**Register Car**

@startuml

package "Register Car" {

actor CarOwner

boundary UI as "RegisterCarForm"

control Controller as "CarController"

control Service as "CarService"

entity Repo as "CarRepository"

entity Car

CarOwner -> UI : open register form

UI --> CarOwner : display input fields

CarOwner -> UI : submit car data

UI -> Controller : validate input

Controller -> Service : process new car (if input valid)

Service -> Car : create new car entity

Service -> Repo : save car

Repo --> Service : confirmation

Service --> Controller : success

Controller --> UI : show success message

UI --> CarOwner : registration complete

Controller --> UI : show error message (if input invalid)

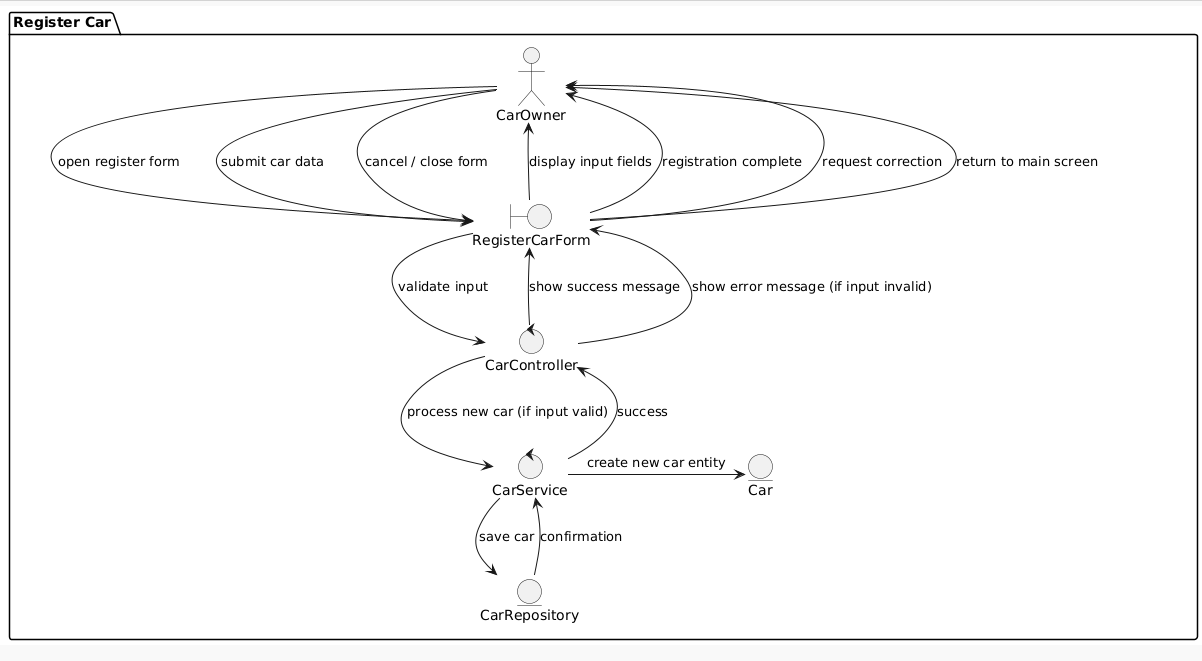
UI --> CarOwner : request correction

CarOwner -> UI : cancel / close form

UI --> CarOwner : return to main screen

}

@enduml



**Update Car Info**

@startuml

package "Update Car Info" {

actor CarOwner

boundary UI

control Controller as "CarController"

control Service as "CarService"

entity Repo as "CarRepository"

entity Car

CarOwner -> UI : select car to update

UI -> Controller : request car info

Controller -> Service : fetch car by ID

Service -> Repo : find car

Repo --> Service : return car

Service --> Controller : car data

Controller --> UI : show data

CarOwner -> UI : submit updated data

UI -> Controller : validate and submit

Controller -> Service : update car

Service -> Repo : save updated car

Repo --> Service : update confirmation

Service --> Controller : success

Controller --> UI : show confirmation

Controller --> UI : show error message

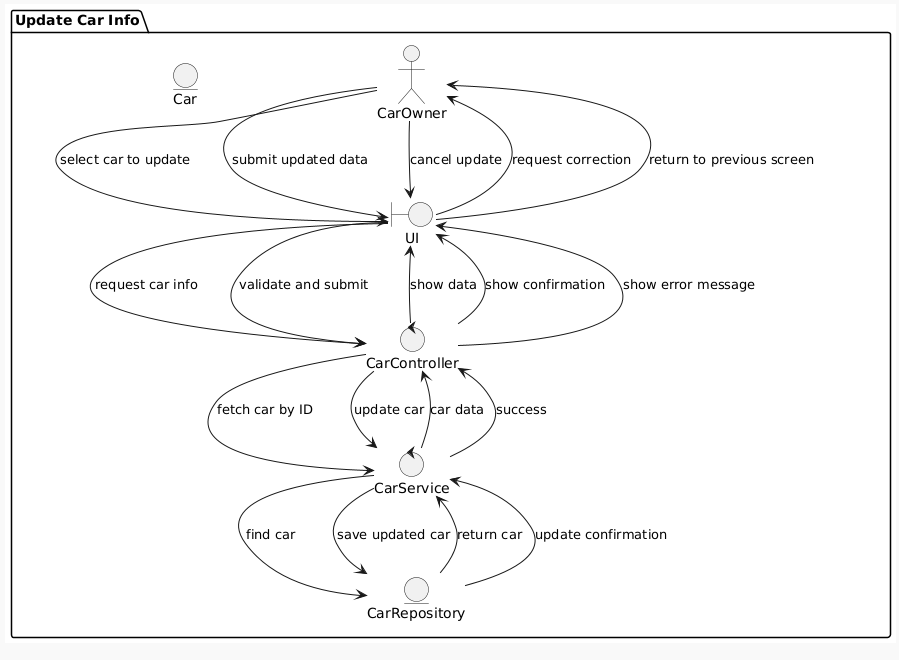
UI --> CarOwner : request correction

CarOwner -> UI : cancel update

UI --> CarOwner : return to previous screen

}

@enduml



**Book Car + Pay for Car**

@startuml

package "Book Car + Pay for Car" {

actor Renter

boundary UI

control BookingController

control BookingService

entity CarRepository

control PaymentService

entity BookingRepository

Renter -> UI : select car + book

UI -> BookingController : request booking

BookingController -> BookingService : create booking

BookingService -> CarRepository : check availability

CarRepository --> BookingService : available

BookingService -> UI : show booking details

Renter -> UI : confirm booking

BookingService -> PaymentService : initiate payment

PaymentService -> Renter : ask payment method + card details

Renter -> PaymentService : provide payment info

PaymentService -> PaymentService : charge card

PaymentService --> BookingService : payment confirmation

BookingService -> BookingRepository : save booking

BookingRepository --> BookingService : confirmation

BookingService --> BookingController : booking successful

BookingController --> UI : show success message

UI --> Renter : booking pending approval

PaymentService --> BookingService : payment failed

BookingService --> BookingController : notify failure

BookingController --> UI : show payment failure

UI --> Renter : payment failed

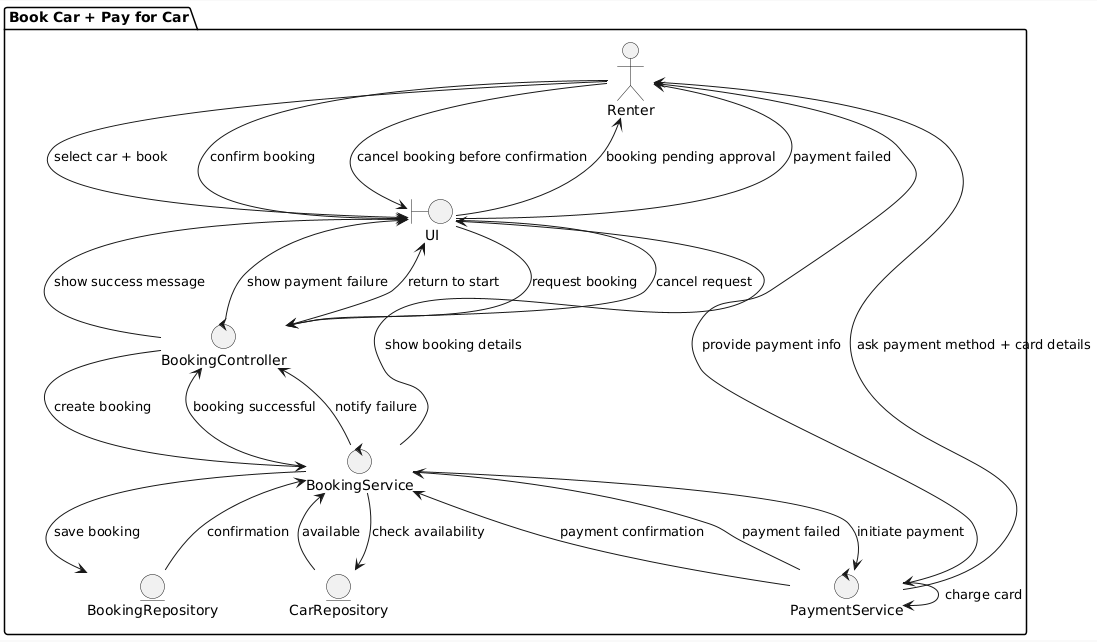
Renter -> UI : cancel booking before confirmation

UI --> BookingController : cancel request

BookingController --> UI : return to start

}

@enduml



**Approve Booking**

@startuml

package "Approve Booking" {

actor CarOwner

boundary UI

control Controller as "BookingController"

control Service as "BookingService"

entity Repo as "BookingRepository"

control NotificationService

CarOwner -> UI : view pending bookings

UI -> Controller : get requests

Controller -> Service : fetch pending bookings

Service -> Repo : query pending

Repo --> Service : list

Service --> Controller : return list

Controller --> UI : show list

CarOwner -> UI : approve booking

UI -> Controller : confirm approval

Controller -> Service : update status = approved

Service -> Repo : update booking status

Repo --> Service : confirmation

Service -> NotificationService : notify renter (approved)

NotificationService --> Service : notification done

Service --> Controller : done

Controller --> UI : show approval success

CarOwner -> UI : reject booking

UI -> Controller : confirm rejection

Controller -> Service : update status = rejected

Service -> Repo : update booking status

Repo --> Service : confirmation

Service -> NotificationService : notify renter (rejected)

NotificationService --> Service : notification done

Service --> Controller : done

Controller --> UI : show rejection success

Service -> Repo : update status = expired

Repo --> Service : confirmation

Service -> NotificationService : notify renter (expired)

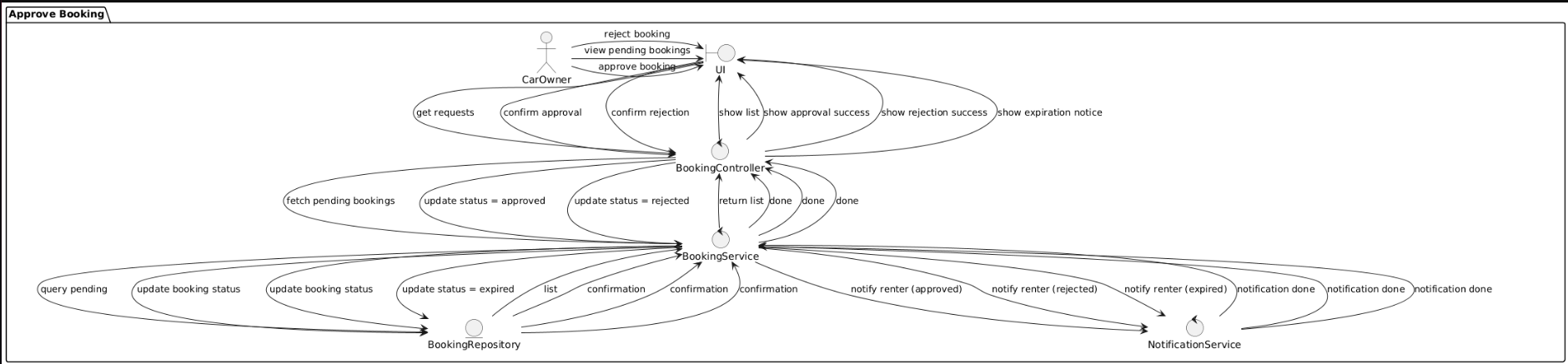
NotificationService --> Service : notification done

Service --> Controller : done

Controller --> UI : show expiration notice

}

@enduml



**Reject Booking**

@startuml

package "Reject Booking" {

actor CarOwner

actor Admin

boundary UI

control Controller as "BookingController"

control Service as "BookingService"

entity Repo as "BookingRepository"

control NotificationService

CarOwner -> UI : view requests

Admin -> UI : view requests

UI -> Controller : get bookings

Controller -> Service : fetch pending bookings

Service -> Repo : fetch data

Repo --> Service : list

Service --> Controller : return list

Controller --> UI : show options

CarOwner -> UI : reject booking + reason

UI -> Controller : submit rejection + reason

Controller -> Service : update status = rejected + save reason

Service -> Repo : update booking status + reason

Repo --> Service : confirmation

Service -> NotificationService : notify renter with rejection + reason

NotificationService --> Service : notification done

Service --> Controller : done

Controller --> UI : show rejection confirmation

Admin -> UI : reject booking + reason "terms violation"

UI -> Controller : submit rejection + reason

Controller -> Service : update status = rejected + save reason

Service -> Repo : update booking status + reason

Repo --> Service : confirmation

Service -> NotificationService : notify renter with rejection + reason

NotificationService --> Service : notification done

Service --> Controller : done

Controller --> UI : show rejection confirmation

CarOwner -> UI : approve booking

UI -> Controller : submit approval

Controller -> Service : update status = approved

Service -> Repo : update booking status

Repo --> Service : confirmation

Service -> NotificationService : notify renter with approval

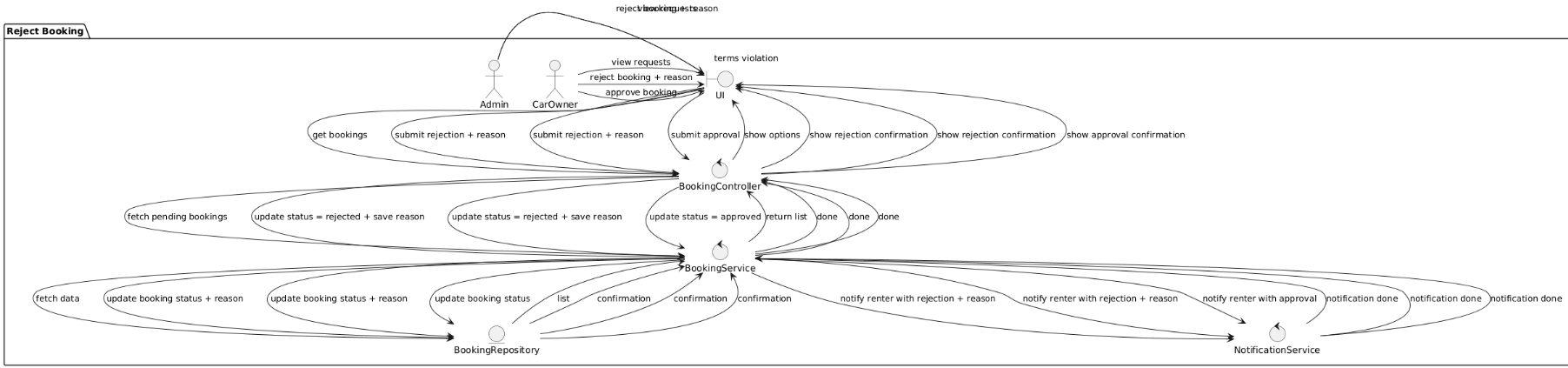
NotificationService --> Service : notification done

Service --> Controller : done

Controller --> UI : show approval confirmation

}

@enduml



**Cancel Booking**

@startuml

package "Cancel Booking" {

actor User

boundary UI

control Controller as "BookingController"

control Service as "BookingService"

entity Repo as "BookingRepository"

control NotificationService

User -> UI : select active booking to cancel

UI -> Controller : request cancellation

Controller -> Service : check booking status and policy

Service -> Repo : find booking

Repo --> Service : booking found

UI -> User : show confirmation prompt

User -> UI : confirm cancellation

UI -> Controller : confirm cancel

Controller -> Service : update to canceled

Service -> Repo : update booking status

Repo --> Service : confirmation

Service -> NotificationService : notify other party

NotificationService --> Service : notification sent

Service --> Controller : cancellation done

Controller --> UI : show canceled message

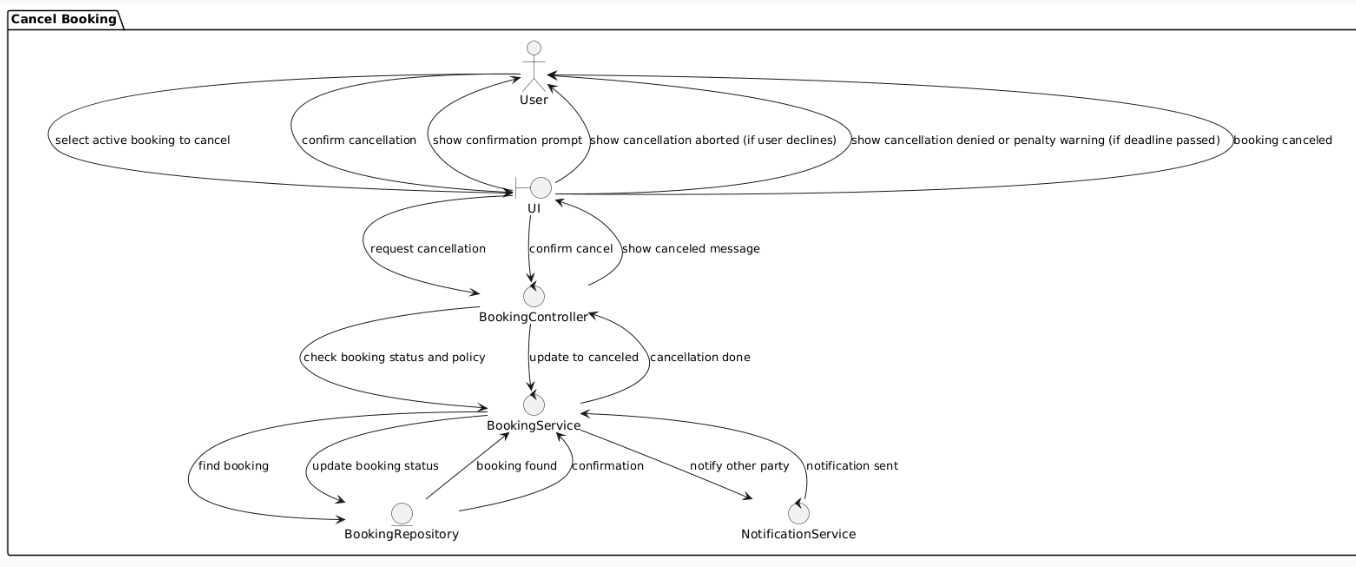
UI --> User : booking canceled

UI -> User : show cancellation aborted (if user declines)

UI -> User : show cancellation denied or penalty warning (if deadline passed)

}

@enduml



**Leave Review**

@startuml

package "Leave Review" {

actor Renter

boundary UI

control Controller as "ReviewController"

control Service as "ReviewService"

entity Repo as "ReviewRepository"

Renter -> UI : leave review

UI -> Controller : submit review data

Controller -> Service : validate and create review

Service -> Repo : save review

Repo --> Service : confirmation

Service --> Controller : done

Controller --> UI : show success

UI --> Renter : cancel review process (no save)

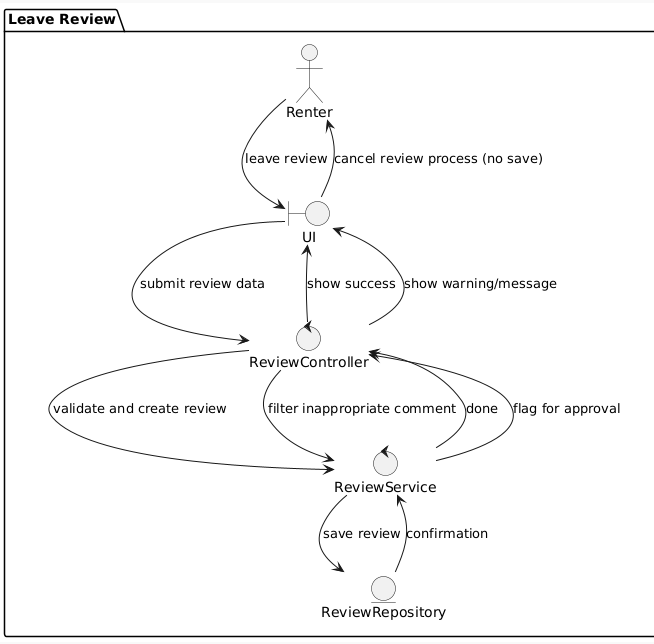
Controller -> Service : filter inappropriate comment

Service --> Controller : flag for approval

Controller --> UI : show warning/message

}

@enduml



**Resolve Dispute**

@startuml

package "Resolve Dispute" {

actor Renter

actor Admin

boundary UI

control DisputeController as Controller

control DisputeService as Service

entity DisputeRepository as Repo

entity Dispute

Renter -> UI : open dispute

UI -> Controller : submit dispute

Controller -> Service : create dispute

Service -> Dispute : create new dispute

Service -> Repo : save dispute

Repo --> Service : confirmation

Service --> Controller : done

Controller -> UI : show confirmation

Admin -> UI : view disputes

UI -> Controller : request disputes

Controller -> Service : fetch disputes

Service -> Repo : find disputes

Repo --> Service : list

Service --> Controller : return list

Controller -> UI : show disputes

Admin -> UI : resolve dispute

UI -> Controller : submit resolution

Controller -> Service : apply resolution

Service -> Repo : update dispute status

Repo --> Service : update confirmed

Service --> Controller : resolution saved

Controller -> UI : show resolution success

UI -> Controller : manual close request

Controller -> Service : close dispute

Service -> Repo : update status closed

Repo --> Service : confirmation

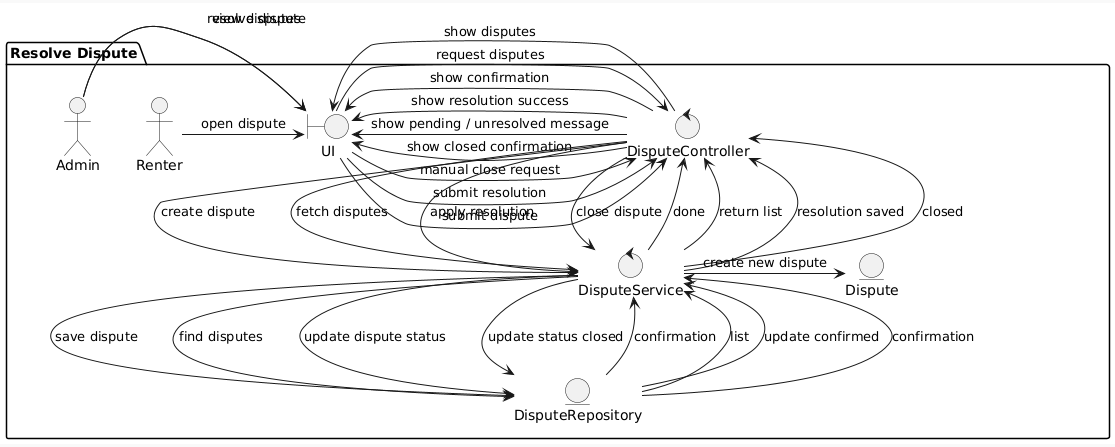
Service --> Controller : closed

Controller -> UI : show closed confirmation

Controller -> UI : show pending / unresolved message

}

@enduml



**Manage Users & Cars**

@startuml

package "Manage Users & Cars" {

actor Admin

boundary UI

control AdminController as Controller

control UserService

control CarService

entity UserRepository as UserRepo

entity CarRepository as CarRepo

entity User

entity Car

Admin -> UI : open management panel

UI -> Controller : request all users and cars

Controller -> UserService : get users

UserService -> UserRepo : find all users

UserRepo --> UserService : user list

Controller -> CarService : get cars

CarService -> CarRepo : find all cars

CarRepo --> CarService : car list

Controller -> UI : return combined data

UI -> Admin : show dashboard

Admin -> UI : select user or car to manage

UI -> Controller : submit management action

Controller -> UserService : block user / update status

Controller -> CarService : hide car / update status

UserService -> UserRepo : update user

CarService -> CarRepo : update car

UserRepo --> UserService : updated

CarRepo --> CarService : updated

UserService --> Controller : user update success

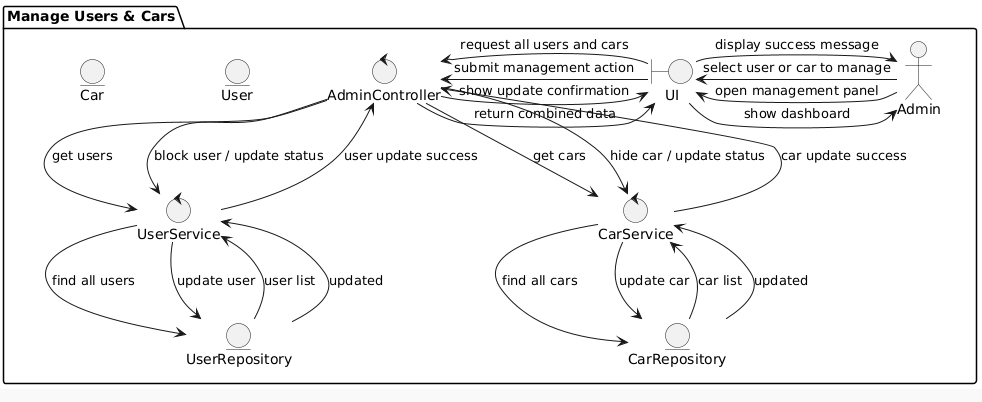
CarService --> Controller : car update success

Controller -> UI : show update confirmation

UI -> Admin : display success message

}

@enduml



**View Reports**

@startuml

package "View Reports" {

actor Admin

boundary UI

control ReportController as Controller

control ReportService as Service

entity BookingRepository as Repo

entity ReportData

Admin -> UI : open reports

UI -> Controller : get stats (optional filters)

Controller -> Service : fetch statistics

Service -> Repo : retrieve booking data

Repo --> Service : data set

Service -> ReportData : generate report

ReportData --> Service : report ready

Service -> Controller : report

Controller -> UI : show report

UI -> Admin : display charts and tables

Admin -> UI : apply filters

UI -> Controller : resubmit filtered request

Controller -> Service : fetch filtered stats

Service -> Repo : retrieve filtered data

Repo --> Service : filtered dataset

Service -> ReportData : generate filtered report

ReportData --> Service : updated report

Service -> Controller : new report

Controller -> UI : update display

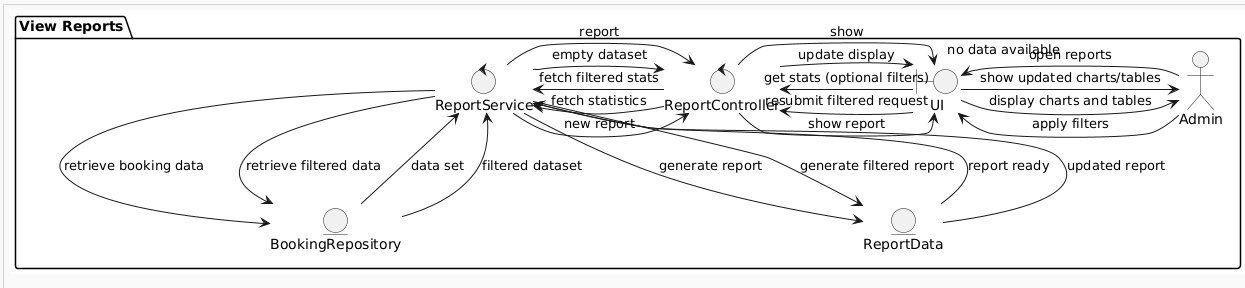
UI -> Admin : show updated charts/tables

Service -> Controller : empty dataset

Controller -> UI : show "no data available"

}

@enduml



**Search Car**

@startuml

package "Search Car" {

actor Renter

boundary UI

control SearchController as Controller

control SearchService as Service

entity CarRepository as Repo

entity Car

Renter -> UI : open search screen

UI -> Controller : display search form

Renter -> UI : enter search criteria

UI -> Controller : submit search

Controller -> SearchService : validate criteria

SearchService -> Repo : search available cars

Repo --> SearchService : list of cars

SearchService -> Controller : return results

Controller -> UI : show available cars

UI -> Renter : display list

SearchService -> Controller : criteria invalid

Controller -> UI : show warning

UI -> Renter : request correction

SearchService -> Controller : no cars found

Controller -> UI : suggest change of criteria

UI -> Renter : show no results message

}

@enduml

