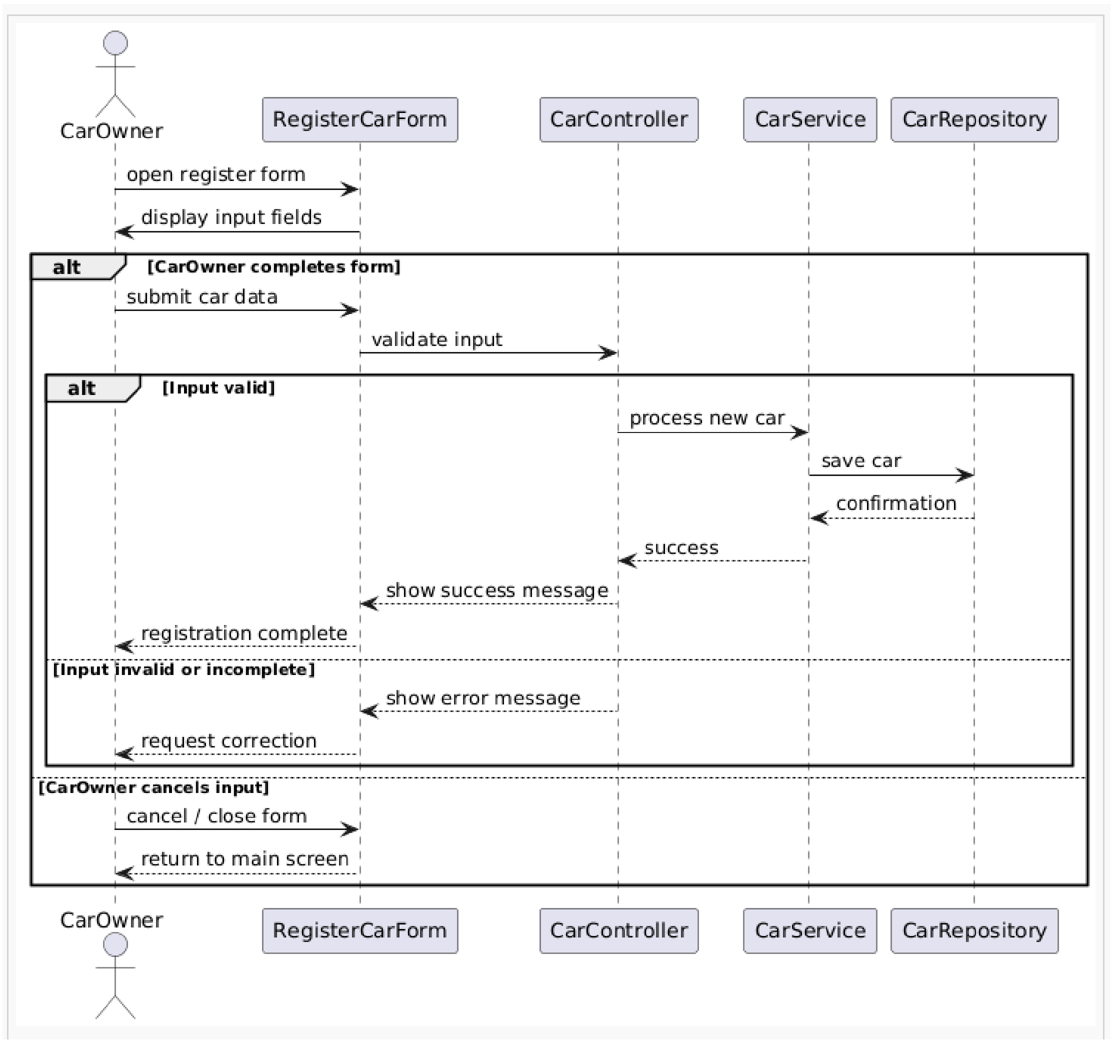
"Register Car"

```
@startuml

actor CarOwner

participant UI as "RegisterCarForm"

participant Controller as "CarController"

participant Service as "CarService"

participant Repo as "CarRepository"


CarOwner -> UI : open register form

UI -> CarOwner : display input fields


alt CarOwner completes form

   CarOwner -> UI : submit car data

   UI -> Controller : validate input


   alt Input valid

      Controller -> Service : process new car

      Service -> Repo : save car

      Repo --> Service : confirmation

      Service --> Controller : success

      Controller --> UI : show success message

      UI --> CarOwner : registration complete

   else Input invalid or incomplete

      Controller --> UI : show error message

      UI --> CarOwner : request correction

   end

else CarOwner cancels input
```

```
    CarOwner -> UI : cancel / close form

    UI --> CarOwner : return to main screen

end

@enduml
```

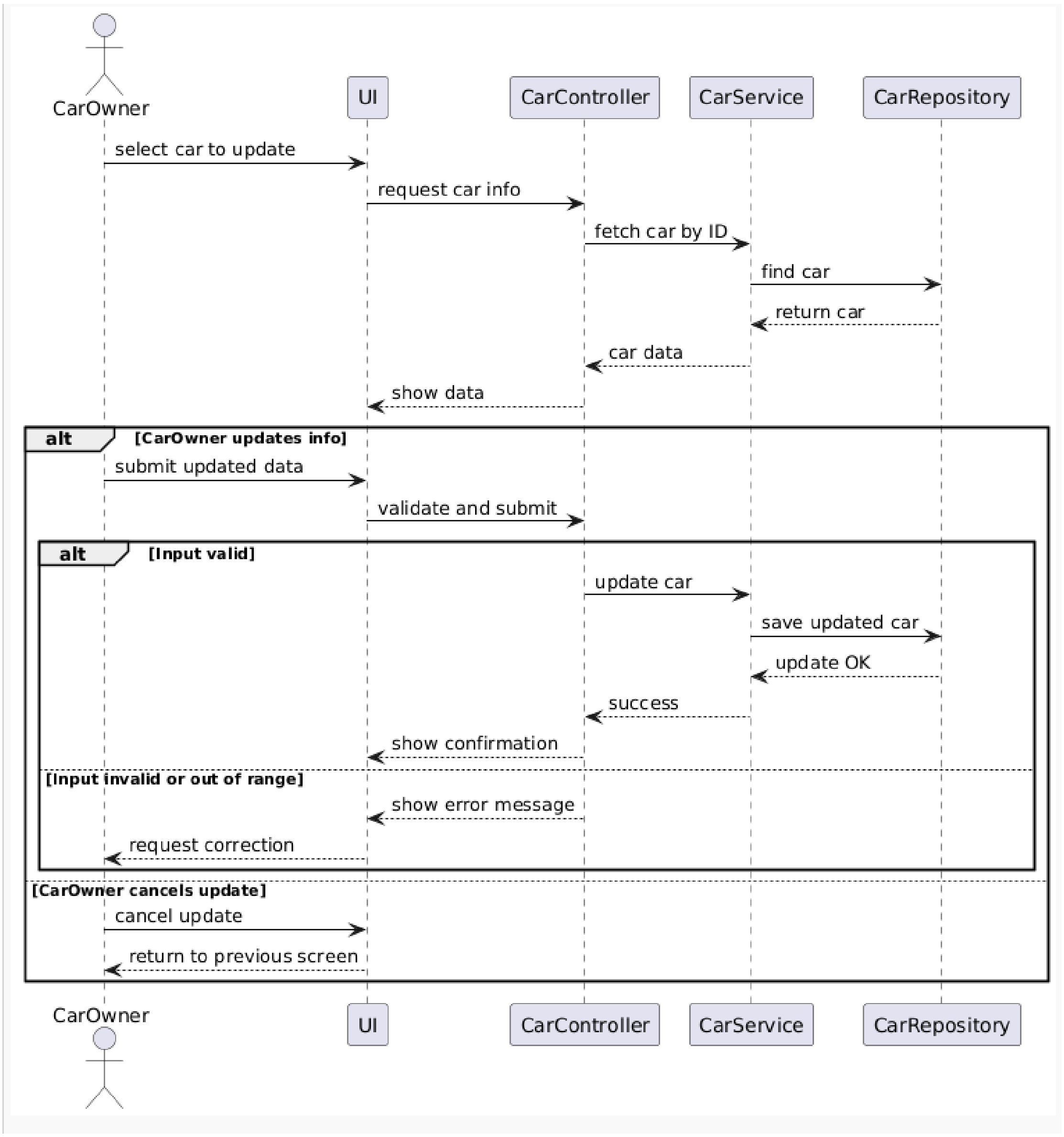


Update Car Info

```
@startuml

actor CarOwner

participant UI

participant Controller as "CarController"

participant Service as "CarService"

participant Repo as "CarRepository"
```

```
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

alt CarOwner updates info
  CarOwner -> UI : submit updated data
  UI -> Controller : validate and submit

  alt Input valid
    Controller -> Service : update car
    Service -> Repo : save updated car
    Repo --> Service : update OK
    Service --> Controller : success
    Controller --> UI : show confirmation
  else Input invalid or out of range
    Controller --> UI : show error message
    UI --> CarOwner : request correction
  end
else CarOwner cancels update
  CarOwner -> UI : cancel update
  UI --> CarOwner : return to previous screen
end
```

@enduml



Book Car + Pay for Car

@startuml

actor Renter

participant UI

participant BookingController

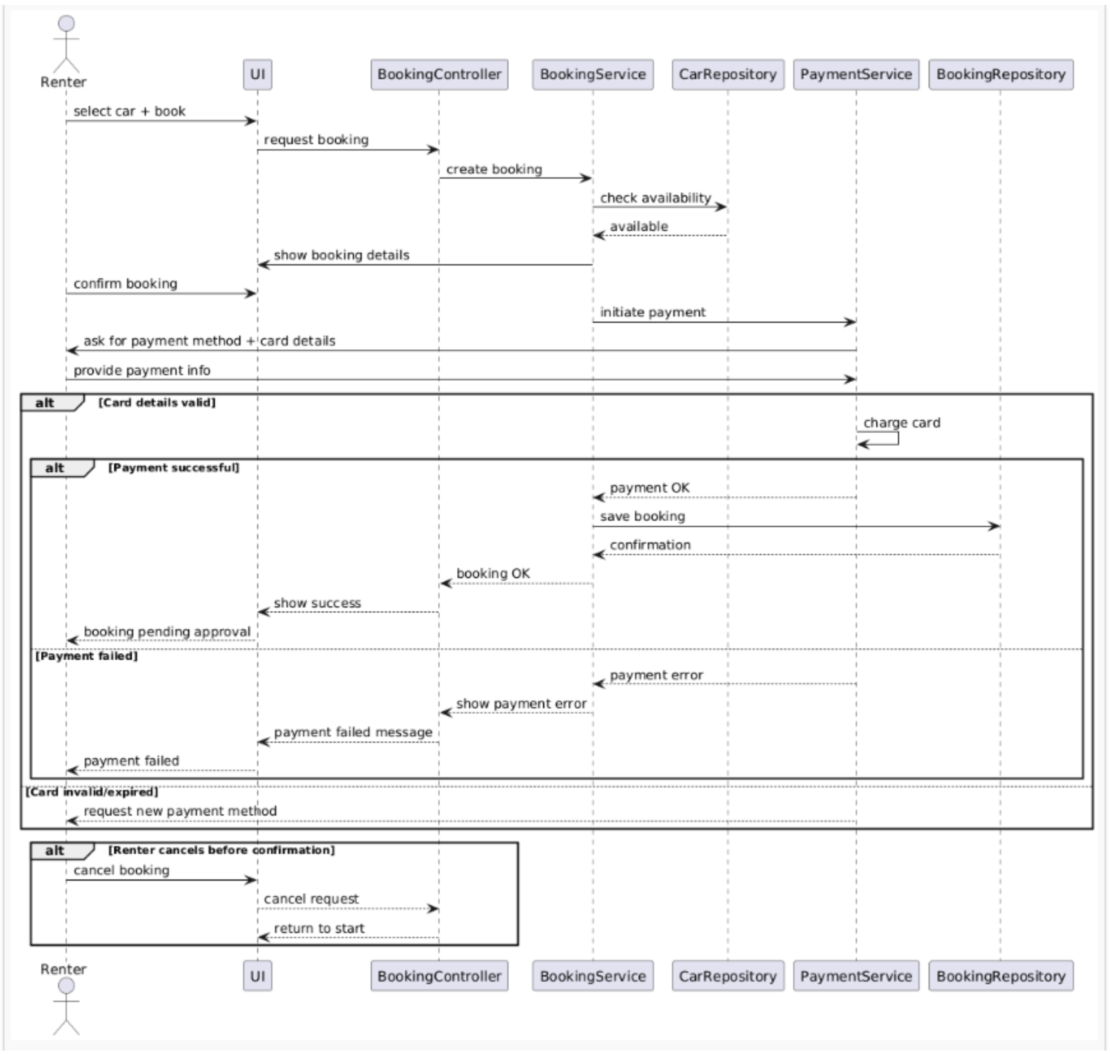
participant BookingService

participant CarRepository

participant PaymentService

```
participant 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 for payment method + card details

Renter -> PaymentService : provide payment info


alt Card details valid

  PaymentService -> PaymentService : charge card

  alt Payment successful

    PaymentService --> BookingService : payment OK

    BookingService -> BookingRepository : save booking

    BookingRepository --> BookingService : confirmation

    BookingService --> BookingController : booking OK

    BookingController --> UI : show success

    UI --> Renter : booking pending approval

  else Payment failed

    PaymentService --> BookingService : payment error

    BookingService --> BookingController : show payment error
```

```
        BookingController --> UI : payment failed message

        UI --> Renter : payment failed

    end

else Card invalid/expired

    PaymentService --> Renter : request new payment method

end


alt Renter cancels before confirmation

    Renter -> UI : cancel booking

    UI --> BookingController : cancel request

    BookingController --> UI : return to start

end

@enduml
```

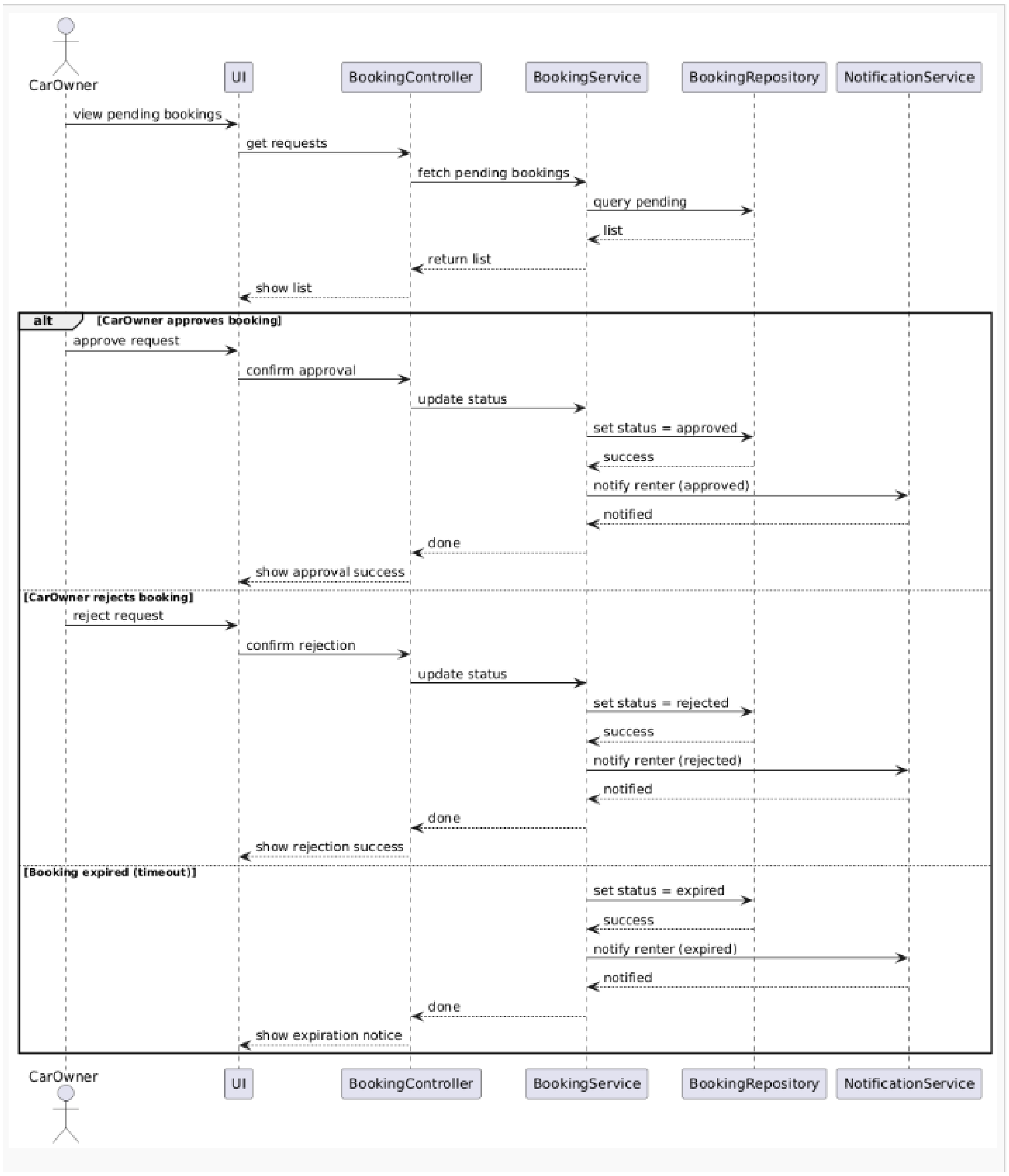


Approve Booking

```
@startuml

actor CarOwner

participant UI

participant Controller as "BookingController"

participant Service as "BookingService"

participant Repo as "BookingRepository"

participant NotificationService


CarOwner -> UI : view pending bookings

UI -> Controller : get requests
```

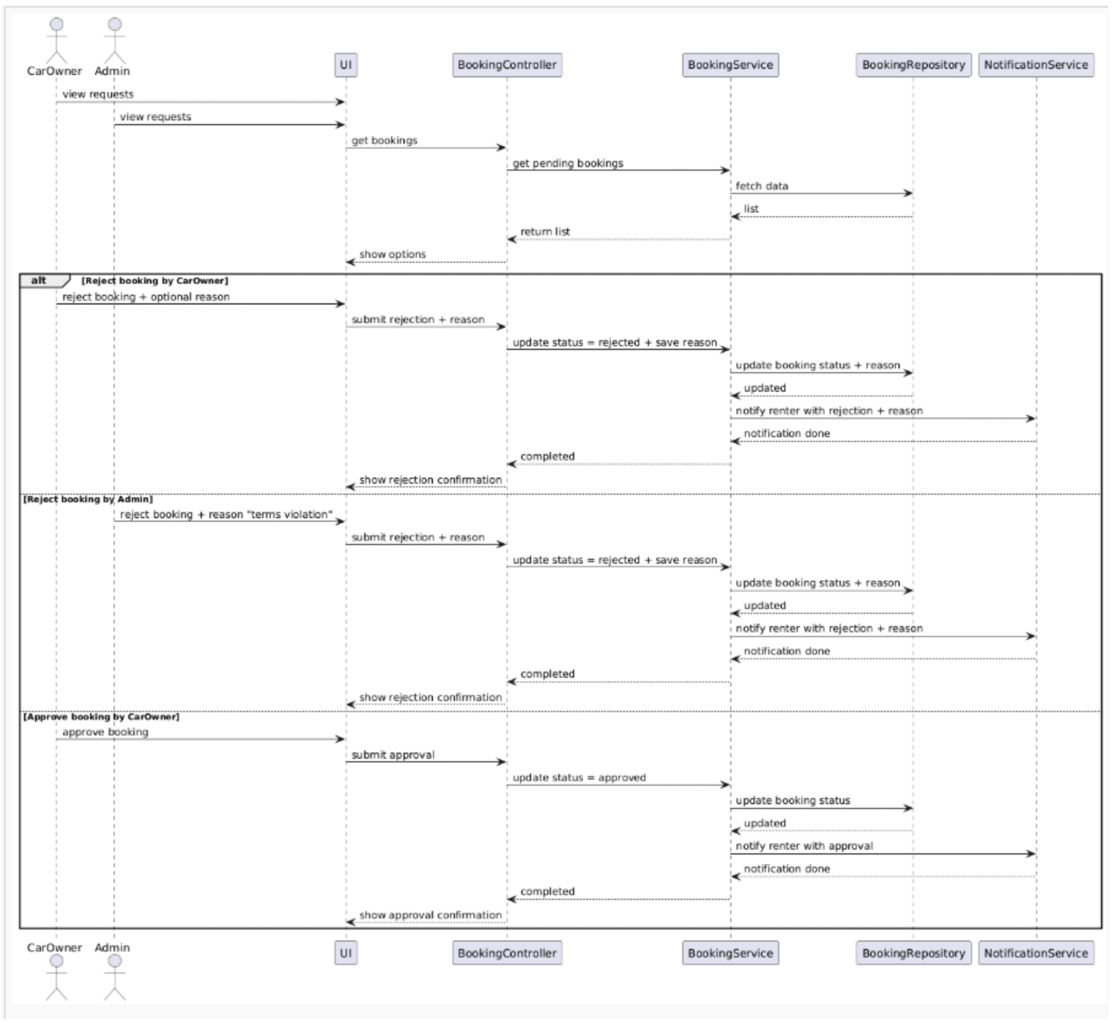
```
Controller -> Service : fetch pending bookings

Service -> Repo : query pending

Repo --> Service : list

Service --> Controller : return list

Controller --> UI : show list


alt CarOwner approves booking

   CarOwner -> UI : approve request

   UI -> Controller : confirm approval

   Controller -> Service : update status

   Service -> Repo : set status = approved

   Repo --> Service : success

   Service -> NotificationService : notify renter (approved)

   NotificationService --> Service : notified

   Service --> Controller : done

   Controller --> UI : show approval success

else CarOwner rejects booking

   CarOwner -> UI : reject request

   UI -> Controller : confirm rejection

   Controller -> Service : update status

   Service -> Repo : set status = rejected

   Repo --> Service : success

   Service -> NotificationService : notify renter (rejected)

   NotificationService --> Service : notified

   Service --> Controller : done

   Controller --> UI : show rejection success

else Booking expired (timeout)
```

```
    Service -> Repo : set status = expired

    Repo --> Service : success

    Service -> NotificationService : notify renter (expired)

    NotificationService --> Service : notified

    Service --> Controller : done

    Controller --> UI : show expiration notice

end

@enduml
```



Reject Booking

```
@startuml

actor CarOwner

actor Admin

participant UI

participant BookingController as Controller

participant BookingService as Service

participant BookingRepository as Repo

participant NotificationService


CarOwner -> UI : view requests

Admin -> UI : view requests

UI -> Controller : get bookings

Controller -> Service : get pending bookings

Service -> Repo : fetch data

Repo --> Service : list

Service --> Controller : return list

Controller --> UI : show options


alt Reject booking by CarOwner

   CarOwner -> UI : reject booking + optional reason

   UI -> Controller : submit rejection + reason

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

   Service -> Repo : update booking status + reason

   Repo --> Service : updated

   Service -> NotificationService : notify renter with rejection + reason

   NotificationService --> Service : notification done
   Service --> Controller : completed
   Controller --> UI : show rejection confirmation
else Reject booking by Admin
   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 : updated
   Service -> NotificationService : notify renter with rejection + reason
   NotificationService --> Service : notification done
   Service --> Controller : completed
   Controller --> UI : show rejection confirmation
else Approve booking by CarOwner
   CarOwner -> UI : approve booking
   UI -> Controller : submit approval
   Controller -> Service : update status = approved
   Service -> Repo : update booking status
   Repo --> Service : updated
   Service -> NotificationService : notify renter with approval
   NotificationService --> Service : notification done
   Service --> Controller : completed
   Controller --> UI : show approval confirmation
end
@enduml
```


CarOwner
Admin
UI
BookingController
BookingService
BookingRepository
NotificationService
view requests
view requests
get bookings
get pending bookings
fetch data
list
return list
show options
alt
[Reject booking by CarOwner]
reject booking + optional reason
submit rejection + reason
update status = rejected + save reason
update booking status + reason
updated
notify renter with rejection + reason
notification done
completed
show rejection confirmation
[Reject booking by Admin]
reject booking + reason "terms violation"
submit rejection + reason
update status = rejected + save reason
update booking status + reason
updated
notify renter with rejection + reason
notification done
completed
show rejection confirmation
[Approve booking by CarOwner]
approve booking
submit approval
update status = approved
update booking status
updated
notify renter with approval
notification done
completed
show approval confirmation

Cancel Booking

@startuml

actor User

participant UI

participant BookingController as Controller

participant BookingService as Service

participant BookingRepository as Repo

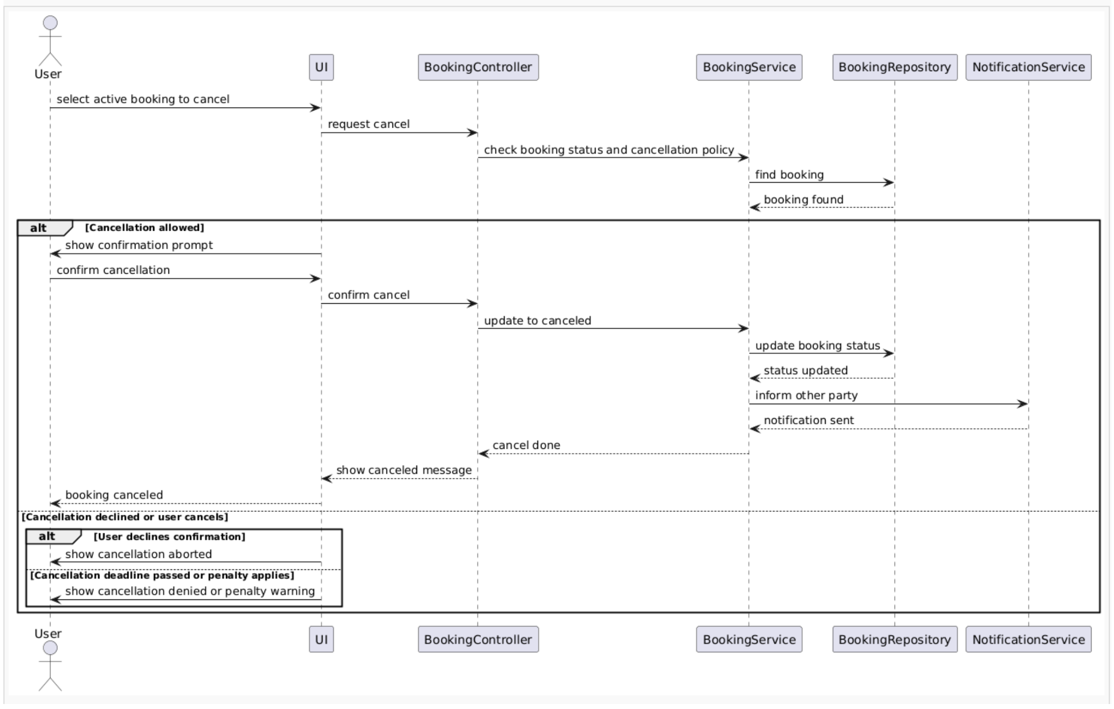
participant NotificationService

User -> UI : select active booking to cancel

UI -> Controller : request cancel

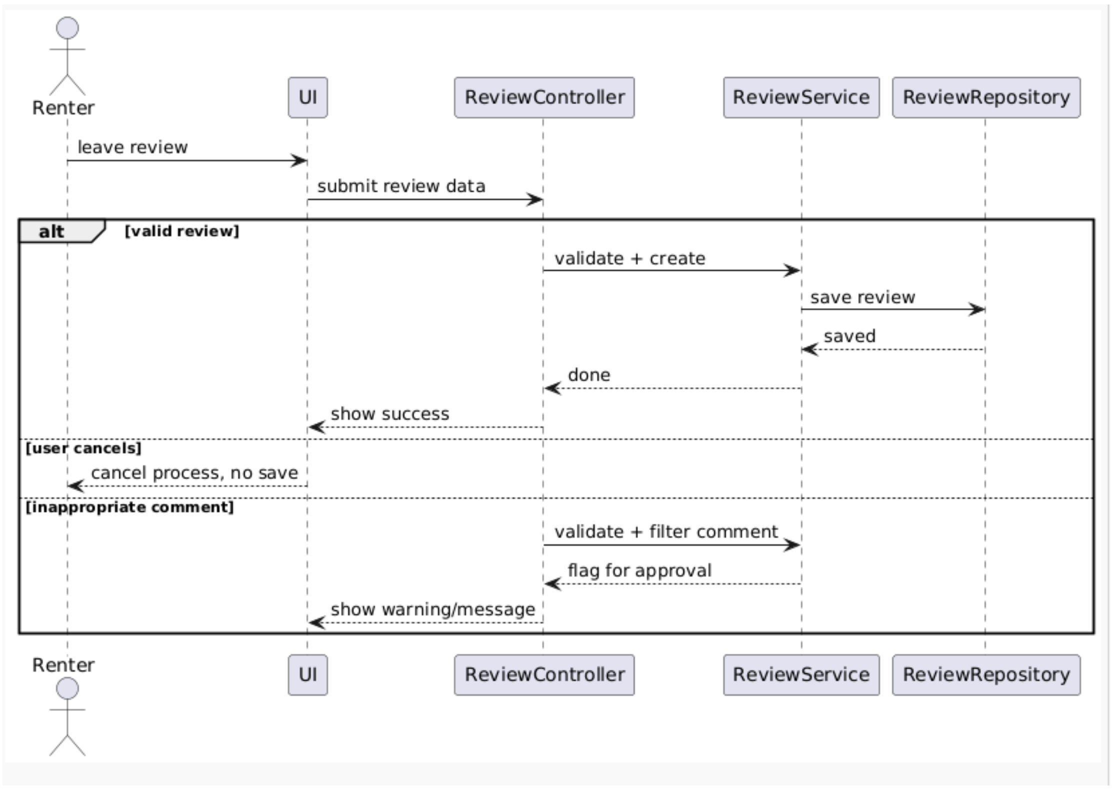
Controller -> Service : check booking status and cancellation policy

```
Service -> Repo : find booking

Repo --> Service : booking found


alt Cancellation allowed

   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 : status updated

   Service -> NotificationService : inform other party

   NotificationService --> Service : notification sent

   Service --> Controller : cancel done

   Controller --> UI : show canceled message

   UI --> User : booking canceled

else Cancellation declined or user cancels

   alt User declines confirmation

      UI -> User : show cancellation aborted

   else Cancellation deadline passed or penalty applies

      UI -> User : show cancellation denied or penalty warning

   end

end

@enduml
```



Leave Review

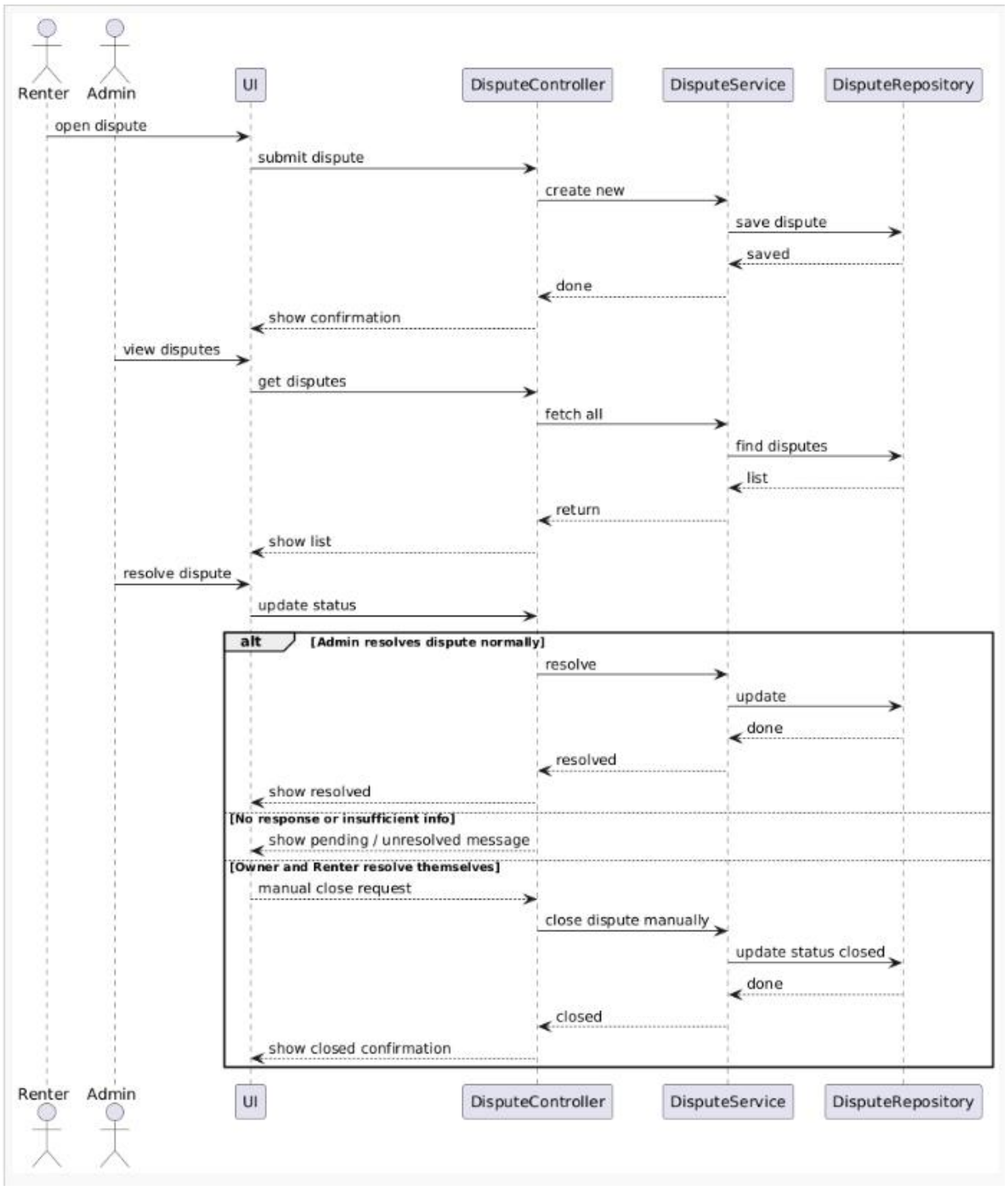
```
@startuml

actor Renter

participant UI

participant ReviewController as Controller

participant ReviewService as Service

participant ReviewRepository as Repo


Renter -> UI : leave review

UI -> Controller : submit review data


alt valid review

  Controller -> Service : validate + create

  Service -> Repo : save review

  Repo --> Service : saved

  Service --> Controller : done
  Controller --> UI : show success

else user cancels

  UI --> Renter : cancel process, no save

else inappropriate comment

  Controller -> Service : validate + filter comment

  Service --> Controller : flag for approval

  Controller --> UI : show warning/message

end


@enduml
```



Resolve Dispute

```
@startuml

actor Renter

actor Admin

participant UI
participant DisputeController as Controller

participant DisputeService as Service

participant DisputeRepository as Repo


Renter -> UI : open dispute

UI -> Controller : submit dispute

Controller -> Service : create new

Service -> Repo : save dispute

Repo --> Service : saved

Service --> Controller : done

Controller --> UI : show confirmation


Admin -> UI : view disputes

UI -> Controller : get disputes

Controller -> Service : fetch all

Service -> Repo : find disputes

Repo --> Service : list

Service --> Controller : return

Controller --> UI : show list


Admin -> UI : resolve dispute

UI -> Controller : update status


alt Admin resolves dispute normally

    Controller -> Service : resolve

    Service -> Repo : update

    Repo --> Service : done
```

```
   Service --> Controller : resolved

   Controller --> UI : show resolved


else No response or insufficient info

   Controller --> UI : show pending / unresolved message


else Owner and Renter resolve themselves

   UI --> Controller : manual close request

   Controller -> Service : close dispute manually

   Service -> Repo : update status closed

   Repo --> Service : done

   Service --> Controller : closed

   Controller --> UI : show closed confirmation

end


@enduml
```



Manage Users & Cars

@startuml

actor Admin

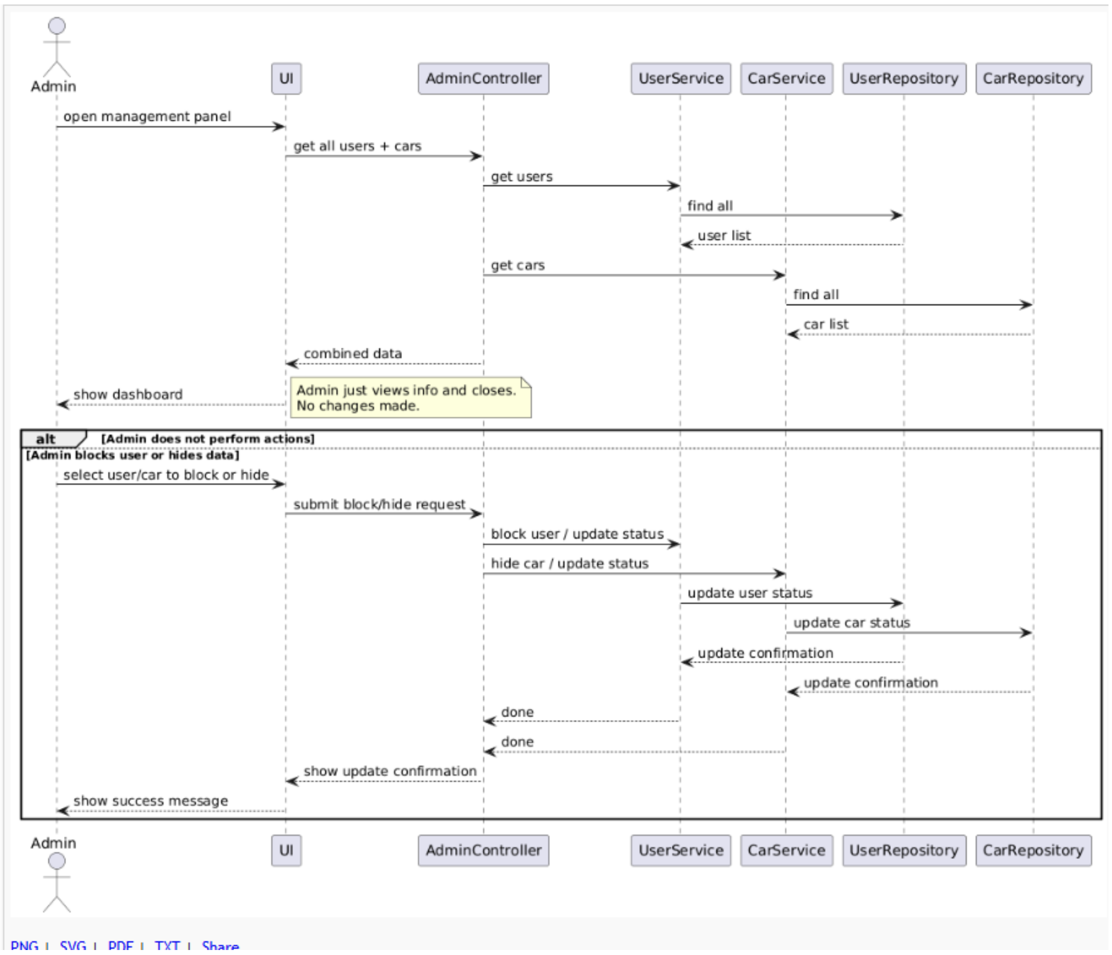
participant UI

participant AdminController as Controller

participant UserService

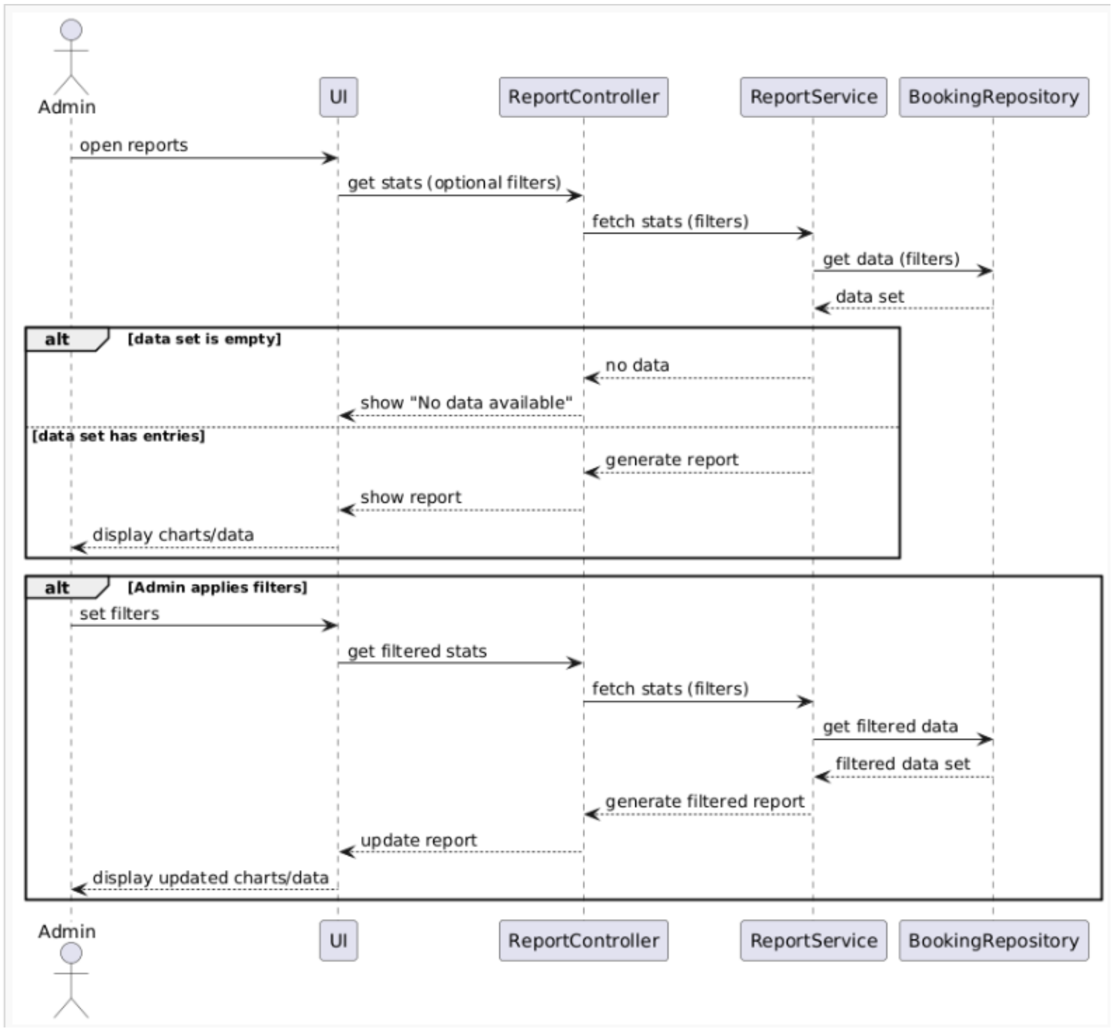
participant CarService

```
participant UserRepository as UserRepo
participant CarRepository as CarRepo

Admin -> UI : open management panel
UI -> Controller : get all users + cars
Controller -> UserService : get users
UserService -> UserRepo : find all
UserRepo --> UserService : user list

Controller -> CarService : get cars
CarService -> CarRepo : find all
CarRepo --> CarService : car list

Controller --> UI : combined data
UI --> Admin : show dashboard

alt Admin does not perform actions
  note right
    Admin just views info and closes.
    No changes made.
  end note

else Admin blocks user or hides data
  Admin -> UI : select user/car to block or hide
  UI -> Controller : submit block/hide request
  Controller -> UserService : block user / update status
  Controller -> CarService : hide car / update status

  UserService -> UserRepo : update user status

  CarService -> CarRepo : update car status

  UserRepo --> UserService : update confirmation

  CarRepo --> CarService : update confirmation

  UserService --> Controller : done

  CarService --> Controller : done

  Controller --> UI : show update confirmation

  UI --> Admin : show success message

end


@enduml
```



View Reports

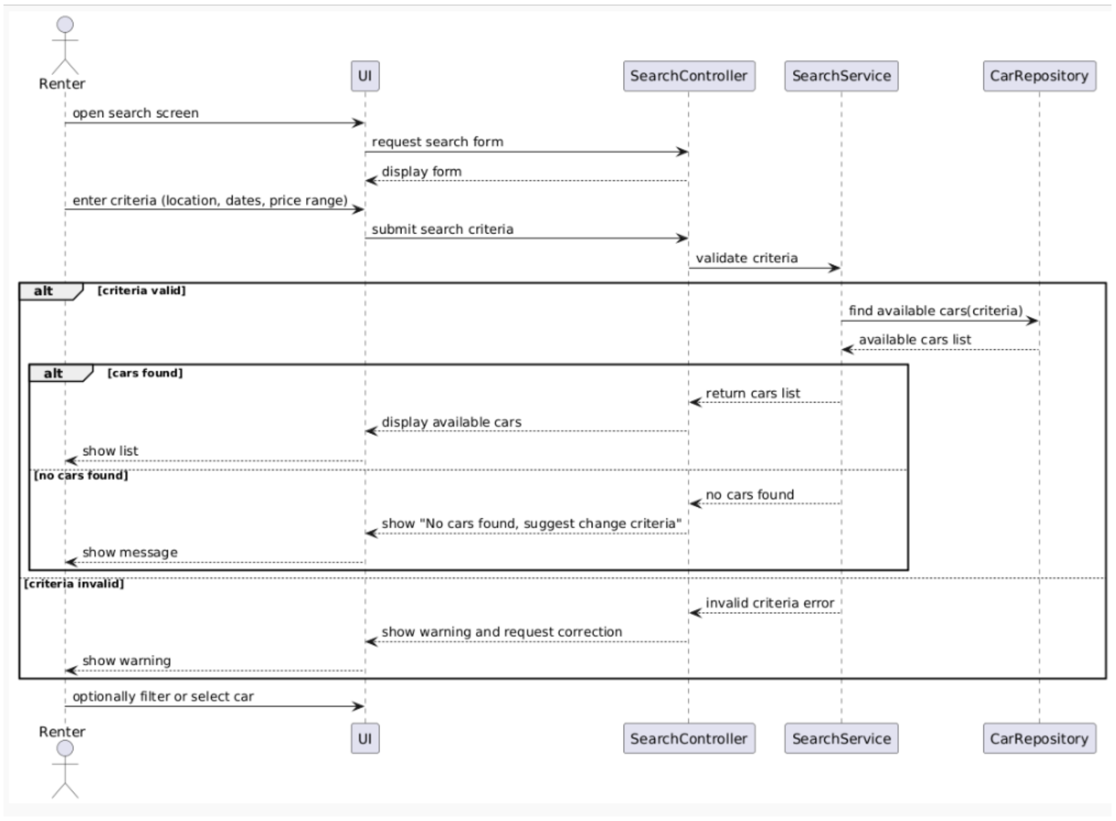
```
@startuml

actor Admin

participant UI

participant ReportController as Controller

participant ReportService as Service

participant BookingRepository as Repo


Admin -> UI : open reports

UI -> Controller : get stats (optional filters)

Controller -> Service : fetch stats (filters)

Service -> Repo : get data (filters)

Repo --> Service : data set


alt data set is empty

  Service --> Controller : no data

  Controller --> UI : show "No data available"

else data set has entries

  Service --> Controller : generate report

  Controller --> UI : show report

  UI --> Admin : display charts/data

end


alt Admin applies filters

  Admin -> UI : set filters

  UI -> Controller : get filtered stats
  Controller -> Service : fetch stats (filters)

  Service -> Repo : get filtered data

  Repo --> Service : filtered data set

  Service --> Controller : generate filtered report

  Controller --> UI : update report

  UI --> Admin : display updated charts/data

end

@enduml
```



Search Car (Renter)

@startuml

```
actor Renter

participant UI

participant SearchController as Controller

participant SearchService as Service

participant CarRepository as Repo


Renter -> UI : open search screen

UI -> Controller : request search form

Controller --> UI : display form


Renter -> UI : enter criteria (location, dates, price range)

UI -> Controller : submit search criteria

Controller -> Service : validate criteria


alt criteria valid

    Service -> Repo : find available cars(criteria)

    Repo --> Service : available cars list


    alt cars found

        Service --> Controller : return cars list

        Controller --> UI : display available cars

        UI --> Renter : show list

    else no cars found

        Service --> Controller : no cars found

        Controller --> UI : show "No cars found, suggest change criteria"

        UI --> Renter : show message
    end

else criteria invalid

    Service --> Controller : invalid criteria error

    Controller --> UI : show warning and request correction

    UI --> Renter : show warning

end


Renter -> UI : optionally filter or select car

@enduml
```


Renter
UI
SearchController
SearchService
CarRepository
open search screen
request search form
display form
enter criteria (location, dates, price range)
submit search criteria
validate criteria
alt [criteria valid]
find available cars(criteria)
available cars list
alt [cars found]
return cars list
display available cars
show list
[no cars found]
no cars found
show "No cars found, suggest change criteria"
show message
[criteria invalid]
invalid criteria error
show warning and request correction
show warning
optionally filter or select car