F28SD COURSEWORK 1 REPORT

Dubai Campus

João Francisco Bianchi Labriola · H00411696 · jsb2000@hw.ac.uk

An Exercise in Designing a Software-Based System: From requirements through to design-level models and scenario test cases

Table of contents

Introduction

This report outlines a range of requirements and design models for the software-based system Vehicle Hire System (VHS). Its purpose is to provide Easy Rentals LTD, a vehicle rental business, with a robust system that handles data, processing and management of vehicle rentals in each of their sites.

I | Assumptions and Expectations

Assumptions

1. A stable and secure connection between the system and external systems, validateMyDVL, validateMyCC and rentalArchive is available.
2. The system takes robust security measures to protect sensitive customer and EASYRENTALS LTD data.
3. The system only shows vehicles that are available for rent.
4. Users, both customers and garage technicians, may supply erroneous data.
5. Only garage technicians will have access to touchscreen in inspection areas
6. Customers can only rent one car at a time.
7. Rental records will only be archived if completed.
8. The system is capable of charging credit cards.

Assumption 1 must be true before the ‘Create Rental’ use case can be triggered. The system relies on assumption 3 being true so customers do not create rentals with unavailable vehicles. Assumption 4 is dealt with by extensions use case ‘EditCustomerDetails' and ‘EditRentalRecord’ as well as ‘’ form validation and clear instructions during use cases on the VHS. The system relies on assumption 5 to ensure that only technicians will complete the return of rentals, as it requires the necessary inspection. Use case ‘ReturnRental’ finds the rental record it requires by matching customer ID and vehicle registration number, therefore relying on assumption 6 to be true.

Expectations

1. The touchscreen and UI provide and friendly interface for users of different technology literacies to be able to use the systems with ease.
2. External systems validateMyDVL and validateMyCC provide accurate and valid data on whether a customer's driver's license and credit card are valid.

3. The system will store and maintain up-to-date and accurate data on vehicles from its respective site.
4. The system will validate DVL and CC before completing rental process, even for returning customers.

Expectation 1 speaks to the experience tied to the system rather than its robustness. Expectation 2 describes a property of a system that the VHS must rely upon, VHS trusts the external systems' validation of DVLs and CCs. The system relies on expectation 3, which assumes is ensured by systems administrators. Expectation 4 must be true to avoid any returning customers that had their data saved from renting a vehicle with a now an invalid DVL and/or CC.

II | Functional and Non-Functional Requirements

ID	Description	Priority
FR1	The VHS shall display only vehicles available during desired dates for selection	M
FR2	The VHS should clearly label minimum age and license category necessary to rent each vehicle displayed	S
FR3	The VHS shall verify unique customer ID is valid	M
FR4	If the user's ID is invalid VHS shall warn the user and provide ID recovery through email option	M
FR5	If customer is new, then VHS should allow user to register	M
FR6	The VHS should provide users with the option to save their personal information and credit card details	S
FR7	The VHS shall verify that user's age is higher than or equals to selected vehicle's minimum age requirement	M

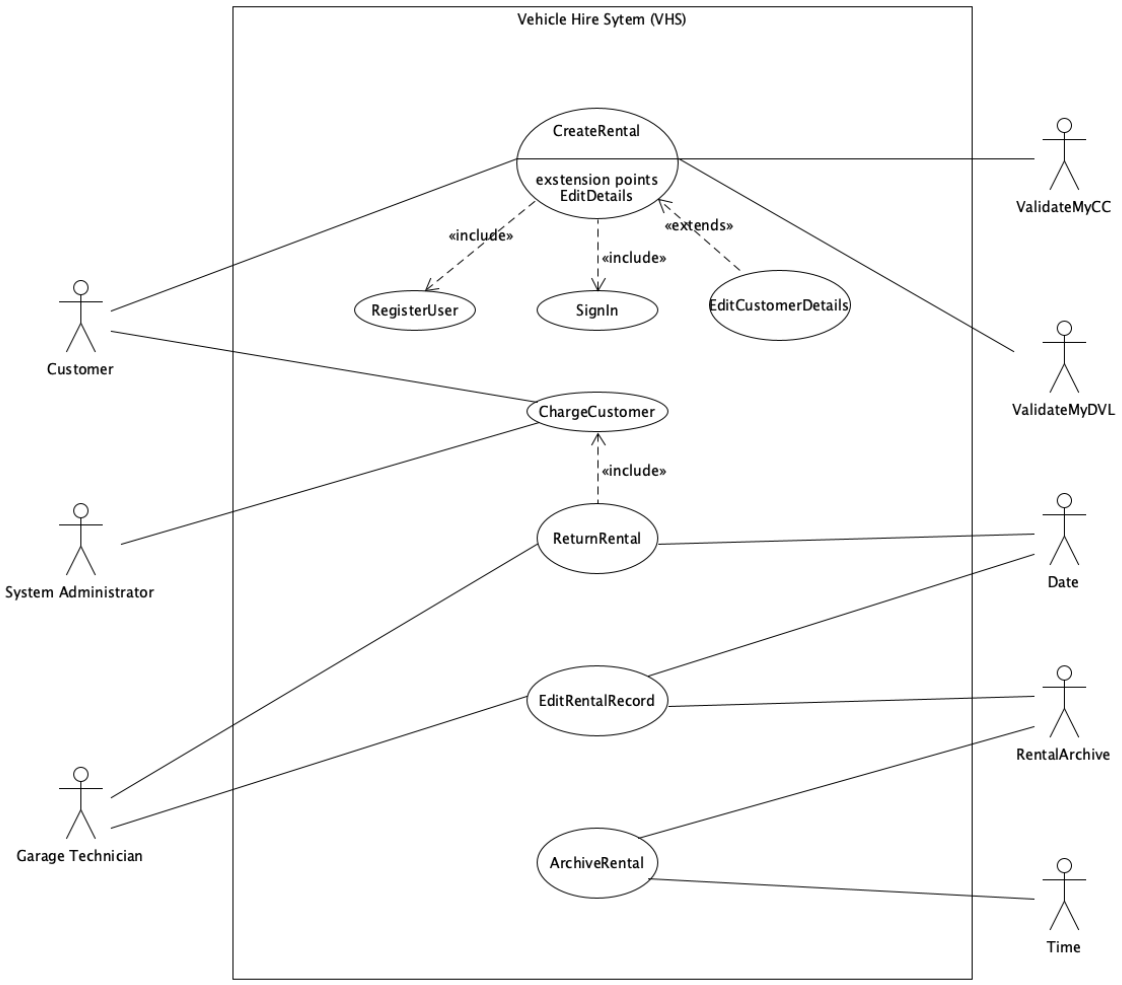
FR8	The VHS shall verify that user’s driving license is the same category selected vehicle's driving license category	M
FR9	The VHS shall verify that the user’s driving license matches the registered name and address	M
FR10	The VHS shall verify that the user’s driving license is valid by, using ValidateMyDVL	M
FR11	The VHS shall verify that the user’s credit card is valid, using ValidateMyCC	M
FR12	The VHS shall provide the user with the option to edit their details	M
FR13	The VHS shall create and store rental record including reference to customer and vehicle, start-date and end-date	M
FR14	The VHS shall send a summary of the rental record to the customer	M
FR15	The VHS shall retrieve rental record that matches registration number AND customer ID given by user	M
FR16	The VHS must calculate the rental charge based on days rented multiplied by daily fee	M
FR17	The VHS should store the rental fee to a local variable named total charge	S
FR18	If fuel level provided by the user is less than 100%, then the VHS shall calculate the fuel charge, add it to the rental total charge variable, and update the rental record to include fuel charge	M
FR19	If the user reports damage to the vehicle, then the VHS shall request a repair charge, add it to rental total charge and update rental record to include damage description and repair charge	M
FR20	The VHS shall update the vehicle’s record, milage and status, according to corresponding input from garage technician	M

FR21	The VHS shall charge the customer's registered credit card the total amount upon completion of rental return	M
FR22	If the customer opted to NOT have their credit card details saved, then the VHS shall delete the customer's credit card information	M
FR23	If the customer opted to NOT have their information saved, then the VHS shall delete all the customer's data	M
FR24	The VHS shall create a secure and stable connection to RentalArchive	M
FR25	The VHS should have access to data stored internally	S
FR26	If a record is completed then the VHS shall select only the necessary data to transfer/store to/in RentalArchive (limited to customer's name, address, vehicle registration, rental start/end dates)	M
FR27	The VHS shall delete all data corresponding to rental record transferred	M
FR28	The VHS should transfer and delete all completed rental records stored internally daily	S
FR29	The VHS should save changes made by user to its internal records	S
FR30	The VHS shall fetch and delete rental data from RentalArchive that matches vehicle registration number and rental return date	M
FR31	If the return date equals current date, then the VHS shall retrieve rental data from internal records that matches vehicle registration number and rental return date	M
FR32	The VHS shall shave all changes made to rental record and store it locally in VHS	M
FR33	If customer's credit card is declined, then the VHS shall update rental record and notify system administrator	M

FR34	The VHS shall create a unique customer ID for new customers	M
NFR1	The VHS should process requests quickly to ensure efficiency	
NFR2	The VHS should be intuitive and accessible to users of all technology literacy levels.	
NFR3	The VHS should comply with legal and regulatory standards related to vehicle rentals and data protection.	
NFR4	The VHS should be capable of managing a large number of operations and vehicle inventory.	

III | Use Case Diagram

The use case diagram represents the interactions between the actors, Customers, System Administrators, Garage Technicians, Time, Date, external systems (validateMyDVL, validateMyCC and rentalArchive), and the Vehicle Hire System (VHS). This diagram serves as a high-level overview of the VHS and its use cases.



IV | Specifications

Use Case: CreateRental

ID: *1*
Goal: The customer's driver's license (DVL) and credit card (CC) are validated, and a rental is created
Primary actor: *Customer*
Secondary actor(s): *validateMyDVL and validateMyCC*
Preconditions: *1. Touchscreen is idle.* *2. A stable and secure connection between the system and external systems, validateMyDVL and validateMyCC is available*
Postconditions: *1. The VHS validated DVL and CC successfully* *2. The VHS updated rented vehicle status to unavailable-for-rental* *3. The VHS created a rental record storing a reference to the customer, a reference to the vehicle, start/end dates for the rental period*
Main flow: *1. VHS requests rental dates (start-end)* *2. User provides rental dates* *3. VHS checks internal record of vehicles and provides user with vehicle choices available (car-economy, car-premium, van-standard and van-large)* *4. User selects desired vehicle* *5. include(RegisterUser) // customer is more likely to be new* *6. User signs in successfully*

7. VHS checks if selected customer minimum age is more than or equal to 22

8. VHS checks if user's driving license category matches vehicle's required license category

9. VHS validates user's DVL using ValidateMyDVL

10. VHS indicates successful validation of DVL

11. VHS validates user's CC using ValidateMyCC

12. VHS indicates successful validation of CC

13. VHS requires user confirmation

14. User confirms booking

15. VHS creates and stores rental record including reference to customer, vehicle, start-date and end-date.

16. VHS updates vehicle's status to unavailable-for-rental

17. VHS sends rental details to user's e-mail

18. VHS indicates successful creation of rental and displays instructions on how to collect vehicle

Alternative flows:

<5a> <User chooses to sign in rather than register>

1. include(SignIn)

2. return to step 6 main flow

<7a> <User is not of appropriate age >

1. VHS indicates user is not of appropriate age

Extension point: EditDetails

2. Return to step seven main flow

<8a> <User's driving licence category does not match required license category for selected vehicle>

1. VHS indicates user's driving licence does not match required license for selected vehicle

Extension point: EditCustomerDetails

2. VHS indicates vehicles available for user

3. User selects vehicle

4. Return to step eight main flow

<10a> <User's DVL does not match customer's details>

1. VHS indicates DVL does not match customer's details (name or address)

2. VHS prompts user to edit DVL

Extension point: EditDetails

3. Return to step to step nine main flow

<10b> <User's DVL is not valid>

1. VHS indicates DVL is not valid

2. VHS prompts user to enter valid DVL

Extension point: EditDetails

3. Return to step nine main flow

<11a> <User signed in with registered details, but does not have credit card details saved>

1. VHS provides form for user fill. (Including credit card number, expiry date and security code). Form has form validation ensuring data type are integer, DATE, integer(3), accordingly.

2. User enters information

3. User is prompted with the option to save credit card details.

4. User selects their choice

5. Users presses continue

6. VHS saves card details to customer account

7. Return to step eleven main flow

<12a> <User's CC is not valid>

1. VHS indicates CC is not valid

2. VHS prompts user to enter valid CC

Extension point: EditDetails

3. Return to step nine main flow

Use Case: ReturnRental

ID: 2
Goal: To complete return of rented vehicle, make the necessary updates to records and complete relevant charges to customer
Primary actor: *Garage technician*
Secondary actor(s): *None*
Preconditions: *1. Touchscreen is idle.* *2. A stable and secure connection to complete payment charges to customer* *3. Garage technician has completed vehicle inspection*
Postconditions: *1. The VHS updated customer, vehicle and rental records* *2. The VHS successfully charged customer for the necessary payments*
Main flow: *1. User starts rental return process* *2. VHS requires user to insert vehicle registration number and customer ID* *3. User enters vehicle registration number and customer ID* *4. VHS retrieves rental record that matches vehicle registration number and customer ID* *5. VHS requires user to provide fuel level* *6.User provides fuel level* *7. VHS requires user to select from two options (DAMAGE/NO DAMAGE)* *8. User selects no damage*

9. VHS sets damage repair charge to 0

10. VHS sets damage description to 'none'

11. VHS requests milage of vehicle

11. User provides vehicle milage

12. VHS retrieves vehicle record using registration number

13. VHS updates vehicle milage

14. VHS request user to provide vehicle status from options: available-for-rental, unavailable-for-rental and out-of-service

15. VHS updates vehicle status

16. User is prompted to 'complete & save' rental return

17. User completes rental return

18. VHS retrieves customer's details

19. Include(ChargeCustomer)

20. VHS sets rental record to completed

21. VHS saves rental records

22. VHS sends a summary of rental record and charges to customer

23. VHS displays successful message on screen

Alternative flows:

<8a> <User selects DAMAGE>

1. *VHS requests a textual description of the damage*
2. *User provides description of damage*
3. *VHS sets damage description to input from user*
4. *VHS request a repair charge amount*

5. *User provides charge amount*

6. *VHS sets repair charge to input from user*

7. *Return to step eleven main flow*

<23a> <Customer opted to not save credit card details>

1. *VHS deletes customer's credit card data*

2. *Return to step twenty-three main flow*

<23b> <Customer opted to NOT save any personal details>

1. *VHS deletes all customer's data*

2. *Return to step twenty-three main flow*

Use Case: ArchiveRental

ID: 3
Goal: To archive rental information from current day to external system RentalArchive
Primary actor: *Time*
Secondary actor(s): *RentalArchive*
Preconditions: *1. There is a secure and stable connection between VHS and RentalArchive* *2. It is past 23:59*
Postconditions: *1. All relevant information of completed rental records are transferred to RentalArchive* *2. All corresponding records are deleted from VHS*

Main flow: *1. VHS retrieves its rental records stored* *2. VHS iterates through rental records* *3. VHS checks if rental is completed* *4. VHS creates archival record containing the customer's name, address, vehicle registration number and rental start/end dates.* *5. System transfers record to RentalArchive* *6. System deletes all data related to corresponding rental record* *System repeats steps 3 to 6 until there are no more rental records stored*
Alternative flows: *<3a> <Rental record is not completed>* *1. VHS skips to next record* *2. return to step three main flow*

Use Case: RegisterUser

ID: 4
Goal: Register a new customer using VHS
Primary actor: *Customer*
Secondary actor(s): *None*
Preconditions: *1. The VHS is functioning properly*
Postconditions: *1. The prospective user is registered with VHS*

2. The user is issued a unique customer ID
Main flow: *1. VHS provides form for user to fill. (Including First Name, Last Name, Date of Birth, address, mobile number, e-mail address, driving license number, driving license category and driving license expiry date). Form has form validation ensuring data type are text, text, DATE, text, alphanumeric code, text, 16-character alphanumeric code, ENUM('Category B', 'Category C1', 'Category C'), DATE, accordingly.* *2. VHS provides form for user fill. (Including credit card number, expiry date and security code). Form has form validation ensuring data type are integer, DATE, integer(3), accordingly.* *3. User enters information* *4. User is prompted with the option to save personal details and option to save credit card details.* *5. User selects choices* *6. User presses register* *7. VHS stores user data* *8. VHS provides user with unique customer ID* *9. VHS signs user in using registered details*
Alternative flows: *None*

Use Case: SignIn

ID: 5
Goal: Sign user in using unique customer ID

Primary actor: *Customer*
Secondary actor(s): *None*
Preconditions: *1. The user has previously registered using the VHS*
Postconditions: *1. User is signed into VHS*
Main flow: *1. VHS requests unique customer ID* *2. User provides unique customer ID* *3. VHS retrieves customer information from local storage matching unique customer ID*
Alternative flows: *<3a> <User provides invalid customer ID>* *1. VHS indicates ID is incorrect* *2. System prompts user to recover ID using email* *3. User enters registered email* *4. VHS sends unique customer ID to registered email* *5. Return to step two main flow*

Use Case: ChargeCustomer

ID: 6
Goal: Charge customer for rental services charge
Primary actor: *Customer*

Secondary actor(s): *None*
Preconditions: *1. Customer has outstanding payment*
Postconditions: *1. VHS charges customer for rentals services*
Main flow: *1. VHS gets total charge from rental record* *2. VHS charges total price of rental to customer's credit card*
Alternative flows: *<2a> <Credit card declined>* *1. VHS updates rental record to unpaid* *2. VHS notifies system administrator of unpaid charges* *Use case terminated*

Extension Use Case: EditCustomerDetails

ID: 7
Goal: Provide user access to their registered details to make changes and save new details
Primary actor: *Customer*
Secondary actor(s): *None*
Segment Preconditions: *User has registered in the VHS*
Segment Postconditions: *User successfully saves their details*

Segment flow: *EditDetails* *1. VHS displays customer details* *2. User makes changes to details* *3. User saves new details* *4. VHS updates customer record*

Use Case: EditRentalRecord

ID: 8
Goal: To allow garage technicians to make changes to an incorrect rental record return
Primary actor: *Garage technician*
Secondary actor(s): *Time*
Preconditions: *1. There is a secure and stable connection between VHS and RentalArchive*
Postconditions: *1. Rental record changes are saved successfully*
Main flow: *1. VHS requests vehicle registration number* *2. User provides vehicle registration number* *3. VHS requires rental return date* *4. User provides rental return date* *5. VHS fetches and deletes rental record matching vehicle registration number and return date*

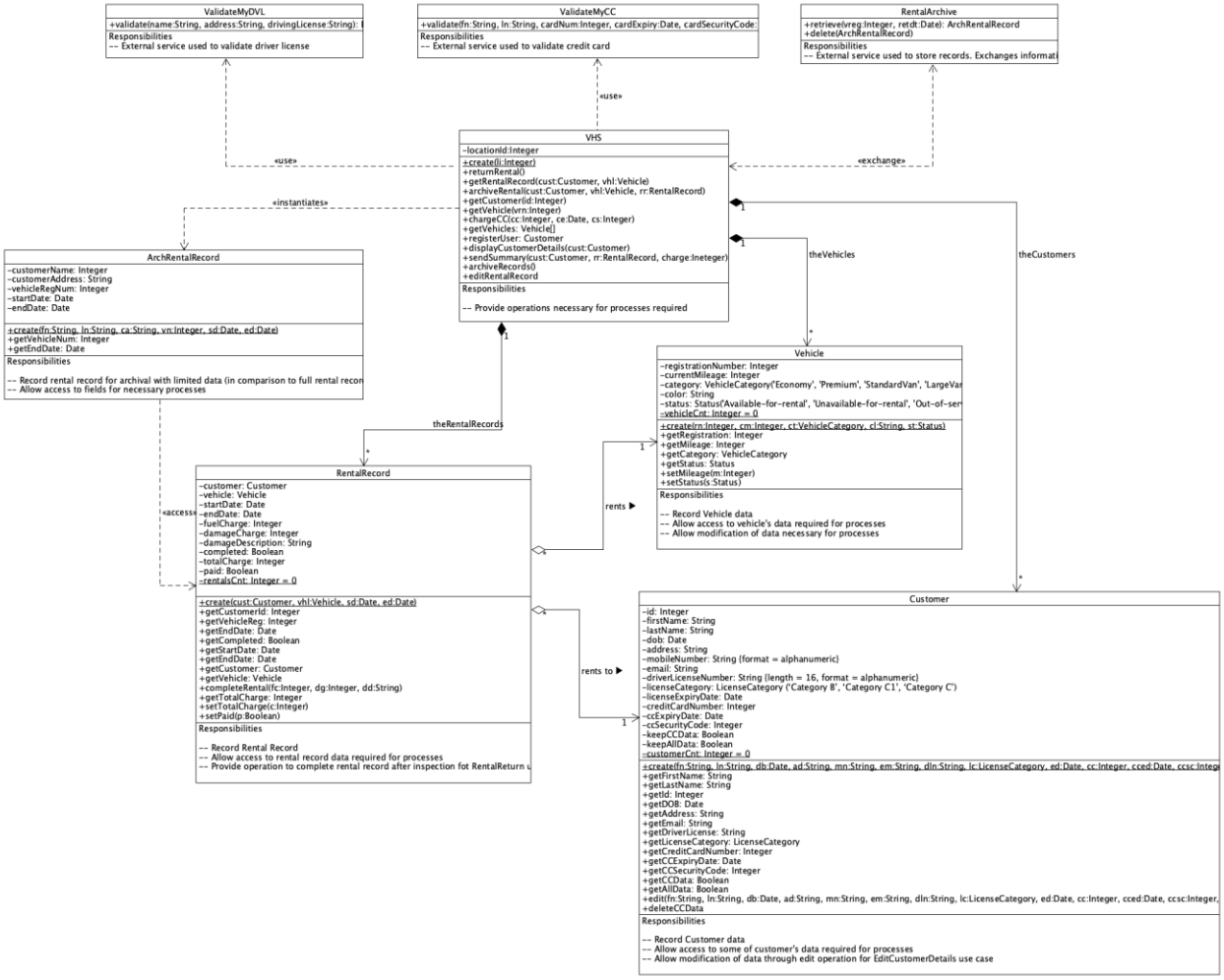
6. VHS provides rental record details *7. User selects field they wish to change* *8. User makes necessary changes* *9. User selects to save changes made* *10. VHS saves changes made and stores in VHS // will be transferred to RentalArchive at end of day*
Alternative flows: *<4a> <Return date is equals to current date>* *1. VHS retrieves rental record matching vehicle registration number and rental return date from internal records* *2. Return to step six main flow*

V | Traceability Matrix

	Use Cases							
ID	1	2	3	4	5	6	7	8
FR1	x							
FR2	x							
FR3					x			
FR4					x			
FR5	x							
FR6				x			x	
FR7	x							
FR8	x							
FR9	x							
FR10	x							
FR11	x							
FR12	x							
FR13	x	x						x
FR14	x	x						x
FR15		x						
FR16						x		

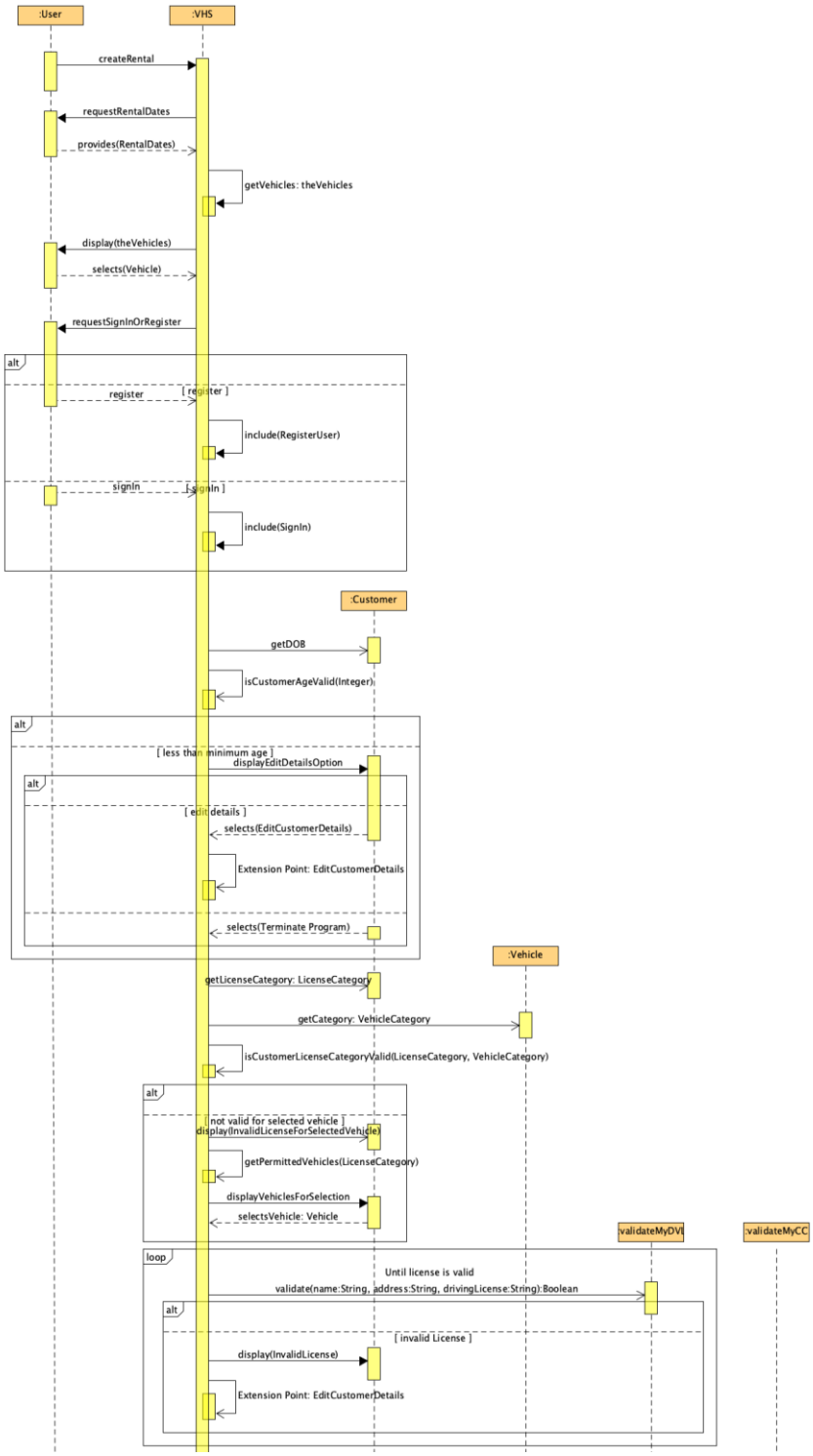
FR17						x		
FR18						x		
FR19						x		
FR20		x						x
FR21		x				x		
FR22		x						
FR23		x						
FR24			x					x
FR25	x	x	x	x	x	x	x	x
FR26			x					
FR27			x					
FR28			x					
FR29							x	x
FR30								x
FR31								x
FR32								x
FR33						x		
FR34				x				

VI | Class Diagram



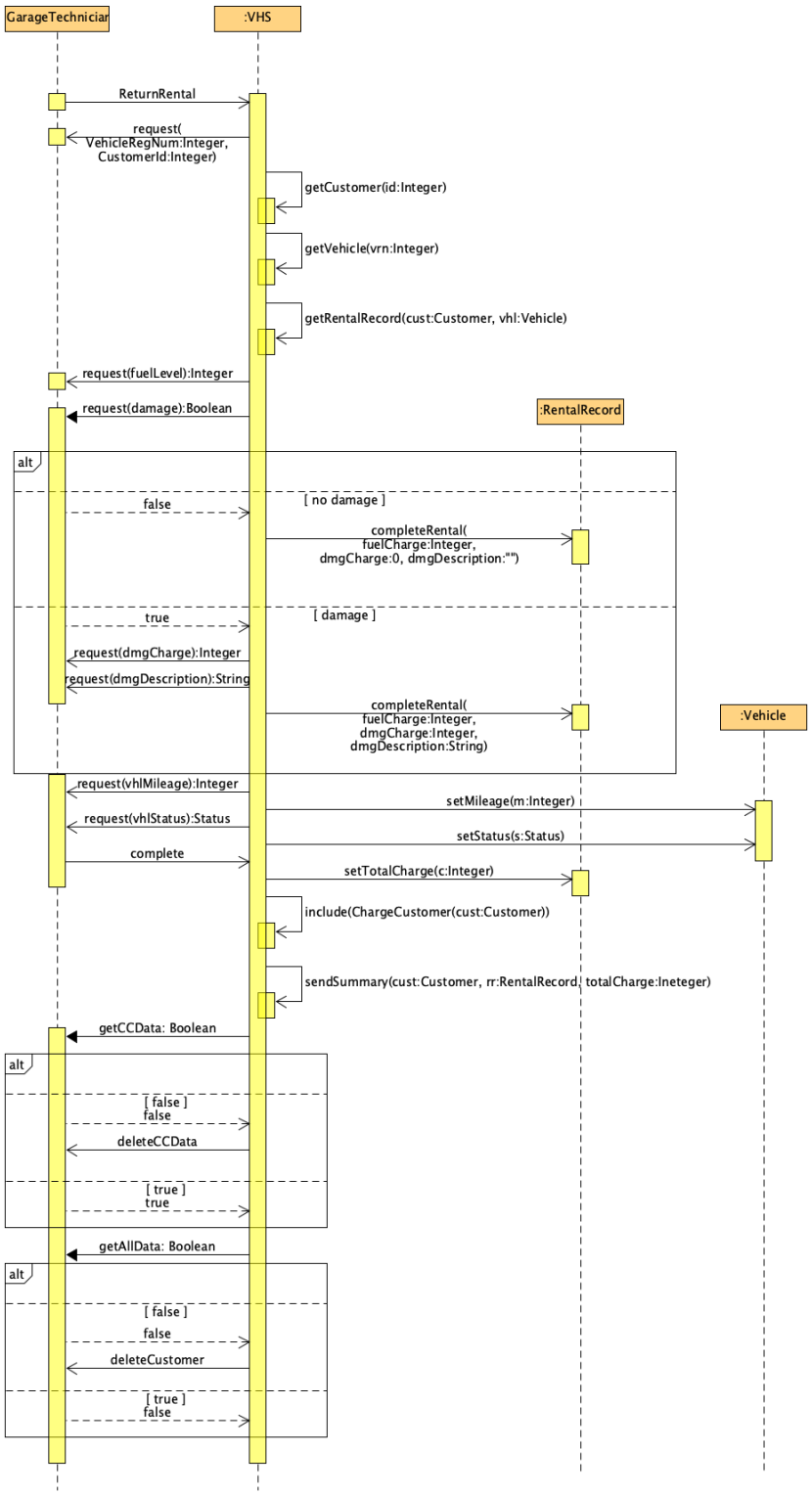
VII | Sequence Diagrams

CreateRental

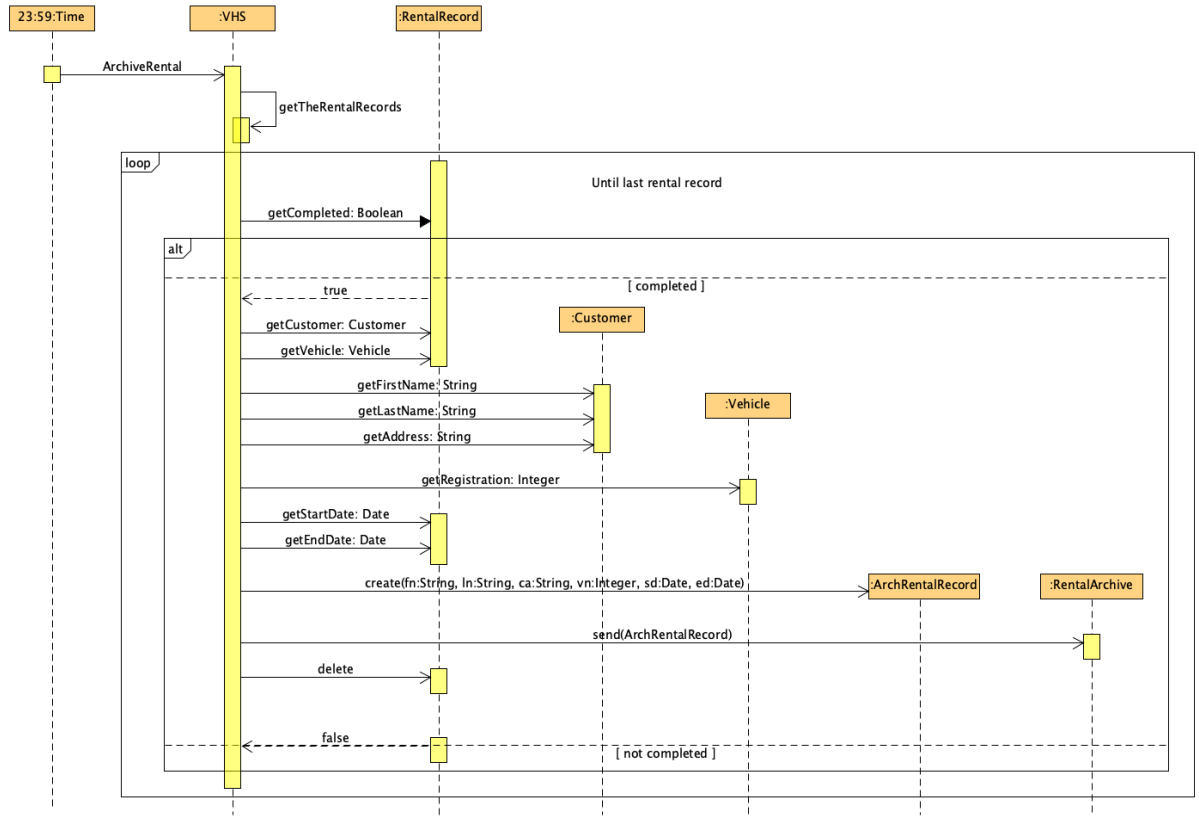



getCreditCardNumber: Integer
alt
[null]
null
Extension Point: EditCustomerDetails
getCreditCardNumber: Integer
loop
Until credit card is valid
validate(fn:String, ln:String, cardNum:Integer, cardExpiry:Date, cardSecurityCode:Integer): Boolen
alt
[invalid License]
display(InvalidCreditCard)
Extension Point: EditCustomerDetails
requestConfirmation
create(cust:Customer, vh:Vehicle, sd:Date, ed:Date)
:RentalRecord
getEmail: String
sendRentalSummaryToCustomerEmail
setStatus('Unavailable-for-rental')
indicateSuccessfulRental

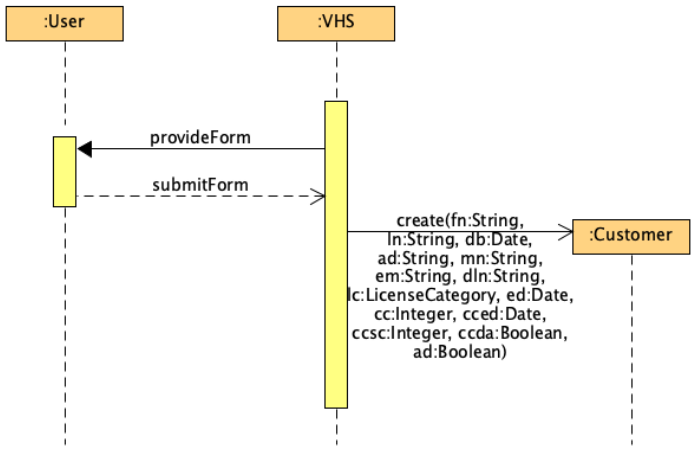
ReturnRental



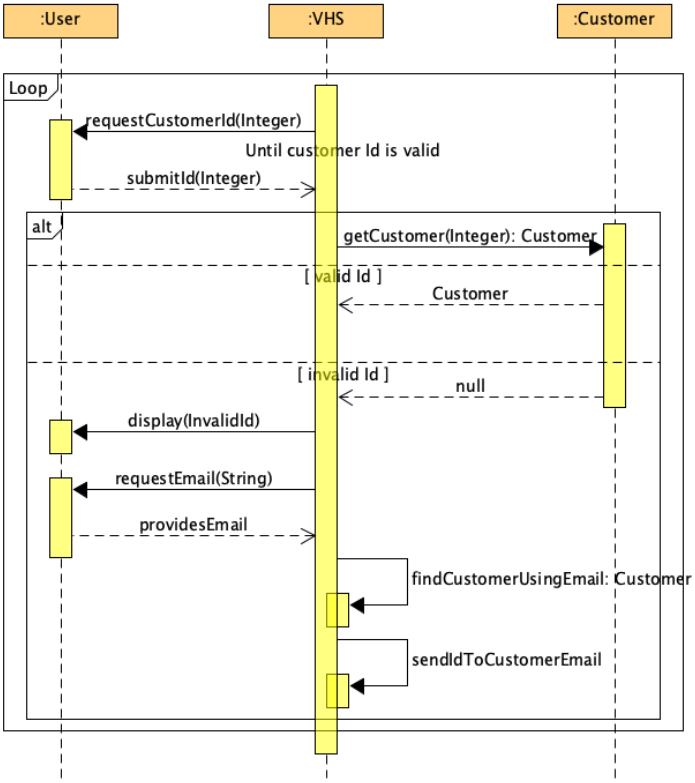
ArchiveRental



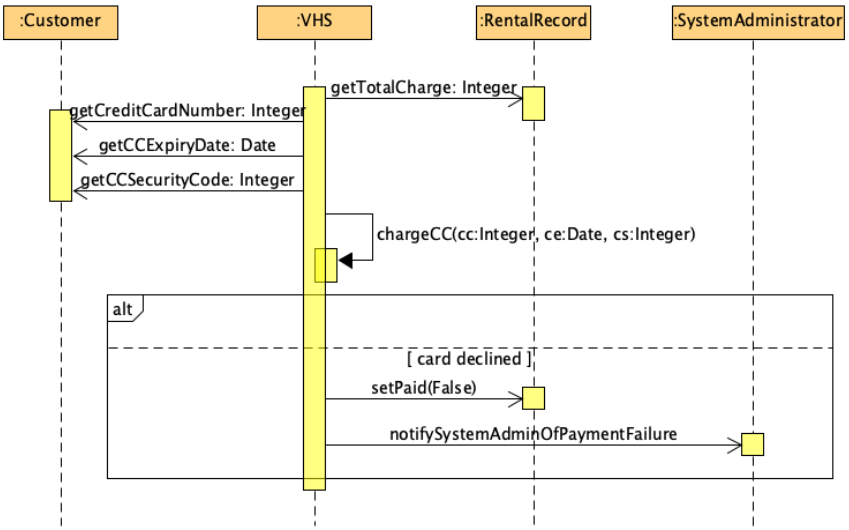
RegisterUser



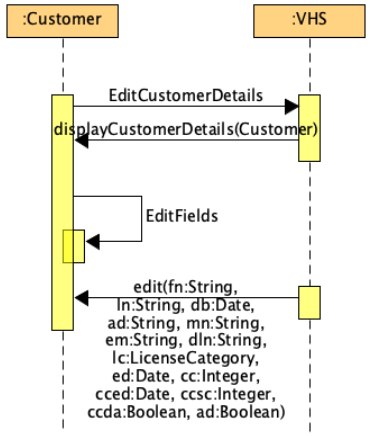
SignIn



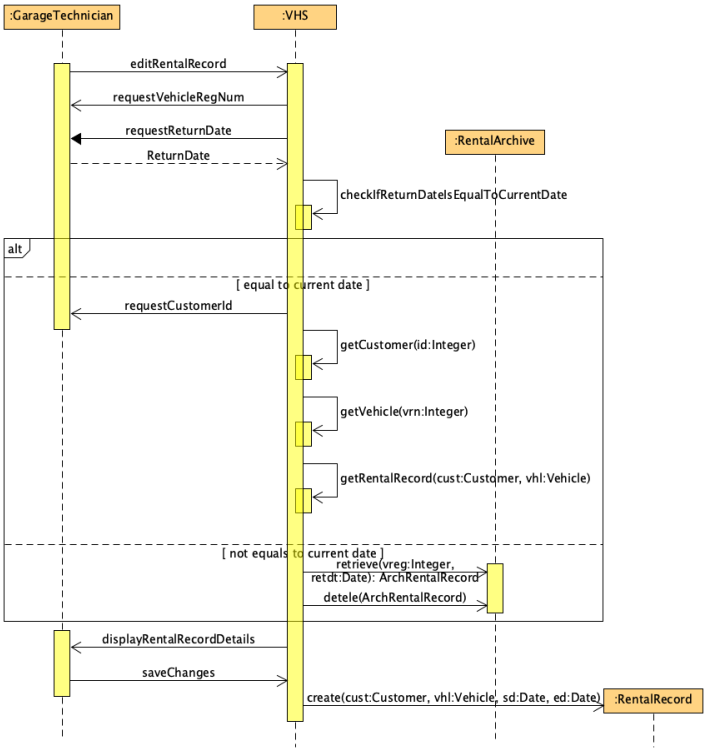
ChargeCustomer



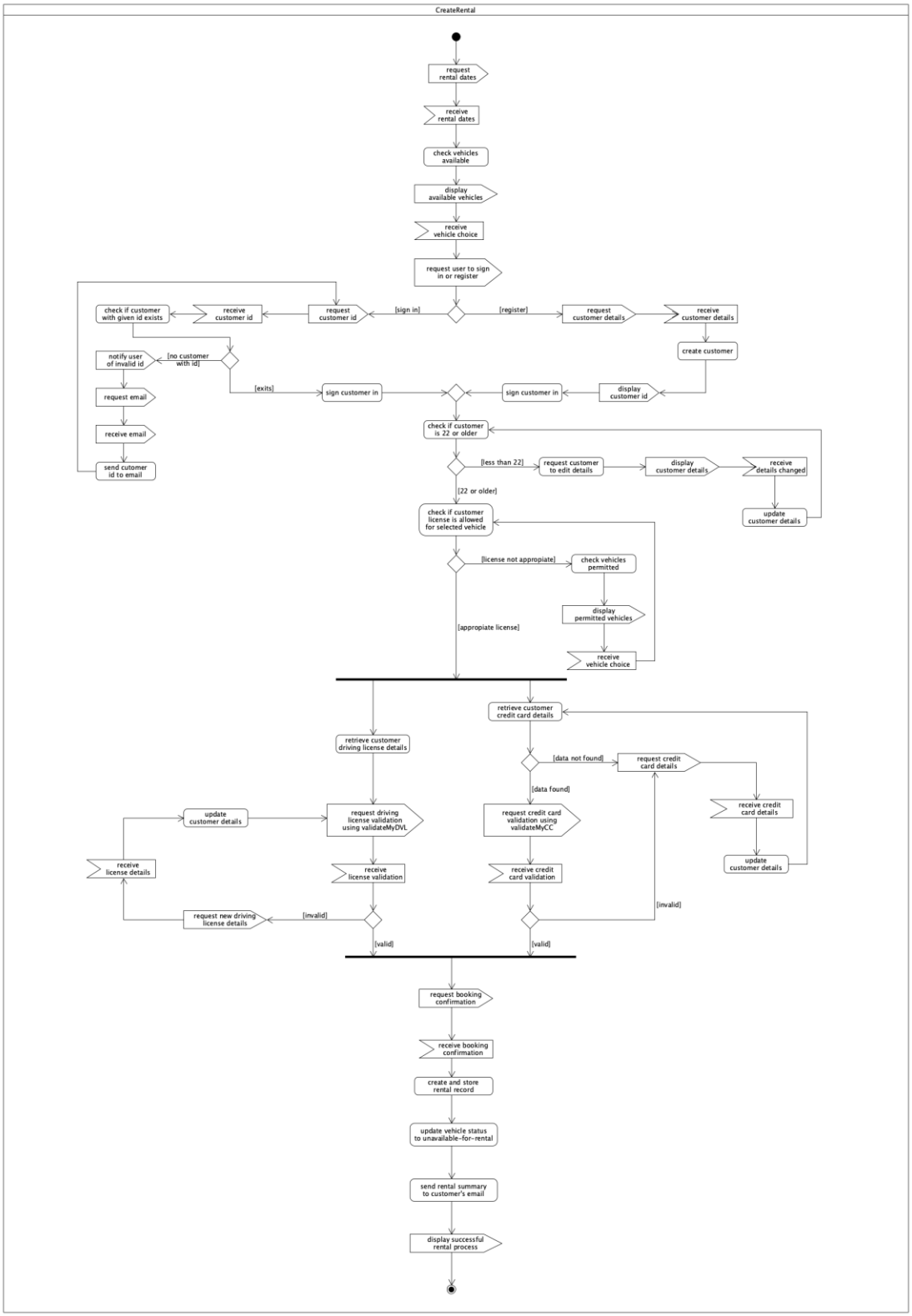
EditCustomerDetails

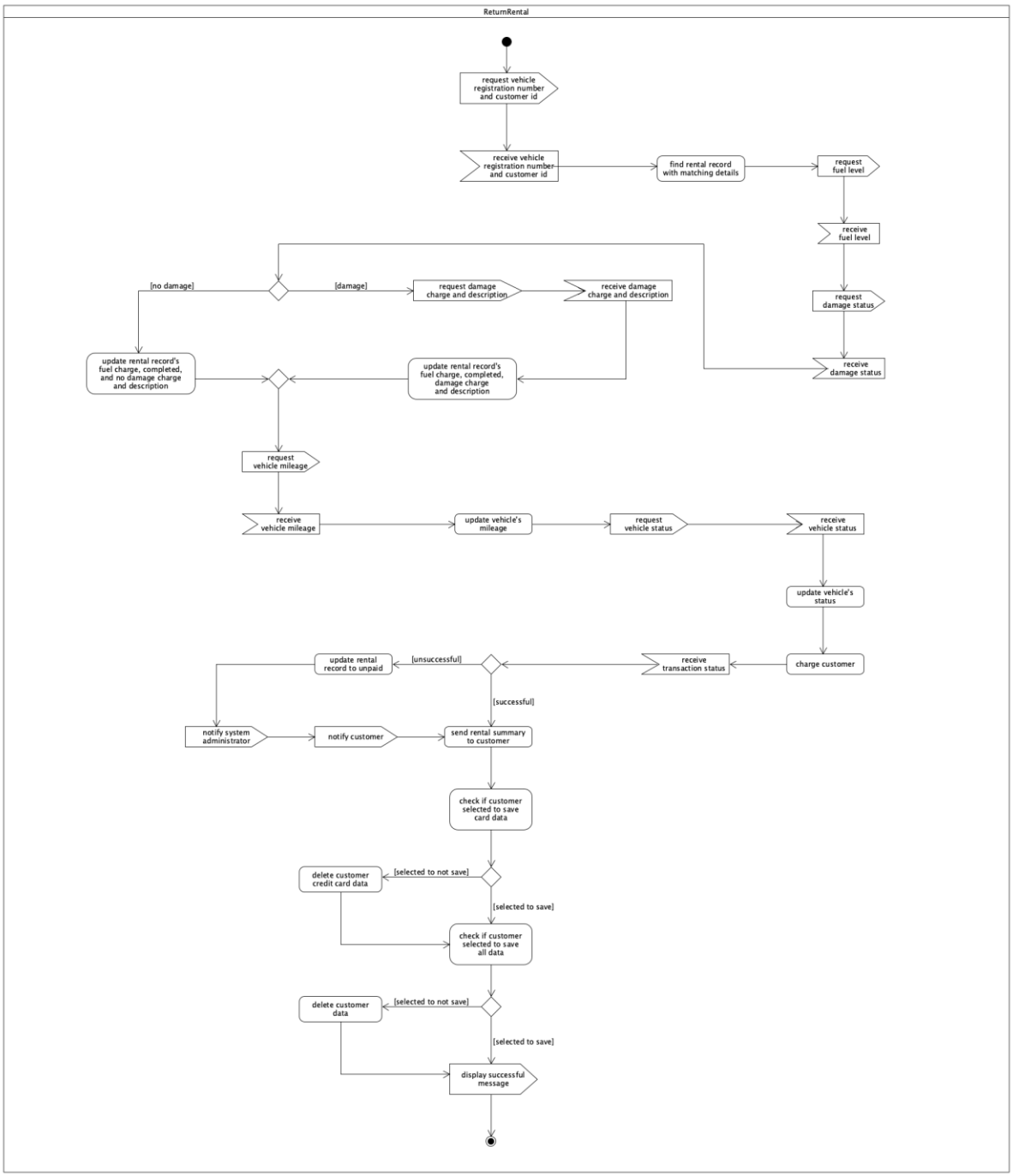


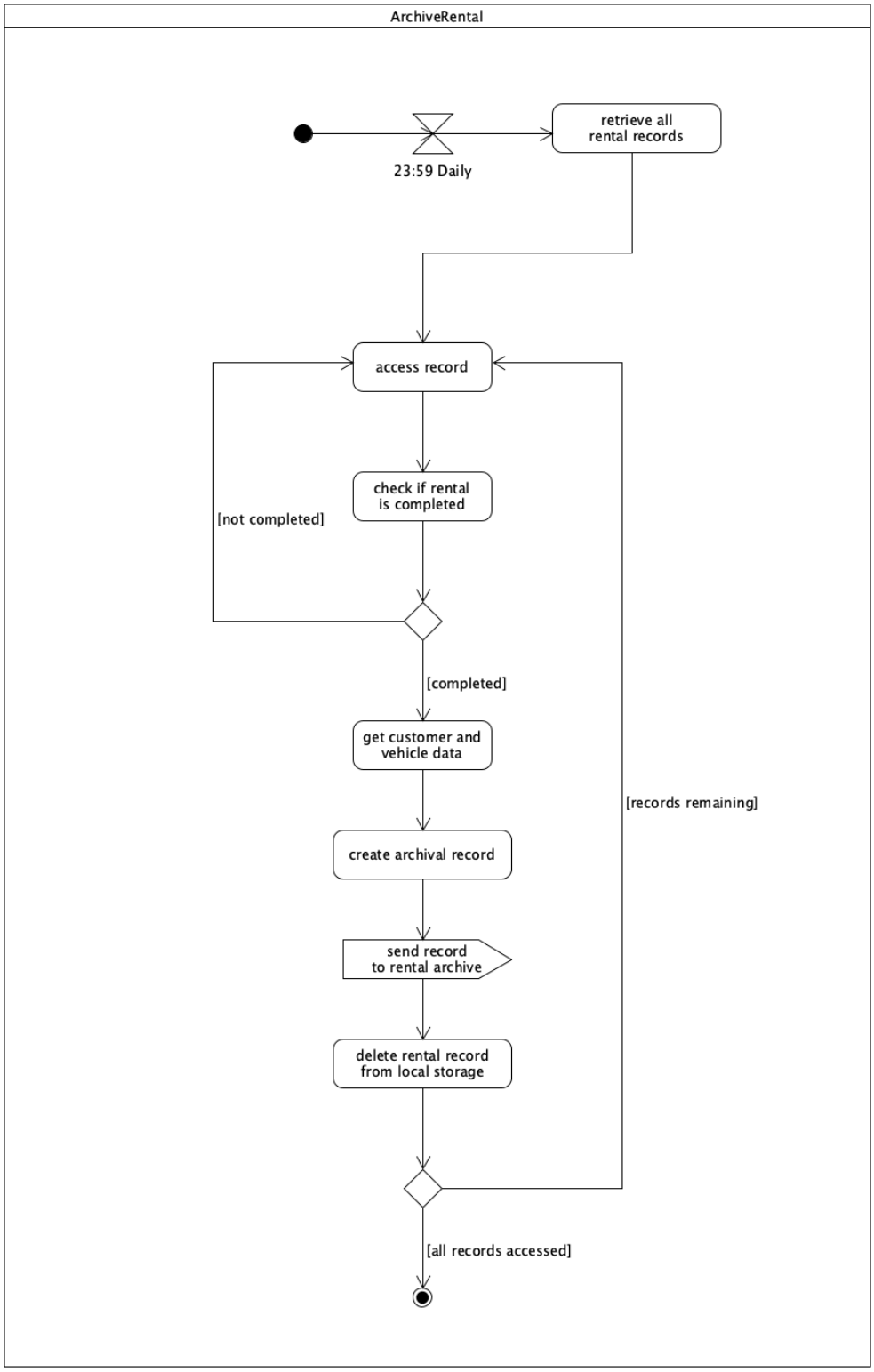
EditRentalRecord



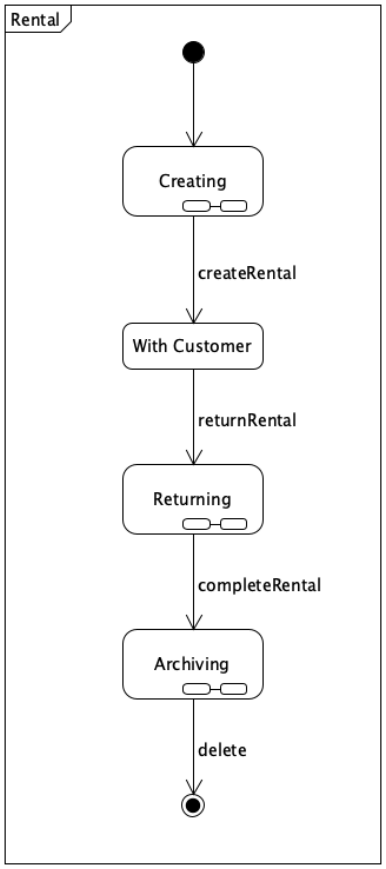
VIII | Activity Diagrams

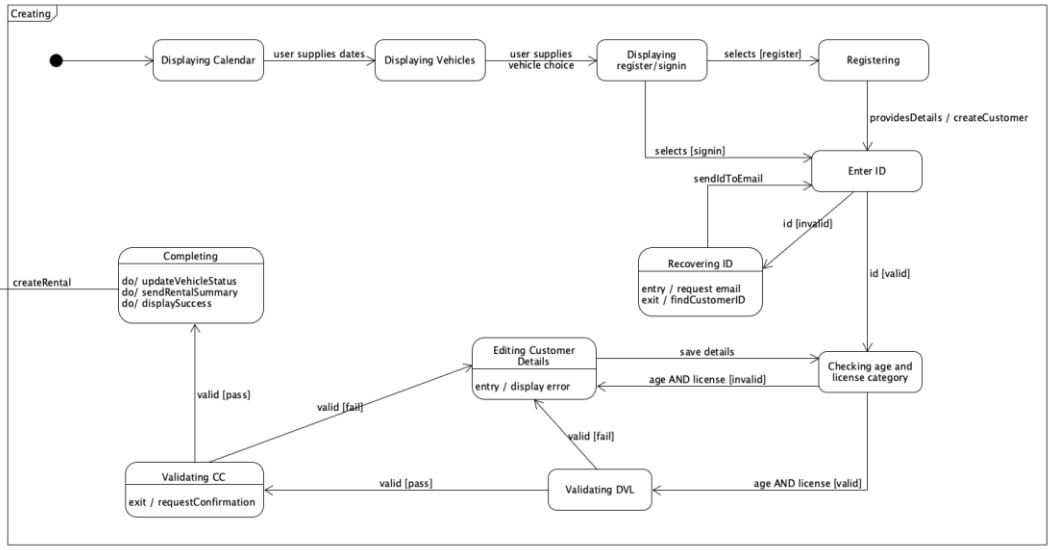

CreateRental
request rental dates
receive rental dates
check vehicles available
display available vehicles
receive vehicle choice
request user to sign in or register
check if customer with given id exists
receive customer id
request customer id
[sign in]
[register]
request customer details
receive customer details
create customer
notify user of invalid id
[no customer with id]
request email
receive email
send cutomer id to email
[exits]
sign customer in
sign customer in
display customer id
check if customer is 22 or older
[less than 22]
request customer to edit details
display customer details
receive details changed
update customer details
[22 or older]
check if customer license is allowed for selected vehicle
[license not appropiate]
check vehicles permitted
display permitted vehicles
receive vehicle choice
[appropiate license]
retrieve customer credit card details
retrieve customer driving license details
[data not found]
request credit card details
receive credit card details
update customer details
[data found]
request credit card validation using validateMyCC
receive credit card validation
[invalid]
[valid]
update customer details
request driving license validation using validateMyDVL
receive license validation
receive license details
request new driving license details
[invalid]
[valid]
request booking confirmation
receive booking confirmation
create and store rental record
update vehicle status to unavailable-for-rental
send rental summary to customer's email
display successful rental process

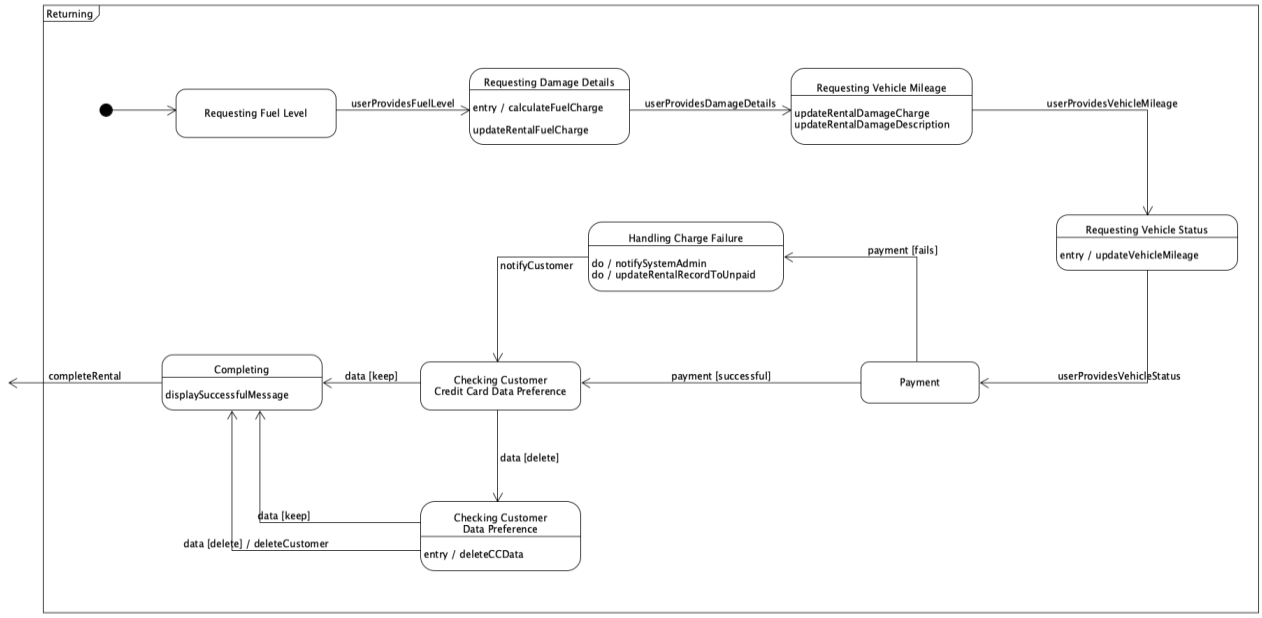

ReturnRental
request vehicle registration number and customer id
receive vehicle registration number and customer id
find rental record with matching details
request fuel level
receive fuel level
request damage status
receive damage status
[no damage]
[damage]
request damage charge and description
receive damage charge and description
update rental record's fuel charge, completed, and no damage charge and description
update rental record's fuel charge, completed, damage charge and description
request vehicle mileage
receive vehicle mileage
update vehicle's mileage
request vehicle status
receive vehicle status
update vehicle's status
charge customer
receive transaction status
[unsuccessful]
update rental record to unpaid
notify system administrator
notify customer
[successful]
send rental summary to customer
check if customer selected to save card data
[selected to not save]
delete customer credit card data
[selected to save]
check if customer selected to save all data
[selected to not save]
delete customer data
[selected to save]
display successful message

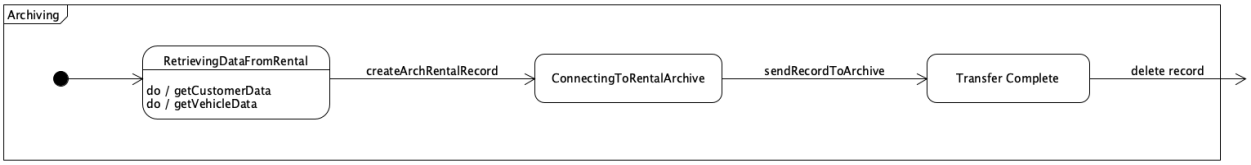

ArchiveRental
retrieve all rental records
23:59 Daily
access record
check if rental is completed
[not completed]
[completed]
get customer and vehicle data
create archival record
send record to rental archive
delete rental record from local storage
[records remaining]
[all records accessed]

IX | State Machine Diagram





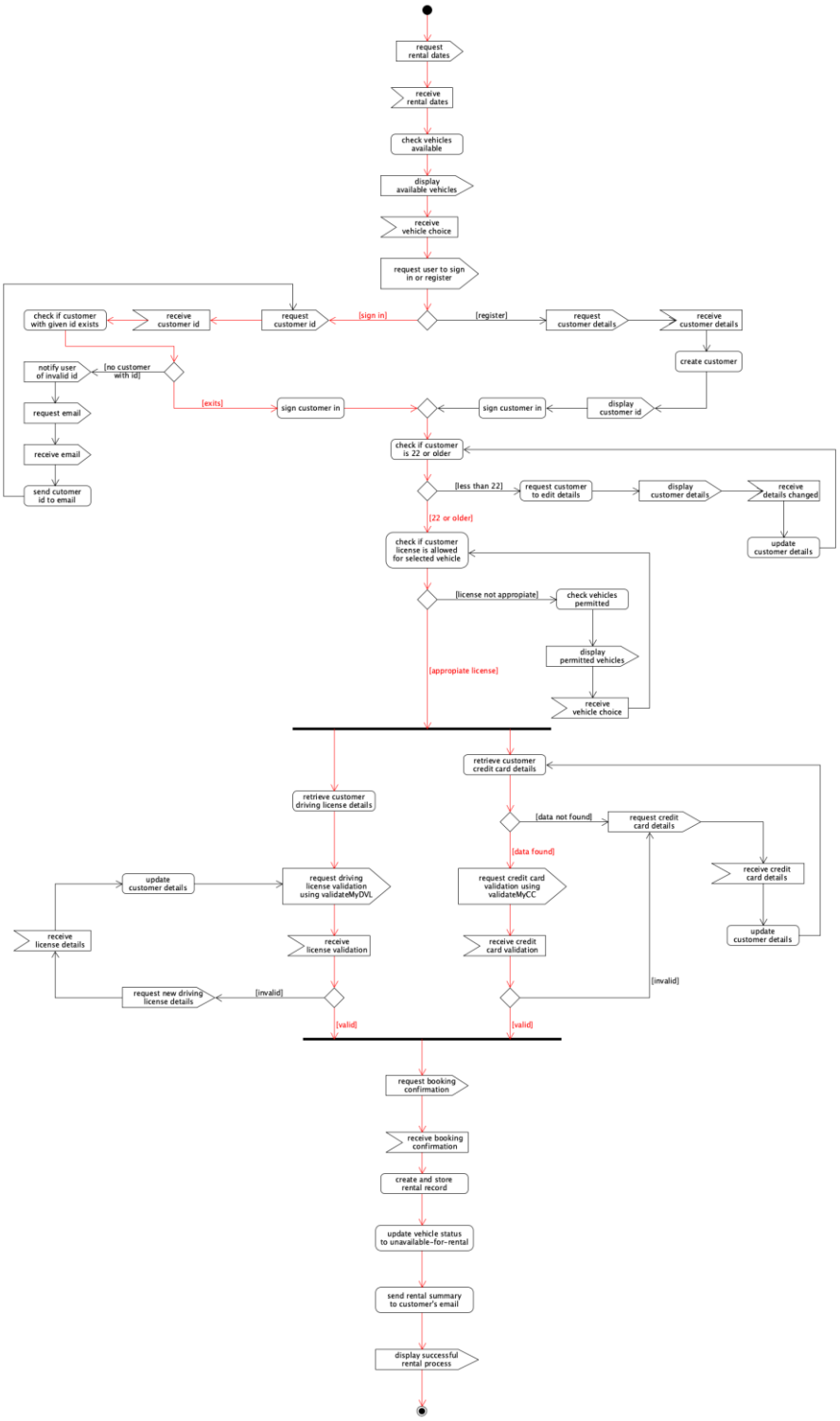




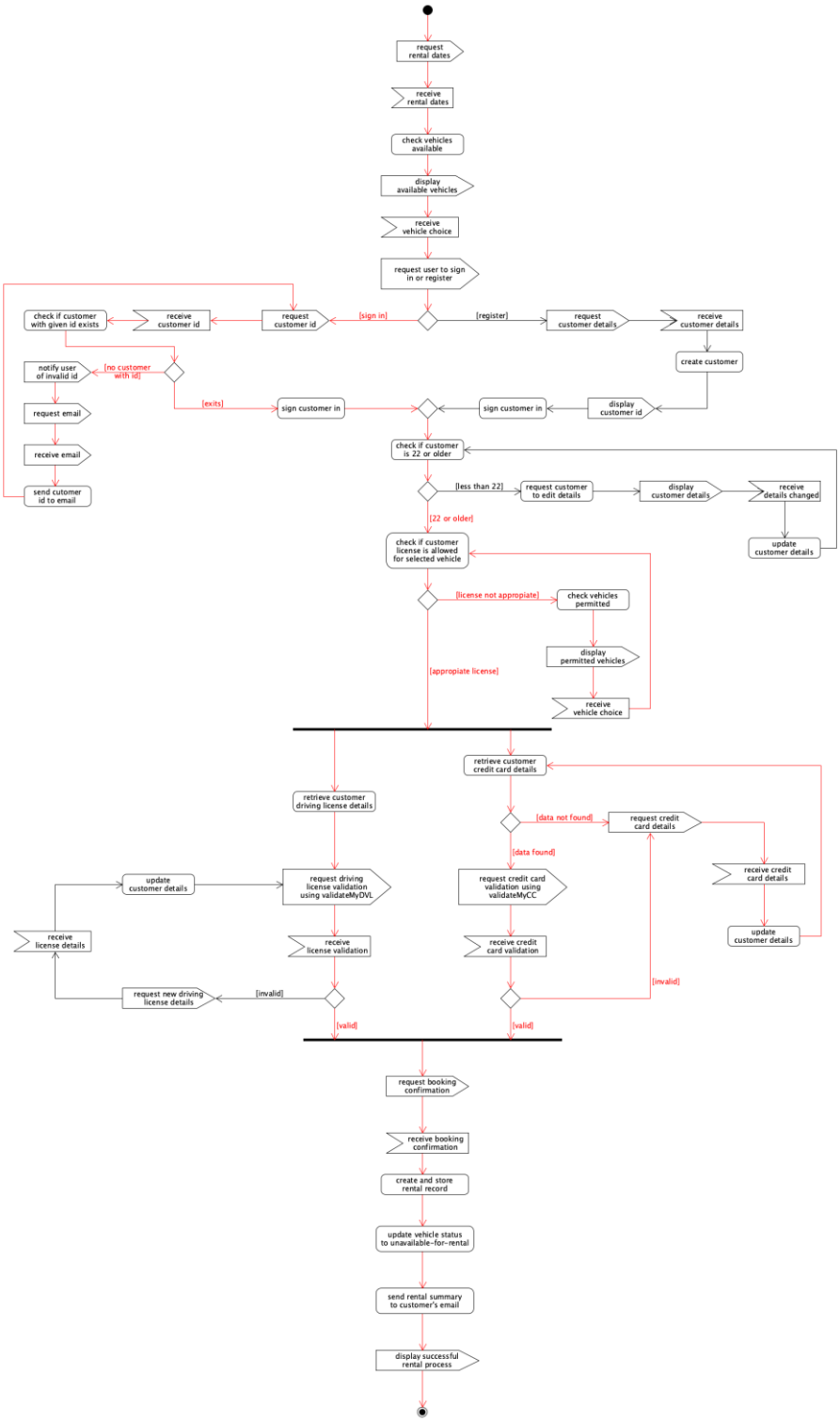
X | Scenario Test Cases

Three sets of scenario test cases were created, one for each Activity Diagram. Each contains 2-4 possible scenarios, it is important to note that each “activity” has multiple decision/conditional points, meaning there could be many different possible scenarios (i.e. CreateRental has 8 conditional points, this could result in hundreds of possible scenarios), thus the ones that are most likely to occur were demonstrated.

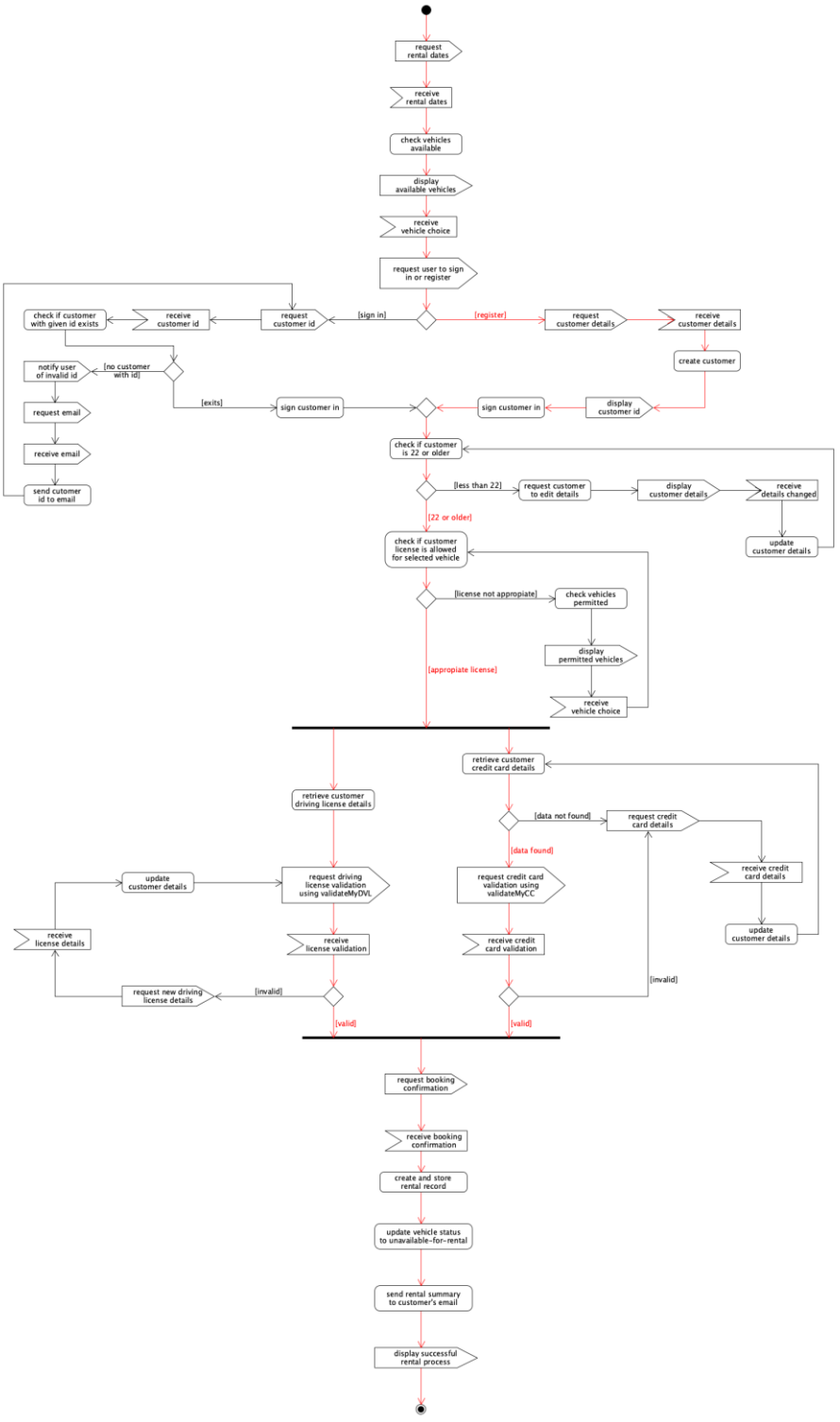
CreateRental


request rental dates
receive rental dates
check vehicles available
display available vehicles
receive vehicle choice
request user to sign in or register
[sign in]
[register]
request customer id
receive customer id
check if customer with given id exists
notify user of invalid id
[no customer with id]
request email
receive email
send cutomer id to email
[exits]
sign customer in
request customer details
receive customer details
create customer
display customer id
sign customer in
check if customer is 22 or older
[less than 22]
request customer to edit details
display customer details
receive details changed
update customer details
[22 or older]
check if customer license is allowed for selected vehicle
[license not appropiate]
check vehicles permitted
display permitted vehicles
receive vehicle choice
[appropiate license]
retrieve customer driving license details
update customer details
request driving license validation using validateMyDVL
receive license validation
receive license details
request new driving license details
[invalid]
[valid]
retrieve customer credit card details
[data not found]
request credit card details
receive credit card details
update customer details
[data found]
request credit card validation using validateMyCC
receive credit card validation
[invalid]
[valid]
request booking confirmation
receive booking confirmation
create and store rental record
update vehicle status to unavailable-for-rental
send rental summary to customer's email
display successful rental process

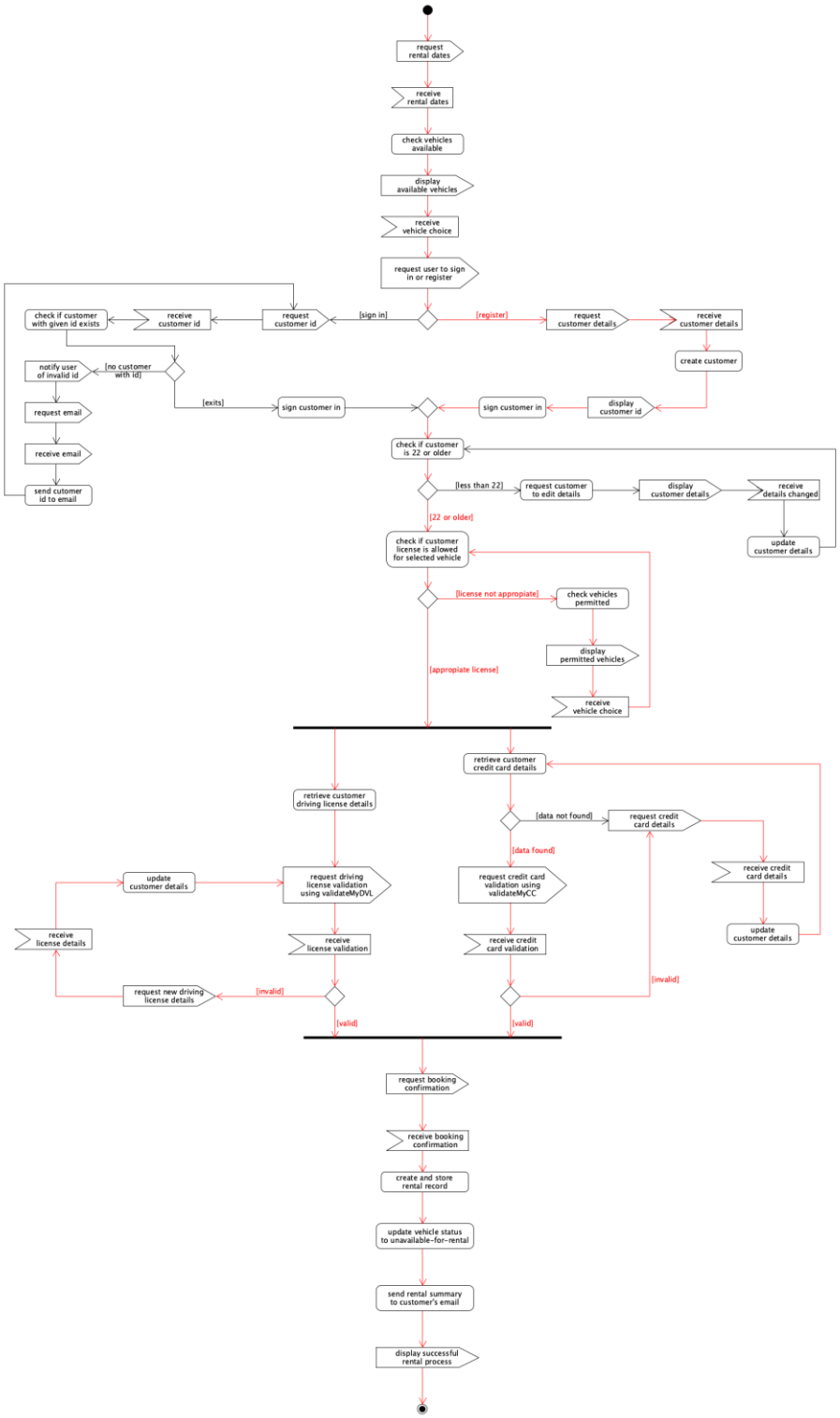
PATH 1


request rental dates
receive rental dates
check vehicles available
display available vehicles
receive vehicle choice
request user to sign in or register
[sign in]
[register]
request customer id
receive customer id
check if customer with given id exists
request customer details
receive customer details
create customer
display customer id
sign customer in
sign customer in
[exits]
[no customer with id]
notify user of invalid id
request email
receive email
send cutomer id to email
check if customer is 22 or older
[less than 22]
request customer to edit details
display customer details
receive details changed
update customer details
[22 or older]
check if customer license is allowed for selected vehicle
[license not appropiate]
check vehicles permitted
display permitted vehicles
receive vehicle choice
[appropiate license]
retrieve customer credit card details
retrieve customer driving license details
[data not found]
request credit card details
receive credit card details
update customer details
[data found]
request credit card validation using validateMyCC
receive credit card validation
[invalid]
[valid]
request driving license validation using validateMyDVL
receive license validation
update customer details
receive license details
request new driving license details
[invalid]
[valid]
request booking confirmation
receive booking confirmation
create and store rental record
update vehicle status to unavailable-for-rental
send rental summary to customer's email
display successful rental process

PATH 2


request rental dates
receive rental dates
check vehicles available
display available vehicles
receive vehicle choice
request user to sign in or register
[sign in]
[register]
request customer id
receive customer id
check if customer with given id exists
notify user of invalid id
[no customer with id]
request email
receive email
send cutomer id to email
[exits]
sign customer in
request customer details
receive customer details
create customer
display customer id
sign customer in
check if customer is 22 or older
[less than 22]
request customer to edit details
display customer details
receive details changed
update customer details
[22 or older]
check if customer license is allowed for selected vehicle
[license not appropiate]
check vehicles permitted
display permitted vehicles
receive vehicle choice
[appropiate license]
retrieve customer driving license details
request driving license validation using validateMyDVL
receive license validation
[invalid]
request new driving license details
update customer details
receive license details
[valid]
retrieve customer credit card details
[data not found]
request credit card details
receive credit card details
update customer details
[data found]
request credit card validation using validateMyCC
receive credit card validation
[invalid]
[valid]
request booking confirmation
receive booking confirmation
create and store rental record
update vehicle status to unavailable-for-rental
send rental summary to customer's email
display successful rental process

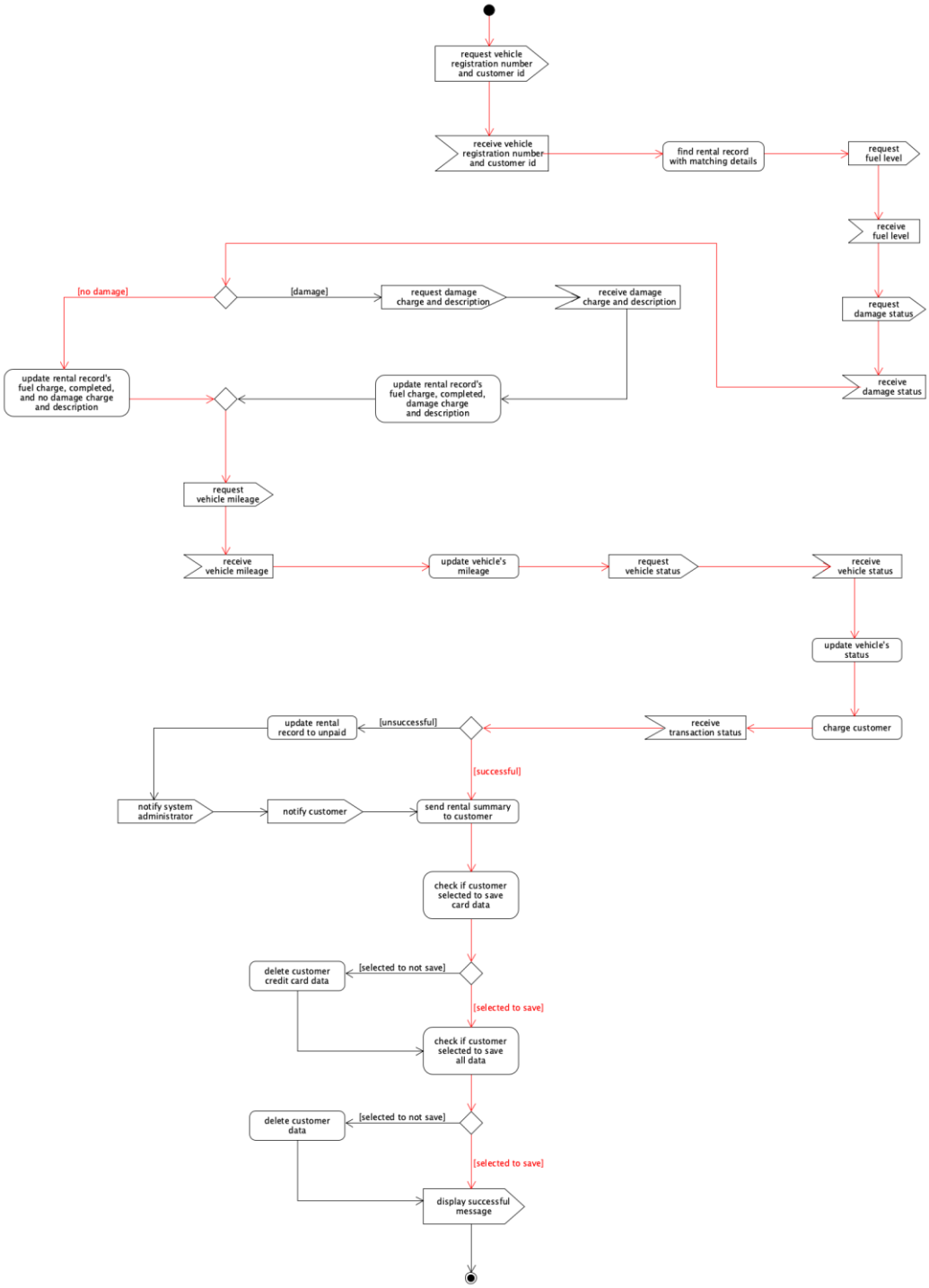
PATH 3


request rental dates
receive rental dates
check vehicles available
display available vehicles
receive vehicle choice
request user to sign in or register
[sign in]
[register]
request customer id
receive customer id
check if customer with given id exists
request customer details
receive customer details
create customer
notify user of invalid id
[no customer with id]
[exits]
request email
receive email
send cutomer id to email
sign customer in
display customer id
check if customer is 22 or older
[less than 22]
request customer to edit details
display customer details
receive details changed
update customer details
[22 or older]
check if customer license is allowed for selected vehicle
[license not appropiate]
check vehicles permitted
display permitted vehicles
receive vehicle choice
[appropiate license]
retrieve customer driving license details
retrieve customer credit card details
[data not found]
request credit card details
receive credit card details
update customer details
[data found]
request driving license validation using validateMyDVL
request credit card validation using validateMyCC
receive license validation
receive credit card validation
receive license details
update customer details
request new driving license details
[invalid]
[valid]
request booking confirmation
receive booking confirmation
create and store rental record
update vehicle status to unavailable-for-rental
send rental summary to customer's email
display successful rental process

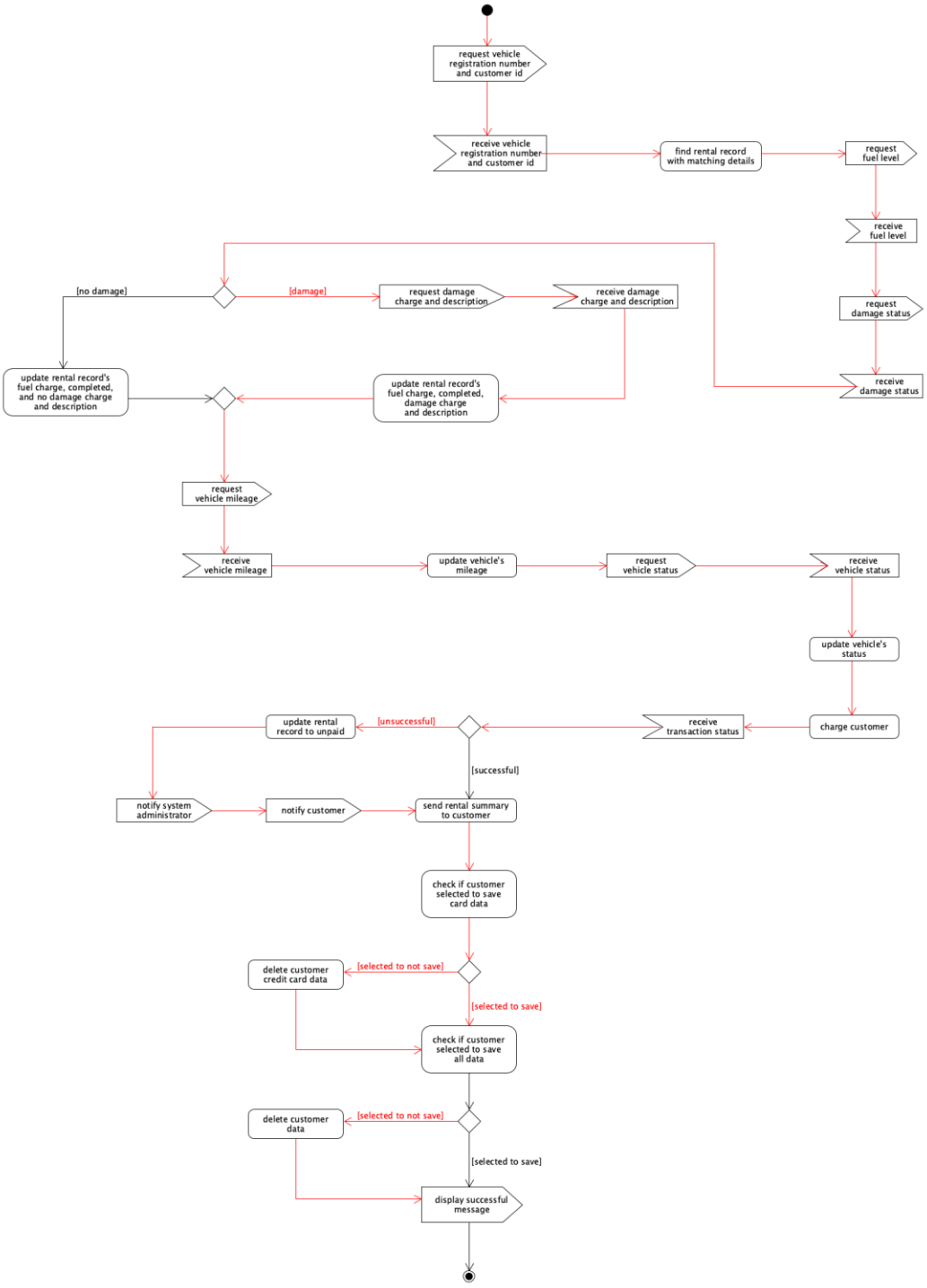
PATH 4

Path	Comment	Path Condition
1	User chooses to sign in, provides a valid ID, is 22 or older, has the appropriate license for the vehicle chosen, has a valid Driving License and has his/her credit card data saved, which is valid.	· choice is **SignIn** · customerId **Exists** · customerAge >= 22 · customerLicense is **Appropriate** · license is **Valid** · cardData is **Found** · card is **Valid**
2	User chooses to signIn, provides an invalid Id, after retrieving, provides a valid one. Customer is 22 or older, does not have the appropriate license for vehicle selected. After selecting a new vehicle the license is appropriate. Customer has a valid license and no credit card data saved. After providing card data, system finds data and given data is invalid. After providing new card data, system finds data and card is valid.	· choice is **SignIn** · customerId Does **NOT Exists** · customerId **Exists** · customerAge >= 22 · customerLicense is **NOT Appropriate** · customerLicense is **Appropriate** · license is **Valid** · cardData is **NOT Found** · cardData is **Found** · card is **inValid** · cardData is **Found** · card is **Valid**
3	User chooses to register, is 22 or older, has the appropriate license for the vehicle chosen, has a valid License and credit card data. System will always find card data if user registered, as it means providing card details, whilst signing in can mean customer has account but no card data saved.	· choice is **Register** · customerAge >= 22 · customerLicense is **Appropriate** · license is **Valid** · cardData is **Found** · card is **Valid**
4	User chooses to register, customer is 22 or older, does not have the appropriate license for vehicle selected. After selecting a new vehicle the license is appropriate. Customer has an invalid license, after providing new license details, system validates license successfully. Card data is found, but invalid. After providing new card data, system finds data and card is validated	· choice is **Register** · customerAge >= 22 · customerLicense is **NOT Appropriate** · customerLicense is **Appropriate** · license is **inValid** · license is **Valid** · cardData is **Found** · card is **inValid** · cardData is **Found** · card is **Valid**

ReturnRental



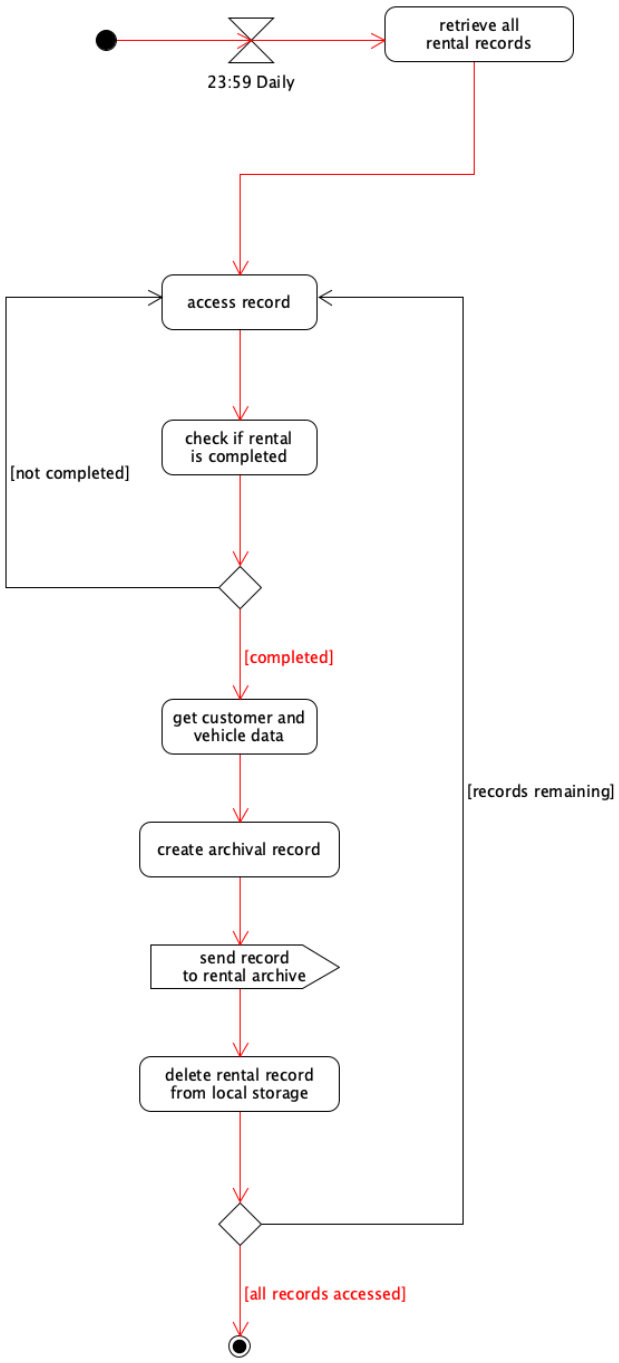
PATH 1


request vehicle registration number and customer id
receive vehicle registration number and customer id
find rental record with matching details
request fuel level
receive fuel level
request damage status
receive damage status
[no damage]
[damage]
request damage charge and description
receive damage charge and description
update rental record's fuel charge, completed, and no damage charge and description
update rental record's fuel charge, completed, damage charge and description
request vehicle mileage
receive vehicle mileage
update vehicle's mileage
request vehicle status
receive vehicle status
update vehicle's status
charge customer
receive transaction status
[unsuccessful]
update rental record to unpaid
notify system administrator
notify customer
[successful]
send rental summary to customer
check if customer selected to save card data
[selected to not save]
delete customer credit card data
[selected to save]
check if customer selected to save all data
[selected to not save]
delete customer data
[selected to save]
display successful message

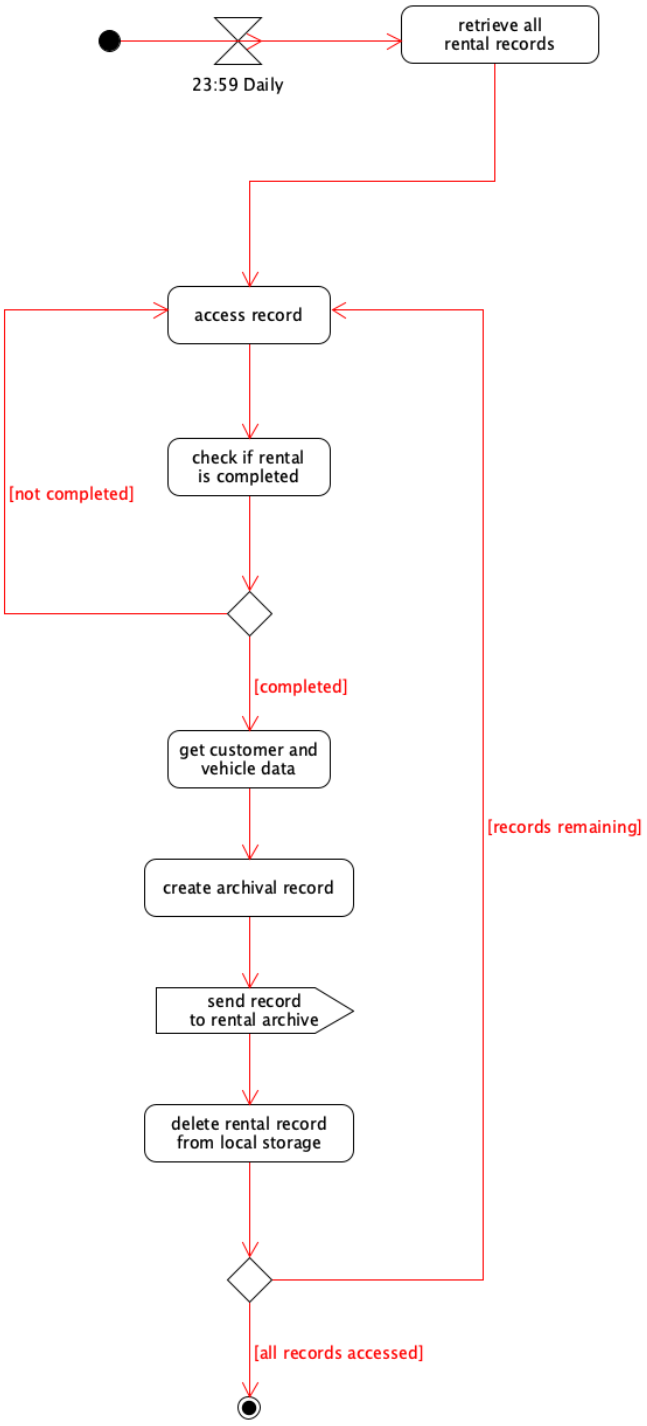
PATH 2

Path	Comment	Path Condition
1	Rental record is successfully completed with no damage, successful payment and all customer data is kept.	· RentalRecord in **Found** · damage is **False** · payment is **Successful** · keepCCData is **True** · keepAllData is **True**
2	Rental record is successfully complete, vehicle has reported damage, payment failed, and all customer data is deleted	· RentalRecord in **Found** · damage is **True** · payment is **NOT Successful** · keepCCData is **False** · keepAllData is **False**

ArchiveRental



PATH 1


retrieve all
rental records
23:59 Daily
access record
check if rental
is completed
[not completed]
[completed]
get customer and
vehicle data
create archival record
send record
to rental archive
delete rental record
from local storage
[records remaining]
[all records accessed]

PATH 2

Path	Comment	Path Condition
1	Archival of all (1) completed records stored in system is completed successfully	· It is 23:59 · RentalAcessed is **Completed** · AllRecordsAccessed is **True**
2	All five records are checked, and all three completed records are archived successfully.	· It is 23:59 · RentalAcessed is **Completed** · AllRecordsAccessed is **False** · RentalAcessed is **Completed** · AllRecordsAccessed is **False** · RentalAcessed is **NOT Completed** · RentalAcessed is **Completed** · AllRecordsAccessed is **False** · RentalAcessed is **NOT Completed** · AllRecordsAccessed is **True**