Faculty of
Computing, Engineering and
the Built Environment

BIRMINGHAM CITY
University

Undergraduate Programme
Academic Year 2014-2015
Coursework: Team Project

Module:	*CMP2515 Software Design UG2*
School:	**Computing, Telecommunication and Networks**
Module Co-ordinator:	**Professor Zhiming Liu**
Setup Date:	**20/02/2015**
Submission Date:	**21/04/2015**

Team Number: 23

Team Leader: Pavendeep Kaur Hayre

Team Members:
Anil Jassi
Ajmal Esakhail
Aaron Joseph
Damani Lana
Uzair Shah

Instruction to Students:

The submission of the final report should contain this cover page with details of the team filled in above.

As part of the project management requirement, each team should have a weekly meeting. A weekly project diary should therefore be maintained to record the attendance of the meetings by team members, together with brief notes about the weekly project tasks allocations and how well individual team members meet the deadlines of their project tasks.

The final report should include the project diary in the appendix.

Table of Contents

CMP2515 Software Design UG2 – Team Project
Team 23

Part I

Introduction

This document contains the software design development for Bvis Car Hire Company (BCHC) who would like to replace their current paper-based system with a new computerised system which carries out the same functions as the existing system. Currently, BCHC provides their service of hiring a vehicle, having it serviced when necessary and having it returned and recorded in a diary. Therefore, this means creating duplicate records if the same car has been hired more than once and if the same customer returns. Due to this, BCHC require a computerized system that allows them to keep track of which vehicles have been hired and which are still within the fleet, when a vehicle needs to be serviced and also to have the customer details stored on the system making it easier for when they return to hire a new vehicle. By allowing this, it will reduce time for the staff when making a transaction as well as having to look up details about a certain registered customer or vehicle. A new system would also help when the management need to lookup details about a certain mechanic and to ensure that they all hold valid driving licences.

Existing System

BCHC currently works with a manual paper-based system for its hire transactions, vehicle records, customer records, mechanic records and the vehicle service records. It works as follows:

- BCHC keeps a file of all of its customers. The following customer information is required:
  - Name
  - Telephone number
  - Address
  - Driving licence number

- Each vehicle's details are recorded and kept; some of these details will be continuously updated such as service dates. The details recorded are:
  - Registration number (unique identifier)
  - Make
  - Mode
  - Engine capacity
  - Hire class (1-6)
  - Date of registration
  - Date of each service
  - Name of mechanic responsible for each service
  - Mileage (updated on return after hire)

- The vehicles have a minor service every 6,000 miles and a major service every 12,000 miles and are serviced at the first available opportunity. The company's mechanics are responsible for the servicing.

- When a vehicle is hired, the customer details are recorded on the hire form along with the vehicle details, and the dates of the hire period (beginning and end). The end date is usually an estimate and will be updated on the return of the vehicle if it does not match what was originally recorded. The mileage at the start of the hire period is also recorded.

- When a vehicle is returned, the eventual return date is recorded on the hire form (if it differs to the original return date); the mileage on return of the vehicle is also recorded. Using the daily hire rate (defined in the hire class of each vehicle) and length of hire period, the cost of hire is calculated and the customer is billed. Only cash payment it accepted, the customer is given a receipt.

- The management requires that every completed hire transaction must contain the details of the customer, the dates of the hire period, details of the vehicle and the cost of hire.

- There is a record kept for each hire class of vehicles which are the daily, weekly and monthly hire rates. These are defined by the hire class.

- The garage keeps a record of each mechanic; it is a requirement that all mechanics hold a current driving licence. The following details are kept:
    - Name
    - Address
    - Telephone number
    - Driving licence number

Recognised Weaknesses

In the age of technology this system can be classed as archaic and outdated. The main issue identified is the fact that the records are entirely paper based. A paper based system can render record-keeping inefficient, one of the causes is that physical documents will need storage space and as more records are accrued the more space will be required. It can also be a slow and arduous process in locating certain files especially if the filing system has not been maintained well as documents can easily be lost. The physical nature of the system hampers productivity.

Physical files are also prone to many forms of damage and are not backed up as easily as a digital file would be, the backup process would be very time consuming. The transportation of documents can also be seen as a weakness as sending physical copies is much slower than sending through email, physically mailing files can take days compared to seconds when electronically mailing them.

A major flaw regarding the paper based system and the nature of the company is the editing of documents. Since the company require details of various items to be constantly updated (e.g. mileage, hire dates, service dates) it is very inefficient to do this on paper compared the relative ease of which this can be achieved electronically. Updating paper based records also means an increase in the amount of paper which goes on to add to the issue of diminishing storage space.

Another concern with paper based filing is security, digital files can be protected with very advanced password and encryption services whereas physical copies of files are basically protected by a door which can be broken. A further concern is the cost of materials as the company will have to purchase a lot of paper; this could be in various sizes and styles of paper (e.g. thickness, colour). Paper will need to be purchased frequently; this is also compounded by printer costs as more ink is used in such a system.

Requirements

A solution to a problem of this nature would need to be developed using **Object-Oriented (OO) programming**. An **object** is a tangible entity based on a real world equivalent; there are multiple items in the system that can be identified as objects, such as Vehicles and People. These objects will

have variable data states as attributes and properties will differ based on each instance of the object, they can share this data with other objects or functions to enable the system to cohesively interact in order to provide services.

Functional Requirements

BCHC would like the following functions implemented in the new computerised system:

- Register a new customer
- Record that a particular car has been hired
- Record that a particular car has been returned
- Calculate the cost of hire based on the daily hire rate
- Display the appropriate details and print out a receipt
- Log a completed hire
- Record a service for a particular car, together with the date of the service, the type of service, and the name of the mechanic responsible
- Remove a customer
- Add a vehicle to the fleet
- Delete a car that is no longer in the hire fleet
- Add a mechanic who has joined the company
- Remove the details of a mechanic who has left the company
- Determine if a particular car is due for a particular service
- List the information (history) about all hires for a specified car
- List the information (history) about all services that a specified car has had

Non-Functional Requirements

Usability:

- A help and support menu is provided when needed

Security:

- Keep data confidential
- Provide more security with the system e.g. username and passwords

Performance:

- The system has a response of no more than 15 seconds when user inputs data
- System should provide a response in more than 60 seconds whether it is a high or low complicated task

Availability:

- View/search data when requested
- System available 24/7

Goals

Based on the requirements, the goals of the system are to:

- Provide a paperless hire system
- Speed up the hire process
- Reduce office costs
- Increase administrative efficiency

- Automatically book vehicles for service if needed
- Allow a more dynamic management of records such as vehicles, customers and personnel.

Essential Use Cases

The following **use cases** were compiled based on the requirements.

Use Case	UC01 – Log in
Actor(s)	All staff
Purpose	Log into system
Overview	The staff members are required to log into the system, each type of staff member has different privileges so it is essential they have different accounts. Clerks cannot carry out functions such as adding/removing vehicles from the fleet. Each staff member will be given a unique username and password.
Precondition	None

Actor action	**System response**
1. The staff member would like to use the system.	2. System prompts the staff member for their username and password.
3. The staff member enters their username and password and clicks 'Log in' or hits the 'Enter' button on the keyboard.	4. The system verifies whether the details are correct.
	5. Upon verification, the system logs into the user's account with their relevant privileges.

Table 1 – Log in

Use Case	UC02 – Register Customer
Actor(s)	Customer (**initiator**), Clerk
Purpose	Register customer details
Overview	A customer would like to hire a vehicle; if not already registered; Staff acquires relevant details (Name, Telephone Number, and Address) from the Customer and inputs them into the system database.
Precondition	UC01

Actor action	**System response**
1. Customer would like to register with the interest to hire a vehicle.	
2. Clerk initiates registration process (e.g. opening relevant program/database)	3. The system presents fields to enter Customer information
4. The Clerk acquires Customer information (Name, Telephone Number, and Address).	
5. Clerk enters and submits the data.	6. The system verifies the data types, upon verification data is saved.
	7. Customer is registered.

Table 2 – Register customer

批注 **[ZL1]:** Meaningful, but too much design details

Use Case	UC03 – Hire out a vehicle
Actor(s)	Customer (initiator), Clerk
Purpose	Hire out a vehicle to a customer.

Overview	A Customer would like to hire a vehicle; the Clerk would take their details (such as name when from when they registered) and the desired vehicle and fill out the required form.
Precondition	UC01, UC02

Actor action	**System response**
1. A customer would like to hire out a vehicle.	
2. The Clerk asks for vehicle and customer details (such as name or ID).	
3. Clerk searches for vehicle.	4. System shows vehicle details.
5. Clerk pulls up the rental page.	6. System presents fields to enter data.
7. Clerk enters customer and vehicle details onto the form.	8. System verifies the data entered to make sure it is correct.
	9. System generates a reference number for the hire.
10. Clerk gives reference number to customer.	
11. Clerk gives vehicle to customer.	12. Vehicle is rented out.

Table 3 – Hire a vehicle

Use Case	UC04 – Process return of a vehicle
Actor(s)	Customer (initiator), Clerk
Purpose	Process the return of a returning vehicle.
Overview	Once the customer's hire period has ended, they will return the vehicle to the company who will process the return and calculate the cost of hire to bill the customer.
Precondition	UC01, UC03

Actor action	**System response**
1. A customer would like return a hired vehicle.	
2. The Clerk asks for customer details. (Unique attributes)	
3. Clerk searches for customer.	4. System shows customer details and also details of hired vehicle.
5. Clerk clicks on a button to accept the return vehicle such as 'End hire' or 'Return'.	
6. Clerk manually inputs the new mileage of the vehicle	7. System displays vehicle as returned and updates mileage count.
	8. The system logs date of return.
	9. System calculates the cost of hire by multiplying the number of days hired by the daily hire rate of the vehicle.
	10. System displays the amount owed by the customer.
11. Clerk bills the customer.	12. Vehicle is returned.
13. Customer pays (cash).	14. A receipt is printed for the customer.
	15. System logs the hire by saving the hire details to appropriate places such as the vehicle's hire history and to the

	customer's history.

Table 4 – Return a vehicle

Use Case	UC05 – Determine service
Actor(s)	Clerk (initiator)
Purpose	To check if a vehicle needs servicing.
Overview	Once a vehicle has been returned to the company, the Clerk will update the vehicle's mileage on the database; if this figure has reached the 6,000 or 12,000 mile interval then it is booked in for a service.
Precondition	UC01, UC04

Actor action	**System response**
1. On return of a vehicle the Clerk inputs the mileage.	2. System calculates if the mileage has passed the 6,000 or 12,000 interval.
	3. If the mileage has not passed the intervals then vehicle is not booked for a service.
	4. If the mileage has passed only passed the 6,000 mile interval it is automatically booked for a minor service.
	5. If the mileage has passed the 12,000 miles interval it is automatically booked in for a major service.

Table 5 – Determine if vehicle needs servicing

Use Case	UC06 – Record service
Actor(s)	Mechanic (initiator)
Purpose	To record the details of a servicing.
Overview	Once a vehicle has been booked in for a service by the system, a mechanic will carry out the service. The following details need to be recorded: date of service, type of service and mechanic who carried out the job.
Precondition	UC01, UC05

Actor action	**System response**
1. Mechanic opens the vehicle record.	2. System shows the vehicle details including the type of service required.
3. Mechanic determines the type of service they need to carry out.	
4. Mechanic carries out the service of the vehicle.	
5. Mechanic inputs the type of service, date of service and their name.	6. System verifies data types and on verification will update the vehicle record to show the servicing details.

Table 6 – Record details of service

Use Case	UC07 – Remove Customer
Actor(s)	Customer (initiator), Clerk
Purpose	Remove the details of a Customer
Overview	A Customer would like to leave the services of the company, the Clerk would find the Customer's details and remove them from the system.
Precondition	UC01, UC02

Actor action	**System response**

1. Customer would like to leave the company.	
2. Clerk initiates the unregistering process by opening relevant program/database.	
3. Clerk searches for the Customer in the database	4. Customer is found and details are presented.
5. Clerk chooses to unregister the customer.	6. System removes the details of the customer.
	7. Customer is unregistered.

Table 7 – Remove customer

Use Case	UC08 – Add vehicle
Actor(s)	Management (initiator)
Purpose	Add new vehicle to fleet
Overview	Management add a new vehicle to the fleet with required attributes being make, model, engine capacity, hire class and daily hire rate.
Precondition	UC01

Actor action	**System response**
1. A new vehicle is acquired, Management want to add the vehicle to its existing fleet.	
2. Management open relevant program/database.	3. The system requests details and provides fields for data entry.
4. Management enters and submits the required details (make, model, engine capacity, hire class, daily hire rate, and date of registration).	5. The system verifies the data types and saves details if correct.
	6. New vehicle has been successfully added to the fleet.

Table 8 – Add vehicle

Use Case	UC09 – Remove vehicle
Actor(s)	Management (initiator)
Purpose	Remove vehicle from fleet
Overview	Management search for the unwanted vehicle and remove it from the system, this could be for a number of reasons e.g. vehicle is damaged, vehicle is too old etc.
Precondition	UC01, UC08

Actor action	**System response**
1. A vehicle is no longer in the fleet for whatever reason, Management want to remove from database.	
2. Management opens relevant program/database.	
3. Management searches for the desired vehicle.	4. The system presents the details of the vehicle.
5. Management chooses to delete/remove from database.	6. System deletes the data.
	7. The vehicle has been successfully removed from the fleet.

Table 9 – Remove vehicle

Use Case	UC10 – Add mechanic	
Actor(s)	Management (initiator), Mechanic	
Purpose	Register a Mechanic with the company	
Overview	Management acquires the details of the newly hired Mechanic and adds them to the company database.	
Precondition	UC01	
Actor action		**System response**
1. Management must keep a record of each mechanic.		
2. Management initiates the registration process (e.g. opening relevant program/database)		3. The system presents fields to enter the Mechanic information.
4. Management acquires information from Mechanic (Name, Address, Home Telephone)		
5. Management checks if the Mechanic holds a current driving license.		
6. Upon confirmation of license, Management enters these details into the system.		7. System verifies data types and saves the information.
		8. Mechanic is registered.

Table 10 – Add mechanic

Use Case	UC11 – Remove mechanic	
Actor(s)	Mechanic (initiator), Management	
Purpose	Remove details of a Mechanic	
Overview	A Mechanic is leaving the company; their details are to be removed. Management will search for the departing Mechanic and remove them from the system.	
Precondition	UC01, UC10	
Actor action		**System response**
1. Mechanic is leaving the company; Management must keep a record of each mechanic.		
2. Management initiates removal process by opening relevant program/database		
3. Management searches for the departing Mechanic's details.		4. The Mechanic's details are presented by the system.
5. Management chooses the remove/delete the departing Mechanic's details		6. System deletes the Mechanic's details.
		7. Mechanic is unlisted/unregistered/removed from company system.

Table 11 – Remove mechanic

Use Case	UC12 – List services
Actor(s)	Management (initiator)

Purpose	To list the details and history of all services.
Overview	If required, the management can look at the data of all the previous services a vehicle has had. This may be useful when deciding whether to keep the vehicle in the fleet or not.
Precondition	UC01, UC06

Actor action	**System response**
1. Manager opens vehicle page	2. Vehicle page is displayed
3. Manager clicks on a button (such as view service history) to view a history of all services carried out on said vehicle.	4. The history of the vehicle's services are displayed, includes information such as date of service, type of service and mechanic who serviced the vehicle.

Table 12 – List service history

Use Case	UC13 – List hires
Actor(s)	Management (initiator)
Purpose	To list the details and history of all hires of a vehicle.
Overview	If required, the management is able to look at the details of all previous hires a vehicle has had. This may be useful when deciding whether to keep a vehicle in the fleet or not, as well as locating specific details e.g. details of a customer who has previously hired the vehicle.
Precondition	UC01, UC04

Actor action	**System response**
1. Manager opens vehicle page	2. Vehicle page is displayed
3. Manager clicks on a button (such as view hire history) to view a history of all hires of said vehicle.	4. The history of the vehicle's hires are displayed, includes information such as dates of hire, customers who hired them, cost of each hire.

Table 13 – List hire history

Use Case Diagram

The following diagram best represents the identified use cases in the software design for BCHC:

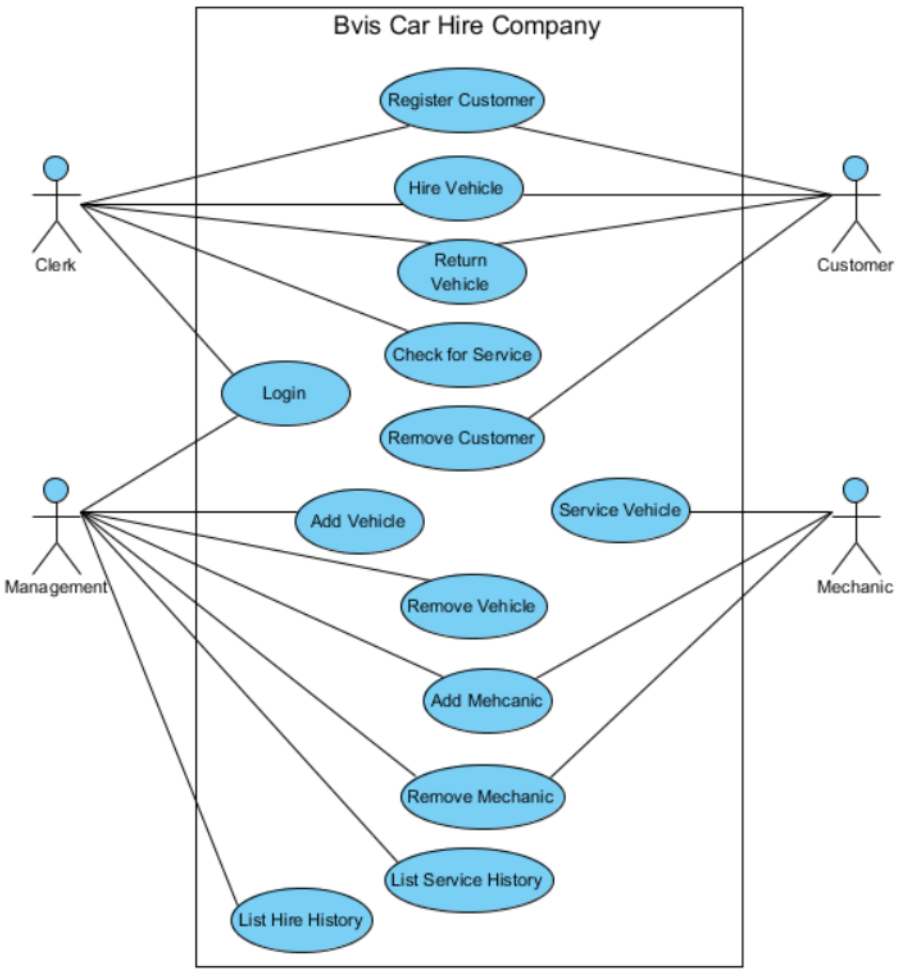


Figure 1 – Use Case Diagram

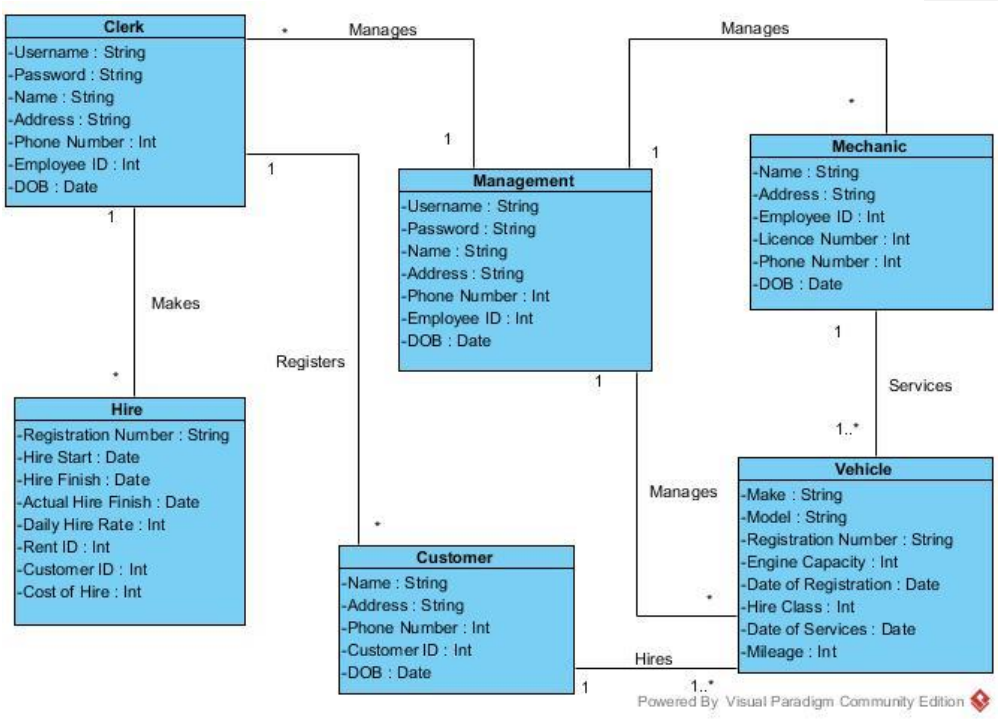
The use case diagram shows how each actor is involved with each use case. In some cases there are two actors involved with a use case, this allows for actors to interact with each other when required.

Conceptual Class Diagram

For the OO design, classes will be required for the functionality of the system, these classes can be conceptualised from the Use Case Diagram. The following classes have been identified:

- Clerk
- Customer
- Hire
- Management
- Mechanic
- Vehicle

Each class will have attributes which will be changed in the different states of the objects created by the classes. This conceptual model shows how the classes interact with each other:



The conceptual class diagram shows how the system is linked and the attributes needed within each section. Alongside this, it shows what data types are needed for each section within the specific table.

批注 **[ZL2]:** Missing important classes and associations

The link between each of these classes is listed below:

- **MANAGEMENT** manages the **CLERKS, MECHANICS AND VEHICLES**
- **MECHANIC** services the **VEHICLES**
- **CUSTOMER** hires the **VEHICLE**
- **CLERK** registers **THE CUSTOMERS** and makes the **HIRE** transaction

Within the Clerk and Management classes, the same information is needed. They need to enter a username and password that would be specific to the user which would be given when the system has been set up. By providing a username and password it limits each user to the amount of data they can access and to what they can enter into the system. These both have the data type of string as this means that the username and password can be made up with either or both numbers and letters. The name and address is also needed within both tables as this will establish what person works for BCHC and where they live as this is also essential information that is needed by the company. Again these are of the string datatypes, the names will only contain letters whereas the address will contain a mixture of letters and numbers as the house number and post code are also needed. The phone number is needed as a form of contact if the management ever wish to get in contact with their staff or vice versa to discuss something important. The employee ID is also required as this differentiates each staff member to one another and allows management to know who has done what transaction within the company when signed in using their unique employee ID number. This can also help with the payroll system as it makes it easier to distinguish who has worked how many hours and how much they need to be paid. These both use the datatype integer as they will both contain numbers and nothing else. Finally they both need to input their D.O.B into the system as this will ensure that the company hire employees that follow the legal ages limit but also helps determine how much they should be getting paid. The datatype for this is Date as this is the format it will be entered in.

Within the Hire class, information needed is about the transaction that would be processed. Firstly, the registration number is needed of the vehicle that will be hired out to the customer. By doing this it helps track where the car is and who it is with. The datatype for the registration number is string as again this will contain both letters and numbers making it a unique attribute to each vehicle. The staff then enter the hire start and finish date and the actual hire finish date (upon return, only if it differs). Again all these need to be entered for the staff of BCHC to know where the cars are, what customer they are with and when they are going to be returned to the company. The datatypes that have been used for this are Date as this would make it easy for the staff to read the dates that are involved in the hire transaction. The staff also need to enter the daily hire rate into the system as each car would have a different price depending on the make and model and any extra features that are included i.e. Sat Nav and Bluetooth. The datatype that is needed for this is Integer as the data that will be entered is the price (numbers) which can go into triple or quadruple figures if necessary. A Rent ID and Customer ID are then needed to be entered into the system as this will make each transaction unique and every customer unique as well. Having a Customer ID will also benefit the company as this will make it easier when creating a new transaction for customer that has already been registered. Therefore this helps save the company time as they will not need to fill in the details again as they will already be saved on the system. Cost of the Hire is also needed as this will show both the customer and the clerk the total price that is needed to be paid for the duration of the hire. The datatypes that are needed for these are Integer as these will only incorporate numbers which help make the IDs unique and for the price.

The Customer class requires similar information to the staff (as people share most attributes) as this will help register every customer that makes a transaction and allows their details to be stored on the system. The details that are required are the name and contact details such as their address and telephone number. These are needed to ensure that BCHC has some form of communication method when needed to get in touch with the customer. The datatypes that are used for the name and address are string as this will allow alphanumeric values to be entered, whereas, the datatype that is used for the phone number is Integer as only numbers need to entered otherwise it would make the field invalid giving a false telephone number. Again the Customer ID is needed which allow every customer to be uniquely identify which will then link to the Hire table ensuring that the same customer has the same ID number throughout the whole process. The datatype that is needed for

this is Integer as this will only incorporate numbers which will help make the ID unique. The clerk also needs to enter the date of birth of the customer as this will ensure that they are of the legal age to carry out the transaction and hire the vehicle. The datatype for this is Date as this is the format it will be entered in.

The Mechanic table again requires similar information to the Management and Clerk tables as they also work for BCHC. Again the name and contact information such as the address and phone number are needed in case the management need to get in touch with them. The datatypes that are used for the name and address are string as this will allow alphanumeric values to be entered, whereas, the datatype that is used for the phone number is Integer as only numbers need to entered otherwise it would make the field invalid giving a false telephone number. As the mechanics work for BCHC they will also require an Employee ID as this will provide them with a unique number at the company which will allow management to search for the mechanics details easily using the Employee ID number. The mechanics also need enter their driving licence number as the management requires a valid driving licence in order to work for the company. By entering this data into the system, it is easy for the management team to track this information and always keep the data updated. As well as this date of birth also needs to be entered into the system as this will ensure that they are of the legal age requirement to work and again this will help with the payroll as it will ensure that they are getting paid the amount they deserve. The datatype for this is Date as this is the format it will be entered in.

The Vehicle class requires information concerning the vehicle that the fleet holds. The team first need to input the make and model and the registration number of the vehicle as this will establish what type of vehicle it is and the sort of price that would be assigned to it when being hired. The registration is a unique ID so this needs to be entered as the team can search for a car in the system using this unique identifier. The datatype that is set for these fields are string as this will allow the team to enter alphanumeric values into the system as some of the model names and the registration numbers may contain numbers. The engine capacity also needs to be entered into the system as this is important information that needs to be registered with the car as well as the mileage having to be entered. This needs to be entered because the company needs to keep track how much the car is being driven by each customer which will help the clerks know when the car needs to be serviced. The datatypes that are used for these fields are Integer as these are only numbers that are have to entered as no letters are involved. The date of registration and the date of services need to be entered as this is crucial for the company as they need to keep track of when the cars need to be serviced again depending on how much mileage they have done. The datatypes that are used for this are in the date format as this will make it easier for the clerks and management to read it when searching for information.

Part II

System Sequence Diagrams

The following System Sequence Diagrams (based on the identified use cases) show how the actors relate to each other and the system. Appropriate methods and operations have been identified for use by classes and objects.

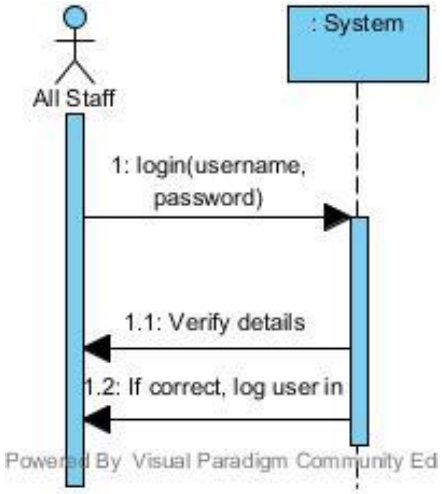


Figure 2 – UC01 Sequence Diagram

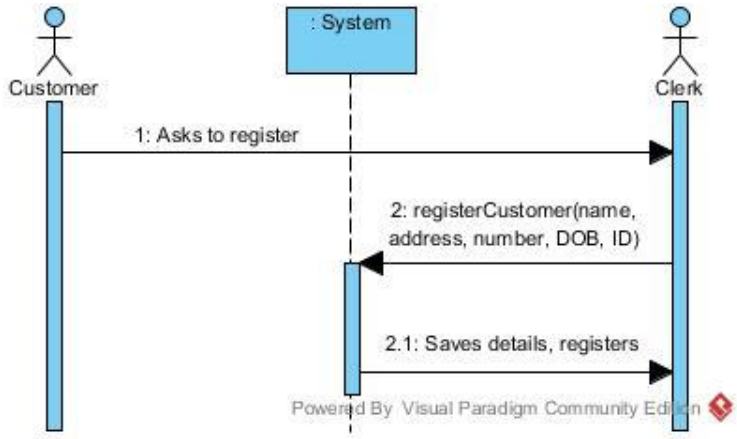


Figure 3 – UC02 Sequence Diagram

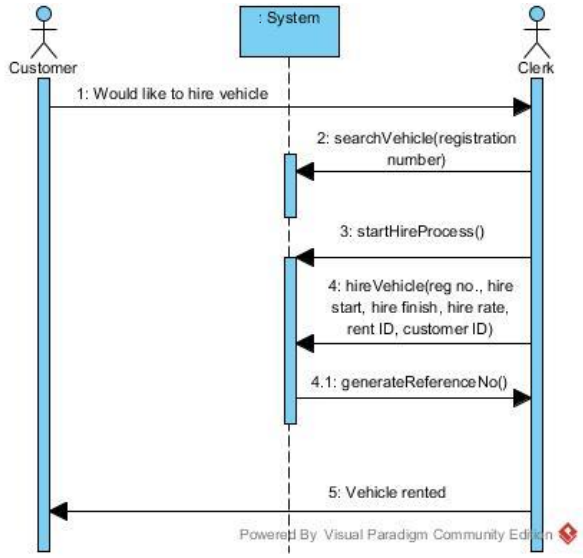


Figure 4 – UC03 Sequence Diagram

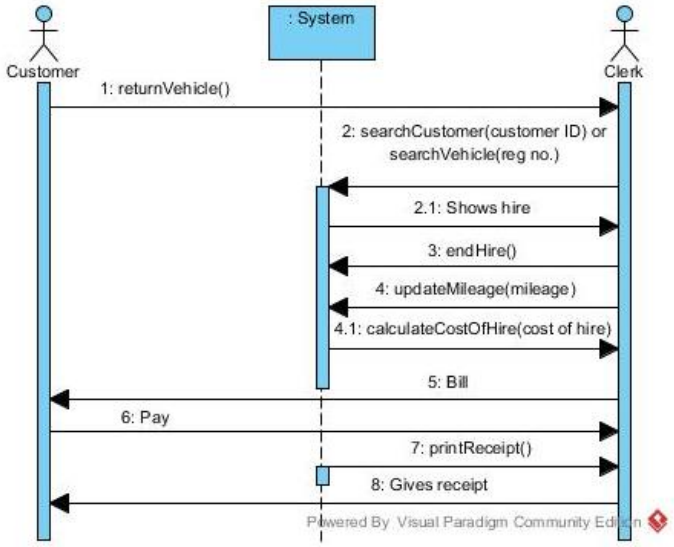


Figure 5 – UC04 Sequence Diagram

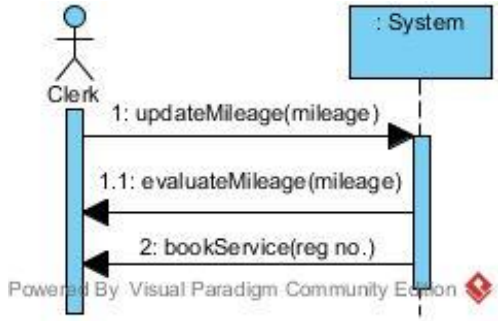


Figure 6 – UC05 Sequence Diagram

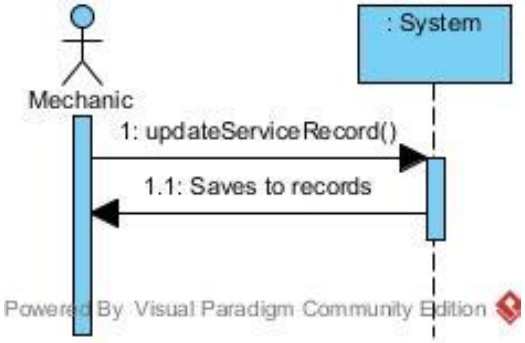


Figure 7 – UC06 Sequence Diagram

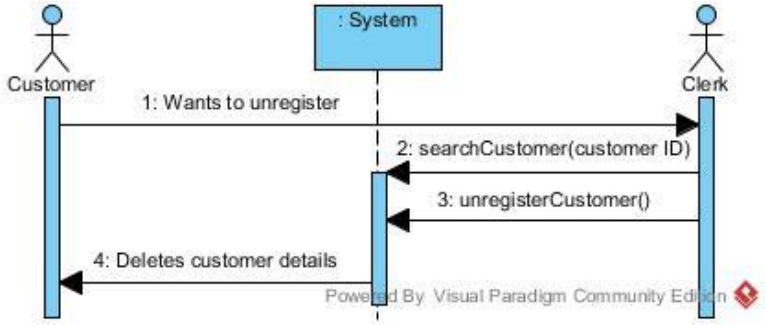


Figure 8 – UC07 Sequence Diagram

批注 [ZL3]: Only show direct actir(s), and interaction with the system

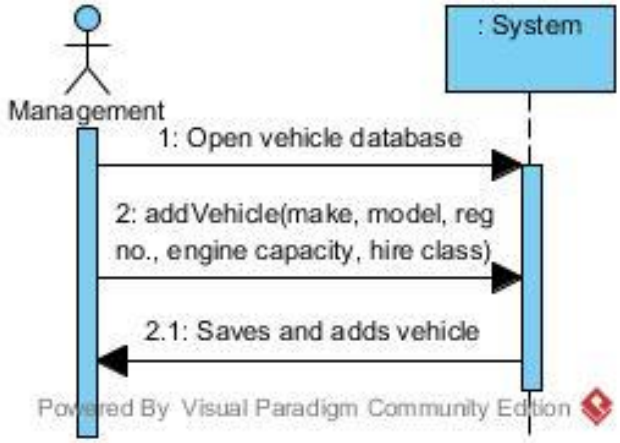


Figure 9 – UC08 Sequence Diagram

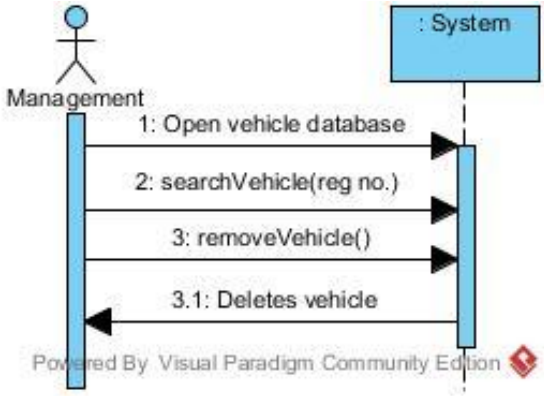


Figure 10 – UC09 Sequence Diagram

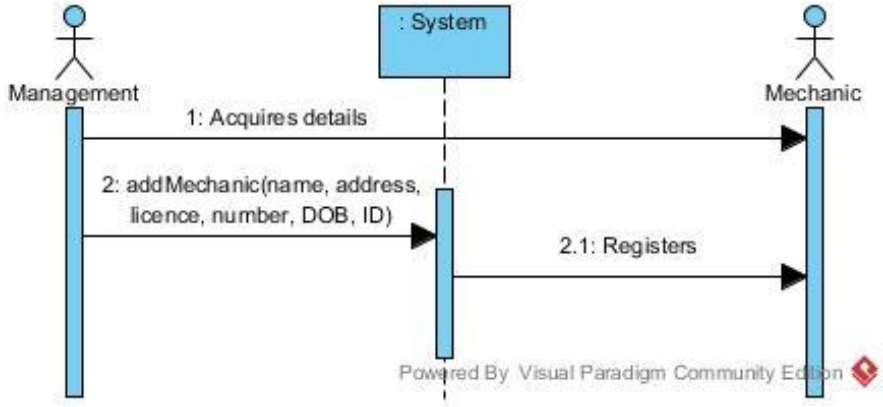


Figure 11 – UC10 Sequence Diagram

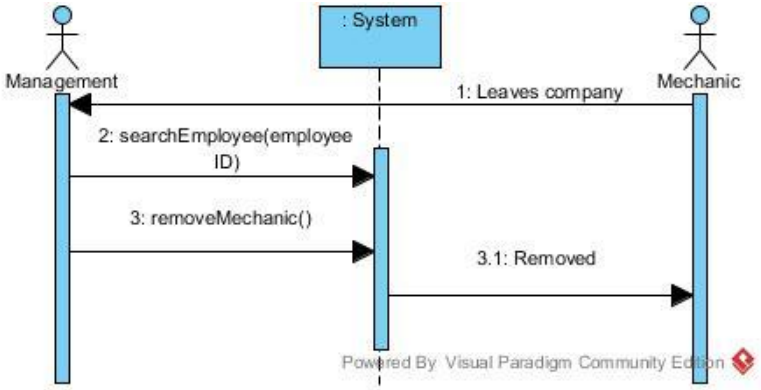


Figure 12 – UC11 Sequence Diagram

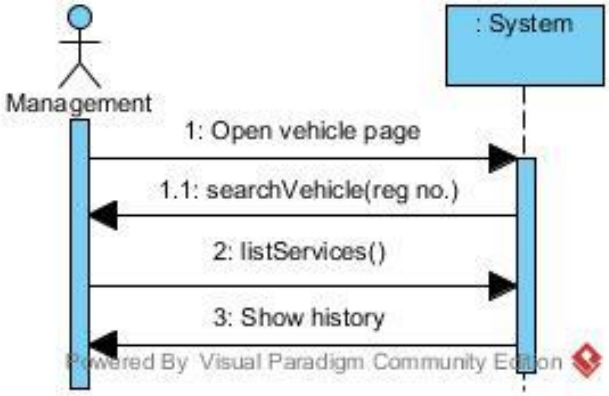


Figure 13 – UC12 Sequence Diagram

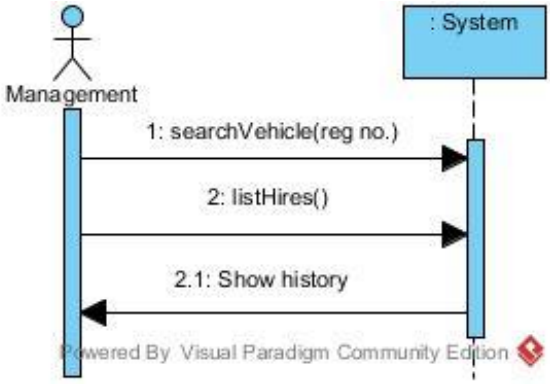


Figure 14 – UC13 Sequence Diagram

Contracts

The operations and methods which were identified from the system sequence diagrams are further explained through the use of contracts.

Contract	login
Name	login(username: String, password: String)
Responsibilities	Allow a user to log into the system.
Type	System
Cross References	UC01
Note	Allows the use of separate accounts for different level employees.
Exceptions	If the username and password combination is not correct, log in will fail.
Output	-
Pre-conditions	Account names and log in details are known to the system
Post-conditions	Account name is set to the employee ID

Table 14 – login()

Contract	registerCustomer
Name	registerCustomer(name: String, address: String, DOB: Date, customerID: Integer)
Responsibilities	Register a customer to the company database to allow them to use the hire services.
Type	System
Cross References	UC02
Note	-
Exceptions	Indicate error if duplicate information.
Output	-
Pre-conditions	
Post-conditions	Customer is created

Table 15 – registerCustomer()

Contract	searchVehicle
Name	searchVehicle(registration number: String)
Responsibilities	Allows employees to find vehicles in the fleet.
Type	System
Cross References	UC03, UC04, UC09, UC12, UC13
Note	-
Exceptions	Indicate error if search parameters return no results.
Output	Display vehicle searched for.
Pre-conditions	
Post-conditions	Vehicle is associated to employee

Table 16 – searchVehicle()

Contract	startHireProcess
Name	startHireProcess()
Responsibilities	Initialises the hire process.
Type	System
Cross References	UC03
Note	-

Exceptions	-
Output	Displays form for vehicle hire.
Pre-conditions	
Post-conditions	Hire instance created

Table 17 – startHireProcess()

Contract	hireVehicle
Name	hireVehicle(registration number: String, hire start: Date, hire finish: Date, hire rate: Integer, rentID: Integer, customerID: Integer)
Responsibilities	Hires a vehicle out to a customer.
Type	System
Cross References	UC03, UC04
Note	-
Exceptions	Indicate error if trying to hire a vehicle already rented out. Indicate error if customer ID is not recognised. Indicate error if vehicle ID is not recognised.
Output	-
Pre-conditions	
Post-conditions	Customer associated to vehicle Customer associated to hire

Table 18 – hireVehicle()

Contract	generateReferenceNo
Name	generateReferenceNo()
Responsibilities	Generates a unique reference number for the transaction.
Type	System
Cross References	UC03
Note	It would be sensible to have the number increment by 1 for each separate hire.
Exceptions	The number cannot be the same as one already generated an in use.
Output	A unique number is generated in reference to the hire transaction.
Pre-conditions	
Post-conditions	

Table 19 – generateReferenceNo()

Contract	searchCustomer
Name	searchCustomer(customerID: Integer)
Responsibilities	Allows employees to search for customers.
Type	System
Cross References	UC04, UC07
Note	-
Exceptions	Indicate error if customer ID is not recognised.
Output	Display customer details.
Pre-conditions	
Post-conditions	Customer is associated to emloyee

Table 20 – searchCustomer()

Contract	endHire
Name	endHire()
Responsibilities	Allows the clerk to end the hire.
Type	System
Cross References	UC04
Note	-
Exceptions	-
Output	-
Pre-conditions	
Post-conditions	Hire is logged, associated with listHires

Table 21 – endHire()

Contract	updateMileage
Name	updateMileage(mileage: Integer)
Responsibilities	Allows the mileage of a vehicle to be updated on return.
Type	System
Cross References	UC04, UC05
Note	-
Exceptions	-
Output	-
Pre-conditions	
Post-conditions	Vehicle.Mileage is changed

Table 22 – updateMileage()

Contract	calculateCostOfHire
Name	calculateCostOfHire()
Responsibilities	Cost of the hire transaction is calculated.
Type	System
Cross References	UC04
Note	Finish date may be different to the one originally agreed upon.
Exceptions	-
Output	Display amount to be charged.
Pre-conditions	
Post-conditions	cost of hire set to output of calculateCostOfHire

Table 23 – calculateCostOfHire()

Contract	printReceipt
Name	printReceipt()
Responsibilities	Prints the details of the billing for the customer.
Type	System
Cross References	UC04
Note	Displays relevant information such as clerk name or ID, date of billing, dates of hire, time, amount charged, hire ID.
Exceptions	-
Output	Generates a receipt of payment.
Pre-conditions	
Post-conditions	

Table 24 – printReceipt()

Contract	evaluateMileage
Name	evaluateMileage(mileage)
Responsibilities	Mileage is evaluated to see if it has reached the 6,000 or 12,000 mile threshold after which a vehicle must be serviced.
Type	System
Cross References	UC05
Note	Should be able to differentiate between a minor and major service as 6,000 is a factor of 12,000 but represents a minor service.
Exceptions	If mileage has not hit an interval of 6,000 then do nothing.
Output	-
Pre-conditions	
Post-conditions	

Table 25 – evaluateMileage()

Contract	bookService
Name	bookService()
Responsibilities	Book a vehicle to be serviced by a mechanic.
Type	System
Cross References	UC05
Note	If after evaluateMileage, mileage has passed an interval of 6,000 then the vehicle is automatically booked in for a service.
Exceptions	-
Output	-
Pre-conditions	
Post-conditions	

Table 26 – bookService()

Contract	updateServiceRecord
Name	updateServiceRecord()
Responsibilities	Allows a mechanic to update the service record of a vehicle as required.
Type	System
Cross References	UC06
Note	-
Exceptions	-
Output	-
Pre-conditions	
Post-conditions	Vehicle is associated to Mechanic Service record updated, listServices associated to Vehicle

Table 27 – updateServiceRecord()

Contract	unregisterCustomer
Name	unregisterCustomer(customerID: Integer)
Responsibilities	Allows a clerk to unregister a customer who no longer wants to use the services of the company.
Type	System
Cross References	UC07
Note	-
Exceptions	Indicate error if customer ID is not recognised.
Output	-

Pre-conditions	
Post-conditions	

Table 28 – unregisterCustomer()

Contract	addVehicle
Name	addVehicle(make: String, model: String, registration number: String, engine capacity: Integer, hire class: Integer)
Responsibilities	Allows a member of the management to add a vehicle to the fleet.
Type	System
Cross References	UC08
Note	-
Exceptions	Indicate error if duplicate record.
Output	-
Pre-conditions	
Post-conditions	Instance of Vehicle created

Table 29 – addVehicle()

Contract	removeVehicle
Name	removeVehicle(registration number: String)
Responsibilities	Allows a member of the management to remove a vehicle from the fleet.
Type	System
Cross References	UC09
Note	-
Exceptions	Indicate error is registration number not recognised.
Output	-
Pre-conditions	
Post-conditions	

Table 30 – removeVehicle()

Contract	addMechanic
Name	addMechanic(name: String, address: String, licence: String, number: Integer, DOB: Date, employeeID: Integer)
Responsibilities	Allows a member of the management to add a mechanic to the company.
Type	System
Cross References	UC10
Note	-
Exceptions	-
Output	-
Pre-conditions	
Post-conditions	Instance of Mechanic created Mechanic associated to Management

Table 31 – addMechanic()

Contract	searchEmployee
Name	searchEmployee(employeeID: Integer)
Responsibilities	Allows the management to search for an employee.
Type	System
Cross References	UC11
Note	Is useful when searching for a mechanic when removing them.

Exceptions	Indicate error if ID not recognised.
Output	Display employee details.
Pre-conditions	
Post-conditions	Employees associated to Management

Table 32 – searchEmployee()

Contract	removeMechanic
Name	removeMechanic()
Responsibilities	Allows a member of the management to remove a mechanic from the company.
Type	System
Cross References	UC11
Note	-
Exceptions	-
Output	-
Pre-conditions	
Post-conditions	

Table 34 – removeMechanic()

Contract	listServices
Name	listServices()
Responsibilities	Lists the service history of a vehicle if any.
Type	System
Cross References	UC12
Note	-
Exceptions	-
Output	Shows a list of services provided to a vehicle.
Pre-conditions	
Post-conditions	listServices associated to Management

Table 35 – listServices()

Contract	listHires
Name	listHires()
Responsibilities	Lists the hire history of a vehicle if any.
Type	System
Cross References	UC13
Note	-
Exceptions	-
Output	Displays the hire history of a vehicle.
Pre-conditions	
Post-conditions	listHires associated to Management

Table 36 – listHires()

批注 **[ZL4]:** Good contracts

Part III

Collaboration Diagrams

Below are the collaboration diagrams which show the interaction between different objects during the use cases.

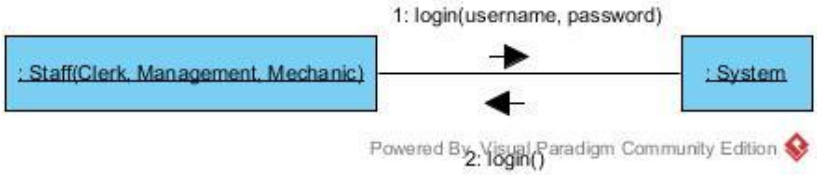


Figure 15 – UC01

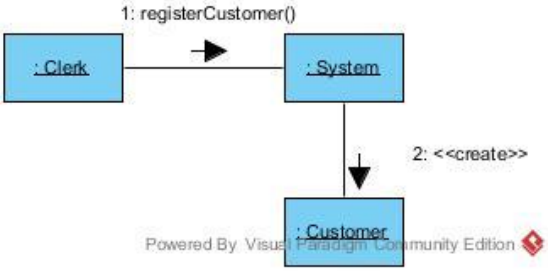


Figure 16 – UC02

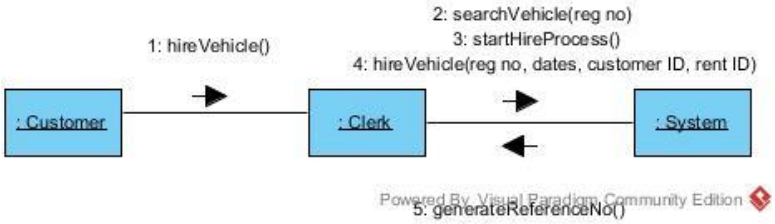


Figure 17 – UC03

批注 **[ZL5]:** Not quite a good one

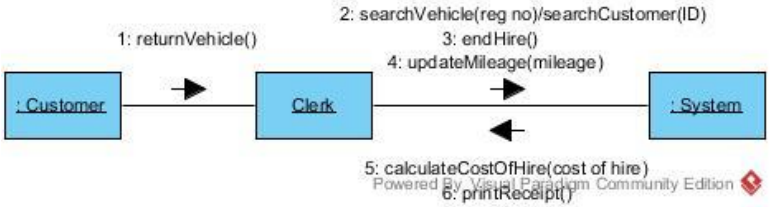


Figure 18 – UC04

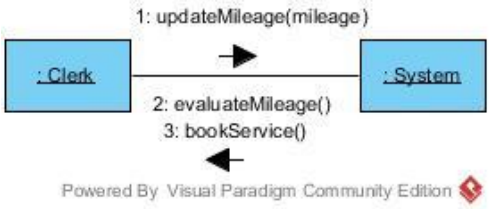


Figure 19 – UC05

Figure 20 – UC06

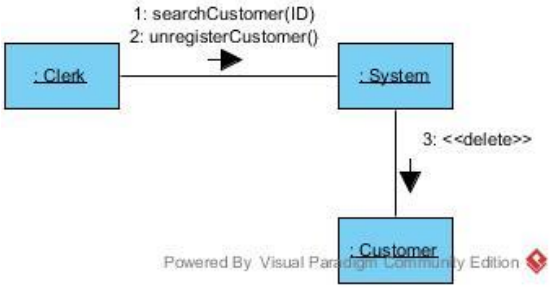


Figure 21 – UC07

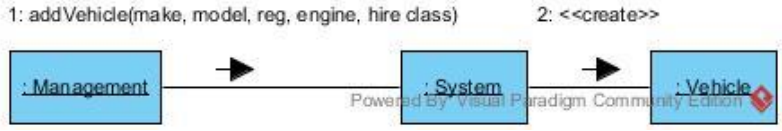


Figure 22 – UC08

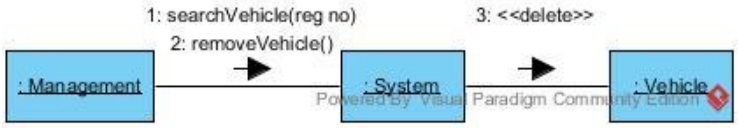


Figure 23 – UC09

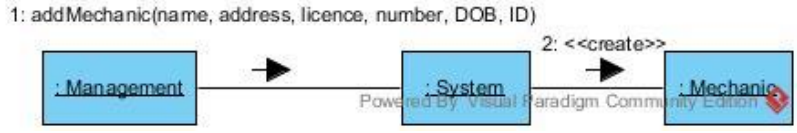


Figure 24 – UC10

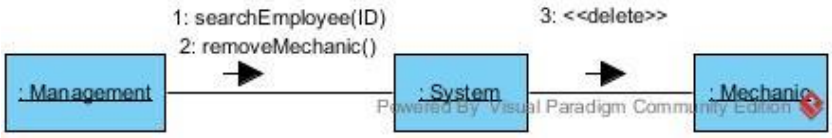


Figure 25 – UC11

Figure 26 – UC12

Figure 27 – UC13

Patterns

Expert – This pattern concerns the delegation of responsibilities, it determines the information needed to fulfil a responsibility and then determines which class has the most information to fulfil the responsibility, thus placing it with that class. There are a number of information experts in the design, these include:

- Clerk – holds information of clerk
- Management – holds information of its own objects
- Mechanics – holds information of its own objects
- Customer – holds information of its own objects
- Vehicle – holds attributes of vehicle objects
- Hire – holds information of the hire, this includes vehicle and customer information as well as information unique to the hire such as cost, dates and a rent ID.

Creator – The Creator pattern concerns the creation of objects when desired; it is one of the most common functionalities of an OO system. In the system design, there are a few creators, these consist of:

- Clerk – can create a customer object (UC02) and an instance when searching for a customer (UC04, UC07). Also creates an instance of Hire (UC03) and Vehicle when searching (UC03, UC04).
- Management – can create a Vehicle object (UC08) and a Mechanic object (UC10), it can create instances of Vehicle when searching for one (UC09, UC12, UC13) and Mechanic (UC11).

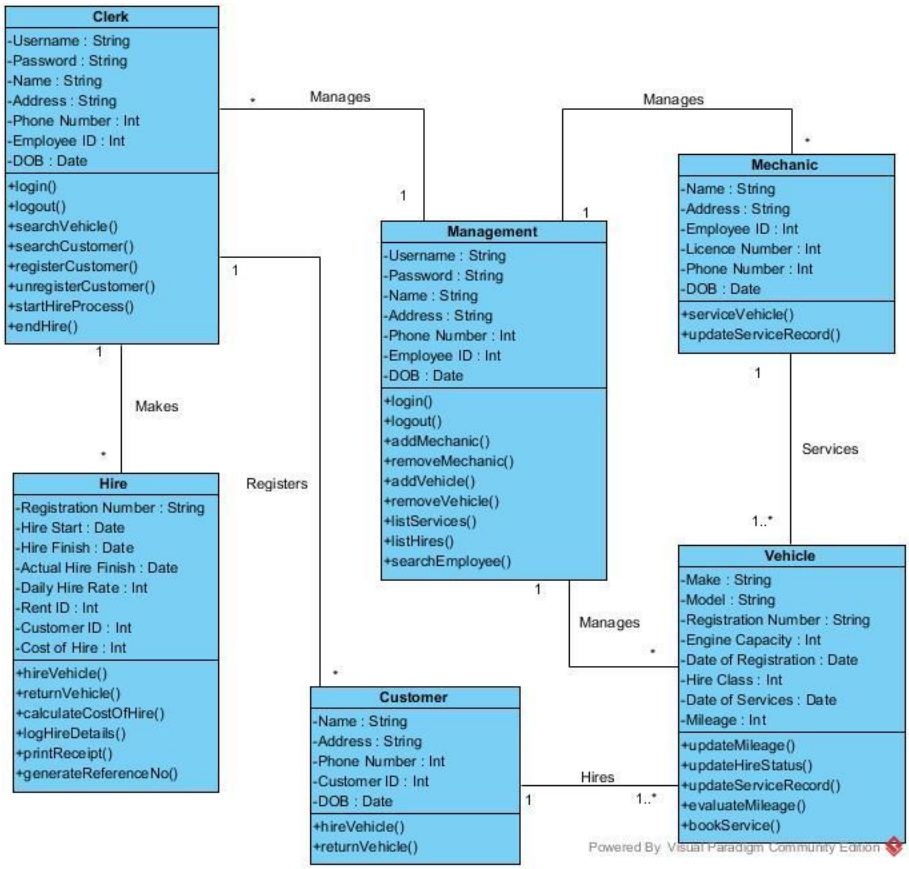
Low Coupling – Low Coupling concerns the assigning of responsibilities between classes to achieve lower dependency between them and for a higher reuse potential. A certain degree of low coupling seems to be achieved with this solution as classes aren't massively dependent on others although this can be improved by the introduction of intermediary objects/classes to reduce the workload of some specific classes (such as Management and Clerk).

High Cohesion – High Cohesion concerns the attempt to keep objects focused and understandable, this can be thought of as having more related responsibilities between classes. Whilst some of the classes carry a number of operations, they are focused rather than irrelevant so there is a factor of high cohesion shown by the design. This can be improved by perhaps splitting responsibilities further to gain further focus.

Controller – The Controller pattern deals with system events in a non-UI class which represents the system as a whole when handling an event. Whilst it hasn't been clearly defined in the collaboration diagrams, the use of a Controller is necessary to facilitate such events as logging in, saving and retrieving data and other maintenance tasks. It can be thought of as a service layer which operates behind the UI.

Class Diagram

From the conceptual class diagram and the collaboration diagrams, it was possible to produce the following class diagram:



Appendix

Project Diary

Date/time	Location	Attended	Points of Discussion
11/02/15 14.00 – 14.30	Student Union	Anil Jassi Ajmal Esakhail Aaron Joseph Damani Lana Uzair Shah Pavendeep Kaur Hayre	• What needs to be achieved by the end of the coursework • What a Use Case is • How to delegate the coursework tasks
18/02/15 14.00 – 14.30	Student Union	Anil Jassi Ajmal Esakhail Aaron Joseph Damani Lana Uzair Shah Pavendeep Kaur Hayre	• Check how far everyone has got with their tasks • Discuss the features of object orientated design and behavior of use cases • Practice using Visual Paradigm as a group
25/02/15 14.00 – 14.30	Student Union	Anil Jassi Ajmal Esakhail Aaron Joseph Damani Lana Uzair Shah Pavendeep Kaur Hayre	• Discuss object sequence diagrams • Methods of objects • How everyone is coping with the coursework
11/03/15 14.00 – 14.30	Student Union	Anil Jassi Ajmal Esakhail Aaron Joseph Damani Lana Uzair Shah Pavendeep Kaur Hayre	• How everyone is coping with the coursework • Discuss class diagrams • Understand the expert pattern
18/03/15 14.00 – 15.00	Student Union	Anil Jassi Ajmal Esakhail Aaron Joseph Damani Lana Uzair Shah Pavendeep Kaur Hayre	• How everyone is coping with the coursework • Know what the coupling pattern is • Understand cohesion pattern
25/03/15 14.00 – 14.30	Student Union	Anil Jassi Ajmal Esakhail Aaron Joseph Damani Lana Uzair Shah Pavendeep Kaur Hayre	• How everyone is coping with the coursework • Help each other with what still needs attention
08/04/15 14.00 – 15.00	Student Union	Anil Jassi Ajmal Esakhail Aaron Joseph Damani Lana Uzair Shah Pavendeep Kaur Hayre	• How everyone is coping with the coursework • Start putting everything together and get it ready for submission

Glossary

Object-Oriented (OO) programming – Programming modelled around objects and data based on the real world.
Object – An instance of a class which holds certain variables.
Use cases – steps defining interactions between actors and the system responses to this.
Actor – Carries out tasks which can affect the response of the system.
Initiator – The actor who takes the first step to initiate a use case.