INFS 2044 Assignment 1

Group Number: 25234

Student name	Student identifiers	Email
Weizhi Chen	Chewy144	chewy144@mymail.unisa.edu.au
Gitae Bae	Baegy002	baegy002@mymail.unisa.edu.au
Daniel Cowdrey	Cowdb001	cowdb001@mymail.unisa.edu.au
Turki Alsolbi	Alsty005	alsty005@mymail.unisa.edu.au

Volatility list

From what we understood the volatilities can be summarized and listed as follows:

Storage type Volatilities

- Customer Data Storage
- Booking Data Storage
- Employee Data Storage
- Issue Data Storage
- Feed Data Storage

Access type Volatilities

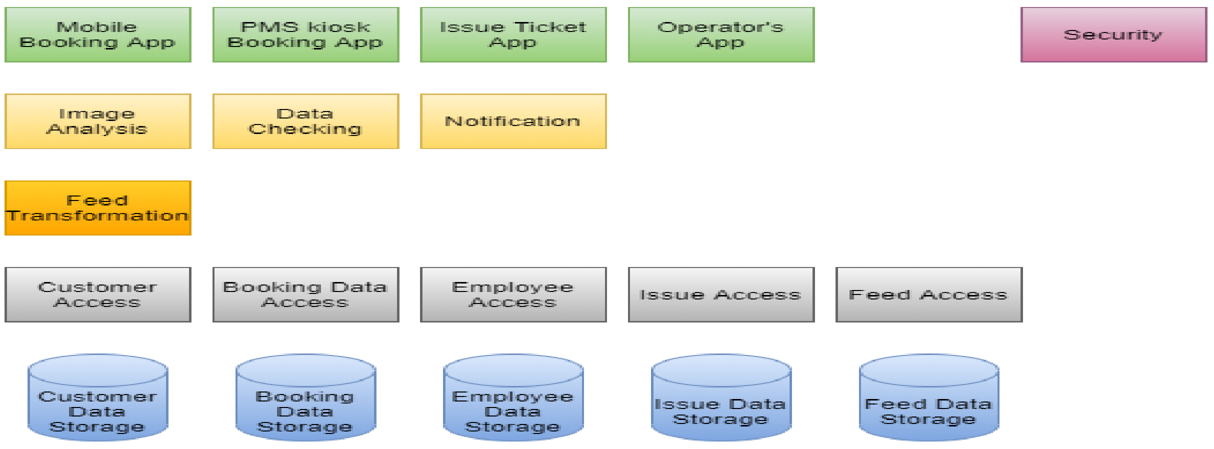
- Customer Access
- Booking Access
- Employee Access
- Issue Access
- Feed Access

Other type of Volatilities

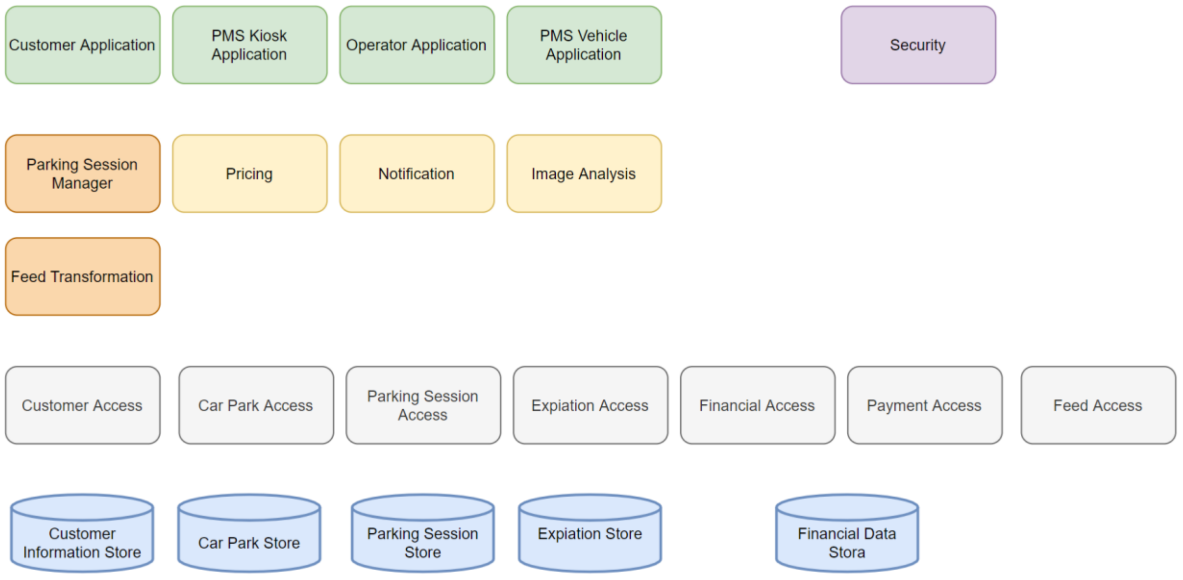
- Mobile Booking app
- PMS kiosk booking app
- Notification
- Image Analysis
- Data Checking

- Issue Ticket App
- Feed Transformation
- View Statistics & Explanation App
- Security

The diagram form of the Volatility list is as follows:



Component Decomposition



Shown above is the component decomposition for the list of volatilities identified earlier. The structure of the design is categorized based on the type of component in each of their respective layers. Components responsible for the interaction with the system are in the topmost client layer. These components will be the ones that the user authorized to use them will interact with the system based on the use cases outlined for it.

The Business logic layer below will undertake the processes requested by the user while using the client layer component. These include processes such as calculating the cost of the session when the customer requests to end their parking session, sending an email or text confirming that a parking session has started and ended or analyzing imagery taken by the cameras on the PMS vehicles as they routinely drive around the car parks.

The Resource Access layer provides access to the resources below it and is responsible for allowing data to be access, modified, or added. Compartmenting the access allow for better re-usability as well as general security, allowing only components (and by extension, users) access to resources that are required to fulfill their responsibility as a component.

The Resource layer contains all the databases thar are responsible for containing all the data that the client and business login components require. They will store data based on the performance, scalability, and business requirements that the owner of the system has.

Below is a further justification and explanation as to the design of the system.

Component Responsibilities

Component	**Responsibilities**
Customer and PMS Kiosk Applications	Present customer user interfaces
Operator Application	Present operator user interface and present live feed
PMS Vehicle Application	Present PMS Vehicle Driver interface
Security	User authentication and authorization
Parking Session Manager	Allow customers to perform booking use cases
Pricing	Calculate costs of sessions and expiations
Notification	Text/Email booking confirmation and expiations
Image Analysis	Process images of number plates and convert to a string to verify current session, identify vehicles and store information
Feed Transformation	Process data into live dashboard for operators

Customer Access	Access customer information
Car Park Access	Retrieve accurate car park locations and information
Parking Session Access	Access information about customer parking sessions
Expiation Access	Access information about customer expiations and any disputes against them
Financial Access	Access session transaction details
Payment Access	Access verification information about payment
Feed Access	Access information for the operator’s live dashboard

Above is a brief description on the client, business logic and access components within the proposed system. Each component has been designed so that it is able to, collectively with other related components, completely resolve a use case set during the settings of the system requirements.

The Customer and PMS Kiosk applications are to provide a user interface to the customers of the PMS system. These interfaces provide both a convenient and highly available option for those who are wishing to initiate a parking session on campus. The mobile application provides the ability for customers to be able to pay for their parking without having to find a parking meter, and the Kiosks located at each parking facility will provide a way for other customers who may not wish to use the mobile application or will want to use physical payment methods, to pay for their parking in this manner.

The Operator application presents a user interface for those who will need to monitor the PMS system’s operation. This includes a live feed in the view of a dashboard, including the statistics about the parked vehicles, any expiation notices that have been handed out and the locations of any PMS active vehicles. This information is updated at least every 5 minutes and allows an operator to quickly intervene if any problems with the system are identified to maintain high availability and verification of potential errors. These dashboards will also allow the operator to filter the information provided to them by the types of information and by the parking location that they wish to reference.

Active PMS Vehicles monitoring the parking locations using the camera mounter to the top of the car will require an application for them to fulfil their functional requirements. The application will allow the camera to take pictures of cars violating their policies and then present them to the driver of the vehicle. The driver of the PMS vehicle will then verify the information presented to him on the PMS Vehicle application to either mark as a false positive, or to accept the notification, lodge the expiation, print a notice off and submit the information to be stored.

The Security component will be responsible for making sure that users are verified and authorized to use the application they are attempting to access. Only operators will have access to the live dashboards and feeds for the parking information, only drivers of the PMS Vehicles will have access to the vehicles, and the on-board tablet application for expiations and only customers with correct login credentials will have access to the information associated to their account.

The Parking Session Manager will manage the components responsible for creating a booking/parking session. This will initiate pricing calculations, notifications to customers and send that information to the correct databases for storage and later reference when required.

The Pricing component will calculate pricing based on the pricing policy that has been set by the university. These policies can include charging amounts dependent on the date and time, or the user that is attempting to place a booking/pay for their parking session.

The Notification component will send emails or text messages based on the user preference when they create a booking or pay for their parking session. This will include information about the date and time of the session, the vehicle that was used during the session as well as the cost of the session along with the relevant payment information used to pay the cost of the session.

For the PMS Vehicles to check the status of a parked vehicles session and whether it is valid, the images will need to be processed into data that can be used to check this information. The Image Analysis component will be responsible for taking the images of the vehicles parked in an area and converted to a string, so that the PMS Vehicle application can query the Parking Session Storage for a current and valid parking session. This component will be able to process the image for this query to be run within 0.5 of a second, in order for all the parked vehicles to be accurately processed while the driver of the PMS vehicle is travelling less than 10 kilometers per hour.

The Feed Transformation component will handle the raw relevant data being sent to the live dashboard feed for the operators. The Operator may query a certain car park location and the Feed Transformation component will display the information requested by the operator in the Operator application using a query to the Parking Session and Expiation Storages through the Expiation Access and Parking Access components.

The Access components are responsible for allowing the other components to access the resources from which they need to either access information for further processing, or if they need to modify/add data based on requests from users through the application interface that they have interacted with.

Volatility Map

Volatility	Encapsulated In
Mobile Booking	Customer Application
PMS Kiosk Booking	PMS Kiosk Application
Notification	Notification
Image Analysis	Image Analysis
Data Checking	Parking Session Manager
Expiation Issue	PMS Vehicle Application and Car Park Access
Feed Transformation	Feed Transformation
View Statistics and Expiations	Operator Application
Security	Security
Customer Access	Customer Access
Booking Access	Parking Session Access
Expiation Access	Expiation Access
Payment Provider	Payment Access
Finance Access	Financial Access
Feed Access	Feed Access
Customer Data Storage	Customer Information Store
Booking Data Storage	Parking Session Store
Expiation Data Storage	Expiation Store
Financial Data Storage	Financial Data Storage

The volatilities outlined earlier in this document have been addressed in the design, as they have been encapsulated in one or more of the components of the system design. User interactions have been encapsulated in their respective applications, based on the function they are wishing to run and what they have been provided access to, access volatilities have been encapsulated within their own respective access components similarly to the storage volatilities.

- Users interacting with the mobile booking application will be encapsulated within the customer application component, managing all requests by customers, such as booking requests, payments, and expiation disputes.
- Users who use the Kiosk Application will be encapsulated by the PMS Kiosk application, allowing for similar interactions to the mobile booking application, just without the ability to sign in and view records.
- The Notifications for customers who make bookings will be encapsulated by the Notifications component, sending out messages based on the event that has occurred, whether it be a booking beginning or nearing the end, or a payment invoice.
- The Data Checking volatility will be handled by the Parking Session manager, as it will provide the ability to provide information about any of the sessions initiated by a customer.
- The Expiation Issue will be encapsulated by both the PMS Vehicle Application as well as the Car Park Access. The PMS Vehicle Application will allow for the expiation to be printed and recorded but will also need information from the Car Park Access to determine the precise location of the incident.
- Feed Transformation will be taken care of through the Feed Transformation component, displaying the relevant information based on the request from the Operator.
- Viewing Statistics and the Expiations recently generated will be encapsulated in the Operator application, making use of the Feed Transformation component to display the requested statistics to the Operator.
- Security will be contained in the security component, making sure that the users are accessing only the components of the system that they are authorized to, only when they have requested it and only when they are in the relevant section of the interface they are using.
- Customer Access will be handled by the Customer Access component, providing access the customer information located in the relevant storage component.
- Booking Access will be handled by the Parking Session Access component, providing access to the information about each parking session.
- Expiation Access will be handled by the Expiation Access component, providing access to the information about each expiation that has been generated.
- Payment Provider will be handled by the Payment Access component, providing information about the verification of financial services.
- Finance Access will be handled by the Finance Access component, providing information about the transactions that have taken place when a customer pays for their parking session.
- Feed Access will be handled by the Feed Access component, providing access to the information displayed on the live dashboard interacted with by the operators.
- Customer Information Storage will hold the customer personal information. This will include their contact details, addresses, vehicles and parking session history.

- The Car Park Storage will hold the Car Park information. This will include the specific location of individual parking spaces, the car park location they are a part of, and the number associated with each one.
- The Parking Session Storage will hold the information about each parking session initiated by customers. This will include information such as the customer that initiated the session, the vehicle used, the date and time of the session, the duration of the session, the cost of the session and the status of payment used to pay for the session.
- The Expiation Storage will hold the information about each expiation generated by the vehicles. This will include the date and time of the expiation, a reference to any expired session, the plate number of the vehicle, the location of the expiation, the result of the manual validation, a ticket number, the expected repayment of the expiation as well as an image confirming the vehicle was at the location at the time specified.
- The Financial Data storage will hold the information about any payments associated with the system. This includes information such as the session being paid for, the customer who paid for the session, the data and time of the payment, the payment method used, the payment method being used, and whether it was verified or not.

UC02 Monitor Vehicles:
PMS vehicles visit the car parks on a regular schedule, monitoring the vehicles parked in each car park via LPR camera technology mounted on the vehicle, and logs the recognized license plates, time, and location information.

- Operation Contracts

PMS Vehicle
enterLoggingDetail()

Security
Login()
Display Error()

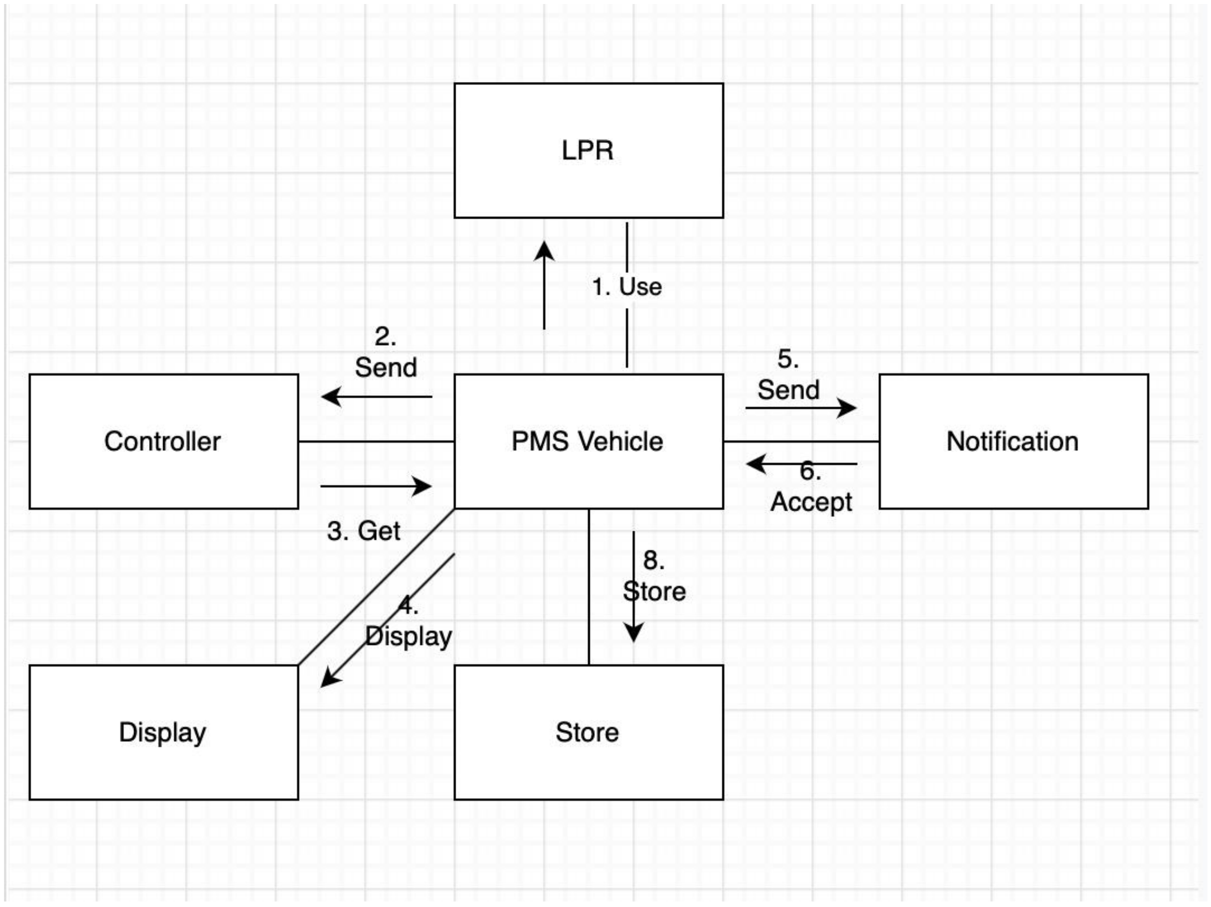
LPR
checkCar(carID: string)
selectCar(carID: string)

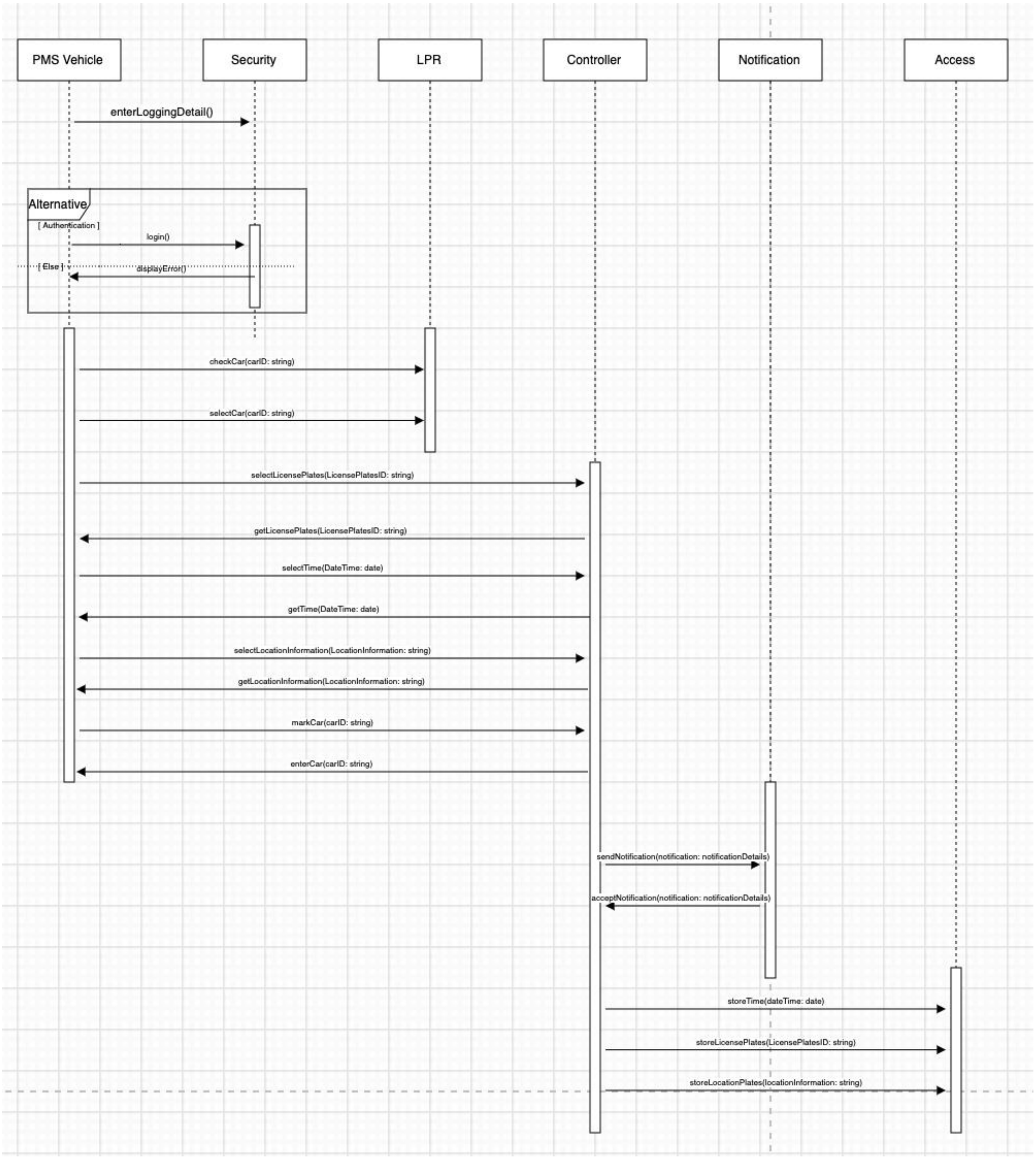
Controller
selectLicensePlates(licensePlatesID: string)
getLicensePlates(licensePlatesID: string)
selectTime(dateTime: date)
getTime(dateTime: date)
selectLocationInformation(locationInformation: string)

getLocationInformation(locationInformation: string)
markCar(carID: string)
enterCar(carID: string)

Notification
sendNotification(notification: notificationDetails)
acceptNotification(notification: notificationDetails)

Access
storeTime(dateTime: date)
storeLicensePlates(licensePlatesID: string)
storeLocationInformation(locationInformation: string)




PMS Vehicle
Security
LPR
Controller
Notification
Access
enterLoggingDetail()
Alternative
[Authentication]
login()
[Else]
displayError()
checkCar(carID: string)
selectCar(carID: string)
selectLicensePlates(LicensePlatesID: string)
getLicensePlates(LicensePlatesID: string)
selectTime(DateTime: date)
getTime(DateTime: date)
selectLocationInformation(LocationInformation: string)
getLocationInformation(LocationInformation: string)
markCar(carID: string)
enterCar(carID: string)
sendNotification(notification: notificationDetails)
acceptNotification(notification: notificationDetails)
storeTime(dateTime: date)
storeLicensePlates(LicensePlatesID: string)
storeLocationPlates(locationInformation: string)

Interaction diagram Refers to diagrams representing interactions between system components and belongs to logical view in 4+1 view. Typical examples are sequence diagrams and communication diagrams, and timing diagrams also belong to this category.

Interactions are specified through messages sent and received between objects and represents how objects interact to accomplish a use case.

This is a sequence diagram and collaboration diagram, and is a diagram belonging to the interaction diagram. The PMS vehicle first accesses and uses the LPR for checking. And if you choose to send information to the controller, send it and then display it. The next step is to send and receive notifications and finally save them. If you look at UC02, the PMS Vehicle examines and checks the cars. Then, the license plates, time and location information are stored. Here, the PMS Vehicle must exist unconditionally, and to log in to the PMS, enter the application and access it. It then monitors using LPR to help check and select cars. There must also be a controller, and the controller is a controlling device, which can be usefully used to identify license plates, time and location information, and this information can be known. Also, by doing this, you can determine whether you are registered with PMS or not, you can record what the car is as a reminder, and save the information you find using Access so you don't lose it.

UC03 Issue Expiation Notice:

Operation Contracts

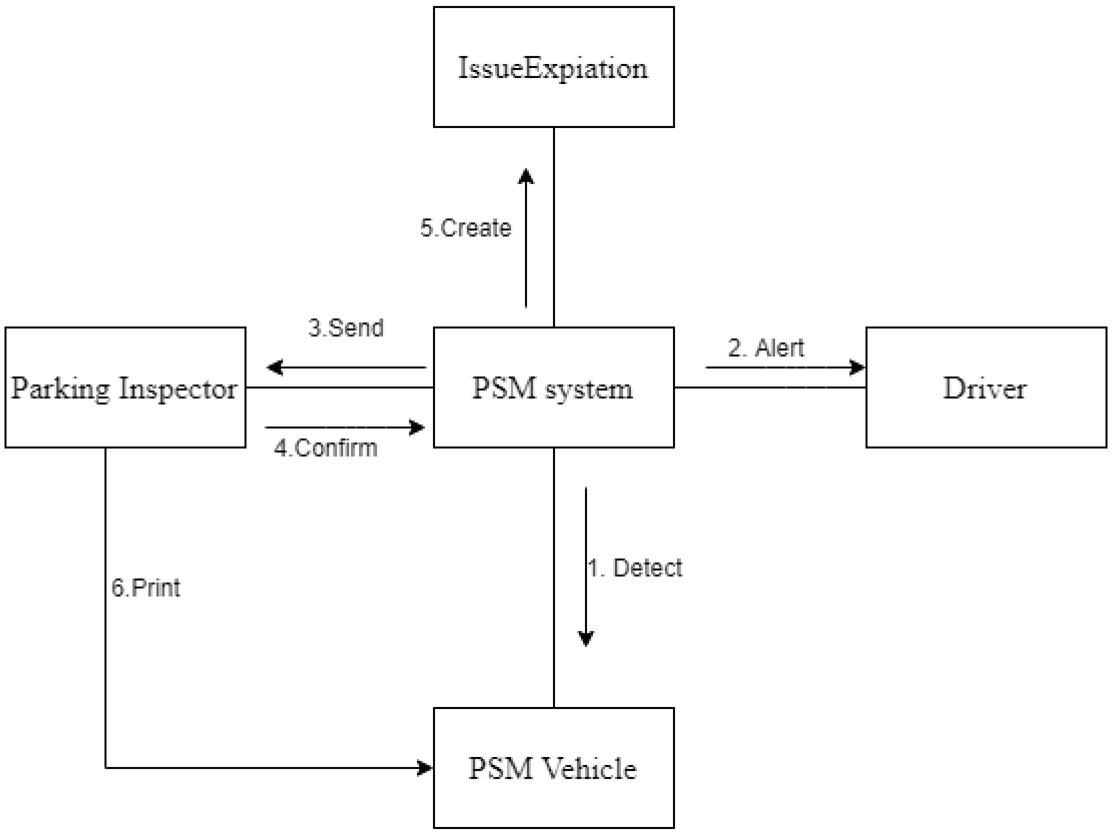
PSM system:
getCar(carID: String)
getLocationInformation(locationInformation: String)

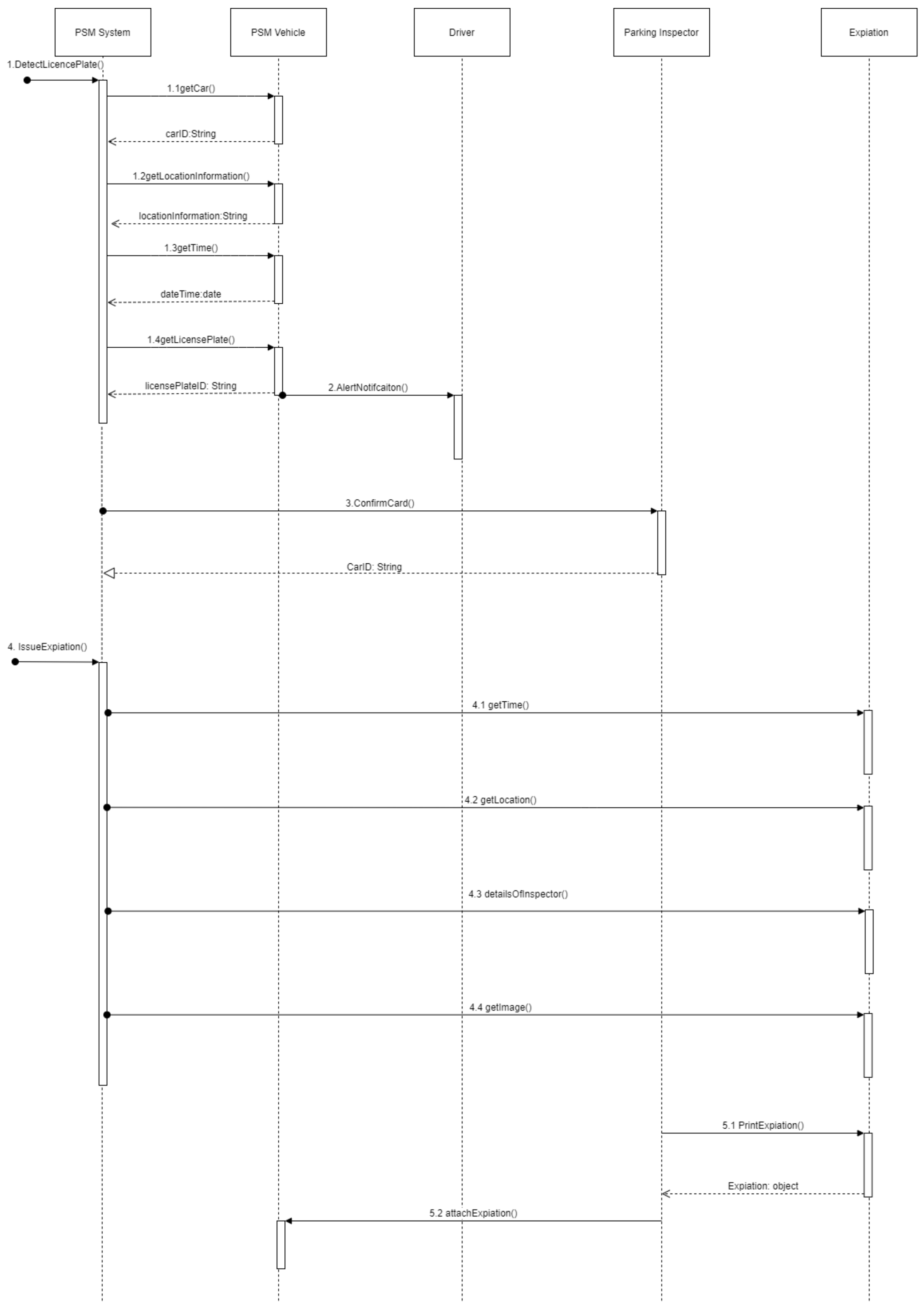
getTime(dateTime: date)
getLicensePlate(licensePlateID: String)

AlertNotification
sendNotification(notifcaitonDetails: Object, driverID: String)

Parking Inspector
confirmCar(carID: String)
printTicket(expiationDetails: String)
attachesExpiation(carLocation: String, expiationDetails: String)

IssueExpiation
getTime(dateTime: date)
getLocation(location: String)
detailsOfInspector(inspectorID: String, inspectorName: String)
getImage(carImage: Image)




PSM System
PSM Vehicle
Driver
Parking Inspector
Expiation
1.DetectLicencePlate()
1.1getCar()
carID:String
1.2getLocationInformation()
locationInformation:String
1.3getTime()
dateTime:date
1.4getLicensePlate()
licensePlateID: String
2.AlertNotifcaiton()
3.ConfirmCard()
CarID: String
4. IssueExpiation()
4.1 getTime()
4.2 getLocation()
4.3 detailsOfInspector()
4.4 getImage()
5.1 PrintExpiation()
Expiation: object
5.2 attachExpiation()

The UC03 Issue Expiation Notice interaction diagram shows that the PSM system will detect the vehicle first with its License Plate. If yes, sending alert and send inspector to check. Once the inspector confirms that the car has not registered, the system will issue an expiation immediately. The expiation includes times, location, inspector details and an image of the unregistered car. The inspector will then print out the expiation and attach it on the car.

UC04 View KPIs: The PMS operators view hourly statistics about parked vehicles, expiation notices issued, and PMS vehicles ’movements (live) on a dashboard, broken down by location and by client. The information shown on the dashboard shall be updated no less than once within any 5-minutes-long time period.

- Operation Contracts

Operator

enterLoggingDetail()

security

Login()
Display Error()

imageAnalysis

requestImage(image: image)

getImage(image: image)

PMS

requestInformation(informationDetails: string)

getCar(carID: string)

getLocationInformation(locationInformation: string)

getTime(dateTime: date)

getLicensePlates(licensePlatesID: string)

Notification

sendNotification(notificationDetails: Object, driverID: String)
acceptNotification(notificationDetail: Object, driverID: String)

feedTransformation

handleExpiationLocation(ExpiationID: string)

requestExpiationLocation(ExpiationID: string)

handleExpiationClient(ExpiationID: string)

requestExpiationClient(ExpiationID: string)

getExpiation(expiationDetails: string)

getMovements(movementsDetails: string)

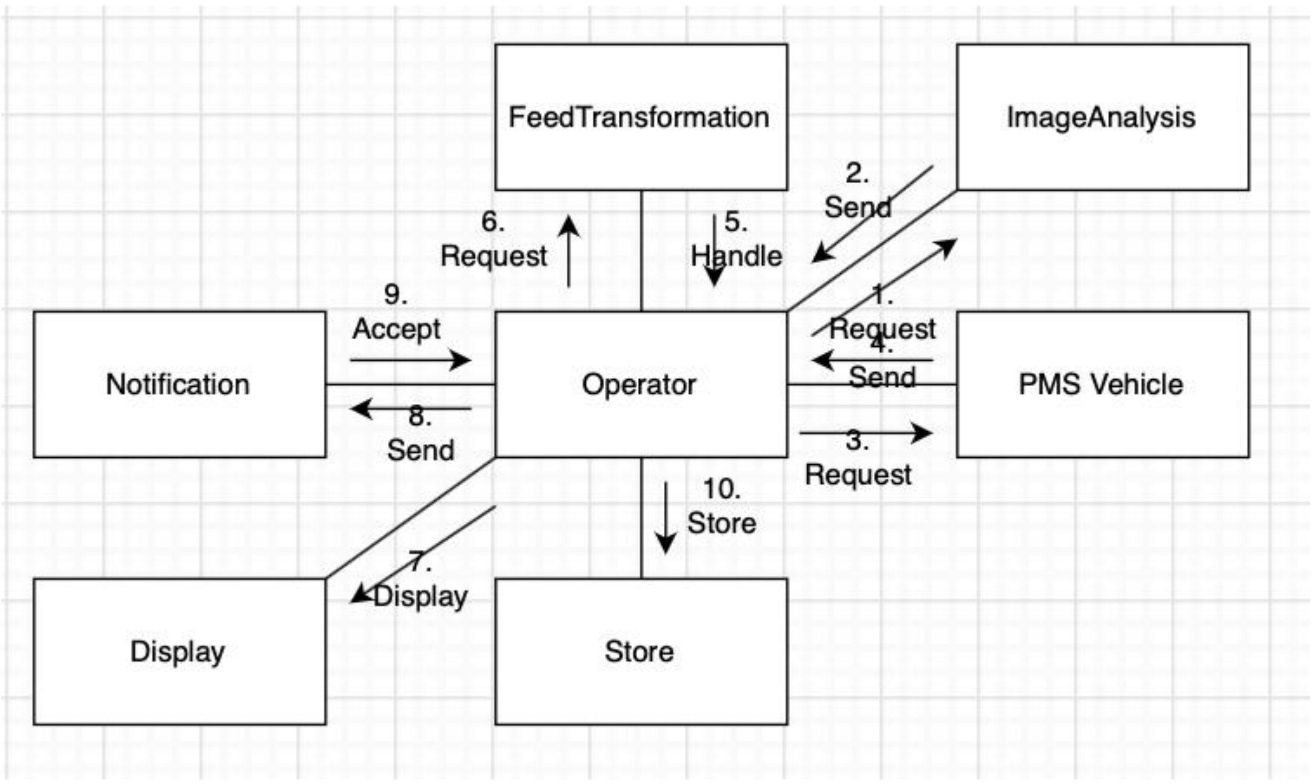
getHourlyStatistics(HourStatisticsDetails: string)

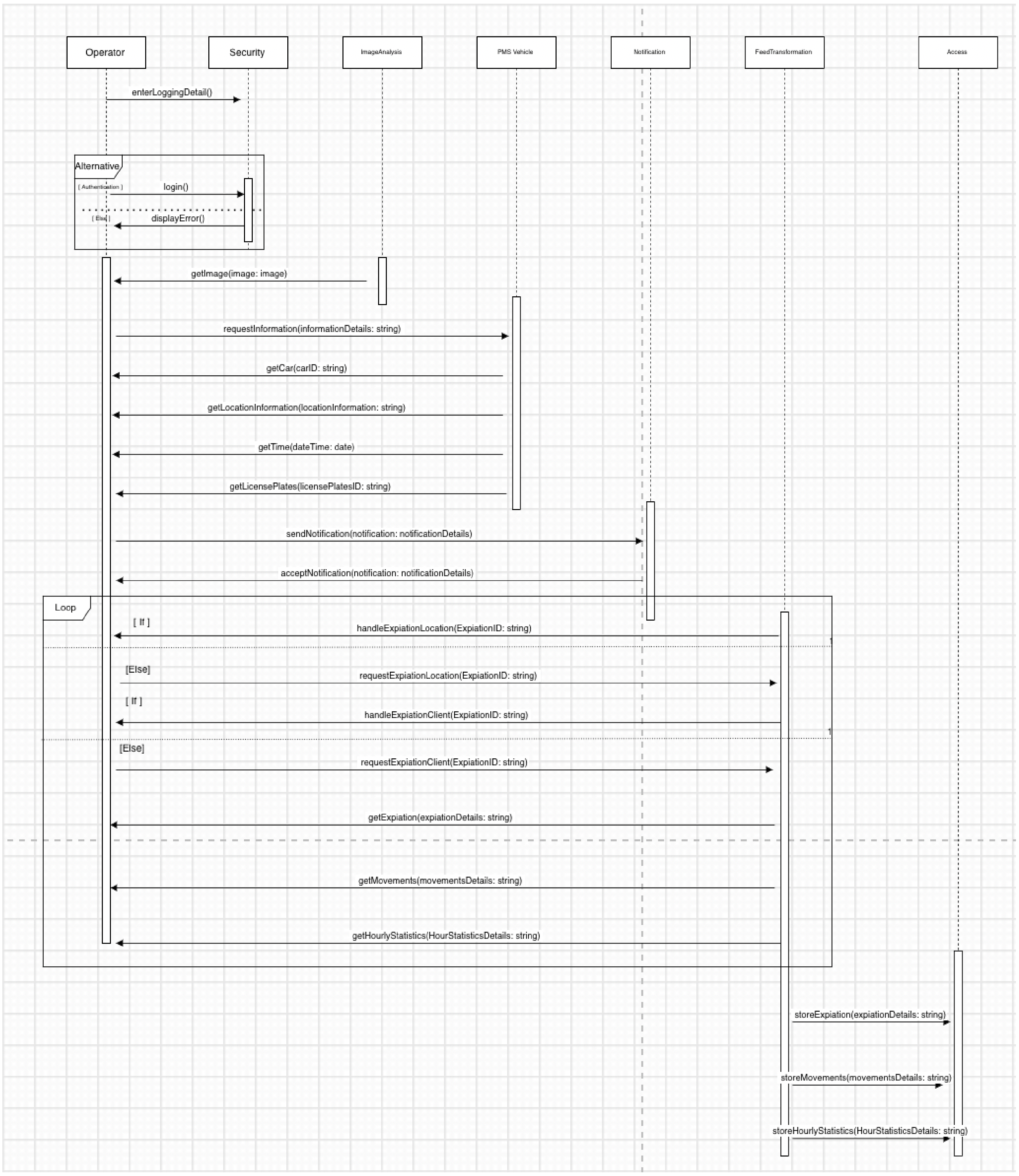
Access

storeExpiation(expiationDetails: string)

storeMovements(movementsDetails: string)

storeHourlyStatistics(HourStatisticsDetails: string)





Operator
Security
ImageAnalysis
PMS Vehicle
Notification
FeedTransformation
Access
enterLoggingDetail()
Alternative
[Authentication]
login()
[Else]
displayError()
getImage(image: image)
requestInformation(informationDetails: string)
getCar(carID: string)
getLocationInformation(locationInformation: string)
getTime(dateTime: date)
getLicensePlates(licensePlatesID: string)
sendNotification(notification: notificationDetails)
acceptNotification(notification: notificationDetails)
Loop
[If]
handleExpiationLocation(ExpiationID: string)
[Else]
requestExpiationLocation(ExpiationID: string)
[If]
handleExpiationClient(ExpiationID: string)
[Else]
requestExpiationClient(ExpiationID: string)
getExpiation(expiationDetails: string)
getMovements(movementsDetails: string)
getHourlyStatistics(HourStatisticsDetails: string)
storeExpiation(expiationDetails: string)
storeMovements(movementsDetails: string)
storeHourlyStatistics(HourStatisticsDetails: string)

This is a sequence diagram and collaboration diagram, and is a diagram belonging to the interaction diagram. Looking at UC04, PMS operators view hourly statistics for parked vehicles, issued atonement notices, and movement ㄴ (live) of PMS vehicles, broken down by location and client on the dashboard.

As you can see, the operator can initially request image analysis and receive the parking image. When received, it requests information from the PMS Vehicle and receives the information. Next, receive and request information on the hourly statistics, atonement notice and PMS vehicle movement from Feed Transformation. It then displays, sends and receives notifications, and finally saves the data.

Security is required to log in. Then, sometimes an error may appear when logging in. Receive images from image analysis and request and receive information from PMS Vehicle (license plates, time, location information)

Then implement notifications to send and receive notifications, handle hourly statistics, issued atonement notices and PMS vehicle movements to feed transformation and request them. Also, you need access to save data.

References

- Baddreddin, O., & Rahad, K. (2018, October). The impact of design and UML modeling on codebase quality and sustainability. In *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering* (pp. 236-244).

- Elaasar, M., Noyrit, F., Badreddin, O., & Gérard, S. (2018, January). Reducing UML Modeling Tool Complexity with Architectural Contexts and Viewpoints. In *MODELSWARD* (pp. 129-138).

- Kusic, G. (2018). *Computer-aided power systems analysis*. CRC Press.

- Naiyapo, W., & Jumpamule, W. (2018, November). An event driven approach to create UML models. In *2018 22nd International Computer Science and Engineering Conference (ICSEC)* (pp. 1-5). IEEE.

- Valacich, J. S., George, J. F., & Valacich, J. S. (2017). *Modern systems analysis and design* (Vol. 9). Boston: Pearson.