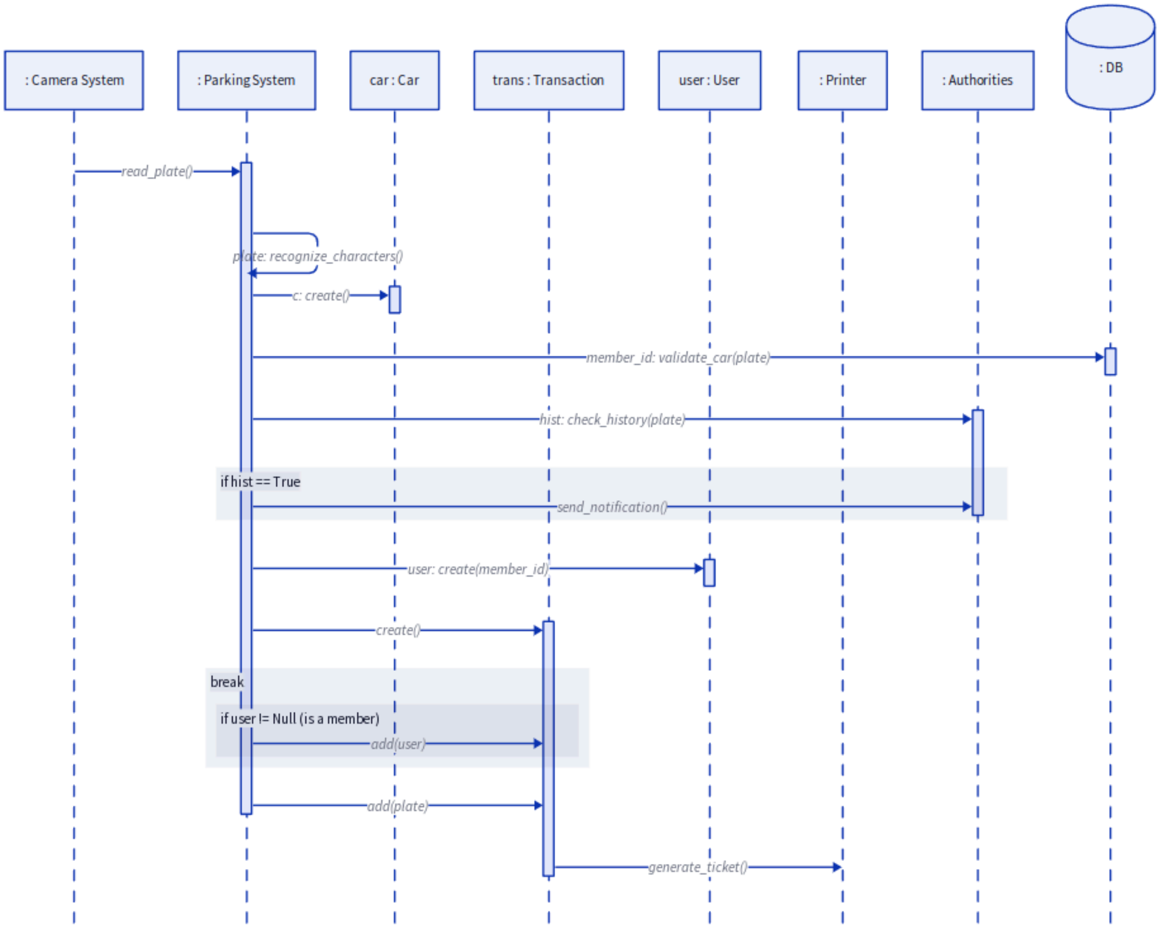
Operation	add_payment_method():
Preconditions	- User has confirmed payment. - Parking management software is ready to process payment.
Postconditions	- Payment instance (payment) is created - Payment status updated - Payment recorded in the transaction

Contract CO4: finish_process()

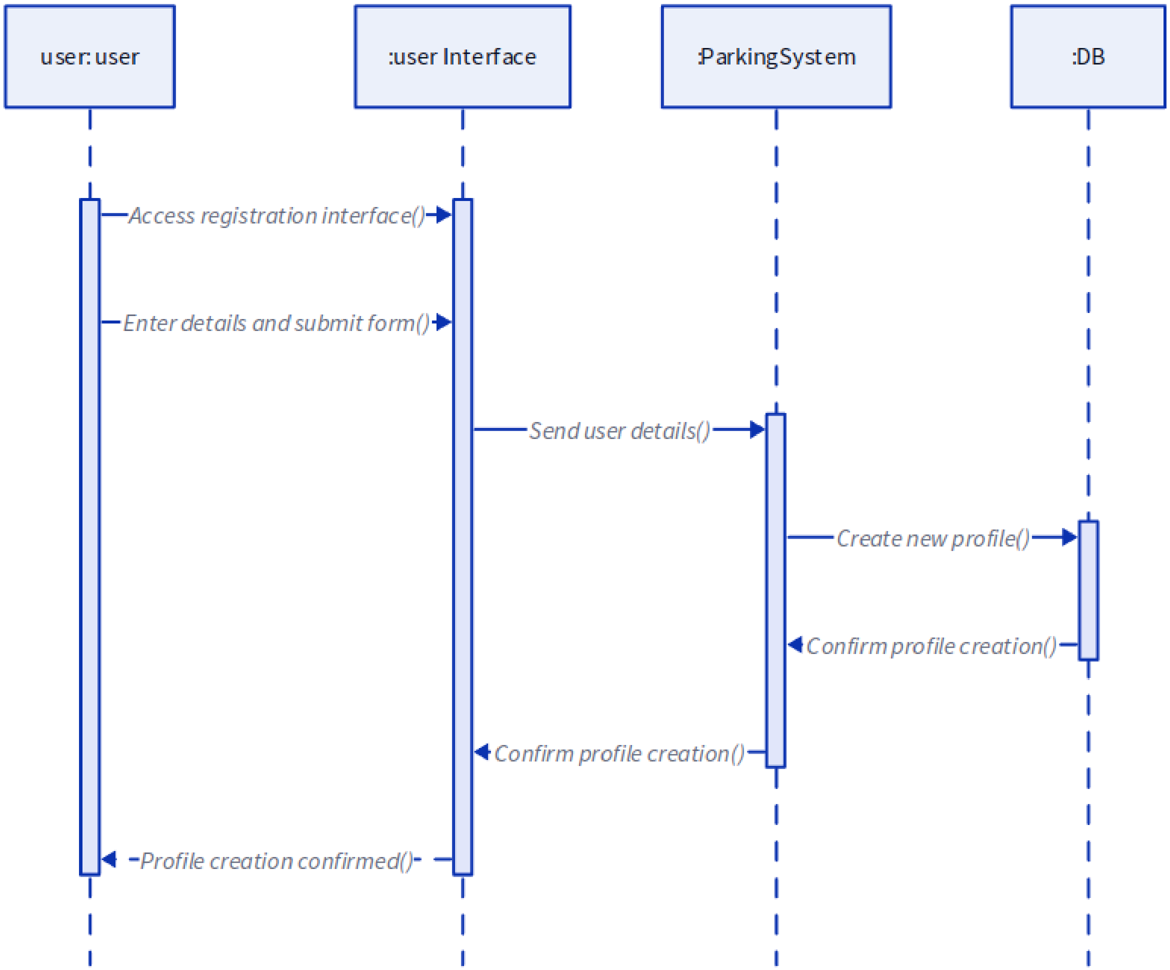
Operation	finish_process()
Preconditions	- The user has completed the parking session and is ready to exit. - The user has a valid payment method on file or is ready to make a payment. - The parking management software, payment gateway, and exit barrier systems are operational.
Postconditions	- The parking fee is successfully processed. - The parking session is marked as completed in the parking records database. - The exit barrier is opened to allow the user to exit. - A confirmation message is sent back to the user interface.

Sequence Diagrams:

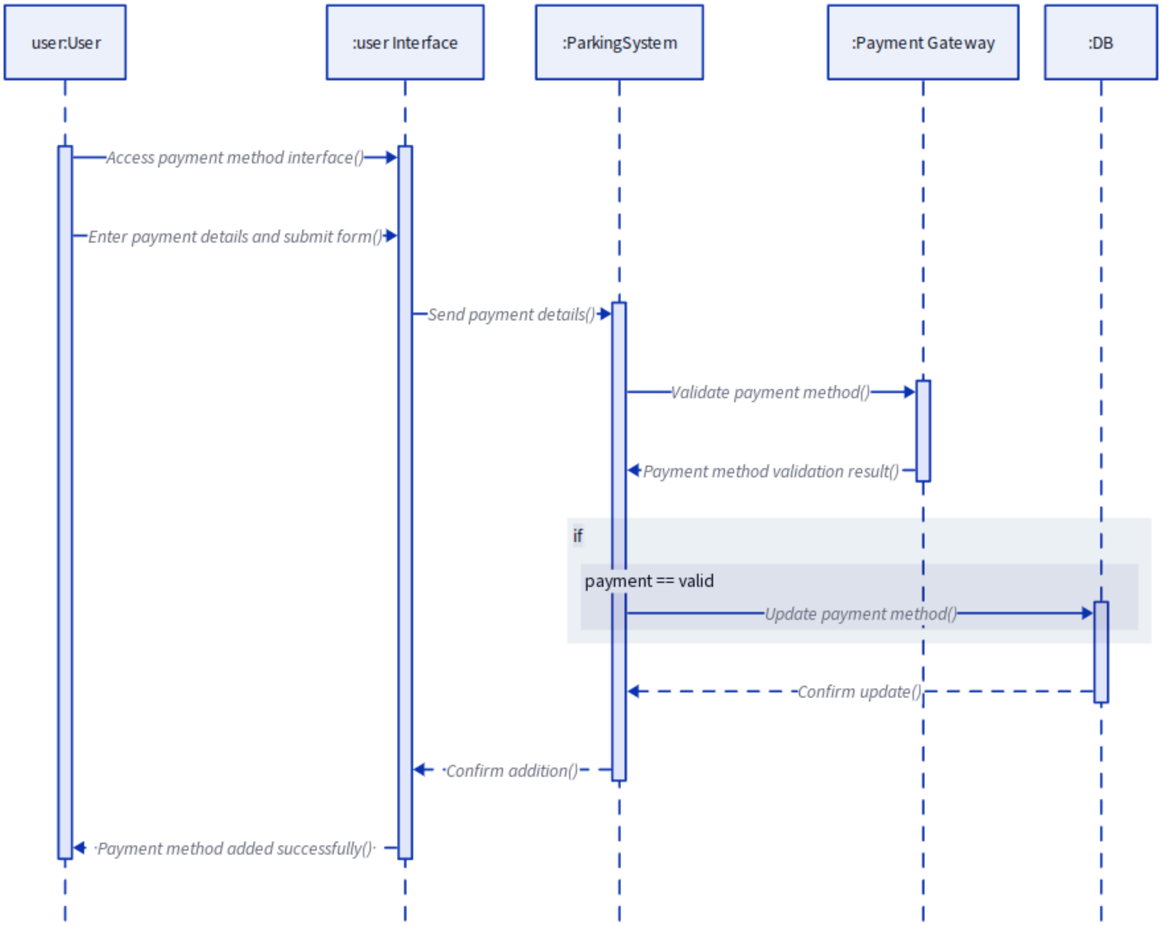
read_plate()



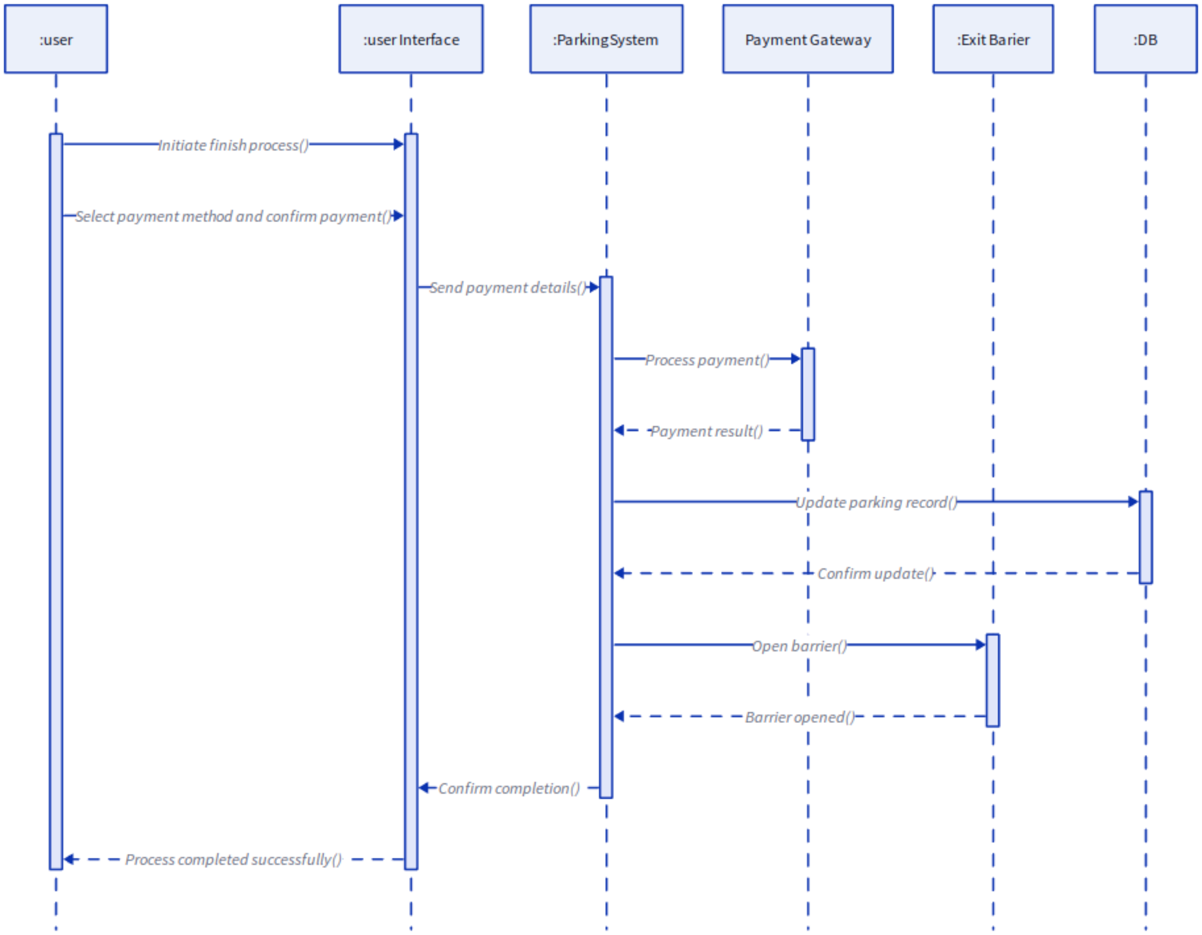
create_profile()


user: user
:user Interface
:ParkingSystem
:DB
Access registration interface()
Enter details and submit form()
Send user details()
Create new profile()
Confirm profile creation()
Confirm profile creation()
Profile creation confirmed()

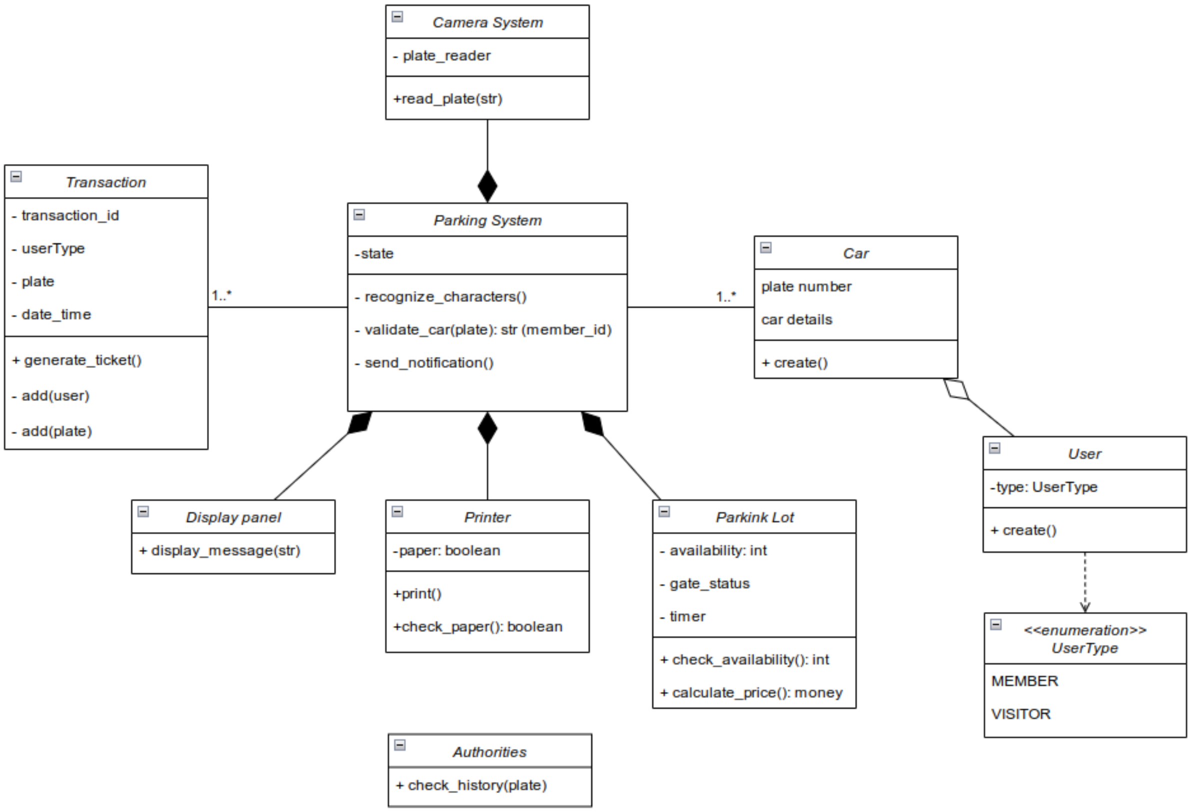
add_payment_(visitor)


user:User
:user Interface
:ParkingSystem
:Payment Gateway
:DB
Access payment method interface()
Enter payment details and submit form()
Send payment details()
Validate payment method()
Payment method validation result()
if
payment == valid
Update payment method()
Confirm update()
Confirm addition()
Payment method added successfully()

finish_process()



Class Diagram:


Camera System
- plate_reader
+read_plate(str)
Transaction
- transaction_id
- userType
- plate
- date_time
+ generate_ticket()
- add(user)
- add(plate)
1..*
Parking System
-state
- recognize_characters()
- validate_car(plate): str (member_id)
- send_notification()
1..*
Car
plate number
car details
+ create()
User
-type: UserType
+ create()
<<enumeration>>
UserType
MEMBER
VISITOR
Display panel
+ display_message(str)
Printer
-paper: boolean
+print()
+check_paper(): boolean
Parkink Lot
- availability: int
- gate_status
- timer
+ check_availability(): int
+ calculate_price(): money
Authorities
+ check_history(plate)