Pure Car Hire

Object-Oriented Analysis and Design

April 21, 2010

Module:
CS4125 Systems Analysis & Design

Authors:
ABC, ID: 1234567; XYZ, ID: 1234568

CS4125: Systems Analysis

Assignment 1: Semester II, 2010-2011

TEAM SUBMISSION : MARKING SCHEME

Updated Version (20th April 2011)

Name:	ID:
Name:	ID:

	Item	**Detailed Description**	**Marks Allocated**		**Marks Awarded**
			Sub-total	Total	
	Presentation	• General Presentation • Adherence to guidelines i.e. front cover sheet, blank marking scheme, table of contents		**2**	
4	Narrative	Narrative description of business scenario		**1**	
5	Software Lifecycle	Is it linear (Waterfall) or iterative (RUP). Discuss risk management strategy?		**1**	
6	Project Plan	Plan specifying timeline, deliverables, and roles.		**1**	
7	System Architecture	System architecture diagram		**2**	
8	Requirement	• Use case diagram(s)	2		
		• Structured use case descriptions(s)	3		
		• Non-functional (quality) attributes	2	**8**	
		• Screen shots / report formats	1		
9	Analysis	• Method used to identify candidate classes	1		
		• Class diagram with generalisation, composition, multiplicity, dialog, control, entity, interface classes, etc.	3		
		• Communication diagram			
		• Sequence diagram	2	**10**	
		• State chart with annotated transition strings	1		
		• Entity relationship diagram with cardinality	2		
			1		
10	Design	• Description of an architectural or design pattern that was evaluated. Cannot use MVC, Broker and Singleton.	2		
		• Pattern incorporated into class diagram	2		
		• Refinement of class diagram from analysis to include MVC architectural pattern.	2	**9**	
		• Refined interaction diagram from analysis with MVC elements	1		
		• Support for concurrency	2		
11	Data dictionary	Fragments to illustrate different artifacts created during requirements, analysis and design.		**1**	
12	References			**1**	
14	Online Assessment	Week 7: Use cases Week 9: Class Diagram		**2**	
	TOTAL			**40**	

Table of Contents

1 Product Description

Our car hire system has been conceived to take the routine out of car rental management, both for the customer and for the company staff, replacing it with a simple, expedient, efficient procedure that minimizes the outlay of time and money. Above all, it will provide a great user experience in a direct, clear style. It will be a web-based system, maximizing ease of access for customers and for staff. Staff will also be able to connect readily to any and all of the data required.

Within the ambit of this operation will be all that the customer can view, amend and confirm with the requisite background procedures to ensure completion. The customer will be able to browse for prices by car type (e.g. eco-friendly, family size), pick-up location and estimated length of hire. A quote will be returned based on the information given.

If the customer wishes to proceed at this point then they will be asked to log in (if they haven't already done so) or register if they are a new customer. There will be help at this stage if the customer has forgotten their username or password or both. The information needed for registering is a password and email address. Any information collected at this stage will have to comply with the relevant data protection legislation.

Every registered customer will be given the opportunity to join a club scheme which will reward the customers for their loyalty as giving the company valuable information about their customers. The reward the customers will receive is a discount on their next reservation after they have already made ten reservations. Club members will be able to inspect and/or update their personal details as well as cancel membership. Club members will be recognised as such after log in.

Once logged in the customer will continue to the reservation bookings stage, any information given before logging in will be retained and presented for verification/amendment. Further choices will be available at this point (e.g. baby seat, GPS tracking, insurance, SatNav). Once the customer is satisfied with the details, they confirm and an email will be sent with the details and a reservation number to cover this transaction.

The customer will also have the option to view, modify and/or cancel an existing reservation (outside a certain period for modification and cancellation), using the reservation number emailed at confirmation. Once the reservation in question has been selected, the reservation will open in a viewing screen. If a cancellation is required, the customer will be asked to confirm that this is what is required before processing the request. For modifying the customer will be taken back through the reservation process as described above. A reminder email will be sent a week before pickup date to confirm and to remind cancellation will not be possible after this time.

The company staff will have all the same options as the customer permitted privileges as well as supplementary selections for processing collections and returns.

When collecting the car, the customer must have a valid driving license. It would be preferable if they were to have a hardcopy of their reservation details but knowing their reservation details or having an electronic copy of them or their club membership card would also be acceptable.

If the company cannot supply the car as per the customers reservation at this point then an upgrade will be offered. Equally if a customer does not arrive when expected and no message is received with an explanation i.e. flight delays, then the reservation will be held for up to 24hrs. After this time the full amount will be charged to the customer.

The customer is expected to return the car on or before the agreed the return date (given as date, month, year). Every 24hrs after the agreed return date will be charged at the regular rate.

The car must be returned in a reasonable condition as per terms agreed at reservation time. A member of staff will do a visual inspection at point and time of return and agreement is required from customer at this point.

At time of reservation, the customer will be charged per the reservation details i.e. for the car and for the expected number of days it is required. Once car has been returned and any extra charges calculated or any refunds calculated, the payments are processed. Payments will be accepted by cash, credit or debit card from personal customers.

Staff will have the ability to create, update and/or delete customer details. They will also have to update club membership details i.e. any information for company use only.

Certain Staff will also be able to create, update and/or delete staff login details.

Company staff will have access to a range of reports. They will be able to view and print customer transaction history, customer future reservations, daily/weekly/monthly sales, club member activity history.

Each of the steps outlined above will have a help screen which will specify information required at each stage, with a link to a FAQ page, a help email address and a helpdesk phone number.

2 Software life-cycle model

The Software Development Life Cycle, also known as Software Development process, is a methodology used to improve the development of a software product. Previously the methodologies and processes in use would have been on an ad hoc & uncoordinated basis and would have been considered unique to each project and/or development team. But, as has been shown in other disciplines, having repeatable procedures helps to reduce the problems caused by common known recurring errors, as any procedure used would highlight and eliminate these before commencement.

One sign that software development is now at a more 'developed' point is that there is an ISO standard for it. ISO 12207 supplies a common structure so that everyone involved with the development can use a common language to communicate.

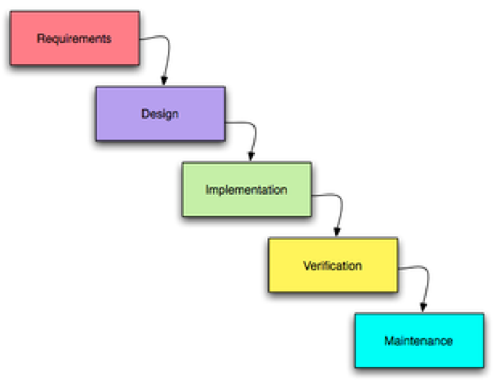
The Software Development Life Cycle methodology may consist of:

- Analysis
- Design
- Implementation
- Testing
- Deployment

One of the main driving forces in the push for a repeatable process for software development was the realisation that the cost of fixing faults in a system increases as the system progresses further through the Software Development Life Cycle.

Several models exist to streamline the development process and these models place greater emphasis on some of the above points at the expense of the others. For this project we considered three of the best known: Waterfall Model, Spiral Model and Agile Development Model.

The Waterfall Model

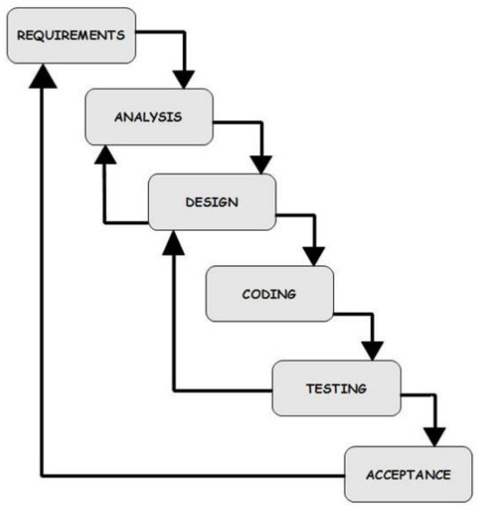


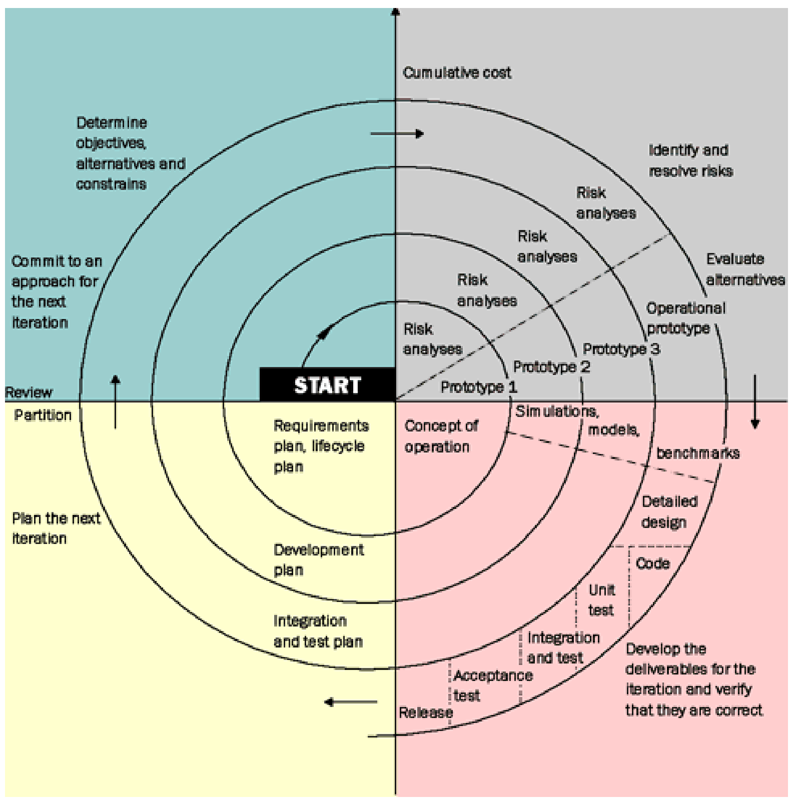
The name comes from the sequential series of steps followed, each one finished before flowing into the next step.

This follows a process where you begin with what the project is supposed to accomplish (Requirements/Analysis). Next you develop a plan for how you will implement the system (Design) and then you write and compile program code (Implementation). After this you run tests to verify that it works as specified (Verification/Testing) and finally it is installed and used for purpose (Maintenance/Deployment).

In the strict Waterfall Model it is not possible to revisit any stage once it is completed. This might be due to its origin in the manufacturing and construction industries [Wikipedia] where cost would prohibit amendments once a stage had been reached. There is a Waterfall Model that allows one back-step for revision before continuing.

With this inflexibility constraining revising and improving the system built-in, other models were created to be flexible.





The Spiral Model

The Spiral Model was defined by Barry Boehm in his 1986 article "A Spiral Model of Software Development and Enhancement"[Wikipedia]. It requires the developer to identify, analyse and resolve any risks inherent in the system under development, build a prototype to check if they have been resolved and to continue until all risks have been removed or neutralised (it might be too expensive to continue attempting to remove a risk that has little impact).

The Spiral Model's use of prototypes allows for Client involvement at an early stage, the Client can test the User Interface prototype and suggest improvements because of this interaction.

One of the disadvantages of this model is the temptation it places in the developers' way to do another iteration and see what happens, rather than attempt to fix whatever doesn't meet requirements.

Because of the time involved and the constant Client feedback leading to further iteration, the Spiral Model may be better suited to large and complicated projects where the expected expense would be justified. This would apply particularly to projects where the developer does not have any experience of the industry the project will be implemented in.

The steps in using a Spiral Model for a project are generally:

1. Gather as much information as possible about what is required, possibly using the SQIRO method (Sampling, Questionnaires, Interviews, Research, Observation) within the Clients company
2. Decide how to implement the requirements thus far identified and create a preliminary design. Take note of any potential risks and how to obviate them. The design may consist

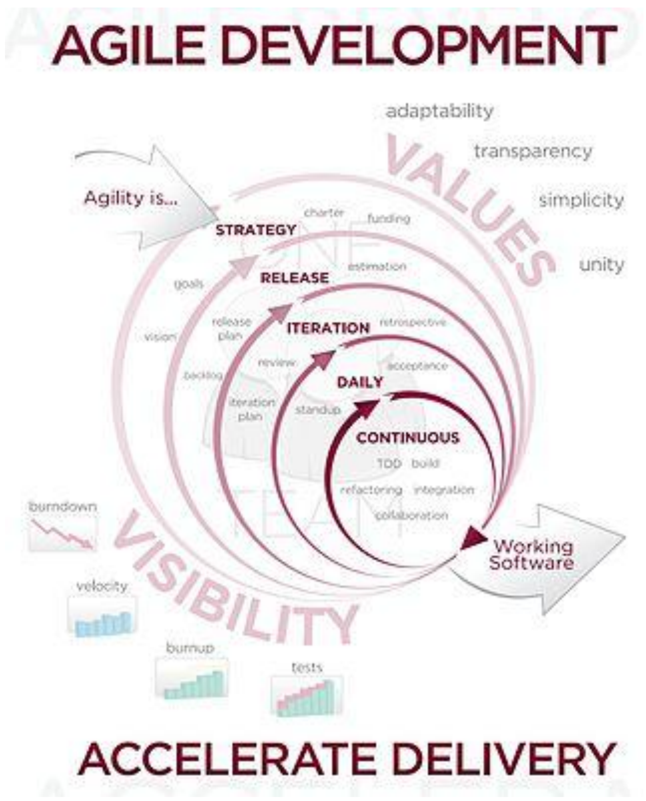
of a class diagram with classes, attributes, operations, associations, etc. which will be used to model the system implementation.

3. Create a prototype based on the design with a User Interface for the Client that will interact with the classes, etc. identified in Step 2 as though this were the finished system.
4. Evaluate the prototype from Client feedback and the developers own observations. If necessary refine the requirements, change the underlying structure for implementation. Then decide what needs to be refined and what needs to be changed from the previous prototype to plan, to make and to test another prototype. This necessitates another iteration through Steps 1-3, leading again to Step 4 and persisting until agreement is reached that the design is the optimum design available.

The Agile Development Model

From the "Manifesto For Agile Software Development" [Kent Beck et al (2001)] which advocates a more people-centric approach to software development.

The twelve principles behind the Agile Manifesto are:



1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

The importance of face-to-face interaction between all members of the team, including Client representatives is what drives Agile Development. Team members report to each other what they did the previous day, what they intend to do today, and what their roadblocks are. This face-to-face communication exposes problems as they arise.

Agile methods break tasks into small increments with minimal planning. Iterations are short time frames that typically last from one to four weeks. For each repeated time frame, a team works through a full software development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing. This is to minimise overall risk and allow the project to adapt to modifications quickly. Stakeholders, e.g. Clients produce documentation as required. Multiple iterations may be required to release a product or new features.

Agile development emphasizes working software as the primary measure of progress. This, combined with the preference for face-to-face communication, produces less written documentation than other methods.

Because we needed to learn about Requirements/Analysis, then about Design and would not be implementing, verifying or maintaining this system we used the waterfall model, or part thereof to design this system.

3 Project Plan

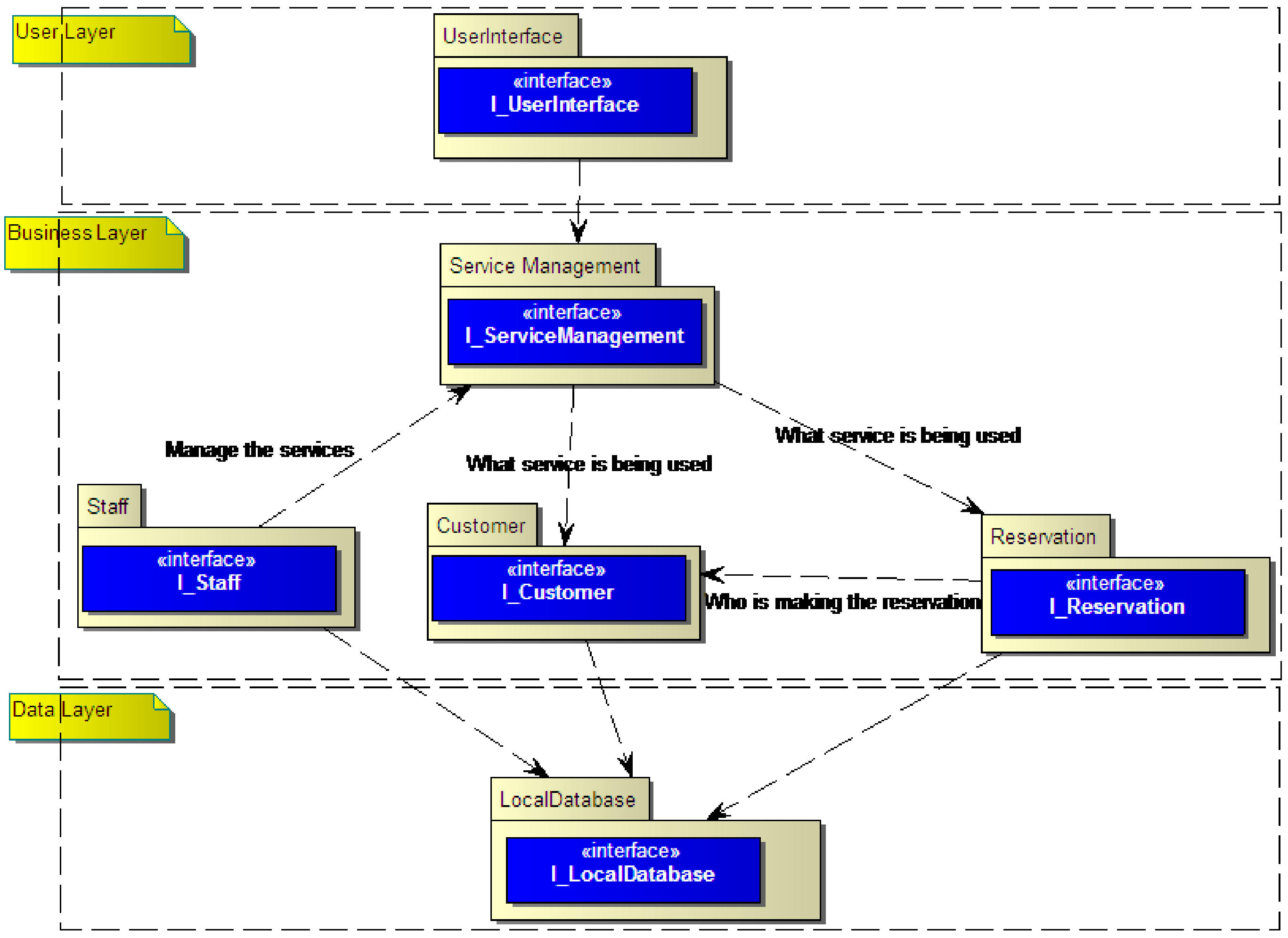
Description		Person	Delivered
Narrative		ABC	Wk 7
Software Lifecycle		ABC	Wk 12
Project plan		Both	Wk 8
System Architecture		XYZ	Wk 12
Requirements	Use Case Diagram	Both	Wk 10
	Description	XYZ	Wk 7
	Non-functional attributes	ABC	Wk 11
	Screen shots / report formats	ABC	Wk 11
Analysis	identify classes	Both	Wk 10
	Class diagram	Both	Wk 9 – wk 12
	Communication diagram	Both	Wk 8 – wk 9
	Sequence diagram	Both	Wk 9 – wk 10
	State chart	XYZ	Wk 11
	E-R diagram	XYZ	Wk 11
Design		Both	Wk 11
Technical architecture		Both	Wk 12
Data dictionary		ABC	Wk 13
Critique		Both	Wk 13
References		Both	Wk 13

4 Discussion of System Architecture

The Car Hire system used on-line by customers can be easily regarded as a client. Then the structure of the system basically is a traditional two-tier or client/server architecture. But for modifiability, extensibility and maintainability needs, the team decided to apply the three-tier architecture for the Car Hire system. Each well-defined and separate layer's process will be running on a different platform:

1) The User Layer is all about user interface running on the user's computer which can be regarded as a client
2) The Business Layer is the functional modules that actually process data and provide services. This tier usually runs on the application server.
3) The Database Layer is a database management system that stores the data required by the Business Layer which runs on a second server called the database server.

Due to the complexity of the system, the team decided to divide the system into sub-systems according to different area of functionality. This approach enhances the modularity of the system and provides an overall view of the system in a neat and clear way. The following is the structure with a list of packages of the system which have been allocated to the three-tier structure:


User Layer
UserInterface
«interface» I_UserInterface
Business Layer
Service Management
«interface» I_ServiceManagement
Manage the services
What service is being used
What service is being used
Staff
«interface» I_Staff
Customer
«interface» I_Customer
Who is making the reservation
Reservation
«interface» I_Reservation
Data Layer
LocalDatabase
«interface» I_LocalDatabase

- UserInterface: responsible for interaction with users which can be a customer or a staff
- Service Management: holds all classes responsible for controlling the functionality of the system for both staff and customer being well dispatched, organized and performed.
- Staff: represents classes of managing the staff information and administrating the whole system
- Customer: holds all the classes for recording all the customer's information and provides an interface of customer service.
- Reservation: is a core package that contains the main business logic. It holds all classes responsible for running the car hire reservation business and recording the reservation information.
- LocalDatabase: responsible for maintaining and controlling the local storage of data for the Car Hire system which can provide services like storing reservations or caching in the event of network or service failure.

  As customers use on-line services which are interfaces that the application server provides, furthermore, a different interface used by staff for overall management is also provided by the same classes in the application server. Thus, it was decided that programming to interface as well as a few programming to implementation would be used according to the requirements. This approach would enhance the flexibility and changeability of the system and allow the system to stay relatively efficient at the same time. Moreover, it enables high reusability by programming to interface as some components would be useful to a third party software.

  Any component that is restricted to the internal system and is unlikely to change will be programming to implementation. This will be only applied to parts of the system that are assumed not to change. Classes that are likely to have additional functionality or new better implementations or could be reused for a third party should use programming to interface, which facilitates the interaction between different components. Similarly, We use a UserInterface package to organize the structure of the user interface well enough to facilitate the interaction between human and computers.

  In the purpose of building a commercial usable, reliable and flexible system, we must have a local data storage mechanism that support the concept of transactions and also have some strategies for corruption protection but still remain low maintenance for the system administrator. As we already have a Fleet Control System sharing the burden of managing the car scheduling and maintenance, and a Accounting System responsible for recording transaction history and sales profit situation, it was deemed that a lightweight database system would be ideal in this scenario. Then MySQL or SQLite would be a good choice.

  The Select Architect is selected during the project as the UML workbench. The Select Architect is widely accepted and recognized as a consistent innovator in the design tools and modeling tool market. The team kept doing the project in the lab using this software. The tool was found to be easy to learn and use once you started to do a few diagrams and find the its logic law. It provides a large bunch of features which not only facilitate the modelling work but also enable us to customise our own characters. However, it's hard to keep a neat class diagram when there are a little more generalizations. And the communication diagram is not good enough as we always get into the situations where operations cluttered together when they were too many. But overall the tool was a good enough for the task.

5 Requirements

5.1 Functional

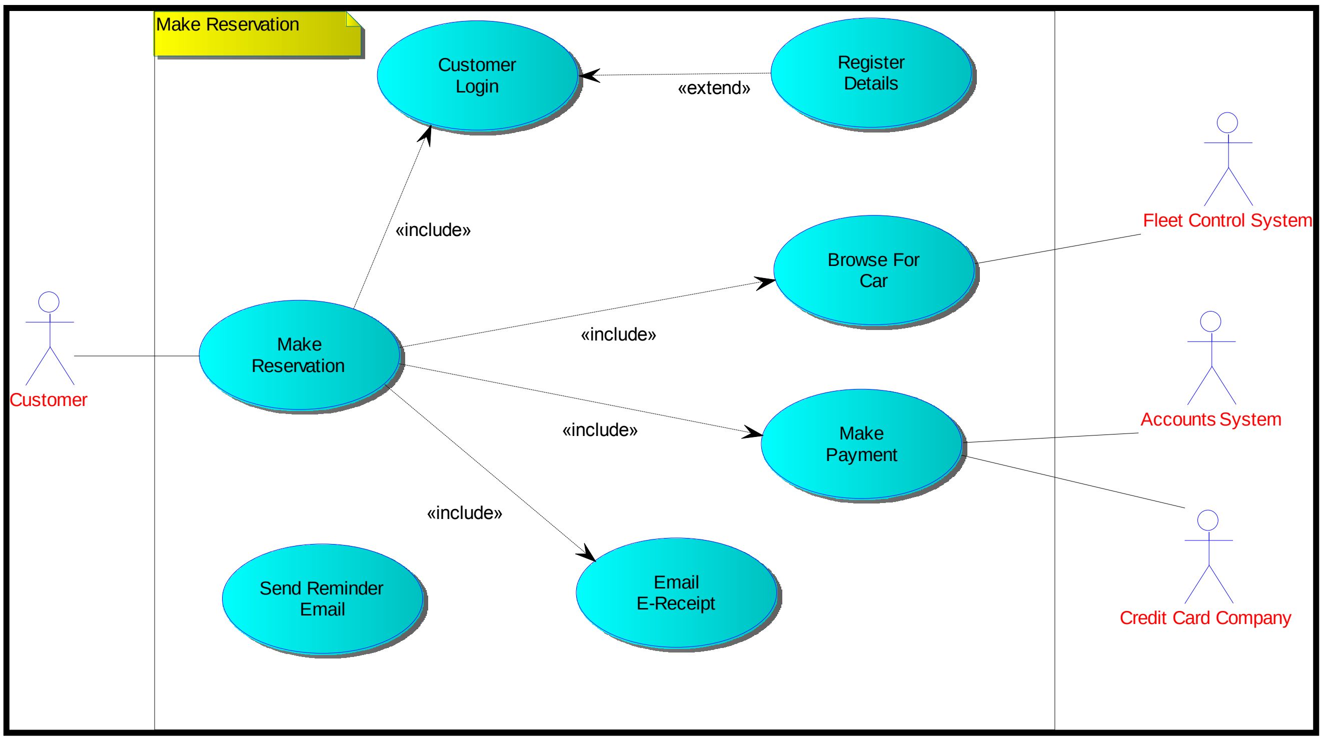
5.1.1 Make Reservation

5.1.1.1 Make reservation Use Case

USE CASE 1	Make reservation	
Goal in Context	Customer wants to find a suitable car to rent and reserves the car in advance, expects to be able to collect the car on arrival	
Scope & Level	Company Summary	
Preconditions	The on-line system is available.	
Success End Condition	Customer has the reservation number and we have deposit for the reservation. The state of the reserved car changes correspondingly in the fleet control system.	
Failed End Condition	Customer don't have the reservation number and has not pay the deposit.	
Primary, Secondary, Actors	Customer, any person who wants to reserve a car. Credit card company, fleet control system, accounting system.	
Trigger	Customer visits the website and requests to find and book a car.	
DESCRIPTION	**Step**	**Action**
	1	Customer browses for car (use case 2)
	2	Customer selects a car
	3	System presents rental details and the additional extras options for customer to choose like GPS, baby seat, etc. Customer finishes choosing and confirms to continue
	4	System presents the new rental details and captures customers personal information like names, email address, etc
	5	Customer makes a payment (use case 4)
	6	System displays the customer a reservation number
	7	System emails a receipt about this reservation to the customers (use case 10)
EXTENSIONS	**Step**	**Branching Action**
	4a	The customer is a club member and has already logged in: 4a1. System just presents the new rental details 4a2. Return to step 5.
	1a	<condition causing branching>: <action or name of sub.use case>
VARIATIONS		**Branching Action**
	1	

RELATED INFORMATION	1. Make reservation
Priority:	Top
Performance	5 minutes for reservation
Frequency	800/day
Channels to actors	Online
OPEN ISSUES	What if no car available during the customer selected date and time? What if extras requested not available?
Due Date	Release 1.0
…any other management information…	None
Superordinates	
Subordinates	Make payment Browse for a car Customer logs in Modify reservation

Use Case Description Template
Courtesy of CSE, Dublin City University



5.1.1.2. Use Case - Browse For Car

Actor Action	System Response
1.The actor selects the location, pick-up and return date&time, and car type.	2. Display list of cars information include car's picture, price, type, and availability etc.
3. Actor confirms choice	4. System displays a quote

Extensions

None

5.1.1.3. Use case - Customer Login

Actor Action	System Response
1. The actor inputs his/her email address and password	2. Verifies the actor's identity.
3. none	4. Display the club member's basic information like name and club level.

Extensions

After step 2, the password is wrong, and the system denies the actor's login request

After step 4, the name or club level displayed is wrong .

5.1.1.4. Use case - Email e-receipt

Actor Action	System Response
1. The actor selects to get a e-receipt	2. Emails a e-receipt to the customer with all details about the reservation

Alternative Course

After step 1, system may automatically send an e-receipt to customer.

5.1.1.4. Use case - Make payment

Actor Action	System Response
1. The actor inputs his/her payment details like Payment method, card no. etc.	2. Capture those payment details and sends them to the credit card company to verify the information
3. None	4. Gets authentication from credit card company
5. None	6. Sends transaction details to accounting system
7. None	8.Completes the payment

Extensions

After step 2, the payment details contain wrong information. System denies the payment.

After step 4, the balance in that credit card is not sufficient and then system denies the payment process.

5.1.1.5. Use case - Register details

Actor Action	System Response
1. The actor inputs email address	2. Checks whether input is valid email address
3. Actor selects password	4. Records customer registration

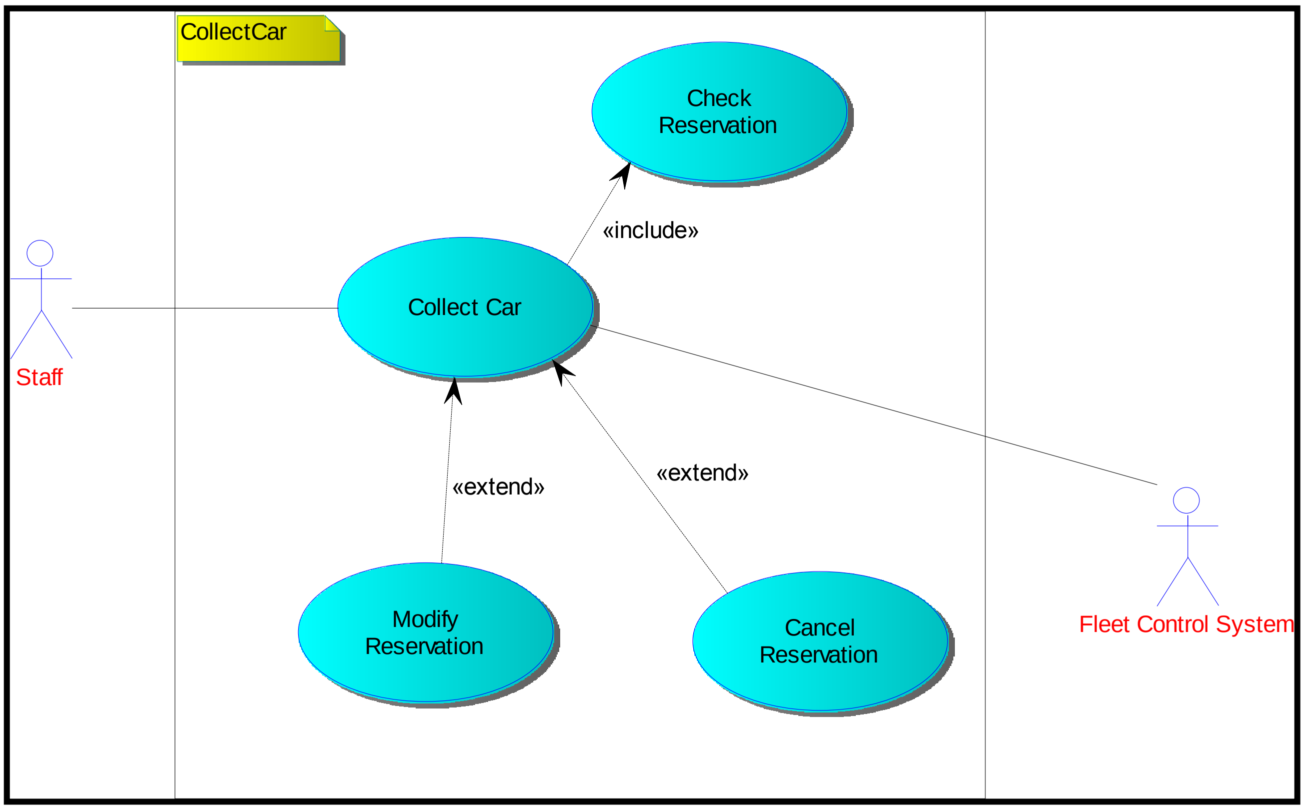
After step 1, customer selects to join club scheme
After step 2, the email address doesn't exist, system asks customer to input it again

5.1.1.6. Use case description: Send reminder email

Actor Action	System Response
1. None	2. Once a day automatically compares the pick-up date of all reservations to the current date once a day
3. None	4. Send a reminder email to customer who has a reservation with a pick-up date 3 days from now.

Extensions
None

5.1.2 Collection



5.1.2.1. Use case - Collect car

Actor Action	System Response
1. Staff asks the customer's reservation number and name, then inputs these to check reservation	2. Displays the reservation details
3. Staff checks the customer's driver license, photo id and credit card	4. System generates contract and captures credit card details
5. Customer checks&signs contract and staff confirms contract signed	6. None
6. Staff gives the customer the car key and confirms that the car has been collected	8. Updates fleet control system

Extensions
After step 3, the customer's driver license is not qualified. Staff cancels the reservation
After step 2, staff upgrades the reservation if the car booked is not available by accident.

5.1.2.2. Use case - Check reservation

Actor Action	System Response
1. The actor inputs the reservation number and his/her name	2. Displays the details for the reservation

Extensions
After step 2, the customer modifies the reservation
After step 2, the customer cancels the reservation
Alternative Course
The customer has already logged in, no need to input reservation number and name.

5.1.2.3. Use case - Cancel reservation

Actor Action	System Response
1. The actor selects cancel reservation	2. Asks the actor to confirm cancellation
3. The actor confirms the cancellation	4. Sends message to fleet control system about the cancellation details
5. None	6. Process refund (use case 7)

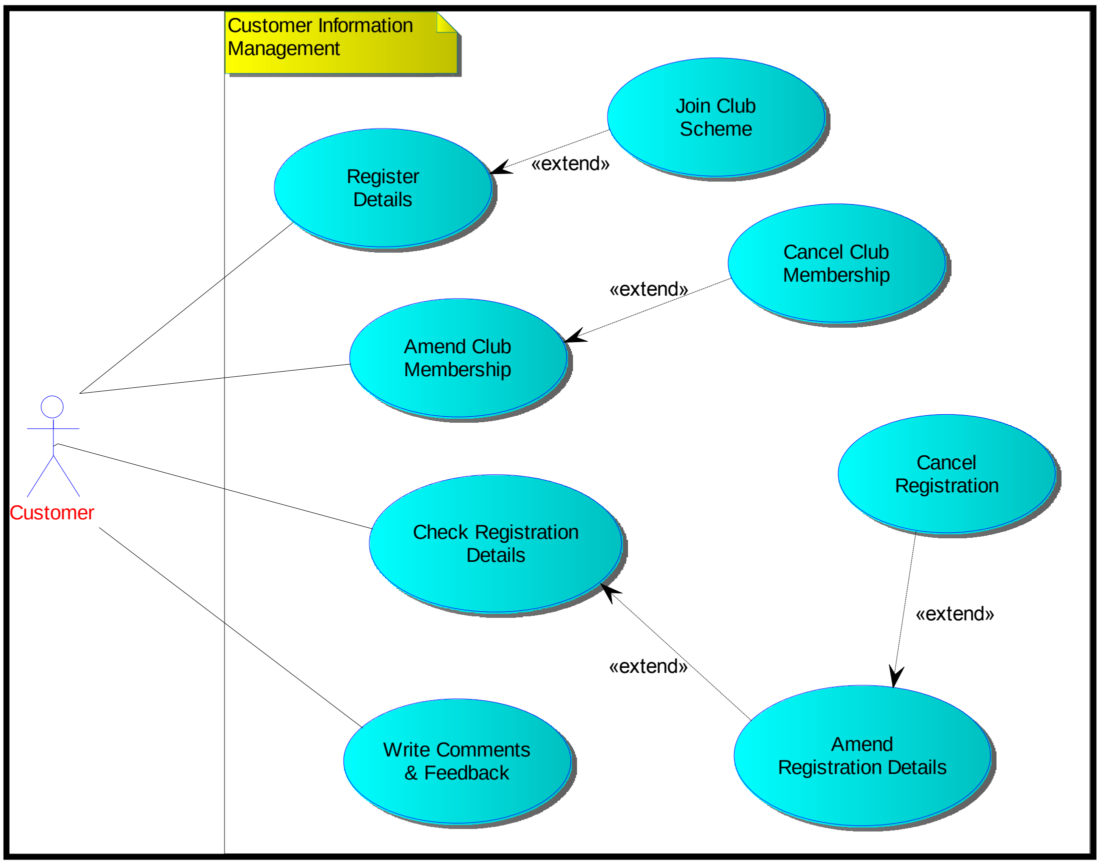
Extensions
None

5.1.2.4. Use case - Modify reservation

Actor Action	System Response
1. The actor selects modify reservation	2. Displays modifiable fields
3. The actor updates selection	4. Displays updated reservation, asks customer to confirm
5. Customer confirms	6. Systems updates fleet control system and sends e-receipt (use case 10)

Extensions
After step 5, customer makes payment if required (use case 4)

5.1.3 Customer Information Management



5.1.3.1. Use case - Amend club membership

Actor Action	System Response
1. Selects to amend membership	2. Displays options for members
3. Updates chosen options	4. Displays new details and requests confirmation
5. Confirms	6. Updates customer membership records
7. None	8. Displays message confirming updates recorded

Extensions
After step 5, customer cancels update

5.1.3.2. Use case - Amend registration details

Actor Action	System Response
1. None	2. Displays amendable fields
3. Makes changes to some fields	4. Updates the database correspondingly and asks customer to confirm
5. Confirms updating	6. Displays message confirming update recorded
Extensions None	

5.1.3.3. Use case - Cancel Club membership

Actor Action	System Response
1. Selects to cancel membership	2. Asks to confirm cancellation
3. Confirms to cancel	
Extensions None	

5.1.3.4. Use case - Cancel registration

Actor Action	System Response
1. Customer requests to cancel registration	2. Displays registration details and requests confirmation of cancellation
3. Customer confirms	4. Cancels registration, removes customer from customer list
5. None	6. Displays message confirming cancellation
Extensions None	

5.1.3.5. Use case - Check registration details

Actor Action	System Response
1. Customer selects to check personal details	2. Displays all the customer's information
Extensions After step 2, customer amends personal details	

5.1.3.6. Use case - Join Club Scheme

Actor Action	System Response
1. The actor inputs detailed information like driver license, contact details etc.	2. Captures these details and assigns the customer a club level
3. None	4. Displays options for customer to customize personal needs
5. Chooses from the options and confirms	6.

Extensions
None

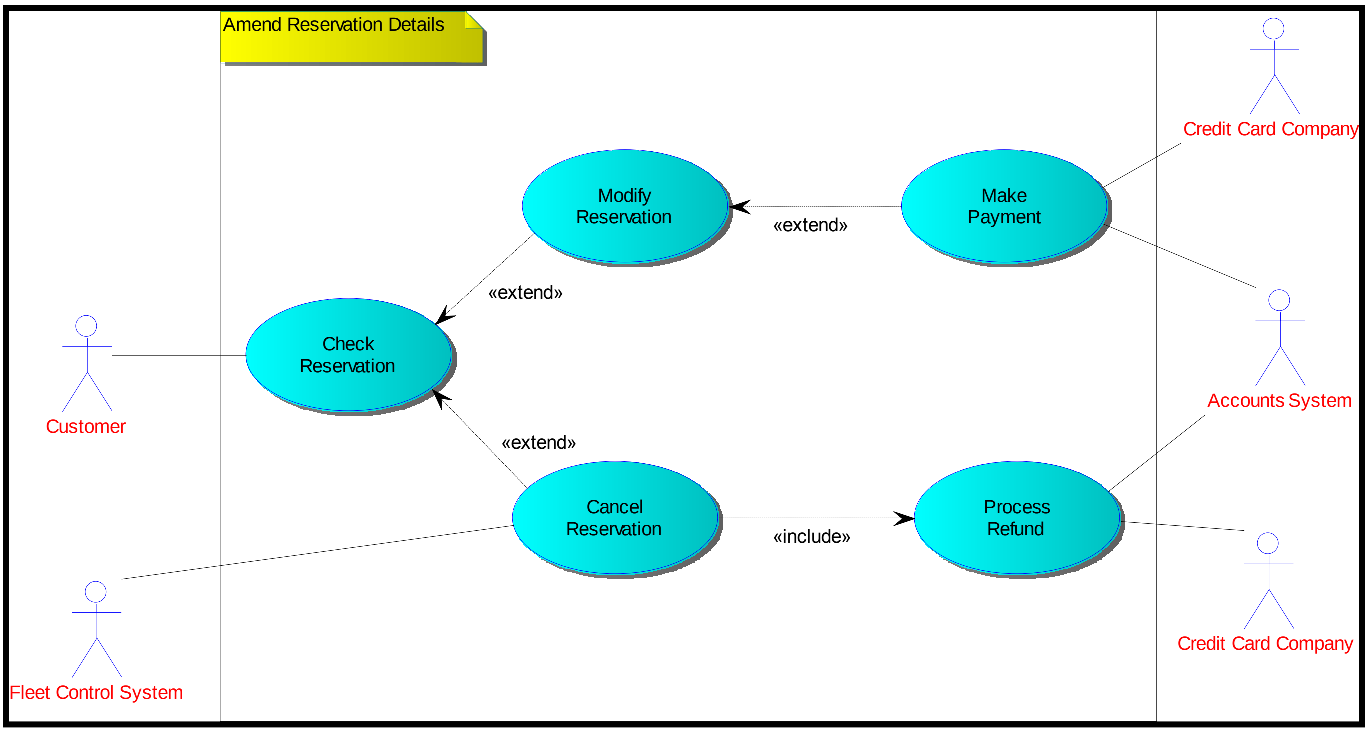
5.1.3.7. Use case - Write comments and feedback

Actor Action	System Response
1. Customer selects comments link on website	2. Displays comments fields
3. Customer updates fields	4. Captures updated information
5. None	6. Displays messages thanking customer for updating comments

Extensions
None

5.1.3.8 Use case – Register details, see 5.1.1.5

5.1.4 Modify Reservation Details



5.1.4.1 Use case - Cancel reservation, see 5.1.2.3.

5.1.4.2 Use case - Check reservation, see 5.1.2.2.

5.1.4.3 Use case - Make payment, see 5.1.1.4.

5.1.4.4 Use case - Modify reservation, see 5.1.2.4

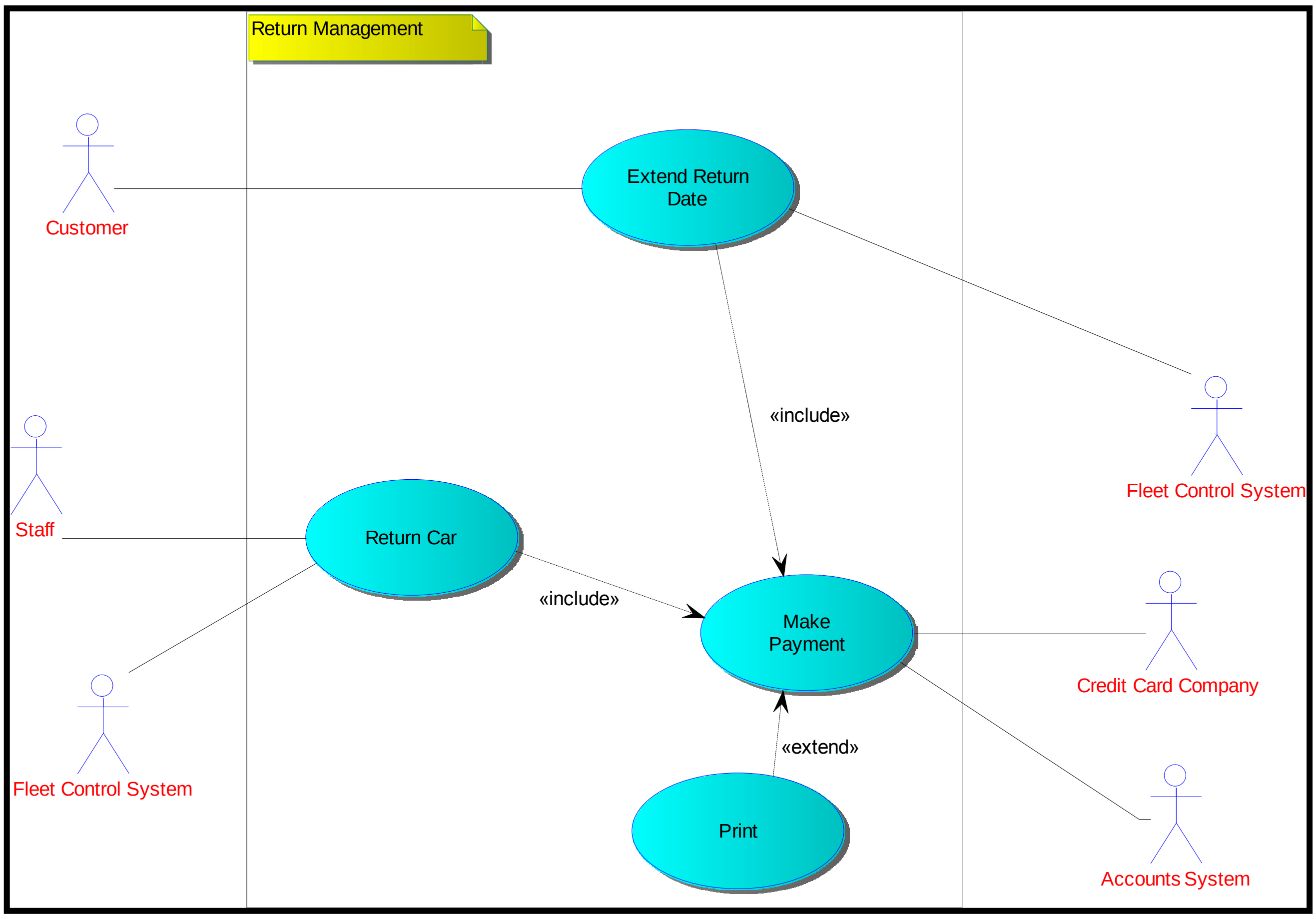
5.1.4.5. Use case - Process refund

Actor Action	System Response
1. None	2. Calculates the amount of money needed to be refund.
3. None	4. Refunds the money to the customer's credit card through the credit card company
5.	6. Updates accounting system

Extensions

None

5.1.5 Return



5.1.5.1. Use case - Extend return date

Actor Action	System Response
1. Customer selects modify reservation link on website	2. Requests reservation number
3. Customer updates	4. Displays reservation
5. Customer amends return date	6. Amends reservation and displays confirmation message
Extensions None	

5.1.5.2 Use case - Make payment, see 5.1.1.4.

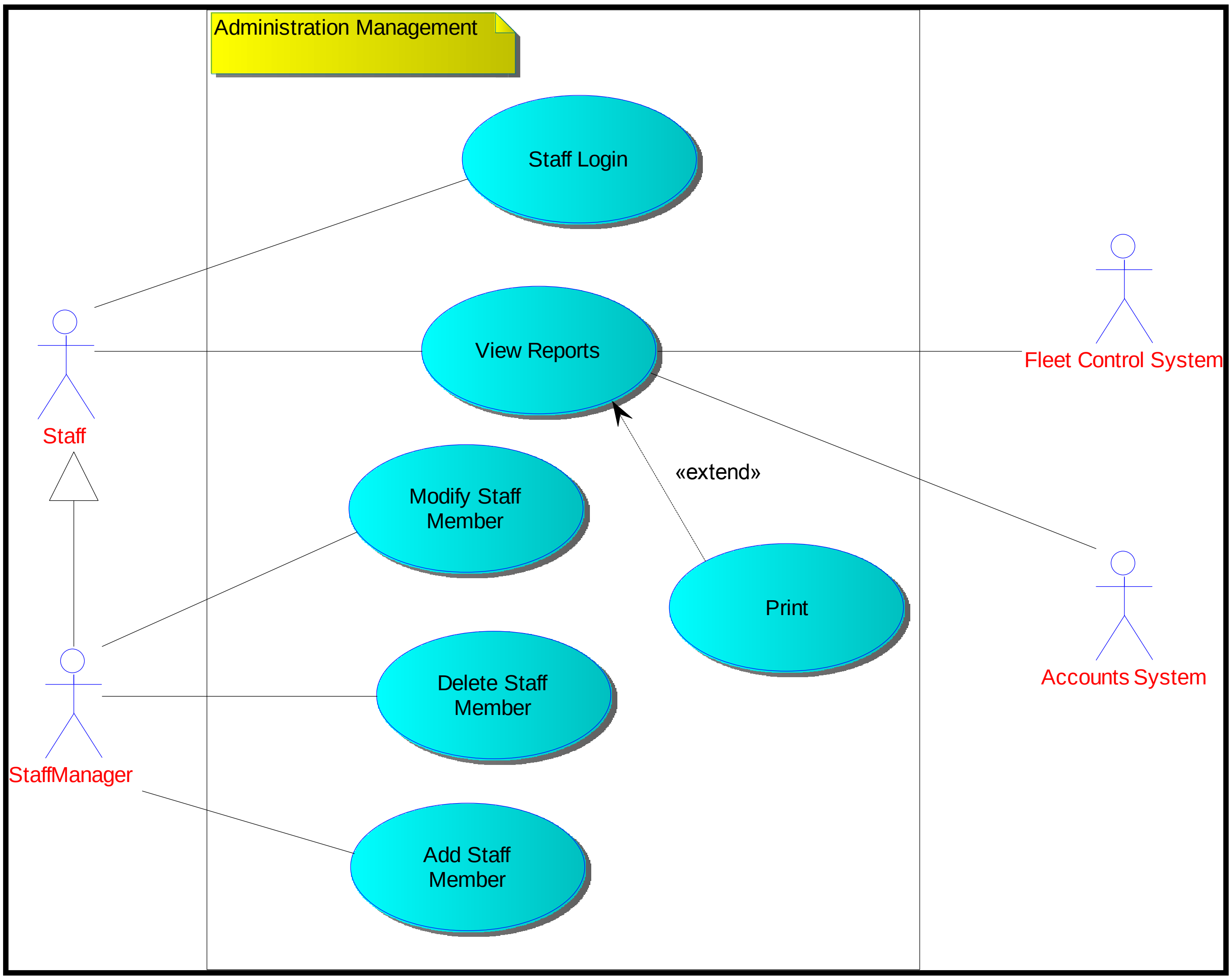
5.1.5.3. Use case - Print

Actor Action	System Response
1. none	2. Shows available printers
3. Chooses printer	4. Sends data to the printer to print
Extensions After step 2, no printer available After step 4, printer offline	

5.1.5.4. Use case description: Return car

Actor Action	System Response
1. Staff asks customer the reservation number and name etc	2. Displays the details about the rental
3. Staff collects the car key	4. Displays car conditions like if damaged or what the mileage is etc.
5. None	6. Automatically calculates the new total rent fees
7. Customer makes payment (use case 4)	8. Records the transaction details
Extensions After step 7, staff prints receipt for customer(use case 19)	

5.1.6 Staff Administration



5.1.6.1. Use case -: Add Staff Member

Actor Action	System Response
1. Staff Manager selects the add staff option	2. Displays required fields
3. Staff Manager inputs required information about that staff member to be added	4. Captures input staff information
5. Staff confirms to create staff	6. creates a staff and adds to staff database and displays message confirming addition

Extensions
After step 3, staff information inputted is not valid. Failed to add staff.

5.1.6.2. Use case - Delete Staff Member

Actor Action	System Response
1. Staff Manager selects delete staff member	2. Displays staff list
3. Staff Manager choose a particular staff	4. none
5. Staff Manager confirms to delete	6. Delete staff from the database and displays confirmation message.

Extensions
None

5.1.6.3. Use case - Modify Staff Member

Actor Action	System Response
1. Staff selects modify staff member	2. Displays staff list
3. Staff Manager choose a particular staff	4. displays detailed information of selected staff
5. make changes to the staff information	6. none

Extensions
None

5.1.6.3 Use case – Print, see 5.1.5.3

5.1.6.4. Use case - Staff Login

Actor Action	System Response
1. Staff inputs user name and password	2. Verifies the user name and password
3. None	4. Displays corresponding user interface that the staff has access to

Extensions
After step 2, the user name and password don't match, system asks staff to input again.

5.1.6.5. Use case - View reports

Actor Action	System Response
1. Staff selects what type of reports he/she wants to view	2. Displays the corresponding reports

Extensions
After step 2, staff prints the report (use case 19)

5.2 Non-Functional

Security – This system will be required to store and access sensitive information such as customer addresses and credit card details. The system will have to ensure that only authorised persons/systems will have access to the relevant data. There are also a number of legal standards to be met, for example the Data Protection Act 1988 and the Data Protection (Amendment) Act 2003 [1]

Reliability – The system would be expected to be available 95% of the time or greater. In the event of network downtime or a service outage, although it would not be possible to progress any further, any information entered would be saved until required again.

Responsiveness - With response times of on average 1 second. Any response exceeding 10 seconds (the limit for keeping the user's attention focused) [2] would need a continuous feedback indicator, for example a percent done animation.

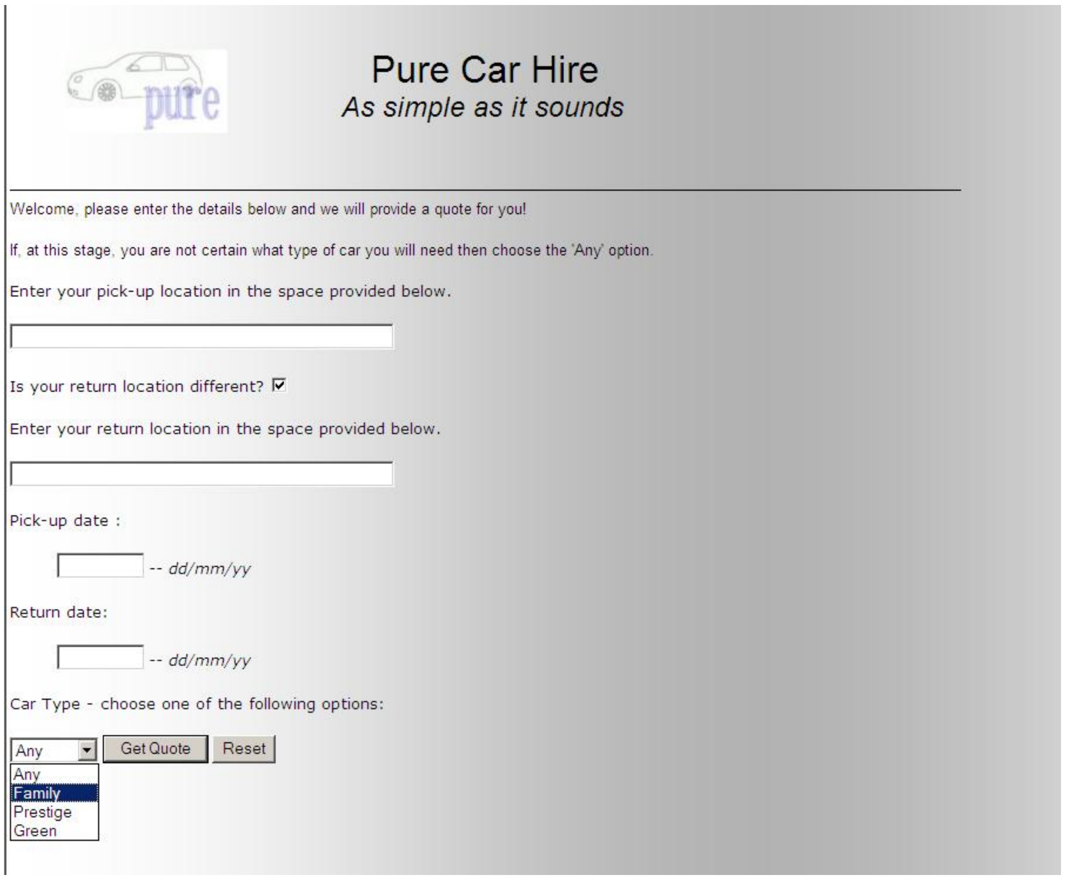
Ease of use – Those unfamiliar with e-commerce websites should still be able to use our system as its layout and options are fairly intuitive. Also a help screen will be offered for each page with a FAQ and step-by-step guidance.

Internationalisation – the system will support at least 5 languages and at least 3 currencies.

Concurrency – Expected to support at least 2000 users

5.3 GUI Prototypes/Reports

Included here are a series of prototype User Interfaces for the front end applications.


Pure Car Hire

As simple as it sounds

Welcome, please enter the details below and we will provide a quote for you!

If, at this stage, you are not certain what type of car you will need then choose the 'Any' option.

Enter your pick-up location in the space provided below.

Is your return location different? ☑

Enter your return location in the space provided below.

Pick-up date :

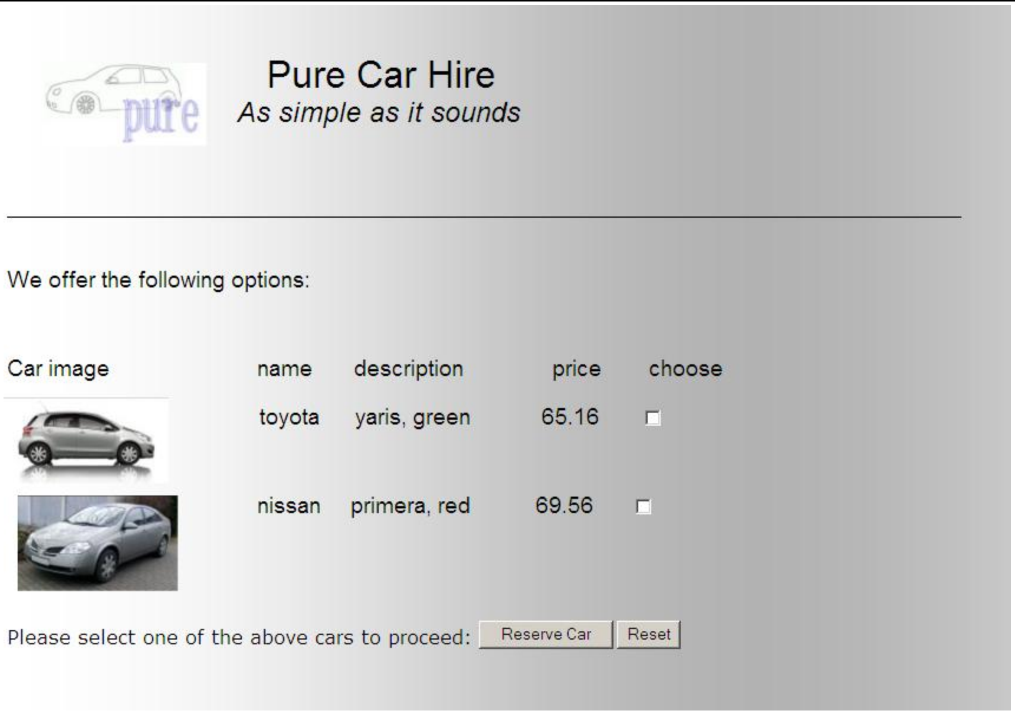
-- *dd/mm/yy*

Return date:

-- *dd/mm/yy*

Car Type - choose one of the following options:

The first screen encountered is where the customer can enter the pickup location and date, a return location (if different from the pickup) and date and choose a car type. The Get Quote button will start the process.



Pure Car Hire

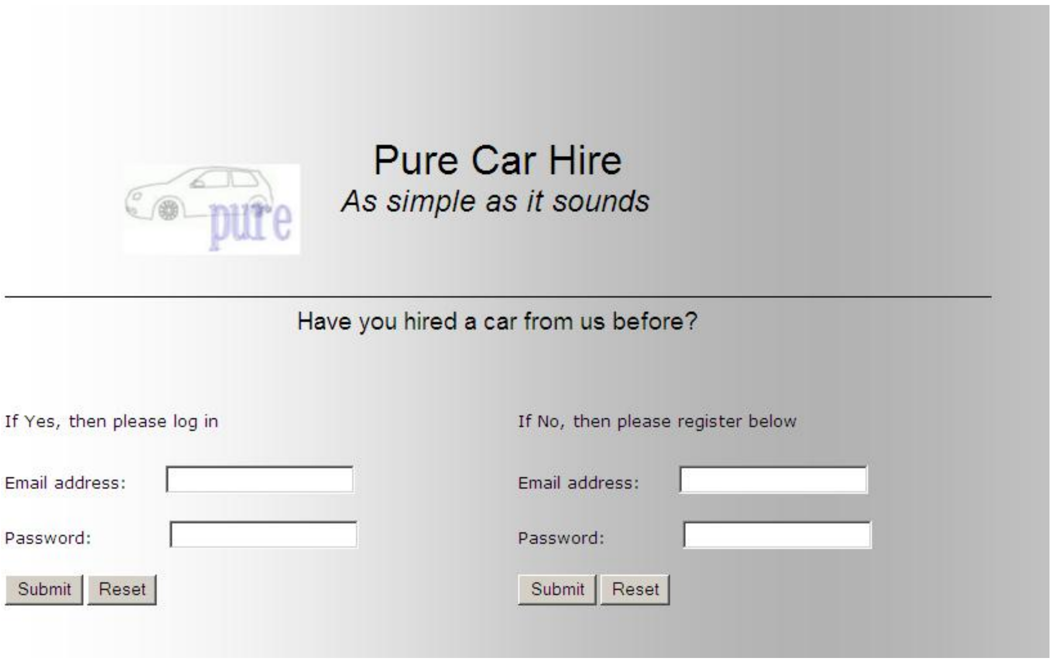
As simple as it sounds

We offer the following options:

Car image	name	description	price	choose
	toyota	yaris, green	65.16	☐
	nissan	primera, red	69.56	☐

Please select one of the above cars to proceed:

Then the options available with their prices are displayed, the customer can then choose an option and click the Reserve Car button to proceed or the Reset button to return to the first screen to enter different options there.



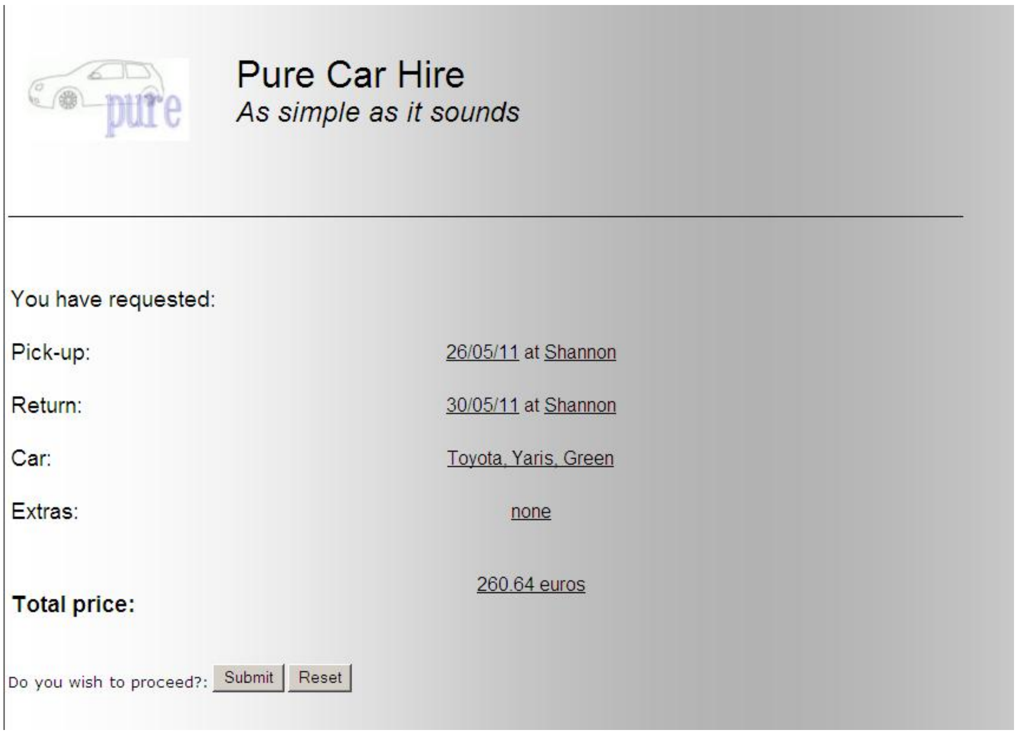
Pure Car Hire

As simple as it sounds

Have you hired a car from us before?

If Yes, then please log in		If No, then please register below	
Email address:		Email address:	
Password:		Password:	

At this point the customer can login, or register if a new customer. Submit will take them to the confirm request screen and reset will take them back to the first screen.


Pure Car Hire

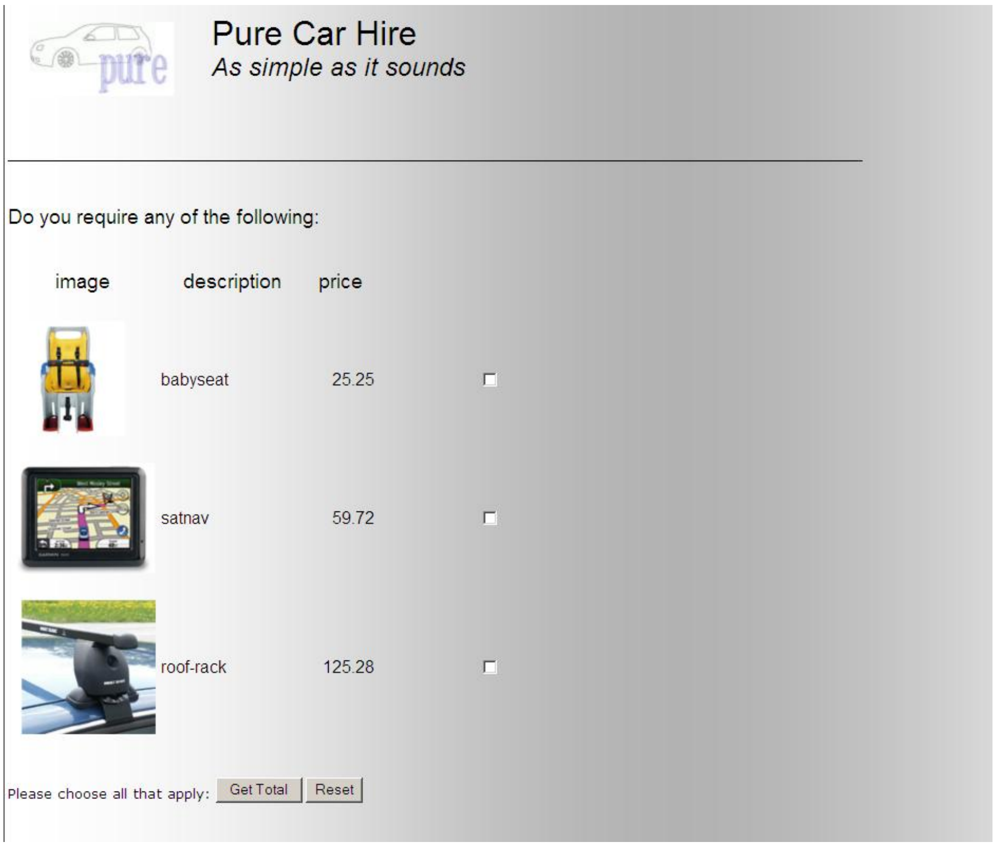
As simple as it sounds

You have requested:

Pick-up:	26/05/11 at Shannon
Return:	30/05/11 at Shannon
Car:	Toyota, Yaris, Green
Extras:	none
Total price:	260.64 euros

Do you wish to proceed?: Submit Reset

The details requested are displayed for confirmation. If this information is all correct and what the customer wants the Submit button is clicked but if there is anything the customer wants to change the Reset button will take them back to the previous screen.


Pure Car Hire

As simple as it sounds

Do you require any of the following:

image	description	price	
	babyseat	25.25	☐
	satnav	59.72	☐
	roof-rack	125.28	☐

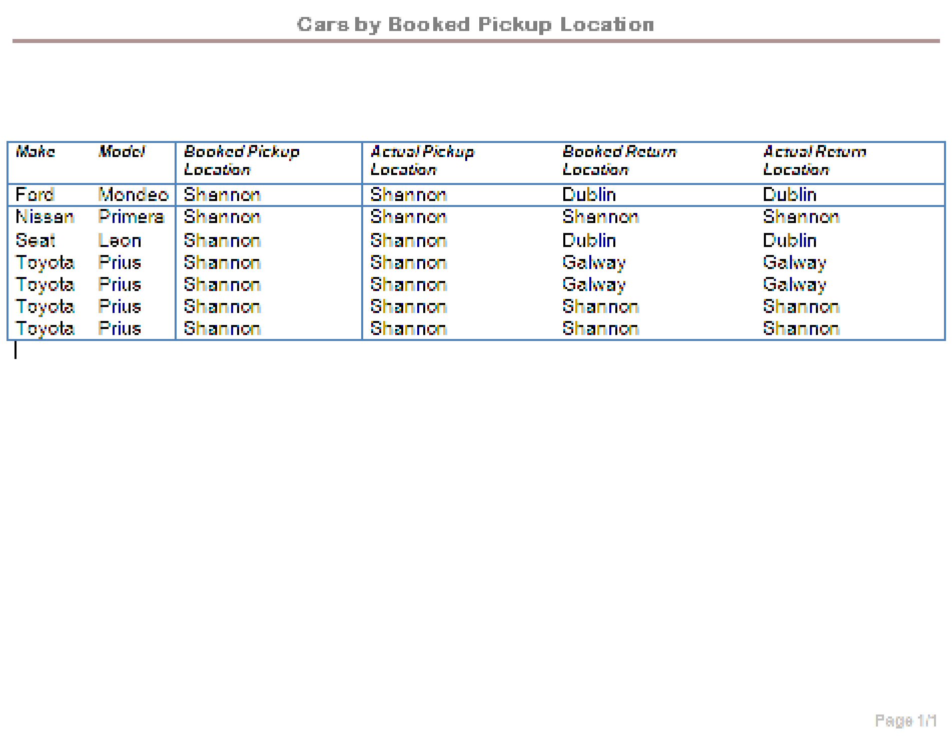
Please choose all that apply: Get Total Reset

Here the customer can choose any extras they need and request the total price. Again if there is a need, they can choose the reset button and go back to the previous screen.

The layout of each screen will be similar for all the customer screens and for the staff screens also.

Staff can access reports based on the transactions carried out, filtered by whatever criteria they need. Some examples follow:

Cars by Location

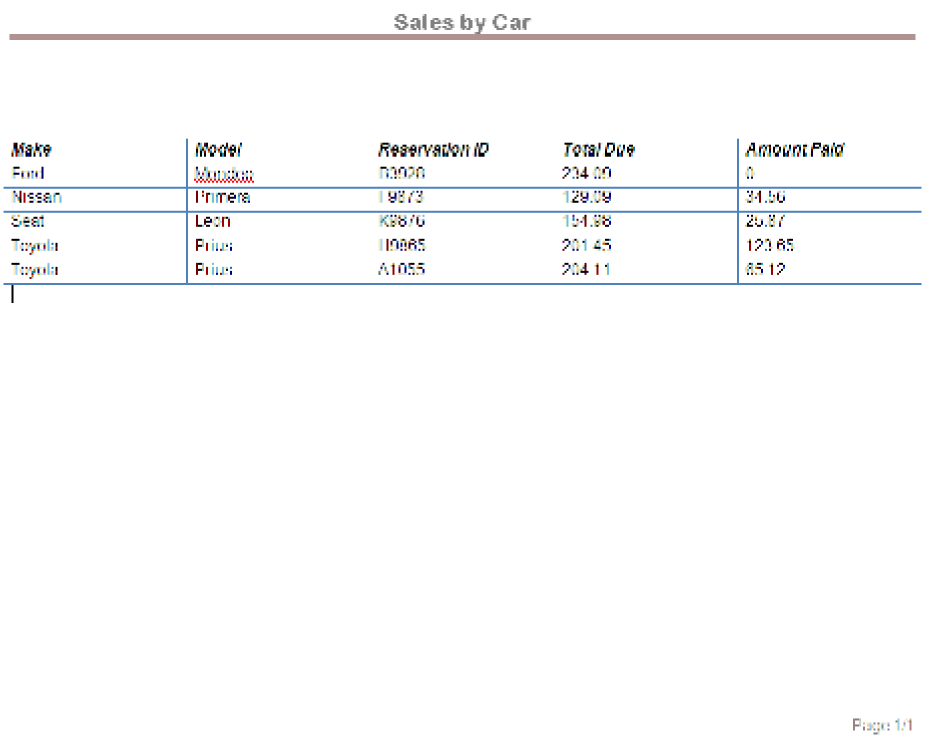


Cars by Booked Pickup Location

Make	Model	Booked Pickup Location	Actual Pickup Location	Booked Return Location	Actual Return Location
Ford	Mondeo	Shannon	Shannon	Dublin	Dublin
Nissan	Primera	Shannon	Shannon	Shannon	Shannon
Seat	Leon	Shannon	Shannon	Dublin	Dublin
Toyota	Prius	Shannon	Shannon	Galway	Galway
Toyota	Prius	Shannon	Shannon	Galway	Galway
Toyota	Prius	Shannon	Shannon	Shannon	Shannon
Toyota	Prius	Shannon	Shannon	Shannon	Shannon

Page 1/1

Sales by Car

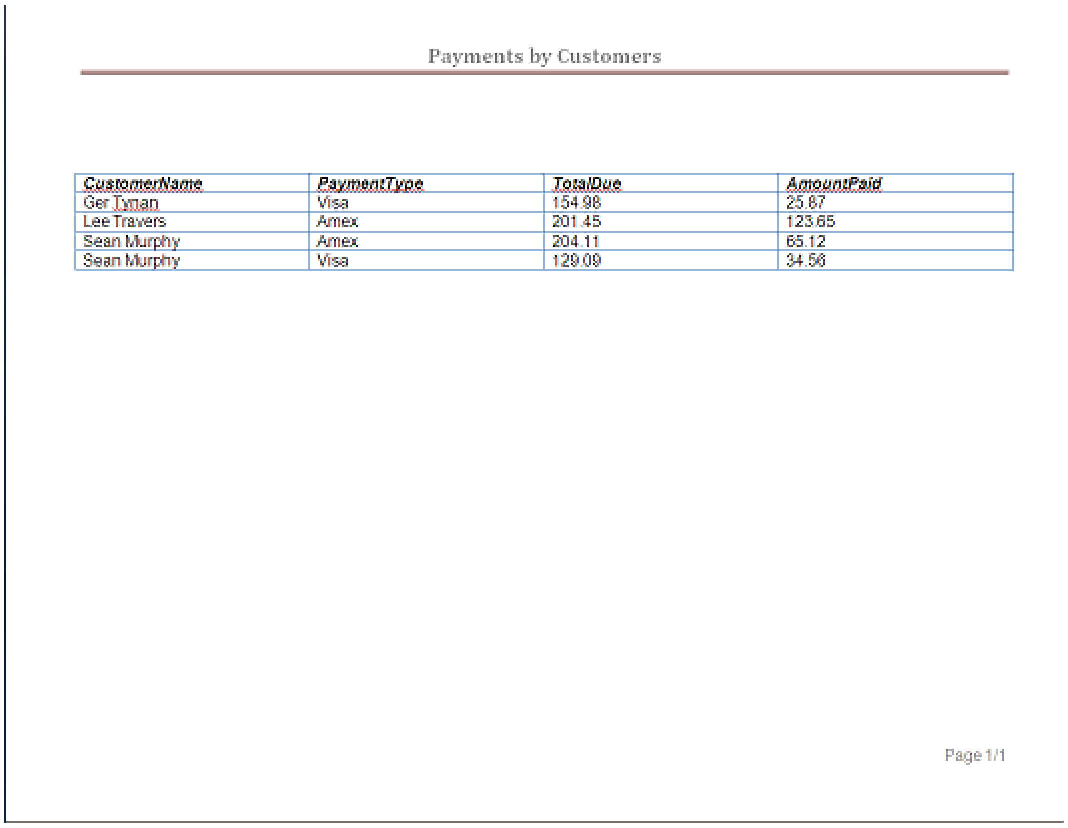


Sales by Car

Make	Model	Reservation ID	Total Due	Amount Paid
Ford	Mondeo	D3020	234.00	0
Nissan	Primera	L9873	129.09	34.56
Seat	Leon	K9870	154.98	26.87
Toyota	Prius	H0965	201.45	123.65
Toyota	Prius	A1055	204.11	65.12

Page 1/1

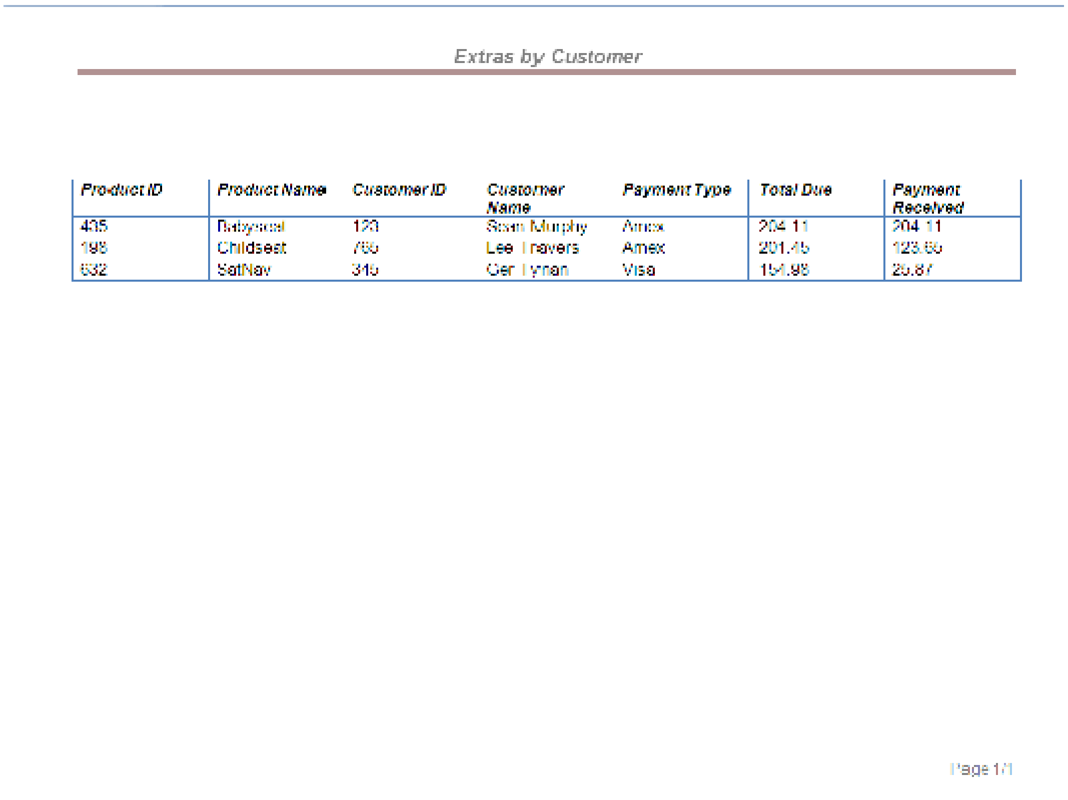
Payments by Customer


Payments by Customers

CustomerName	PaymentType	TotalDue	AmountPaid
Ger Tynan	Visa	154.98	25.87
Lee Travers	Amex	201.45	123.65
Sean Murphy	Amex	204.11	65.12
Sean Murphy	Visa	129.09	34.56

Page 1/1

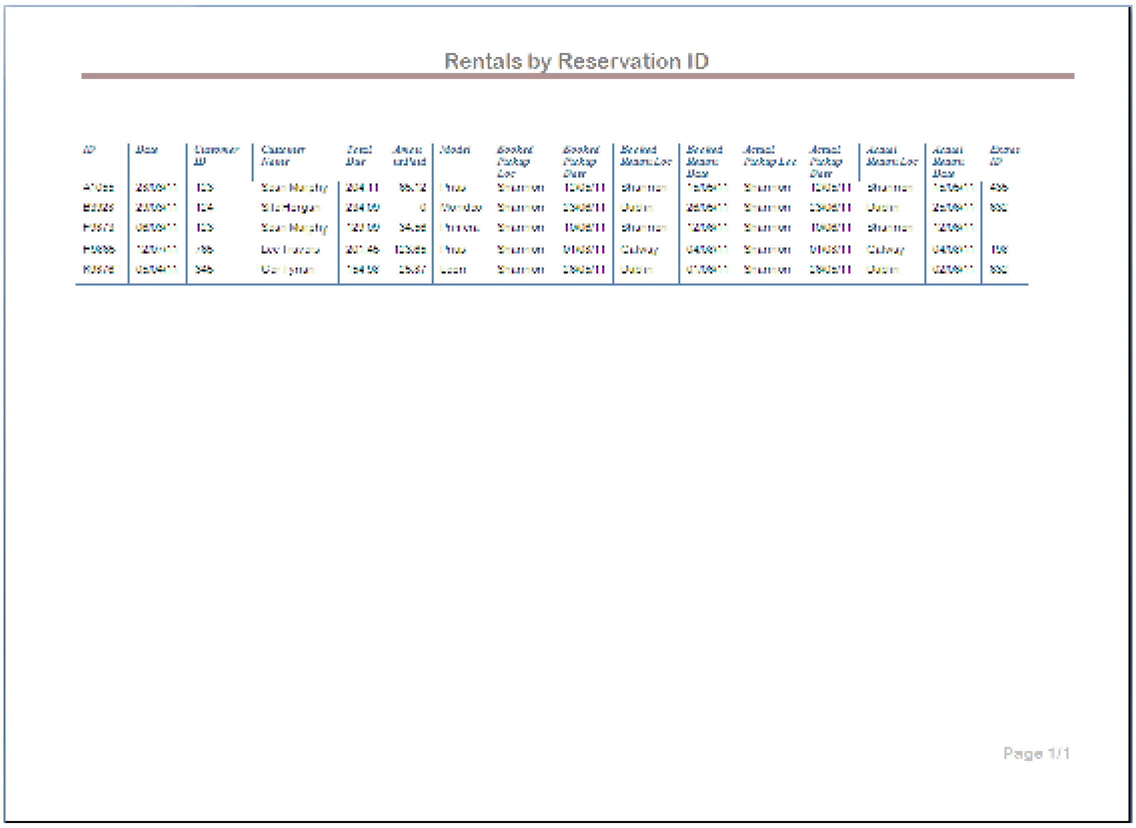
Extras by Customer


Extras by Customer

Product ID	*Product Name*	*Customer ID*	*Customer Name*	*Payment Type*	*Total Due*	*Payment Received*
435	Babyseat	123	Sean Murphy	Amex	204.11	204.11
198	Childseat	766	Lee Travers	Amex	201.45	123.65
632	SatNav	345	Ger Tynan	Visa	154.98	25.87

Page 1/1

Rentals by Reservation ID

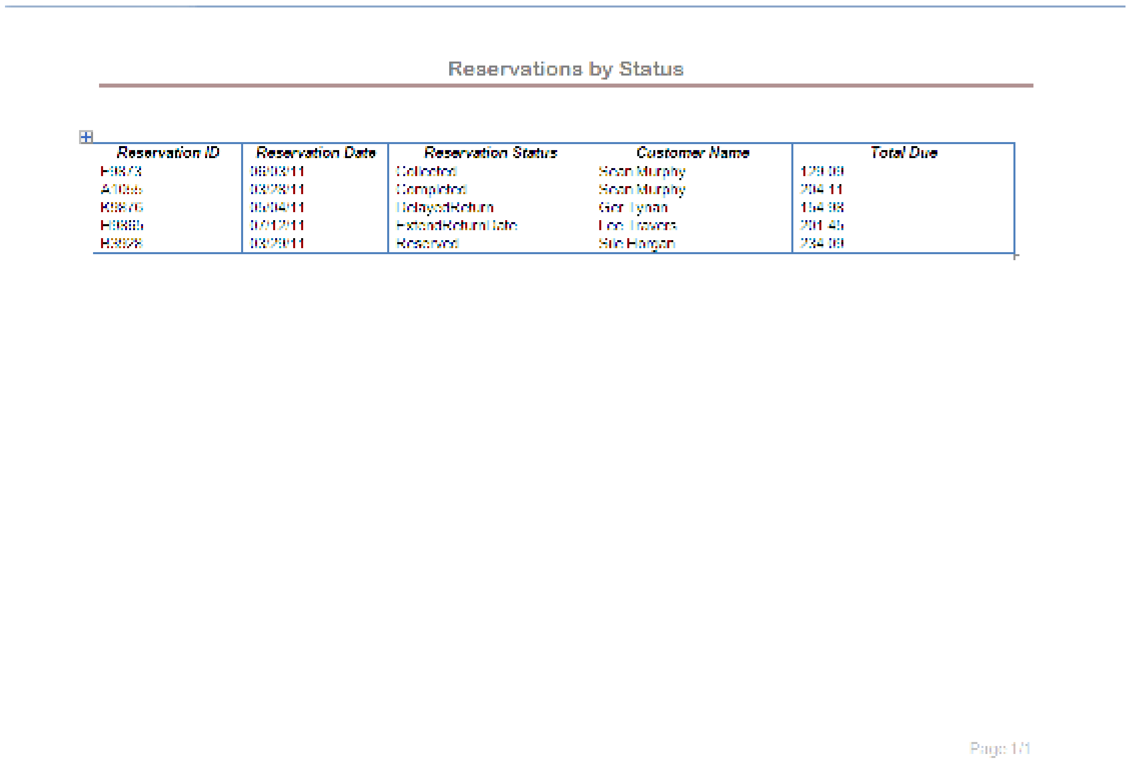


Rentals by Reservation ID

ID	Date	Customer ID	Customer Name	Total Due	Amount Paid	Model	Booked Pickup Loc	Booked Pickup Date	Booked Return Loc	Booked Return Date	Actual Pickup Loc	Actual Pickup Date	Actual Return Loc	Actual Return Date	Driver ID
A1066	23/03/11	123	Sean Murphy	204.11	65.72	Prius	Shannon	12/05/11	Shannon	15/05/11	Shannon	12/05/11	Shannon	15/05/11	435
B3028	23/03/11	124	Sile Hargan	234.00	0	Mondeo	Shannon	23/08/11	Dublin	26/08/11	Shannon	23/08/11	Dublin	25/08/11	632
F9873	08/03/11	123	Sean Murphy	129.00	34.98	Primera	Shannon	10/08/11	Shannon	12/08/11	Shannon	10/08/11	Shannon	12/08/11	
H9886	12/01/11	785	Lee Travers	201.45	123.65	Prius	Shannon	01/08/11	Galway	04/08/11	Shannon	01/08/11	Galway	04/08/11	158
K9876	05/04/11	345	Ger Lynan	154.08	25.87	Leon	Shannon	29/08/11	Dublin	01/09/11	Shannon	29/08/11	Dublin	02/09/11	632

Page 1/1

Reservations by Status



Reservations by Status

Reservation ID	Reservation Date	Reservation Status	Customer Name	Total Due
F9873	08/03/11	Collected	Sean Murphy	129.00
A1066	03/23/11	Completed	Sean Murphy	204.11
K9876	05/04/11	DelayedReturn	Ger Lynan	154.08
H9886	01/12/11	ExtendReturnDate	Lee Travers	201.45
B3028	03/23/11	Reserved	Sile Hargan	234.00

Page 1/1

6 Analysis

6.1 Candidate Classes

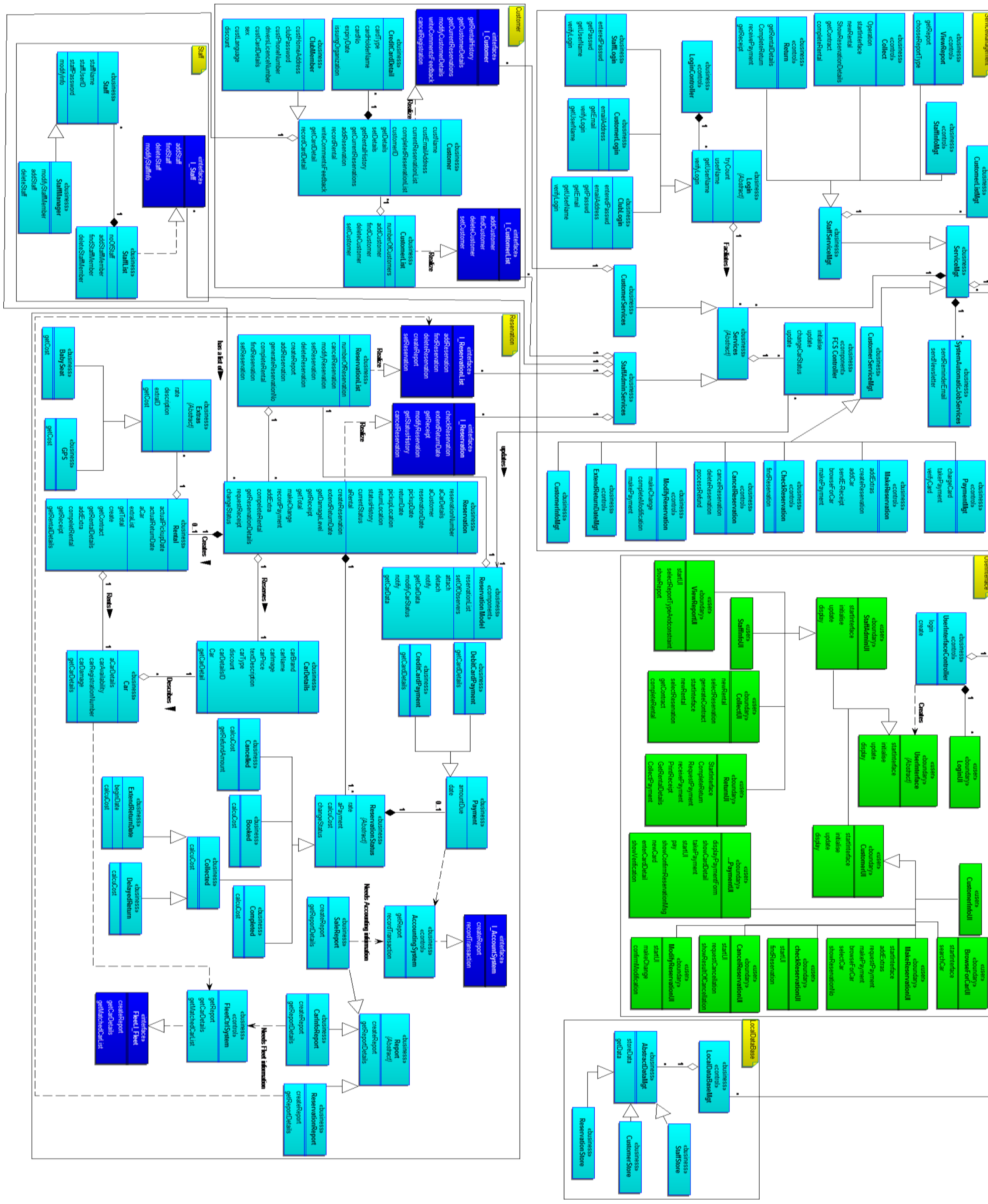
We used the noun identification technique for identifying initial candidate classes. By examining the use case descriptions we came up with the following list:

Customer	Pick-Up Date	User Interface
Car	Reservations	Reports
Rental Details	Current Date	Printers
Customers Personal Information	Details	Data
Payment	Email Address	Customer's Reservation Number
Reservation	Club Scheme	Reservation Details
Receipt	Details	Contract
Club Member	Club Level	Rental Information
Actor	Club Membership	Car Key
Quote	Members	Rental
Credit Card Company	Confirmation	Car Conditions
Accounting System	Message	Total
Actor	Membership	Transaction Details
Fleet Control System	Personal Details	Comments And Feedback
Refund	Customer's Information	Comments
Amount Of Money	Database	Website
E-Receipt	Staff	Fields
Reminder Email	User Name	Updated Information
Day	Password	Registration

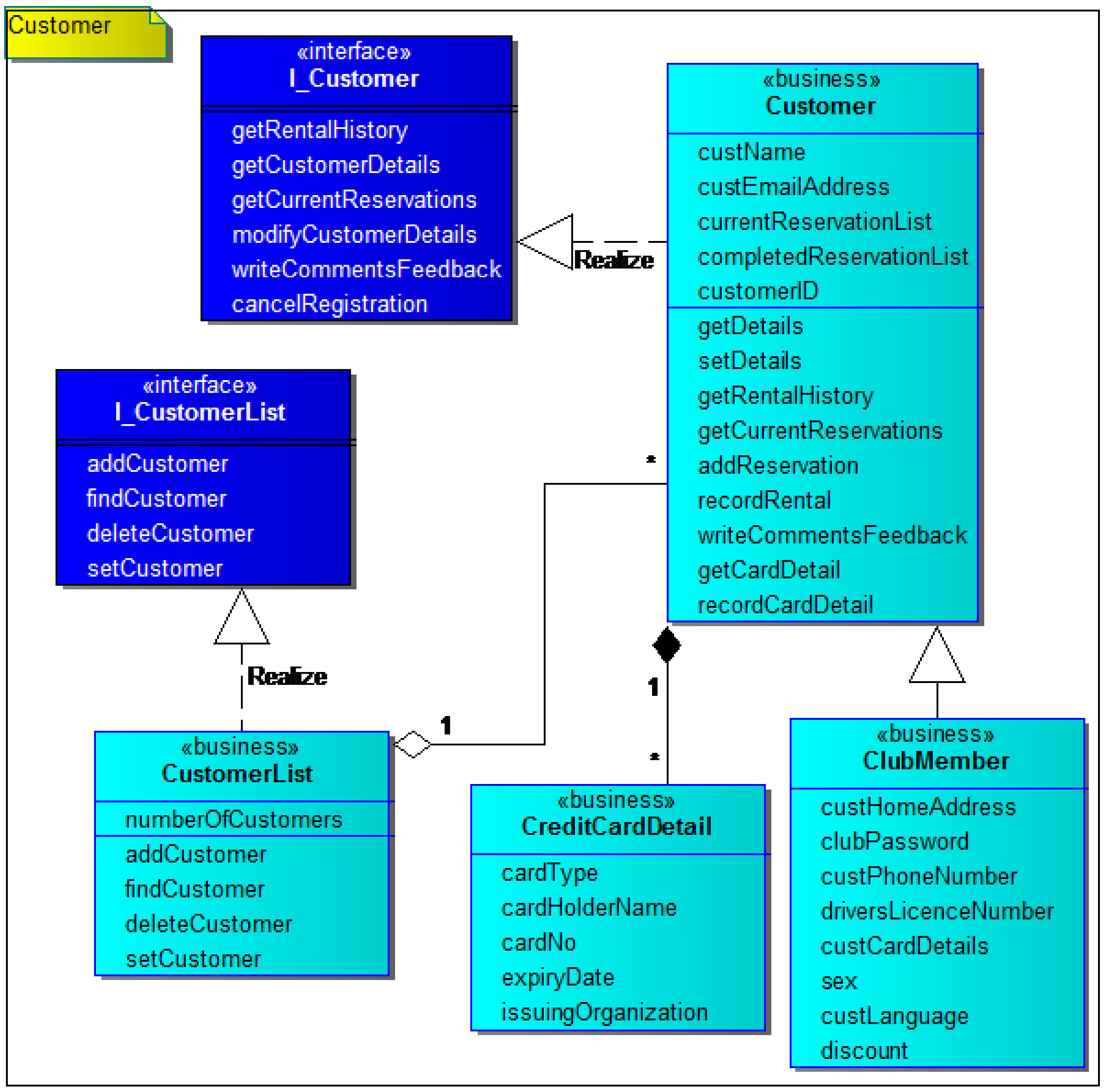
As there is a great deal of duplication and extraneous candidates in this list we further evaluated these, before refining again during the design phase to end up with the final list of classes.

6.2 Class Diagram

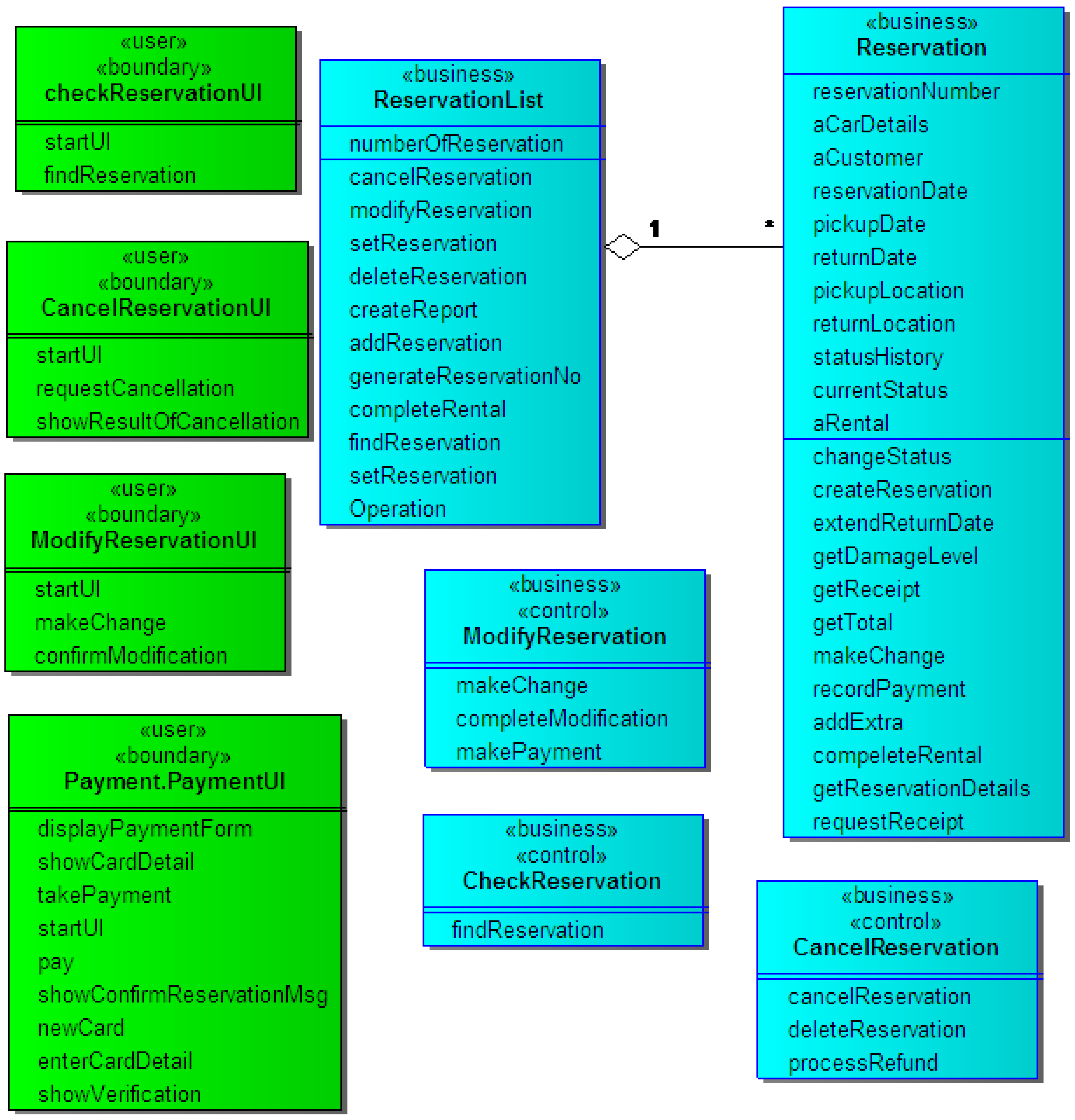
6.2.1 Overall Class Diagram



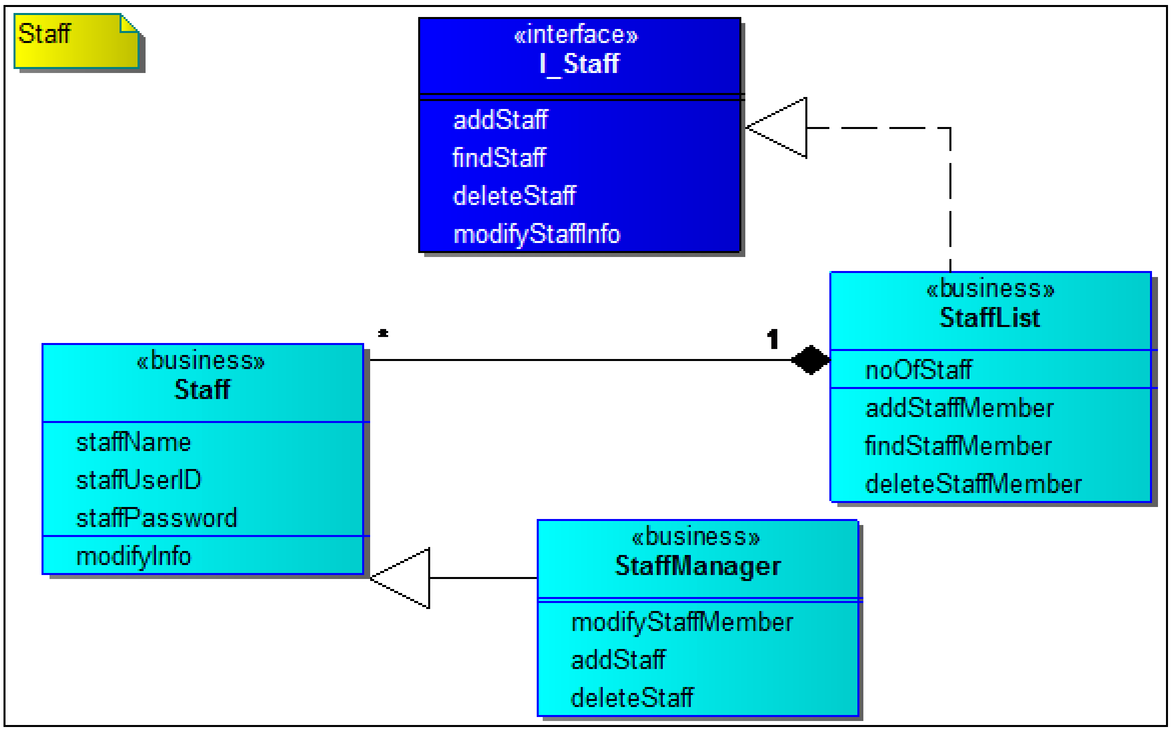
6.2.2 Customer Information



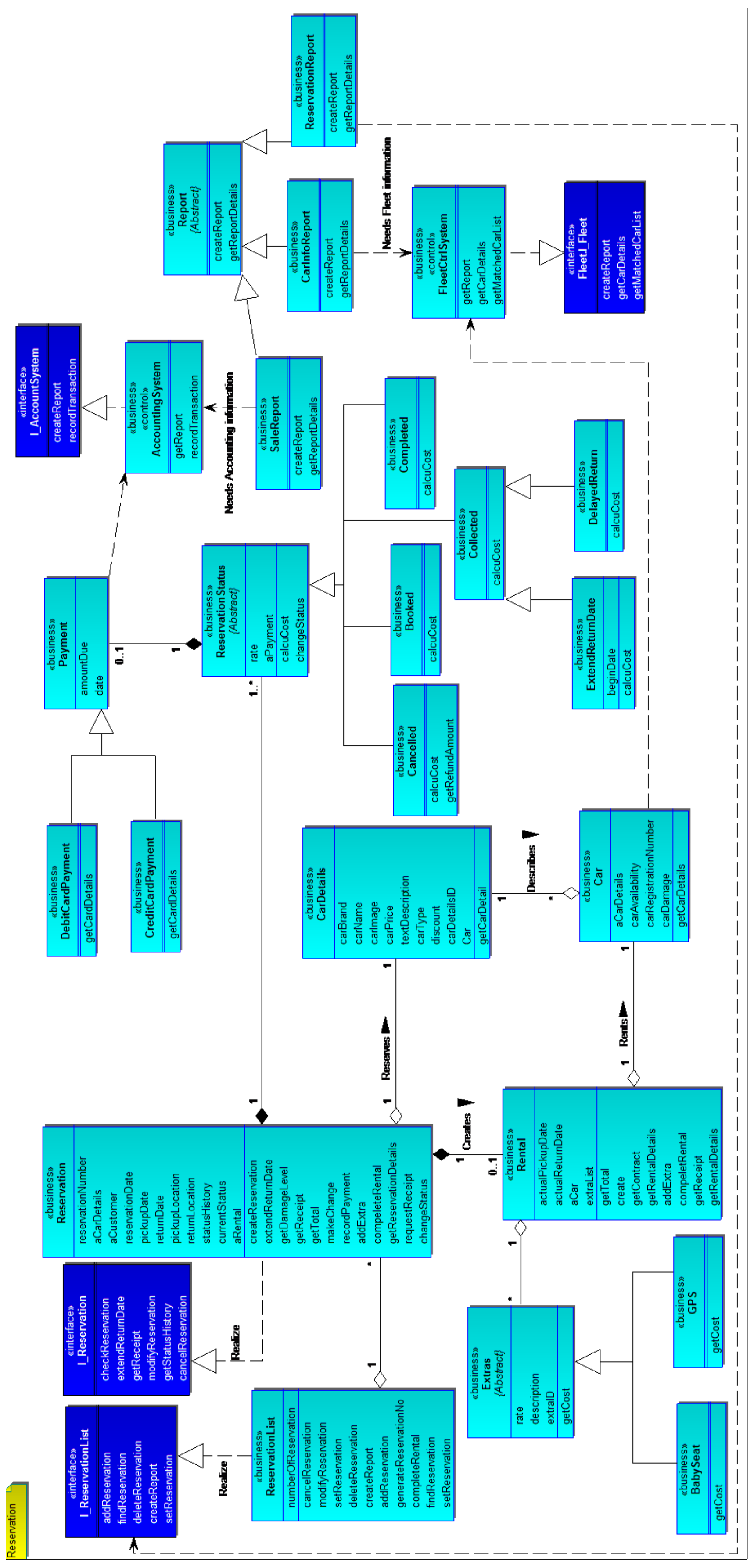
6.2.3 Amend Reservation



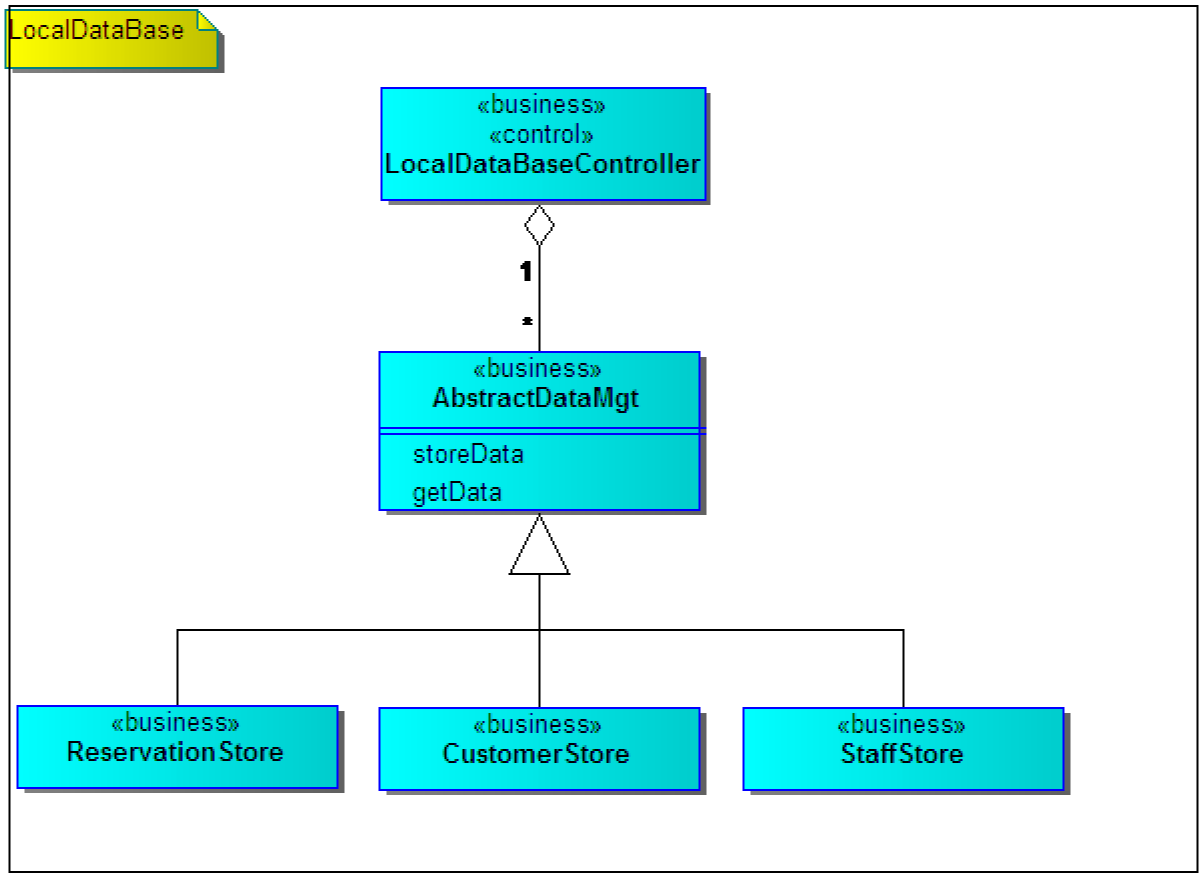
6.2.4 Staff Administration



6.2.5 Reservation Class Diagram

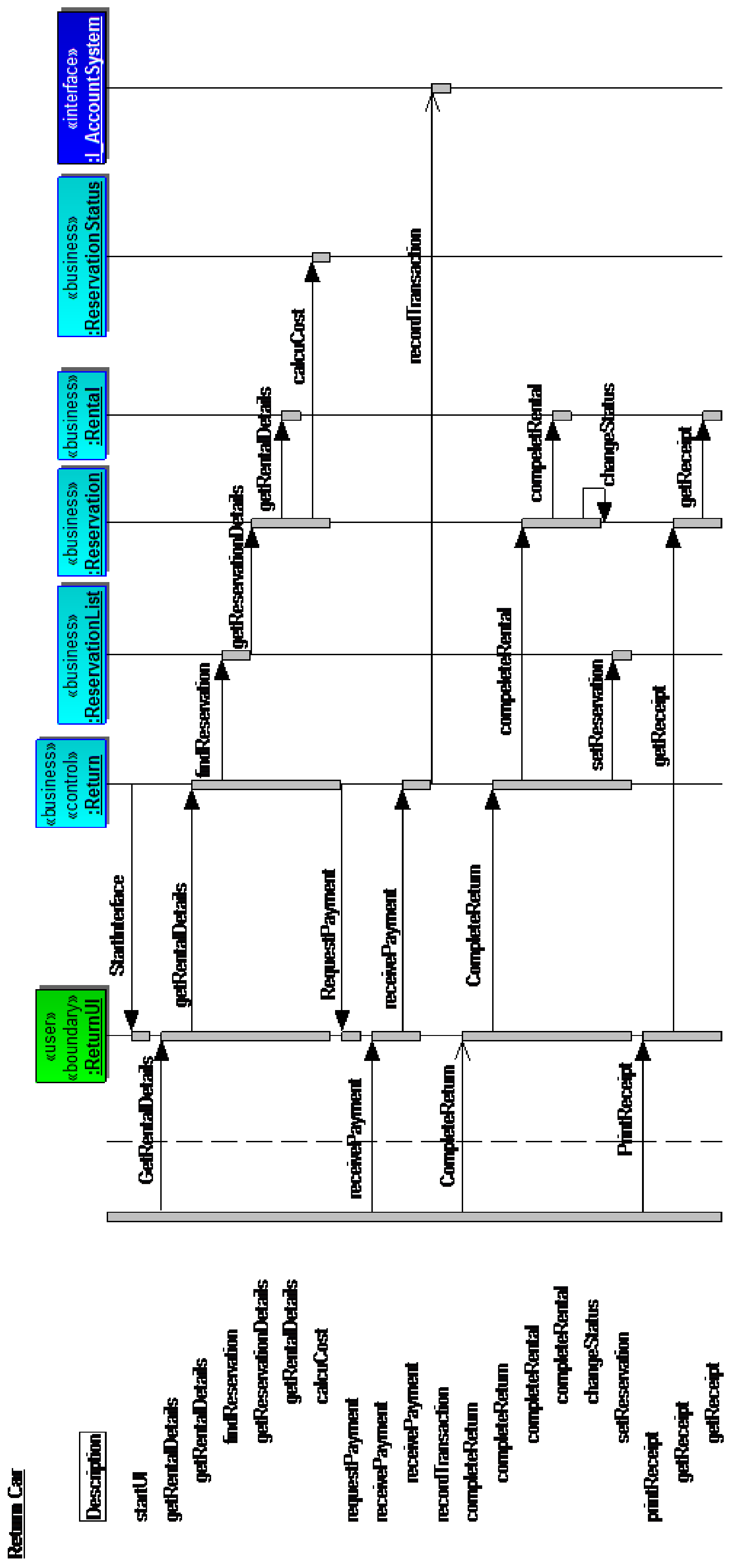


6.2.6 Local Database



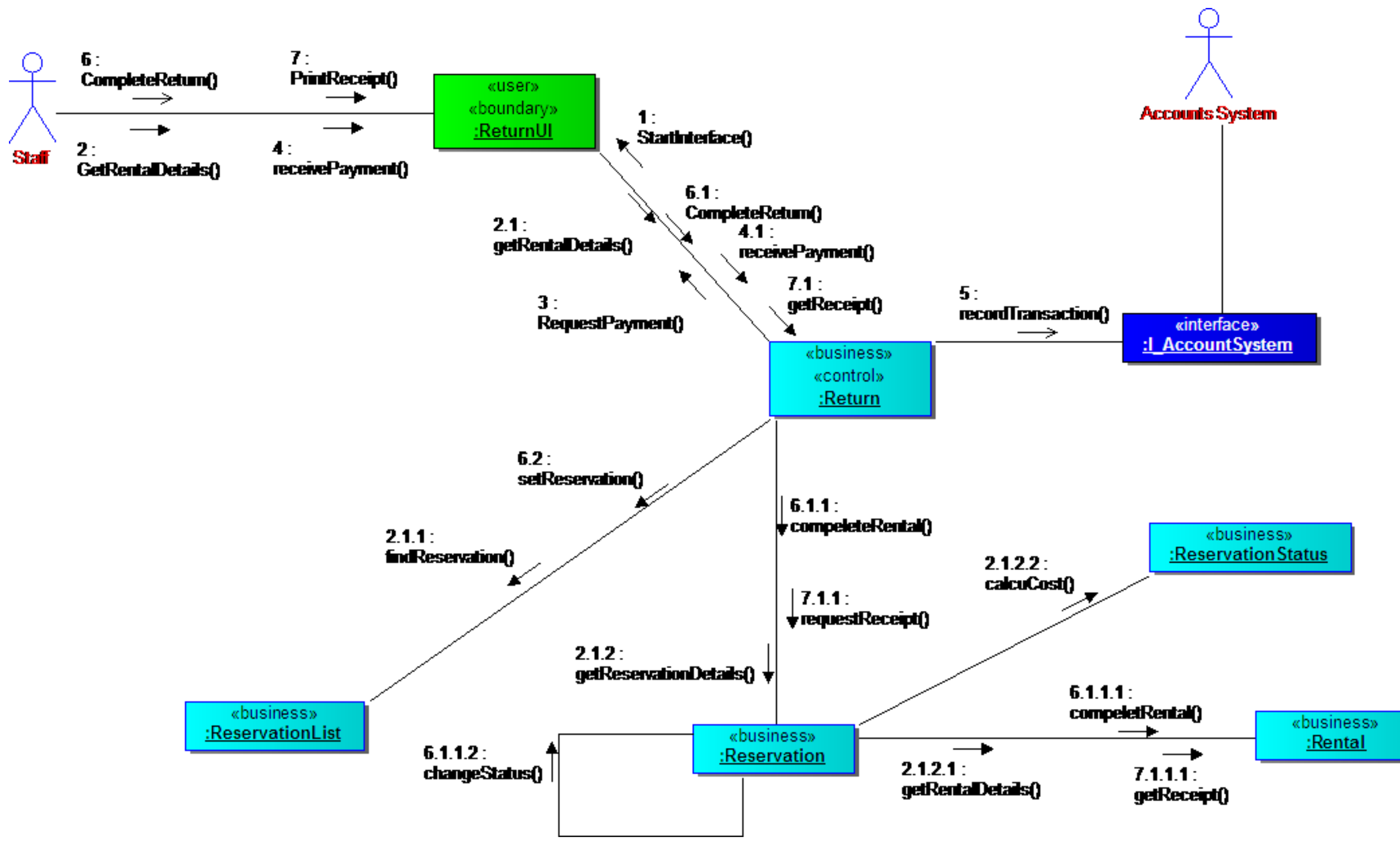
6.3 Sequence Diagram

Return car sequence diagram:



6.4 Communication Diagram

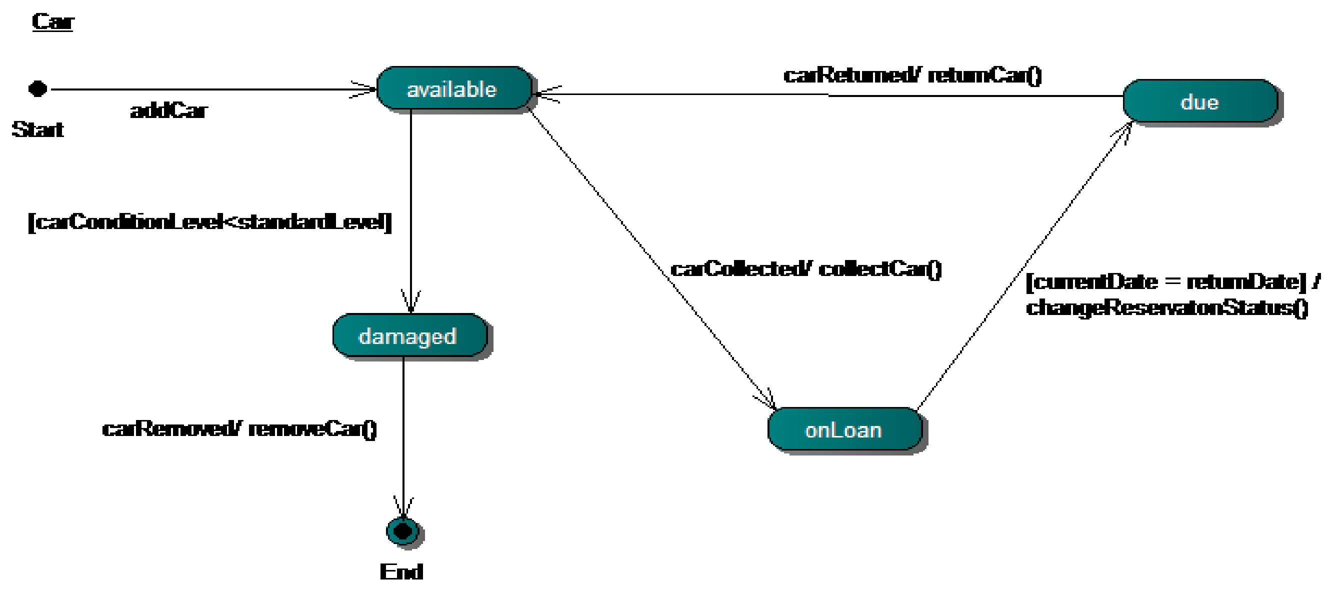
Return car:



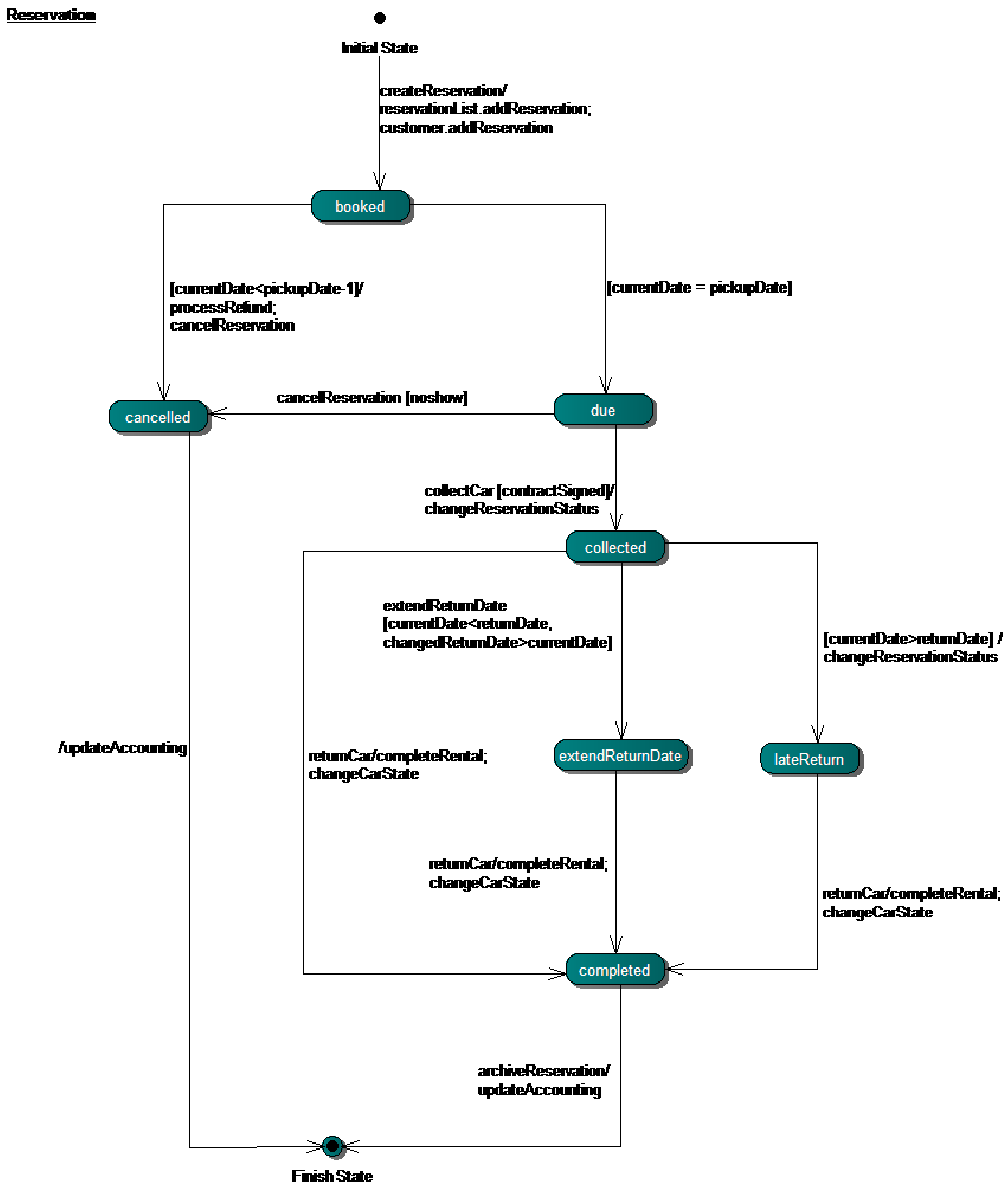
6.5 State Chart

UML uses state machines to model states and state dependent behaviour for objects' and for interactions. All objects have a state and a state describes a particular condition that a modelled object may occupy for a period of time while it awaits some event or trigger. Here we present state chart for the class car and reservation.

6.5.1 Car State Chart


Car
Start
addCar
available
carReturned/ returnCar()
due
[carConditionLevel<standardLevel]
carCollected/ collectCar()
[currentDate = returnDate] / changeReservatonStatus()
damaged
carRemoved/ removeCar()
onLoan
End

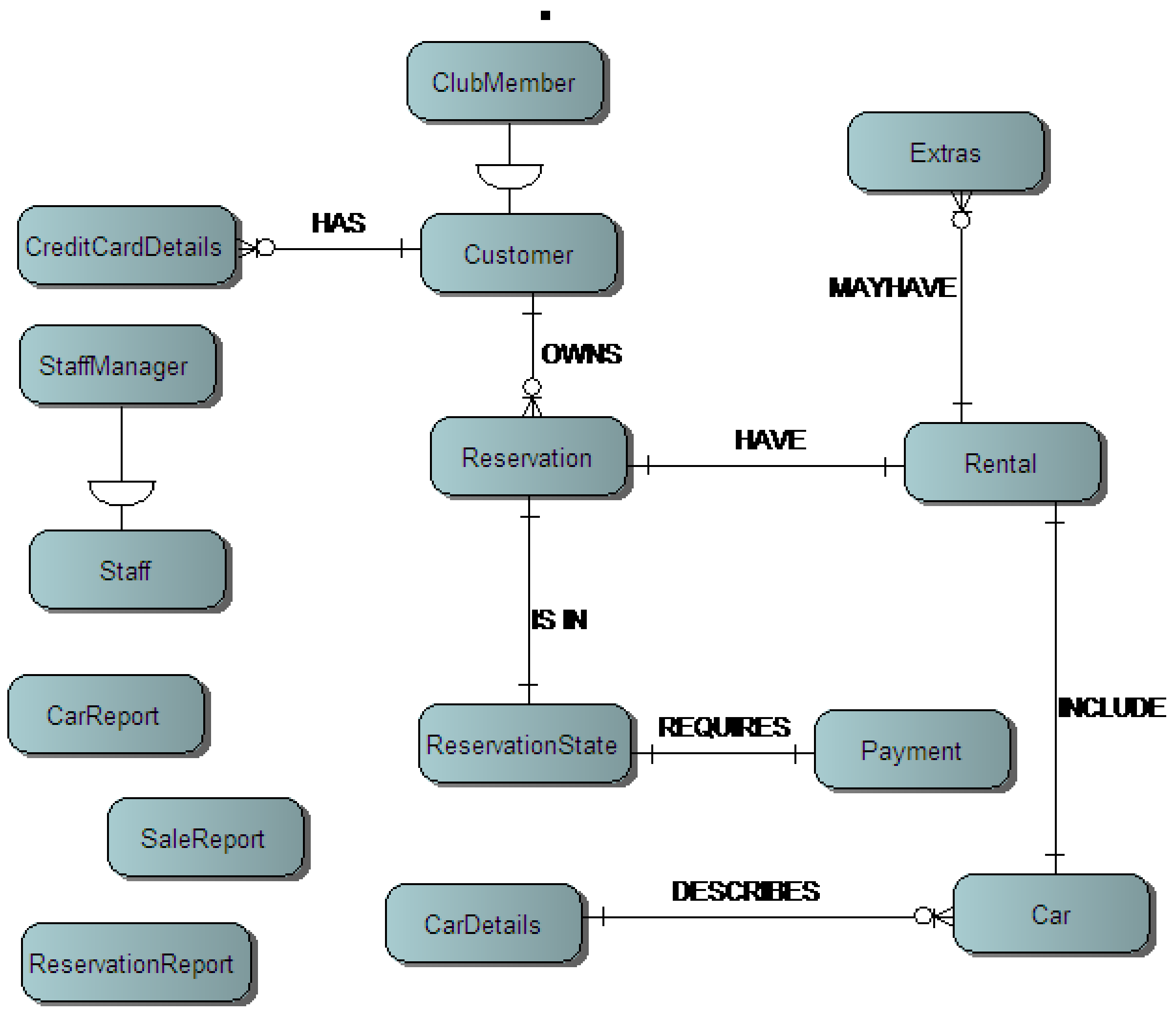
6.5.2 Reservation State Chart


Reservation
Initial State
createReservation/ reservationList.addReservation; customer.addReservation
booked
[currentDate<pickupDate-1]/ processRefund; cancelReservation
[currentDate = pickupDate]
cancelled
cancelReservation [noshow]
due
collectCar [contractSigned]/ changeReservationStatus
collected
extendReturnDate [currentDate<returnDate, changedReturnDate>currentDate]
[currentDate>returnDate] / changeReservationStatus
/updateAccounting
returnCar/completeRental; changeCarState
extendReturnDate
lateReturn
returnCar/completeRental; changeCarState
returnCar/completeRental; changeCarState
completed
archiveReservation/ updateAccounting
Finish State

6.6 Entity Relationship Diagram

Here the entity relationship diagram shows the relationships between data objects within the Car Hire System. The diagram is quite solid for the main entities which are:

- Reservation
- Rental
- Customer



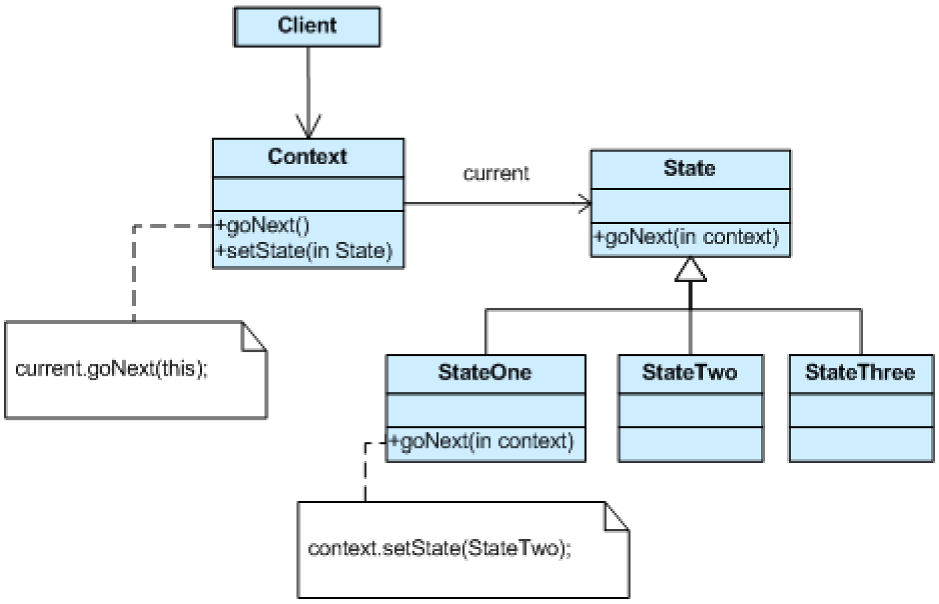
7 Design

7.1 Discussion of Evaluated Patterns

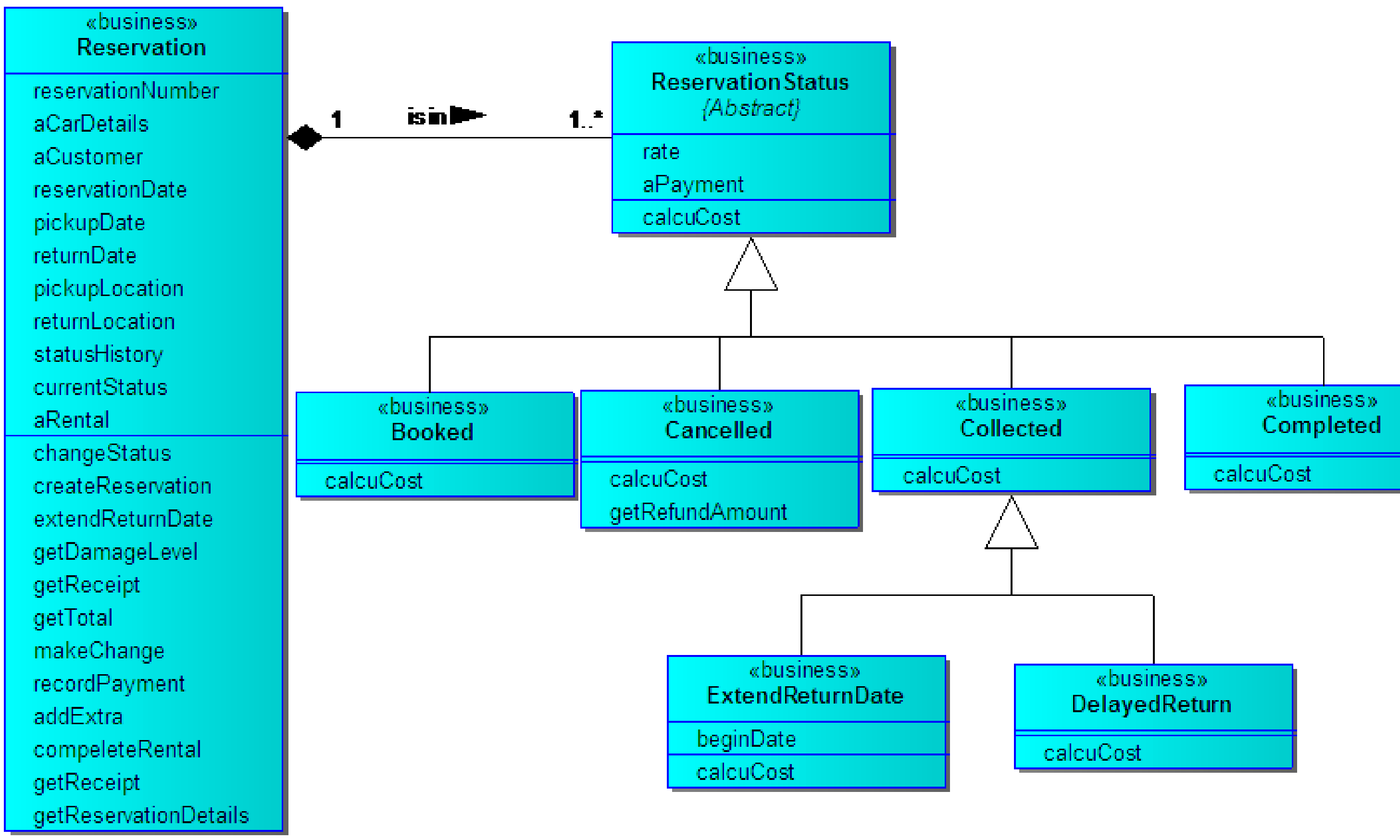
Design patterns describe the core of simple and elegant solutions to specific software design problems in a particular context. They captures solutions that have developed and evolved over time. Our team evaluated many of the design patterns from the Gang of Four in their book Design Patterns. Many patterns are seemed to be applicable for the Car Hire System but here we chose some of the most related patterns evaluated to discuss.

7.1.1 The State Pattern

The state pattern facilitates object to have state-specific behaviour depending on its current state. It enables the object to be appeared to change its class to alter its behaviour at run-time. The state pattern isolates the divergent functionality into swappable classes implementing a common interface or an abstract class. The main object simply delegates to its internal state object at the appropriate times. This pattern is better to be applied in cases where an object's behaviour depends on its state and it must change its behaviour at run-time. State pattern makes state transitions explicit and reduces duplication by eliminating repeated if/else or switch statement. It increase cohesion by delegation and enhance extensibility.

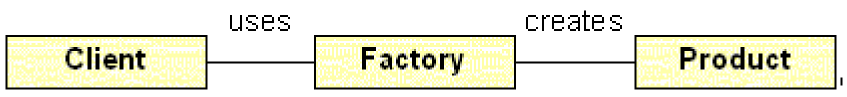


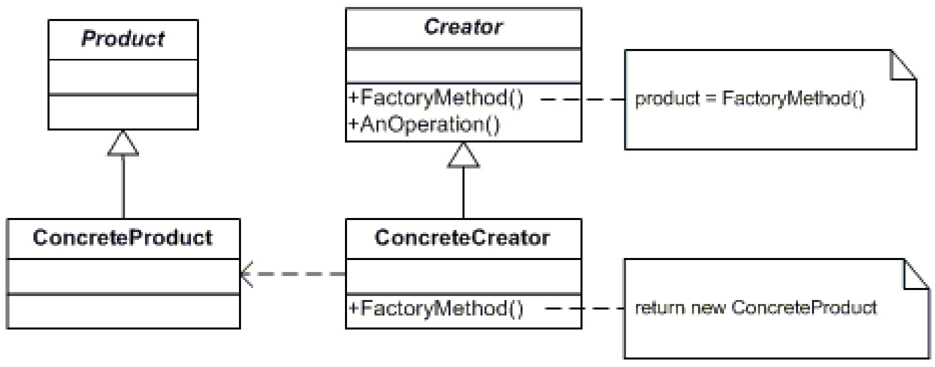
Due to the different reservation statuses resulting in different rate of charge, it was decided to apply the state pattern in the Car Hire System. This largely helps the system to keep a record of the processes of a reservation, i.e. which stages has the reservation being through and the status history kept in a reservation helps to calculate the final total. After a customer make a reservation, the reservation is at booked status that will calculate the payment (deposit)at this stage. When it is cancelled by customer, the reservation is in cancelled status which will help to process refund instead as it will calculate the refund amount. While the reservation is in lateReturn or extendReturnDate status, it can calculate the total according to how long it has been in that status and the rate for that status.



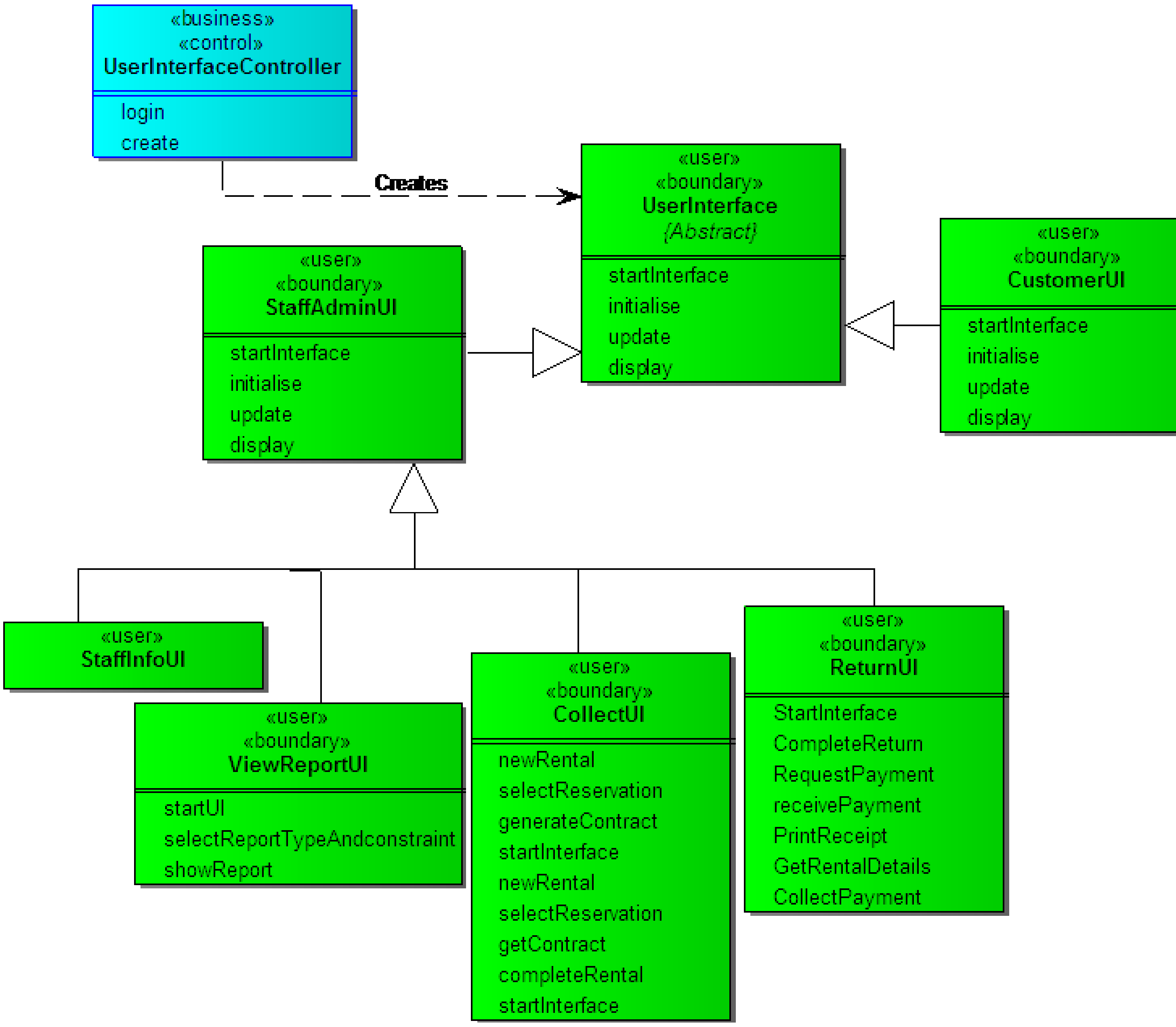
7.1.2 The Factory Pattern

The factory pattern is widely used in software development. Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory pattern lets a class defer instantiation to subclasses. It refers to the newly created object through a common interface.





When considering our Car Hire System, we apply this pattern to do the user interface job. In different use cases there are different user interfaces needed and also each one corresponding to a specific controller class. Here we use the factory pattern to do the matching, , and we don't bother which controller starts which user interface. The following figure is part of its use in our Car Hire System:



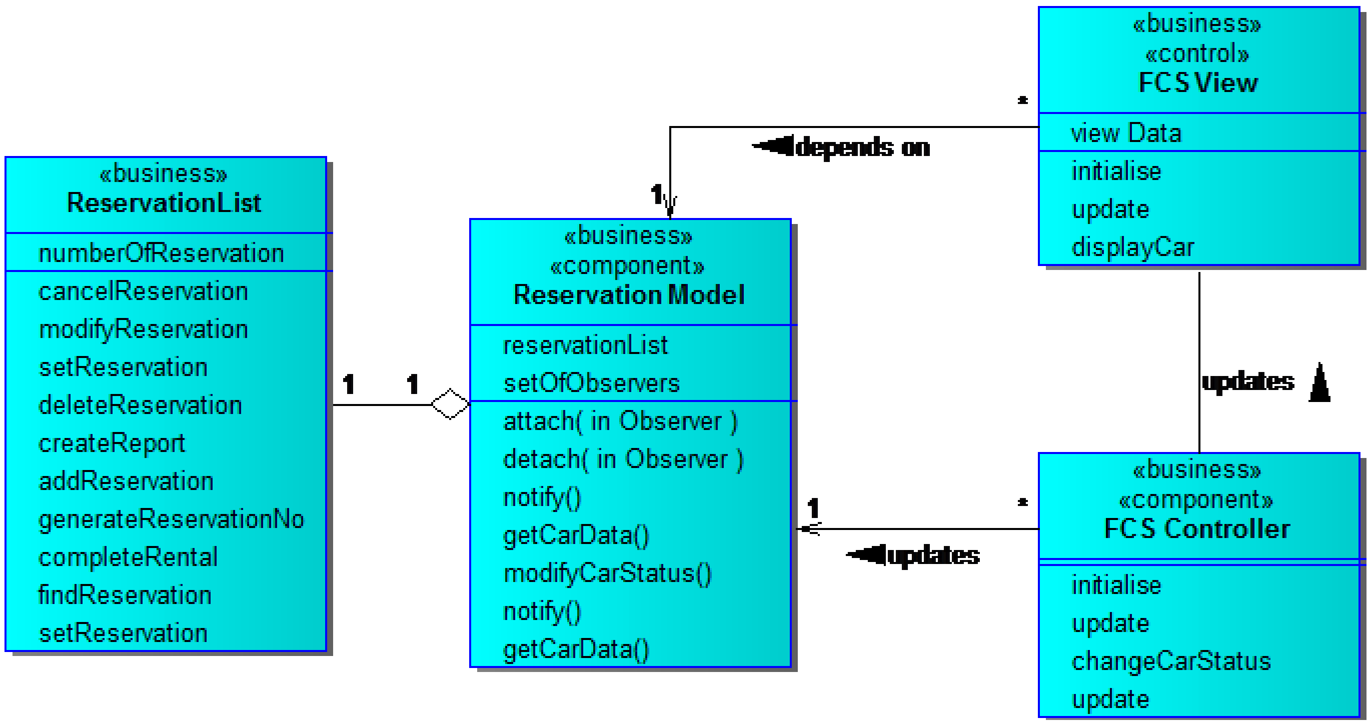
7.2 Model View Controller Architectural Pattern

Model-View-Controller is an architectural pattern that decouples views from models and thus increases flexibility and reusability. The MVC pattern is used when software is needed to represent the same data to different clients and also allows you change the way a view responds to user input without changing its visual presentation. It encapsulates the response mechanism in a Controller object. Both the view and the controller depend on the model. The idea behind MVC is that is establishing a subscribe/notify protocol between views and models.

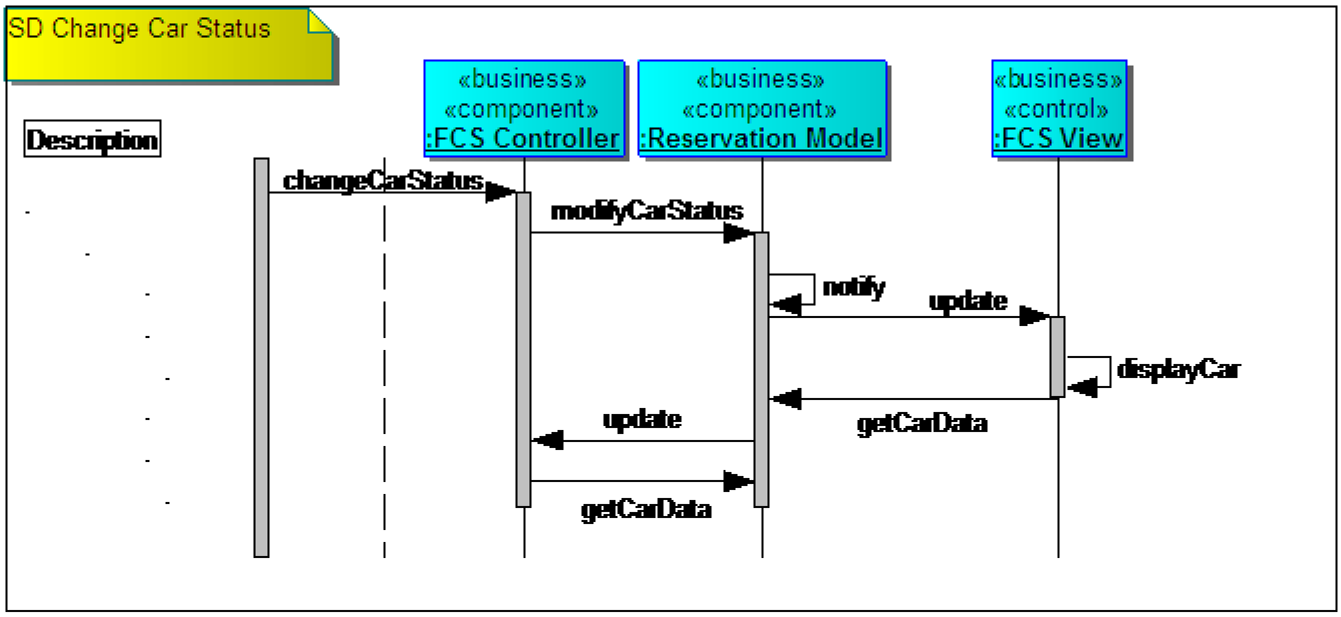
In our system, the car information, where most importantly is its availability, intuitively has many views to different users such as the operator of Fleet Control System and customer browsing car on-line. The change of car state is caused by reservation status. Thus we apply this pattern firstly to the reservation model.

Ideally we should use a class hierarchy of controllers and viewers to make it easy to create a new controller as there are some other situations need the reservation model but to limit the scope we just apply the MVC to the reservation model and Fleet Control System view.

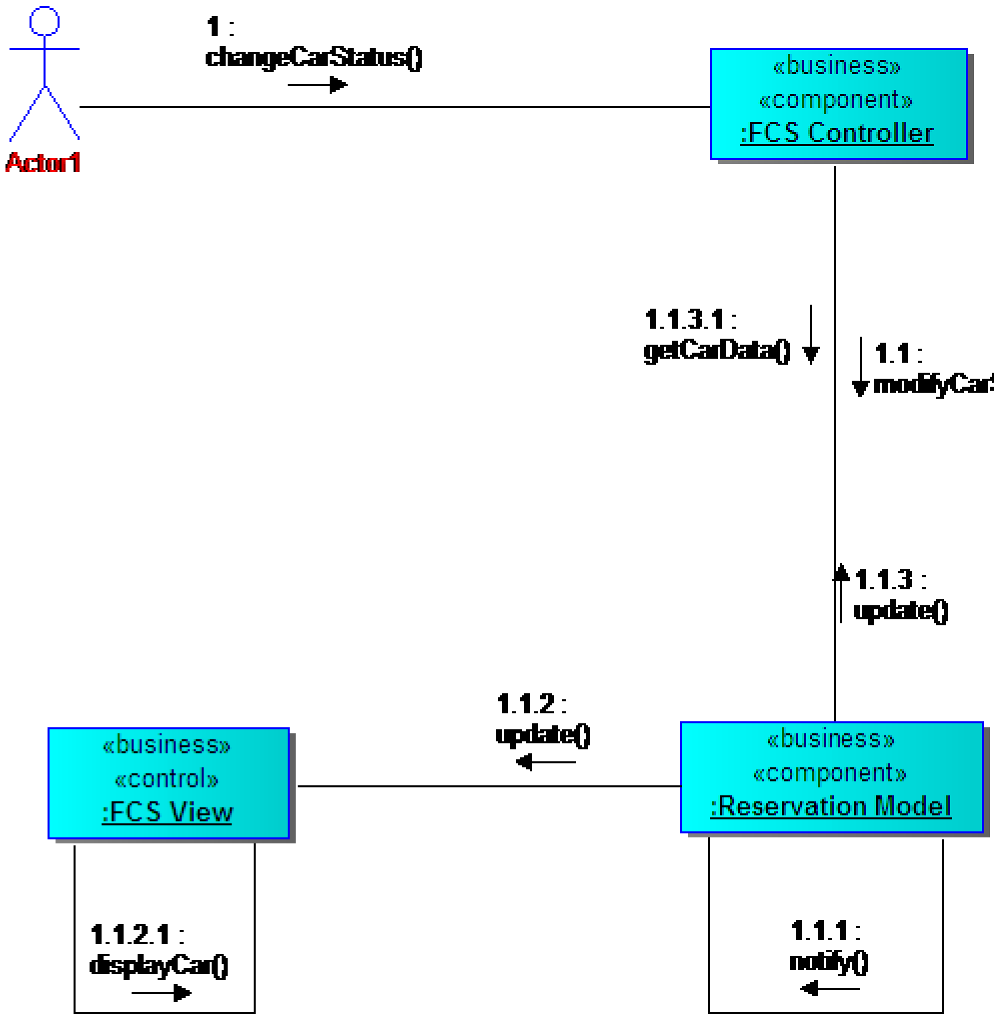
7.2.1 Class Diagram



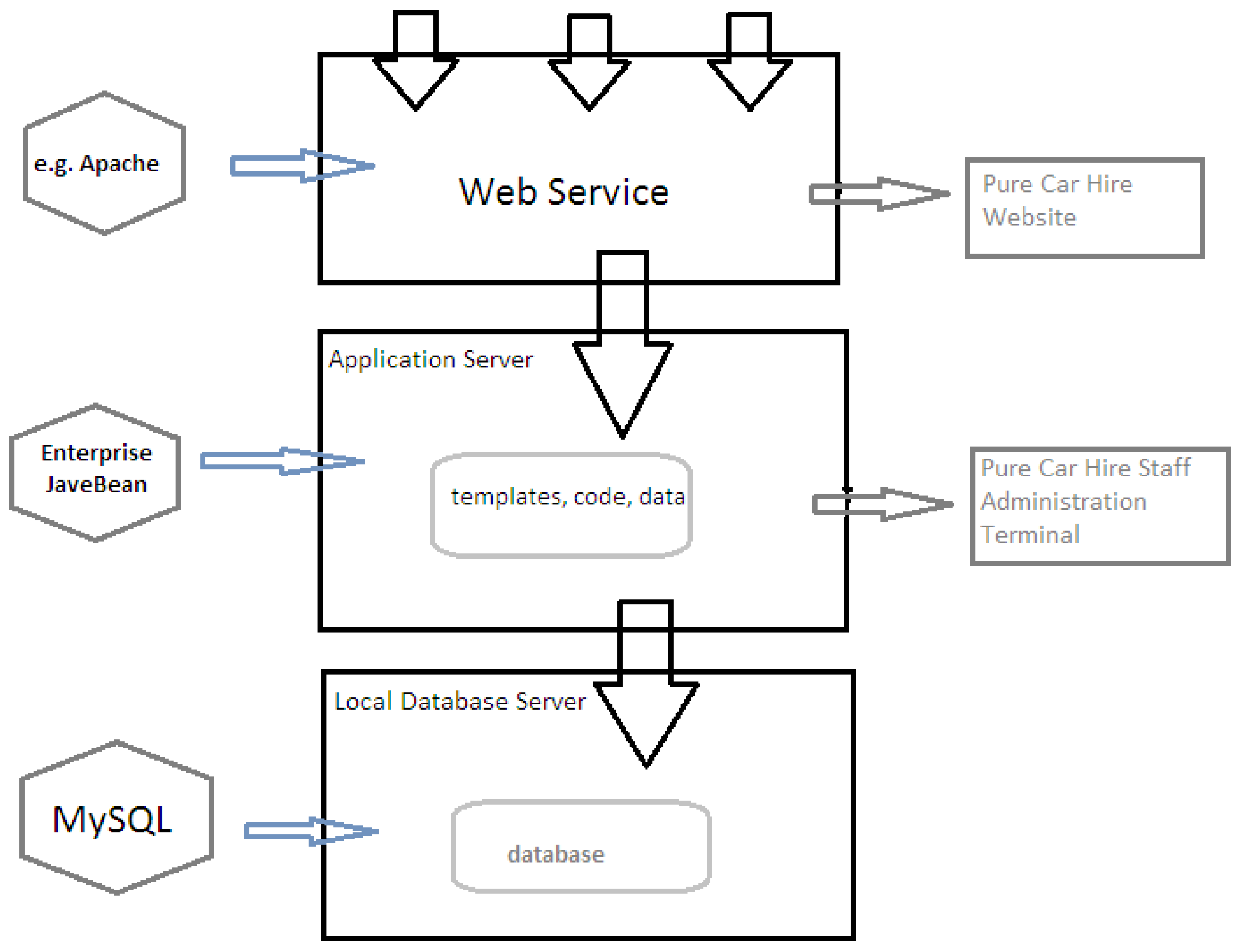
7.2.2 Sequence Diagram


SD Change Car Status
«business» «component» :FCS Controller
«business» «component» :Reservation Model
«business» «control» :FCS View
Description
changeCarStatus
modifyCarStatus
notify
update
displayCar
getCarData
update
getCarData

7.2.3 Communication Diagram



8 Technical Architecture



9 Data Dictionary

Name: Accounts System
Type: External Actor
Description: Accounting System records all the transaction records.

Name: Credit Card Company
Type: External Actor
Description: Authorises credit Card payments.

Name: Customer
Type: Actor
Description: Customer is either a Club Member or someone who has reserved a Car.

Name: Fleet Control System
Type: External Actor
Description: Fleet Control System records all details of Cars in the company. It is also responsible for scheduling the Cars.

Name: Staff
Type: Actor
Description: Staff processes Collections & Returns and views Reports.

Name: StaffManager
Type: Actor
Description: StaffManager manages the deployment of staff, adding staff etc.

Name: Creates [Reservation - Rental]
Type: Association
Description: When a Collection is being processed a Rental is created and stored as an attribute of Reservation

Name: Describes [Car - CarDetails]
Type: Association
Description: Several Cars can have similar CarDetails but are different instances, the details describe the Car.

Name: has a list of [Customer - Reservation]
Type: Association
Description: Each Customer will have a list of their Reservations

Name: Reserves [Reservation - CarDetails]
Type: Association
Description: Once a Reservation is created it associates to a particular Car type, represented by its CarDetails

Name: currentReservationList [Customer$]
Type: Attribute
Description: A Customers current Reservations

Name: completedReservationList [Customer$]
Type: Attribute
Description: A Customers completed Reservations

Name: currentStatus [Reservation$]
Type: Attribute
Description: The current status of a Reservation, whether completed, booked, collected, etc.

Name: Car
Type: Class
Description: The Car class holds the information about a Car

Name: ClubMember
Type: Class
Description: A specialisation of Customer, to refer to Club Members only

Name: CustomerList
Type: Class
Description: The Customer list class keeps note of how many Customer there are and is used to create/modify/delete Customers

Name: Extras
Type: Class
Description: The Extras class holds the information about an Extra

Name: Rental
Type: Class
Description: The Rentals class holds information about a Rental

Name: Report
Type: Class
Description: The Report class holds information about a Report that the Staff can generate

Name: Reservation
Type: Class
Description: The Reservation class holds the information about a Reservation from creation to addition of Rental and on to completion

Name: ReservationList
Type: Class
Description: The Reservation list class keeps note of how many Reservations there are and at what state they are at. It is also used for searching for a Reservation.

Name: Return
Type: Class
Description: The Return class holds the information for a when a Car is returned

Name: I_Customer [Customer.]
Type: Interface
Description: Provides the functionality for accessing Customer information

Name: I_Fleet [Fleet.]
Type: Interface
Description: Provides the functionality for updating and accessing Fleet information

Name: I_OnlineAuthorisation [OnlineAuthorisation.]
Type: Class
Description: Provides the functionality for accessing Credit Card authorisation information.

Name: StaffList
Type: Class
Description: The StaffList class keeps track of the number of Staff members and is used to create/modify/delete a Staff member.

Name: ReservationStatus
Type: Abstract Class
Description: The Reservation can be in 6 different states – booked, cancelled, collected, completed, ExtendReturnDate and DelayedReturn. The ReservationStatus class is used to show the attributes and methods needed.

10 Critique

We believe the design we have come up with is a good one, with the caveat that it is based on no client input at requirements or design stages – or indeed any stage.
We tried to group packages logically so that there would be a high level of cohesion and to minimise dependency between packages and we think we have achieved this. As we learnt more in this module we amended and updated our design until we were left with this final version.

11 References

Websites:

http://www.avis.ie/ Avis car rental

Pictures:

Screenshot of Extras options
www.mysatnav.ie for satNav picture
www.micksgarage.ie for roof rack picture
www.babyaccessories.ie for baby seat picture
Software Development Life Cycle section
www.f14testing.com for strict Waterfall Model picture
www.buzzle.com for one step revision Waterfall Model picture
www.usep-ic.forumsmotions.com for Spiral Model picture
Agile-Software-Development-Poster-En.pdf by VersionOne Inc
Screenshot of Car Options
www.toyota.com for Yaris picture
Rudolf Stricker (on Wikipedia) for Primera picture

Text:

1. Data Protection Commissioner Ireland, www.dataprotection.ie [Accessed 19/04/2011]
2. Nielsen, J. (1993). Usability Engineering. Morgan Kaufman
3. Gamma, E., et al. (1995). Design Patterns. Addison-Wesley
4. Kent Beck et al (2001), "Manifesto for Agile Software Development". Agile Alliance. http://agilemanifesto.org/ [Retrieved 21/04/2011]