Group work delvierables

Coursework 2

The Wizards

Software engineering

Table of Contents

[**Project Management:** 3](#_Toc511300244)

[**Project Plans:** 3](#_Toc511300245)

[*Expected Project Timeline:* 3](#_Toc511300246)

[*Actual Project Timeline:* 3](#_Toc511300247)

[**Minutes for the meetings:** 5](#_Toc511300248)

[**Requirements Documentation:** 6](#_Toc511300249)

[**Introduction:** 6](#_Toc511300250)

[**Actors Definitions:** 6](#_Toc511300251)

[**Specification of core use-cases:** 9](#_Toc511300252)

[*Isaac Rickwood – Part Ordering Mammogrid* 9](#_Toc511300253)

[*Fintan Kennedy – Car Service Mammogrid* 10](#_Toc511300254)

[*Spas Spasov – Car Sales report Mammogrid* 12](#_Toc511300255)

[*Vuong Khac Duy – Part Sales Mammogrid* 14](#_Toc511300256)

[Zheenyar Karim *– Car Sales Mammogrid* 15](#_Toc511300257)

[**System models:** 18](#_Toc511300258)

[*Class diagram* 18](#_Toc511300259)

[*Use-case views* 19](#_Toc511300260)

[*Sequence diagrams* 21](#_Toc511300261)

[**Non-functional requirements:** 23](#_Toc511300262)

[*Product non-functional requirements* 23](#_Toc511300263)

[*Organisational non-functional requirements* 25](#_Toc511300264)

[*External non-functional requirements* 28](#_Toc511300265)

[**Solution Architecture:** 29](#_Toc511300266)

[**Manage part Stock:** 30](#_Toc511300267)

[**Manage Car Stock:** 30](#_Toc511300268)

[**Search database:** 31](#_Toc511300269)

[**Person details management:** 31](#_Toc511300270)

[**Service management:** 31](#_Toc511300271)

[**Software Prototype:** 32](#_Toc511300272)

[**Software Test Specification:** 32](#_Toc511300273)

[**Part Ordering Testing:** 32](#_Toc511300274)

[**Part Sales Testing:** 32](#_Toc511300275)

[**Car Sales Testing:** 33](#_Toc511300276)

[**Car Service Testing:** 34](#_Toc511300277)

[**Car Sales Report Testing:** 34](#_Toc511300278)

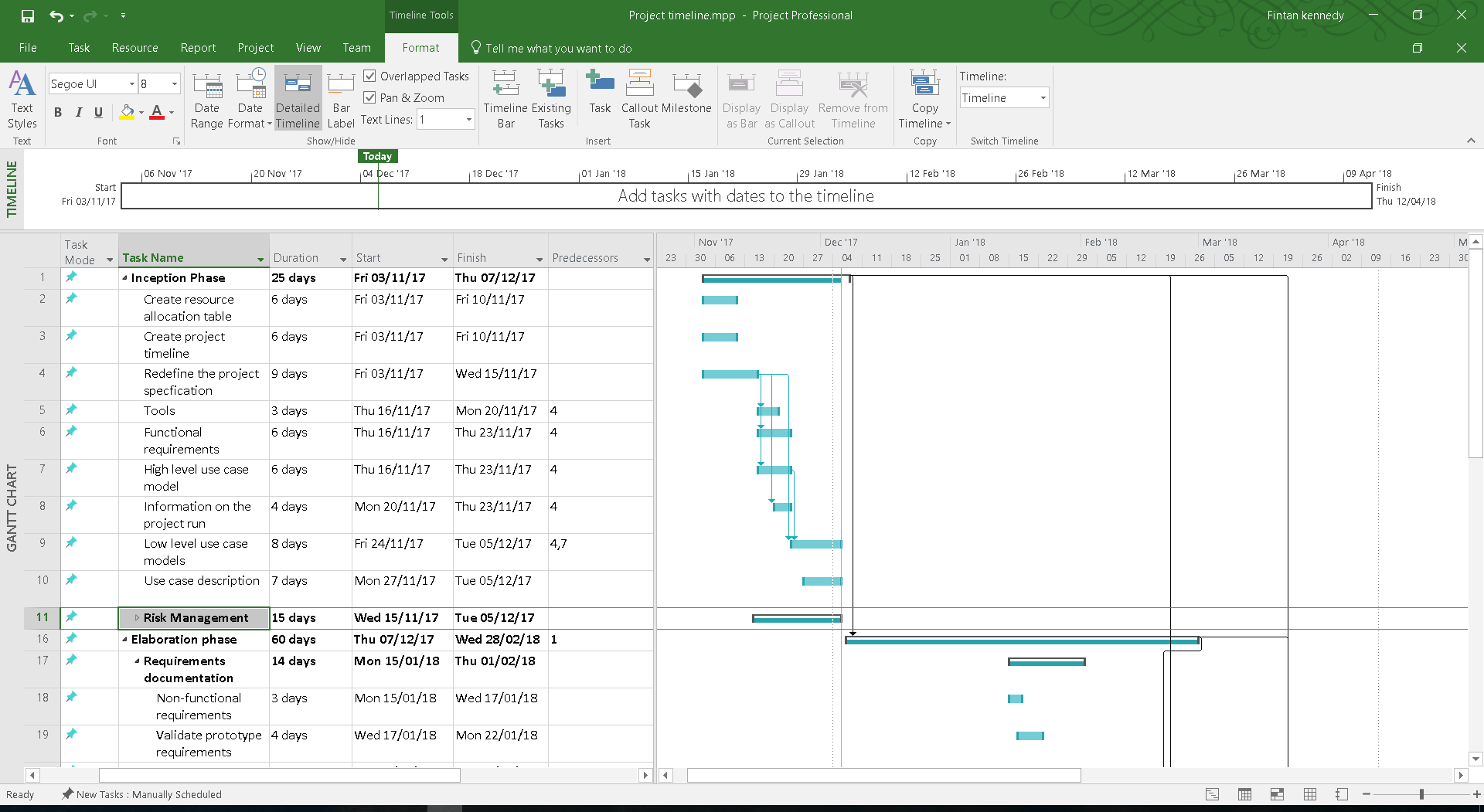
[**Example tests:** 35](#_Toc511300279)

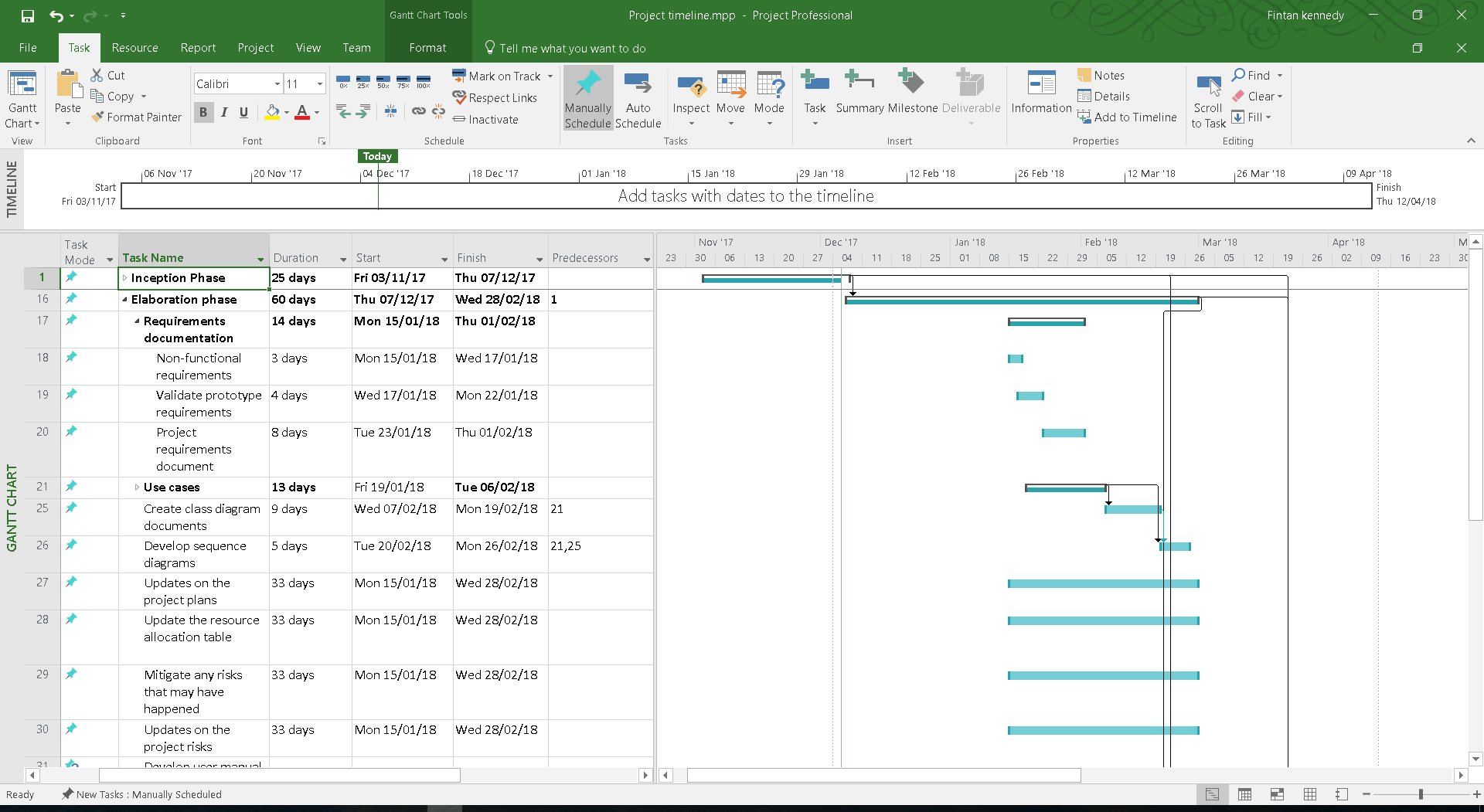
# **Project Management:**

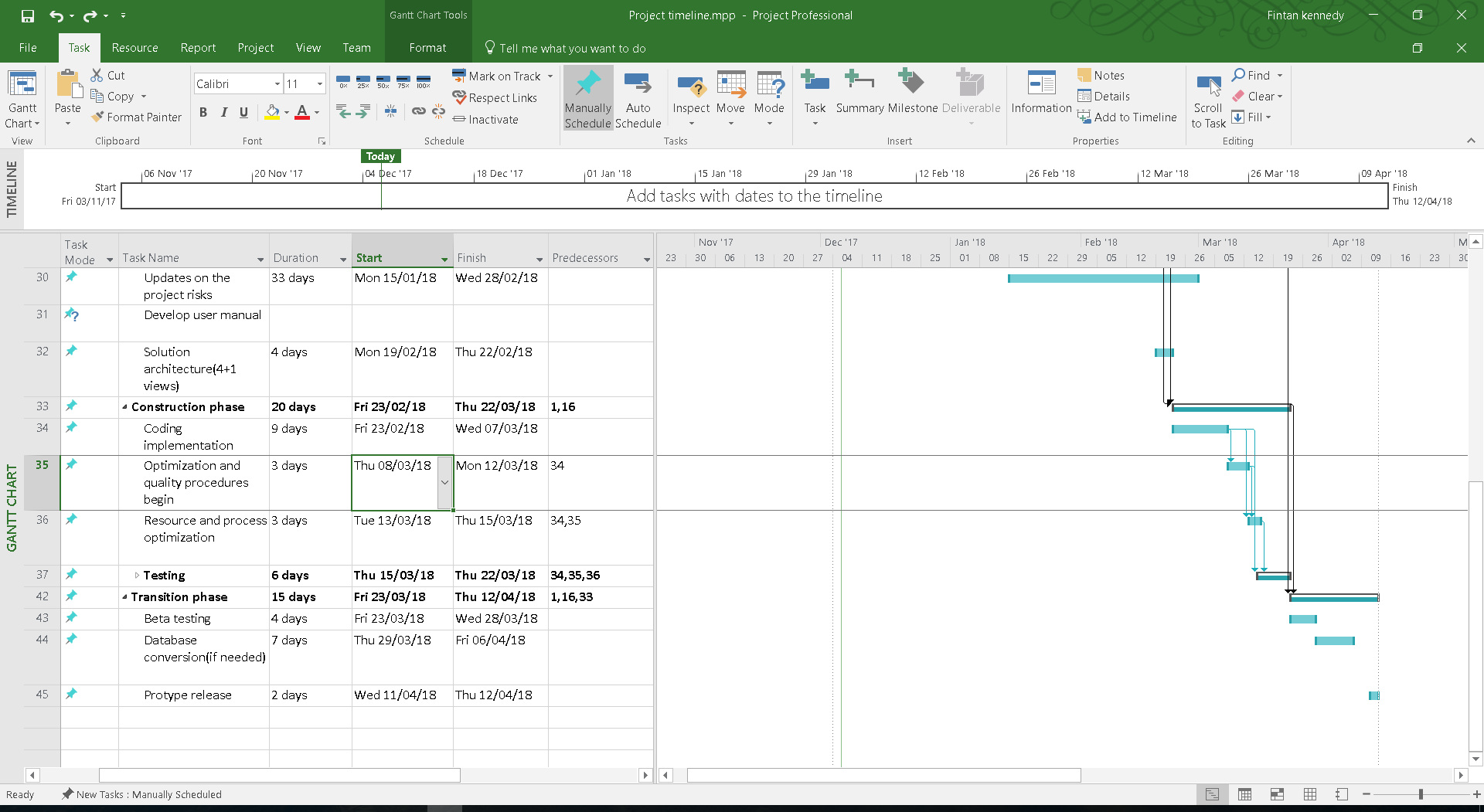
## 

## **Project Plans:**

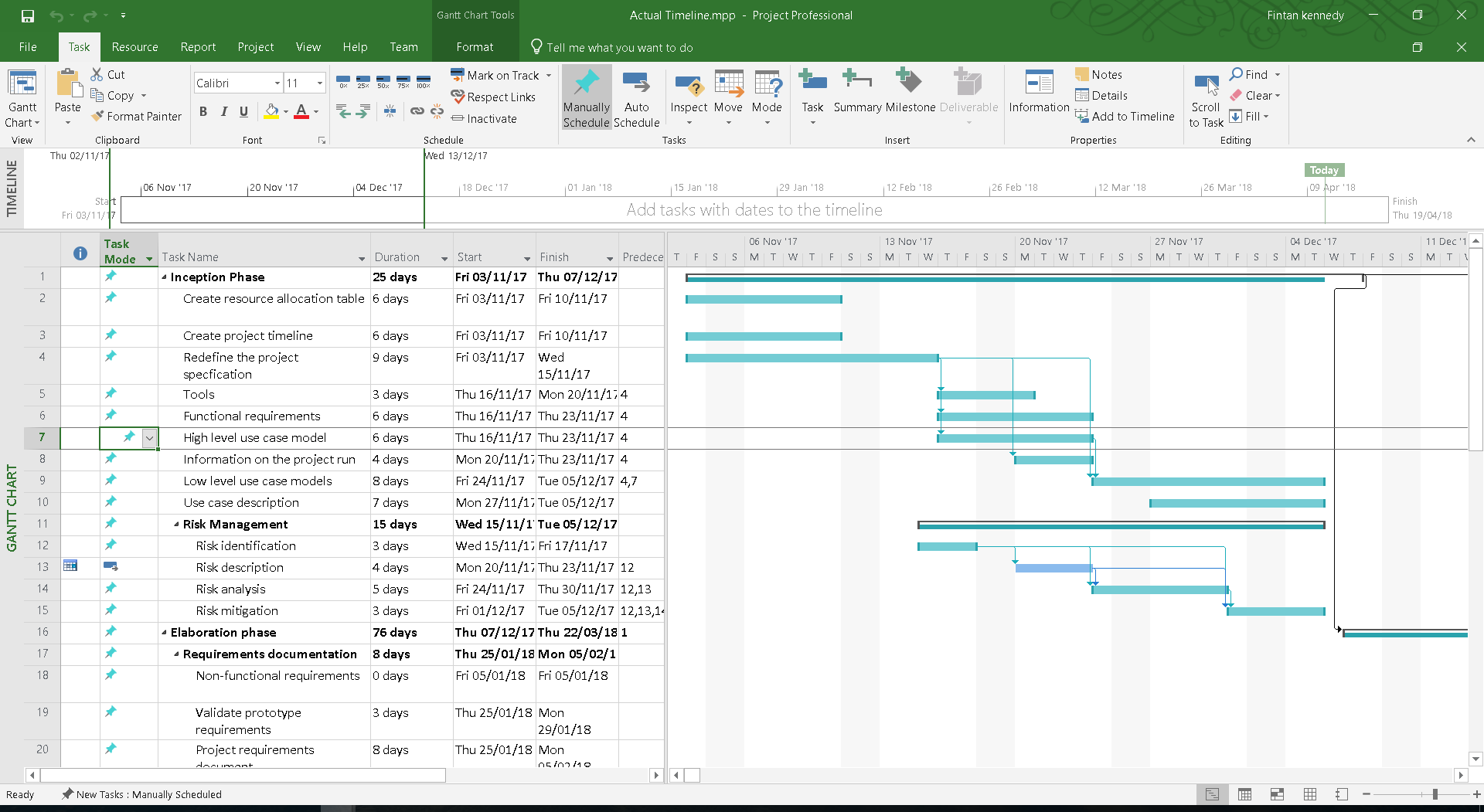
### *Expected Project Timeline:*

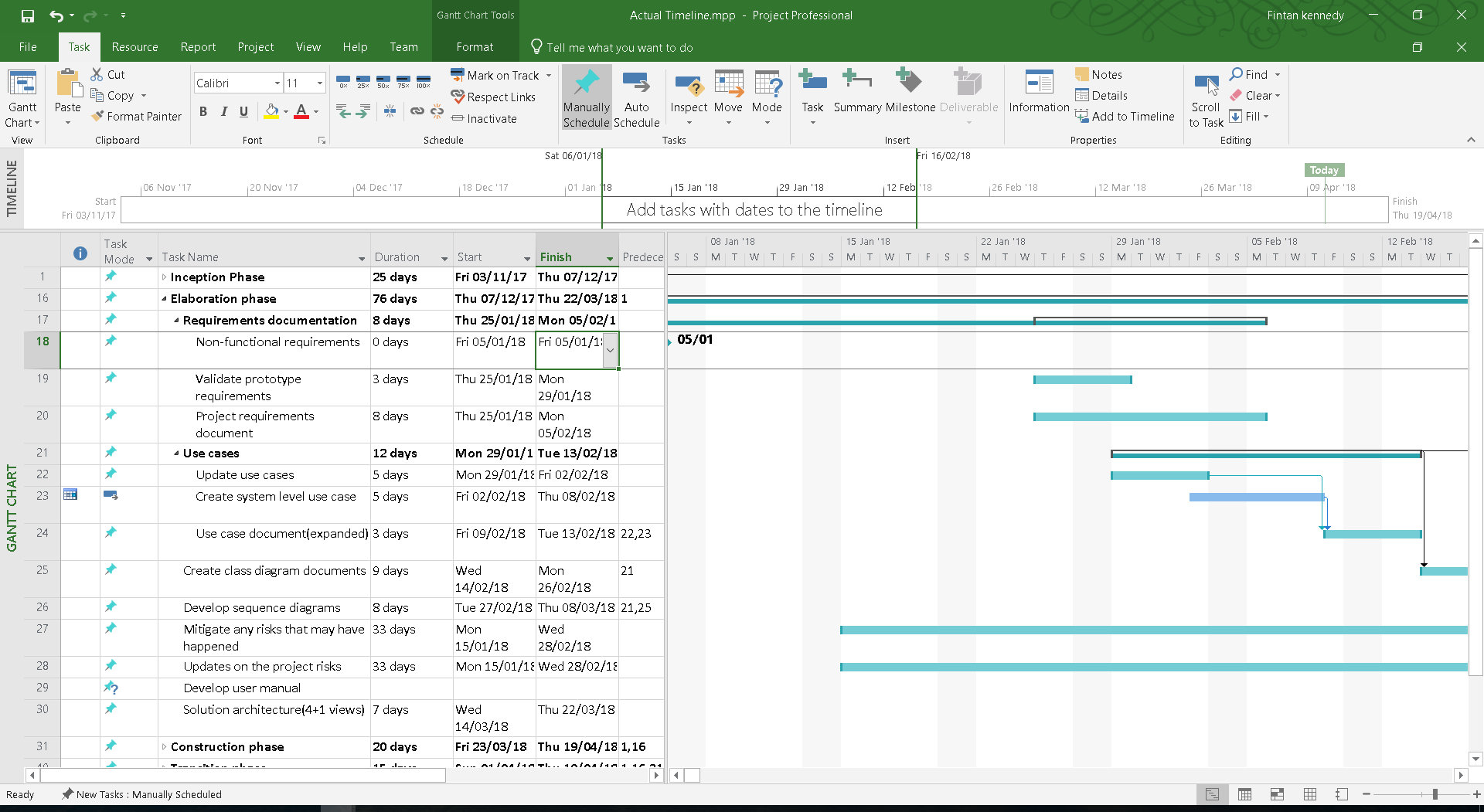


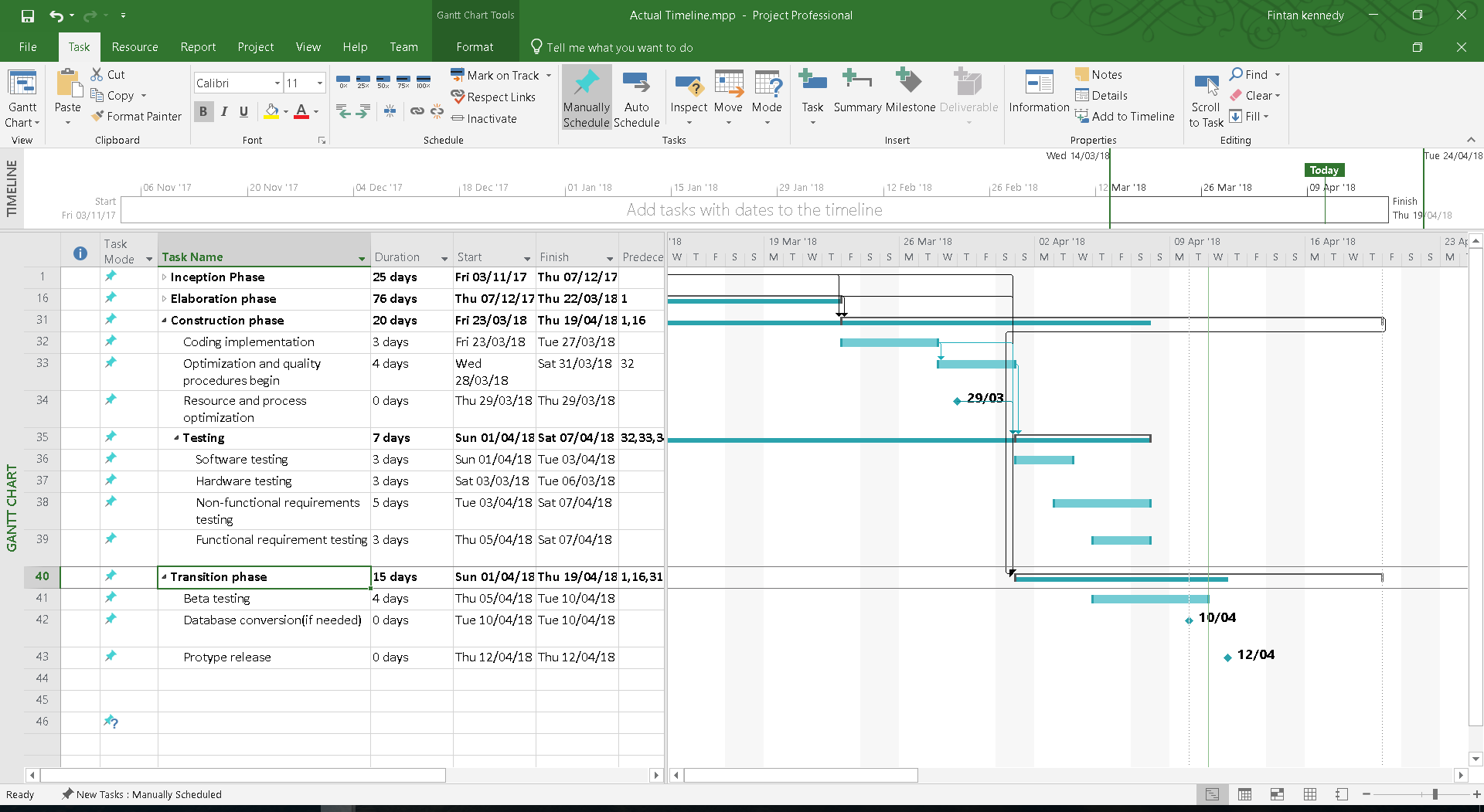
a



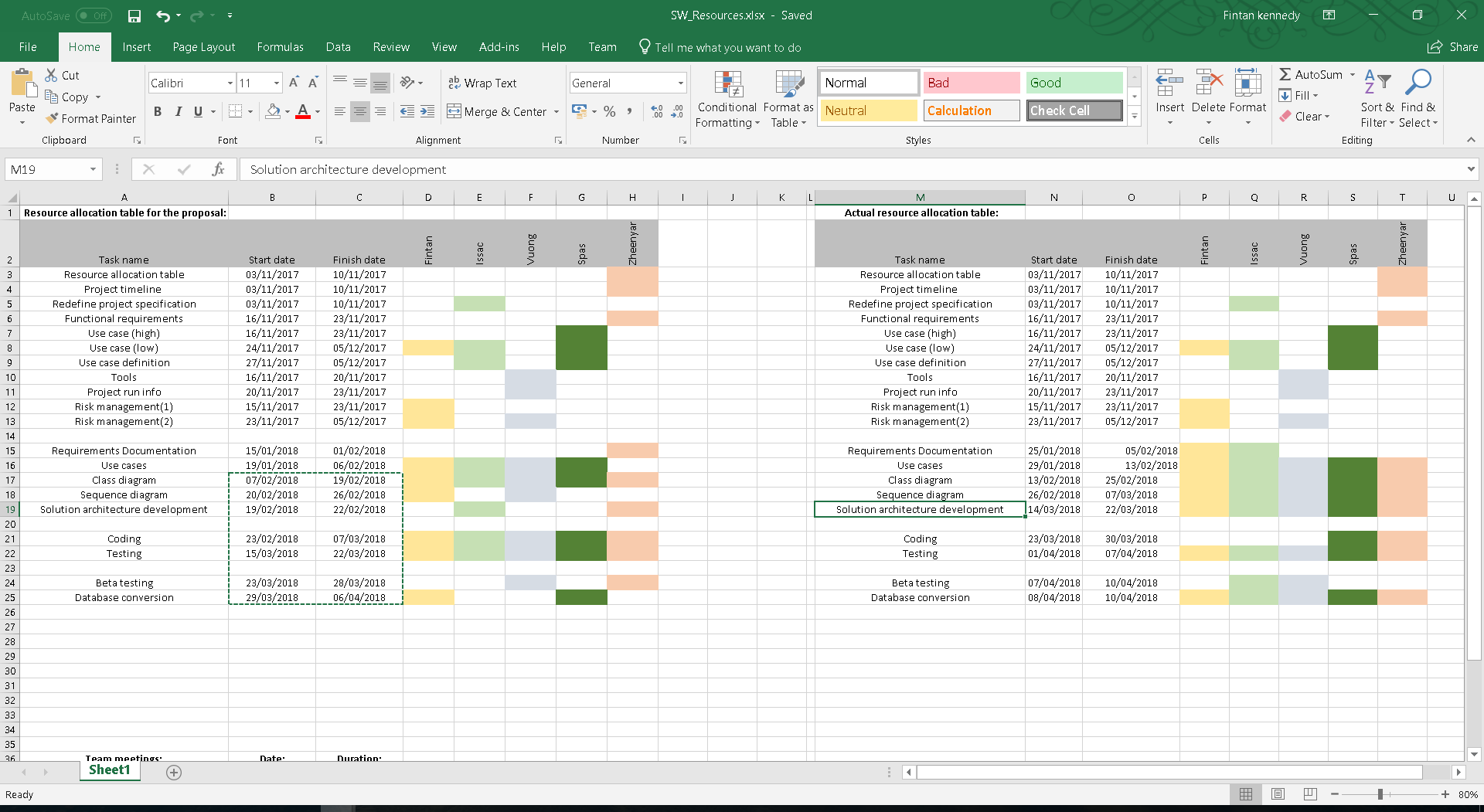
### *Actual Project Timeline:*



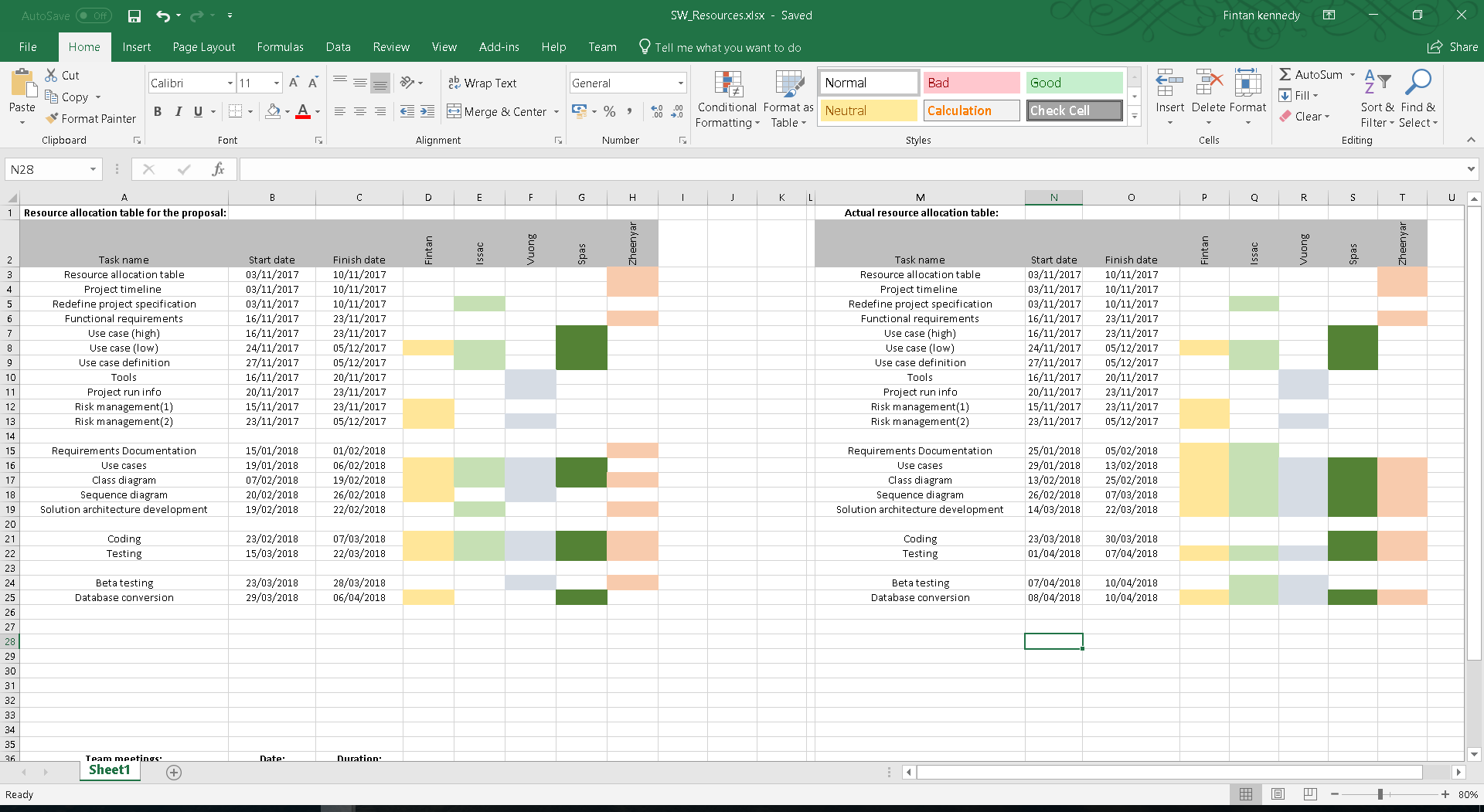




*Expected Resource Table:*



*Actual Resource Table:*



The colours used in the resource allocation tables are just to help distinguish whom completed which task. They do not represent different amounts of effort used per person per task.

## **Minutes for the meetings:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Team meeting** | **Date** | **Duration** | **With Maryam or without** | **Signature** |
| Meeting 1 | 16/11/2017 | 53mins | Without | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |
| Meeting 2 | 30/11/2017 | 40mins | Without | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |
| Meeting 3 | 04/12/2017 | 2.5hours | Without | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |
| Meeting 4 | 15/02/2018 | 10mins | With | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |
| Meeting 5 | 22/02/2018 | 17mins | With | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |
| Meeting 6 | 08/03/2018 | 15mins | With | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |
| Meeting 7 | 15/03/2018 | 20mins | With | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |
| Meeting 8 | 10/04/2018 | 1.5hours | Without | https://scontent-lht6-1.xx.fbcdn.net/v/t1.15752-9/30653112_2061860797176631_8907184232514715648_n.jpg?_nc_cat=0&oh=f8cc18f2731ecba6879dc86e4531693d&oe=5B755DB0 |

|  |  |
| --- | --- |
| **Members of the Wizards** | **UWE ID** |
| Fintan kennedy | 14022421 |
| Issac Rickwood | 16009810 |
| Spas Spasov | 15038097 |
| Vuong Khac Duy | 17036268 |
| Zheenyar karim | 15020327 |

# **Requirements Documentation:**

## **Introduction:**

For our project we have been assigned to create a piece of software to help the company Successful Brothers manage many different aspects of their company. The first thing that we have been made aware of is that this software must function individually in five different locations throughout the country, but must also be all managed by a single network that links all the five dealerships from the main office within Bristol. Due to this we must design, not only an individual piece of software, but also a network software that will enable all the dealerships to be linked together as one. As for the functionality of the software there are many areas that need to be addressed. Logging car’s services and new part deliveries/sales is something the system must be able to do. There will need to be a database that will store all the services that have been done on cars (including details of the service) as well as recording deliveries of new parts and the sale of parts to customers. These aspects will require a database to function as this is the best method of storing them. Processing payments is another aspect of the software that needs to be heavily considered. The DMS must be able to invoice customers, record all transactions that take place for both customers and part buying as well as have a database that is specific to each dealership, so they can keep their own financial records. Managing stock is another aspect of the software that must be addressed. Using the DMS system, each dealership will need to be able to check what stock they have available at any given time. On top of this, if something becomes low/out of stock then the DMS should be able to automatically order new stock once the stock levels reach a certain amount and refill it to another specific amount. Another function of the DMS is to be able to search for cars to be sold to customers that require them. The DMS should be able to be used to find a car from another dealership if one dealership doesn’t have access to it. It should then be able to get that car relocated to the dealership that requires it as well as manage any possible servicing or delivery that might be required. The DMS is only required to process cash payments and not online ones. The final function of the DMS is to be able to view employee performance for any given dealership. This function will allow for someone at a dealership with the correct permissions to view the records of a certain employee in terms of sales, behaviour, attendance and work ethic. This can then be used to come up with plans to improve employee performance depending on which areas are lacking.

## **Actors Definitions:**

**ACT0: DMS Administrator:**

*The main job for the DMS administrator is to maintain the administration system. The DMS admin is the middle point between the customer and the system, logging in the customers details and the car service details (both about the car and what’s wrong with the car).*

**ACT1: DMS Customer Database Administrator:**

*The customer database Administrator has the job of ensuring that the database featuring all of the customers information is working properly without any problems. The customer database manager must ensure that new data is being added correctly and the older data is being stored without being compromised. The DMS Customer Database admin should be able to search for a customer and be able to view the customers contact information as well as how the customer interacted with the system (Was it a service, sale etc.)*

**ACT2: Parts Database Administrator:**

*The parts database Administrator must ensure the database featuring the parts data is working properly and doesn’t contain any bugs. They must make sure that the database is storing the correct information about each part and is ordering new parts when stock starts to run low. Since the ordering of stock is automatic the parts database maintainer must be fully aware all the time that this feature is working, and Successful brothers always have the necessary stock numbers. The parts database administrator will have to be able to search the dealerships parts databases for a certain part and view the relevant information on each individual part.*

**ACT3: Service** **Database Administrator:**

*The service database administrator must ensure that the service database is running smoothly allowing the ‘Successful Brothers’ staff to run at maximum capacity without any delays. This database administrator must keep the service database from failing ensuring the relative information is stored and easy to view. This can range from data on the car being serviced and the parts needed to who is physically servicing the car.*

**ACT4: Car Service Manager:**

*This actor oversees the running of each individual service. While the ‘Service Database Administrator’ is concentrating on the storing and displaying of the data, the car service manager is concentrating on the booking and running of each service. The car service manager will have to book the mechanic for each service along with the locating of the needed parts. When the service is completed the car service manager marks the service removing it from the relative list.*

**ACT5: Car Service Report Manager:**

*The car service report manager should be able to generate the car service reports at any point and view any information regarding services that they may need. Once the service has been completed by the car service manager and logged into the system the report is automatically created and available for viewing.*

**ACT6: Car Sales Manager:**

*The car seller’s job is to log in the selling of a car once the keys have exchanged hands. When the customer has finished the necessary paper work and the car is officially sold then the Car Seller logs the car into the system and marks the car as sold. The car seller must know the necessary information on the car ensuring that before being sold the car has passed all the tests, that it’s road ready and what dealership the car is in.*

**ACT7: Part Sales Manager:**

*The parts sales manager is in charge of the selling of car parts and the logging of those sales. They must ensure that information on each sale is placed into the correct databases, so the other employees will be able to do their job. They must be in constant communication with the ‘parts database administrator’ ensuring that the stock numbers are correct and if not then these errors are fixed.*

**ACT 8: Car Sales Report Manager:**

*The car sales report manager should be able to generate the car sales reports at any point and view any information regarding sales that they may need. Once the sale has been completed by the car sales manager and logged into the system the report is automatically created and available for viewing.*

**ACT 9: Parts Sales Report Manager:**

*The parts sales report manager will view the part sales reports and can see all the information needed to complete their job. Again once the sale has been completed by the parts sales manager and logged into the system the report is automatically created and available for viewing.*

**ACT 10: Branch Manager:**

*There should be five branch managers present in this system, one for each of the dealerships. These branch managers are in charge of their respective dealership keeping an eye on the everyday running of the branch. They do this by looking over the dealerships reports and talking to the employees. The branch manager should be able to look at their branches sales and purchases ensuring that the branch is working within their means and is making a profit.*

**ACT 11: Company Director:**

*The company director has the job of overseeing everything that happens in the company. They ensure the company as a whole is working smoothly keeping an eye on the strong and weak points of each dealership. The Company Director will do this by viewing all the reports and the accounting system. Here they should be able to look at all the sales and purchases for each branch and therefore should be able to see where the dealership can improve.*

**ACT 12: DMS Accounting System Manager:**

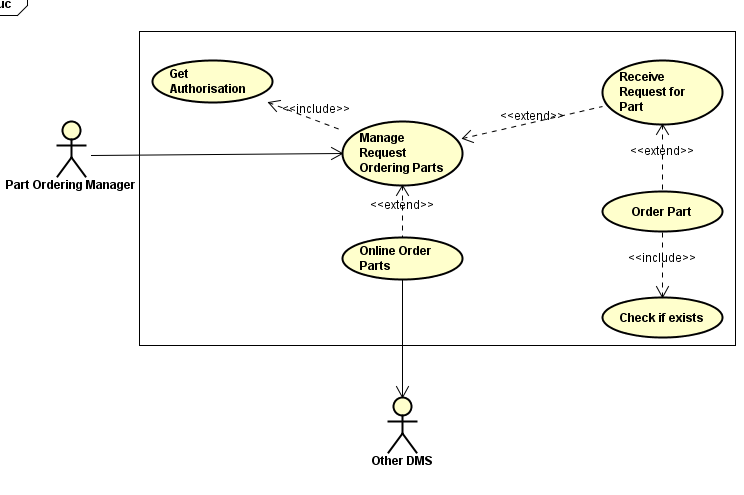
*The DMS Accounting System Manager is basically the accountant for the ‘Successful Brothers’. They are in charge of the businesses finances ensuring the business is going in the right direction and is making money. They must ensure that the company’s finances are stable comparing all the sales reports against the expenditure reports. The Accounting system manager must also compare the expected sales and the actual sales to help prevent any money being lost or stolen.*

**ACT13: Part Ordering Manager:**

*The part ordering manager is a very important actor in this system. Without them the auto ordering system may be broken and by the time its realised the stock shelves could be empty. They ensure that once the stock reaches a certain level new stock is ordered. The part ordering manager must have access to the part ordering database.*

## **Specification of core use-cases:**

### *Isaac Rickwood – Part Ordering Mammogrid*



**Description:**

This use case shows the ‘part ordering use case’ and how the DMS system will order new parts when a dealerships requires them. The use case clearly shows just how each stage of the process will be managed as well as what actors are involved in it and what stages can be accessed and which stages are mandatory. This use case shows when either the system will automatically order new parts or when the ‘part ordering manager’ will manually order new parts when required.

**Actors:**

* Part Ordering Manager – The user who will order the parts
* Other DMS – The other dealerships that can transfer the dealership in question the parts required.

**Flow of events:**

**Where there is a basic flow:**

The flow of ordering a new part for the dealership is as follows. The user ‘Part Ordering Manager’ will access the part of the DMS associated with ordering new parts. They will first get authorisation to order parts through logging into the system. Then they will receive the parts that are needing to be ordered. They will then attempt to order the new part by checking if it first exists within the part database as a part that has been previously obtained. Then, before the online order is completed, other DMS dealerships will be checked to see if they have the part. If another dealership has spares then the order won’t be processed and the dealership will send the part, otherwise an online order will be processed for the part.

**Alternative flow (if something went wrong)**

The following is an alternative flow of events in the case that a part doesn’t exist when checked against the existing parts. The flow will start in much the same way, the user ‘Part Ordering Manager’ will access the part of the DMS associated with ordering new parts. They will first get authorisation to order parts through logging into the system. Then they will receive the parts that are needing to be ordered. Upon checking if the part exists and receiving confirmation that it doesn’t the system will do the following. The user will get a message that the part doesn’t exists and will get an opportunity to check the part requested to see if a simple human error has been made. If there is no human error then the user will check the part against the other DMS’s databases to see if they know of it. In the situation that all of these checks fail then the user will be unable to order the part and will have to get verification from other sources to see if an external error has occurred.

**Non-functional:**

For the non-functional requirements for this use case they will be exactly the same as the non-functional for the rest of the project in the ‘product non-functional requirements part of the report’.

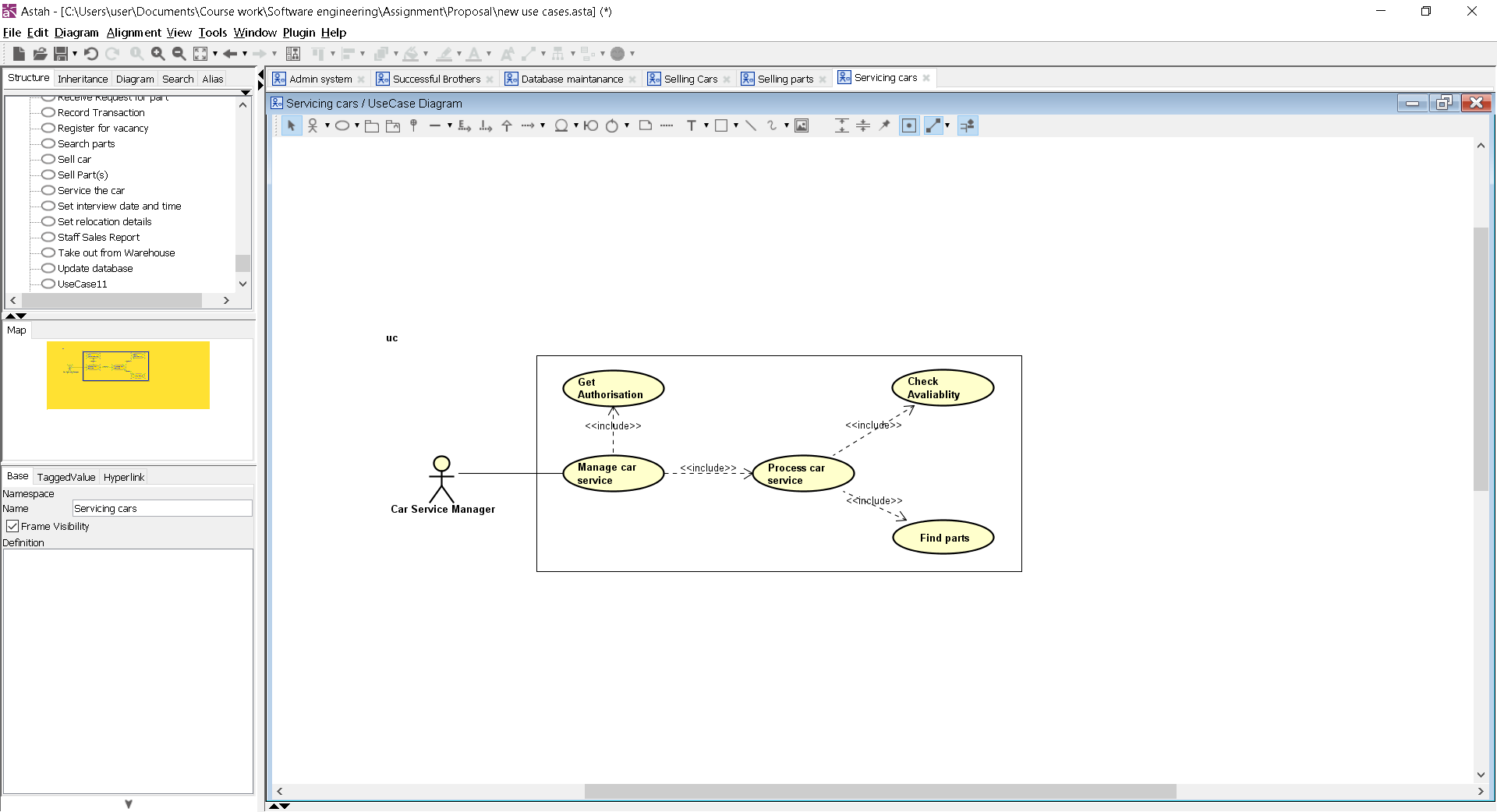
**Pre-conditions:**

* The Part Ordering Manager will be alerted to a part that has a low stock level within the dealership.
* The database will be checked to see if the part exists.
* The database will be checked to see if the part is in stock or not.
* The database will be constantly checked by the system to see if a part has gone completely out of stock or not.

**Post conditions**

* A part will be ordered online successfully and delivered to the dealership, updating the stock count in the database.
* A part will be ordered from another dealership and delivered to the dealership, updating the stock count in the database.
* A part will be unavailable for order from anywhere and the Part Ordering Manger will inform the appropriate people of this information.
* The part didn’t exist within the database and the Part Ordering Manager will inform the appropriate people of this.

### *Fintan Kennedy – Car Service Mammogrid*



**Description:**

This use case describes the task that the ‘Car Service Manager’ actor may undertake to log a car in for service for the ‘Successful Brothers’ company. Whenever a customer has a problem with their car and they need a fix then they will bring it into the garage. When the car arrives, it will be logged into the system with the suitable information. The car will then go through certain steps to try and overcome the error until a fix is found.

**Actors:**

Car Service Manager:

*This actor oversees the running of each individual service. While the ‘Service Database Administrator’ is concentrating on the storing and displaying of the data, the car service manager is concentrating on the booking and running of each service. The car service manager will have to book the mechanic for each service along with the locating of the needed parts. When the service is completed the car service manager marks the service removing it from the relative list.*

**Flow of events:**

**Basic flow:**

* ‘Car Sales Report Manager’ logs into the system
* This report manager begins building a report
* The car sales report manager selects the dates that they wish to search within
* The system builds the desired report with all the sales within the dates accurately showing all the necessary information
* The manager views the report, assessing the required information

**Alternative flows:**

* The following is an alternative flow of events in the case that a part doesn’t exist when Part is out of stock
  + The mechanic searches the ‘parts database’
    - * The necessary part for the service is out of stock in all locations
      * He contacts the ‘Parts Database Administrator’ about the ordering of the new part
      * The ‘Parts Database Administrator’ searches the previous orders to see if the part has been re-ordered by the automatic ordering system
      * The part is found to be in transit to the garages as it has already been ordered
      * Part arrives, and the service continues
  + Receptionist enters the wrong contact details into the system
    - Receptionist enters the wrong details for the customer (exception not alternative flow)
    - Receptionist enters the wrong contact details upon car arrival
    - The car is unable to be returned to the customer until the customer contacts the shop
    - Car must remain in the garage and use case is stopped until collection

**Non-functional:**

All non-functional requirements mentioned in the previous section of the project is still needed here

**Pre-conditions:**

A car is needed to be serviced

There is car that needs a service and is sitting in the garage. Without the necessary car there would be no need for this use case to be run.

The ‘Car Service Manager’ can log into the system to log the service

The ‘Car Service Manager’ must be able to log into the system. We are assuming that the ‘Car Service Manager’ can access the relevant documents so that he can begin doing his jobs.

**Post conditions**

The mechanic has successfully found a fix for the car

The car has been successfully serviced and the mechanic has been able to find the cause of the problem and fix it using the necessary parts. The car is then returned to the customer.

Customer has paid for the service

The customer comes in to collect the car and is given a quote for the servicing. The customer completes the payment using cash which has been taken note off and a receipt and invoice is issued. A note of the payment is placed into the ‘DMS Accounting System’ reports for future reference.

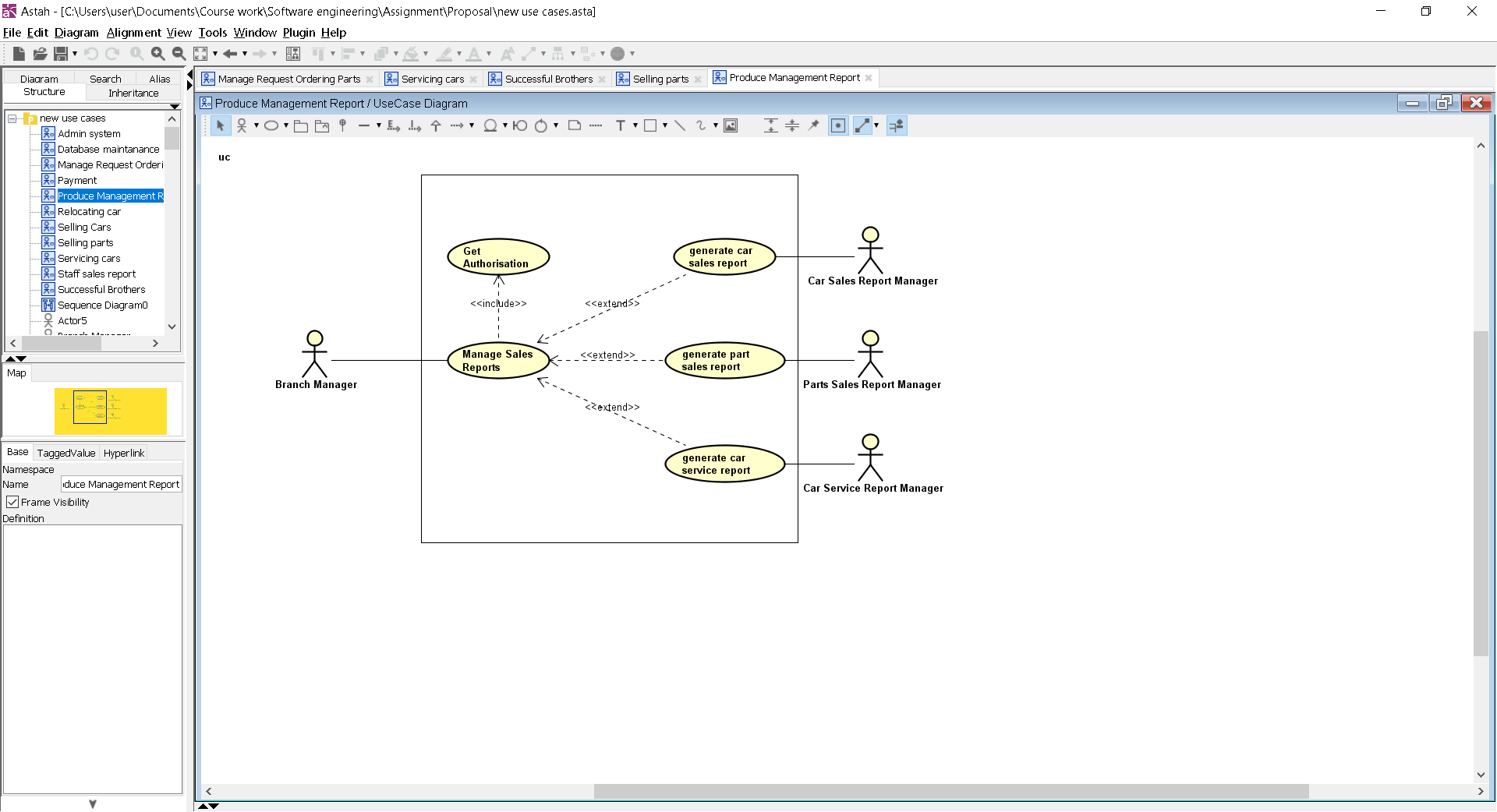
Car service is marked as complete in all the reports

Once the servicing has been completed the service will have to be marked as complete in all of the necessary reports.

Databases have all been updated

As per most services, a cart part was needed to fix the problem the customer was having. Whenever a part is used by the mechanic, the same part should be deducted from the stock levels. This is so the ‘successful brothers’ can keep track of stock levels and the automatic ordering of parts can work without any issues. Therefore, we must assume that at the end of the use case the parts have been removed from the ‘parts database’ and the service is updated on the ‘Service Database’.

### *Spas Spasov – Car Sales report Mammogrid*



**Description:**

In this use case the ‘Car sales report manager’ actor should be able to review the sales of cars in reports containing the relevant information so that they can complete their job. When a car gets sold a report should be automatically generated by the system allowing Successful Brothers to keep track of all incomings and outgoings. These reports must be stored so that they can be located at a later date

**Actors:**

Car Sales Report Manager:

*The car sales report manager should be able to generate the car sales reports at any point and view any information regarding sales that they may need. Once the sale has been completed by the car sales manager and logged into the system the report is automatically created and available for viewing.*

**Flow of events:**

**Basic flow:**

* ‘Car Sales Report Manager’ logs into the system
* The system asks the user whether they would like to enter dates to refine the report
* The ‘Car Sales Report Manager’ enters the desired dates into the system
* System searches through car sales and creates a sales report
* The report is displayed where the ‘Car Sales Report Manager’ can review the relevant information

**Alternative flows:**

* No sales within dates
  + The car sales report manager searches sales between two dates
    - ‘car sales report manager’ attempts to build a sales report between two dates
    - System searches for sales between these dates and returns an alert message warning the user that the following report will be empty and asks whether they would like to search again
    - The ‘car sales report manager’ enters a new timeframe and searches again
    - The correct report is displayed for the ‘car sales report manager’ to view
* Report displayed contains incorrect details
  + The system displays data that is either inaccurate or incorrect
    - The ‘Car Sales Report Manager’ opens up a report with the desired guidelines. The data that the system has included in the report is seen to be wrong
    - The ‘Car Sales Report Manager’ acknowledges that the data is wrong is exits the report
    - The report manager then re-enters the system and enters in the parameters again
    - The system re opens the report, this time containing the correct information

**Non-functional:**

All non-functional requirements mentioned in the previous section of the project is still needed here

**Pre-conditions:**

A car sales report has been successfully generated after sale

Once the sale is completed a sale report must be automatically generated. For the sake of this use case we are to assume that this feature is working properly and therefore don’t have to worry about this problem as an alternative path

The ‘Car Sales Report Manager’ can log into the system to log the sale

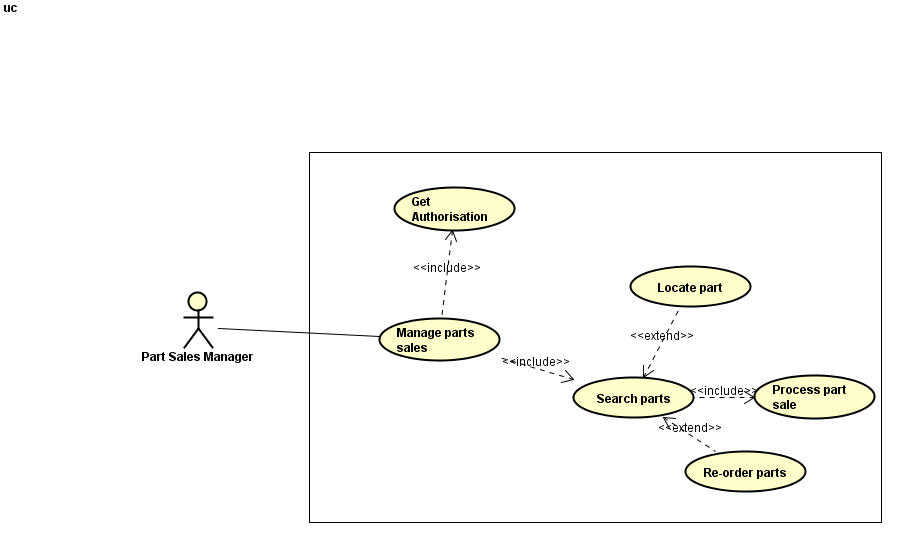
The ‘Car Sales Report Manager’ must be able to log into the system. We are assuming that the ‘Car Sales Report Manager’ can access the relevant documents so that he can begin doing his jobs.

**Post conditions**

A car sales report is made within the desired parameters

The car sales report has been successfully generated and the car sales report manager is happy that the report correctly displays all the sales within the desired timeframe without any errors being shown.

### *Vuong Khac Duy – Part Sales Mammogrid*



**Description:**

This use case shows how the parts sales manager will interact with the system to sell different parts to the customers that require them. They will need to search for the part as well as process the sale once completed.

**Actors:**

Part Sales Manager:

*The parts sales manager is in charge of the selling of car parts and the logging of those sales. They must ensure that information on each sale is placed into the correct databases, so the other employees will be able to do their job. They must be in constant communication with the ‘parts database administrator’ ensuring that the stock numbers are correct and if not then these errors are fixed.*

**Flow of events:**

**Basic flow:**

* ‘Part Sales Manager’ processes the customers and the part sale details
* The part has been successfully logged into the system
* This Part Sales Manager begins by authorising a part sale
* Part Sales Manager search the parts which the customers require
* ‘Part Sales Manager’ contacts the customer to get the detail and storing the order
* The manager views the report, assessing the required information

**Alternative flows:**

* Part is out of stock
  + The mechanic searches the ‘parts database’
    - The necessary part for the service is out of stock in all locations
    - He contacts the ‘Parts Database Administrator’ about the ordering of the new part
    - The ‘Parts Database Administrator’ searches the previous orders to see if the part has been re-ordered by the automatic ordering system
    - The part is found to be in transit to the place as it has already been ordered
* Authorization is declined
  + The sales person requests authorisation while managing a car sale.
    - The authorisation is denied
    - With the part sales manager, the issue is resolved
* Sales person logs the part sale wrong
  + Sales person enters the wrong details for the part sale (exception not alternative flow)
    - Sales person enters the wrong part log details upon a part sale
    - The part sale cannot go ahead without correct logging

**Non-functional:**

All non-functional requirements mentioned in the previous section of the project is still needed here

**Pre-conditions:**

A part is needed to be serviced

There is parts that needs a service and is sitting in the area. Without the necessary parts there would be no need for this use case to be run.

The ‘Part Sales Manager’ can log into the system

The ‘Part Sales Manager’ must be able to log into the system. We are assuming that the ‘Part Sales Manager’ can access the relevant documents so that he can begin doing his jobs.

**Post conditions**

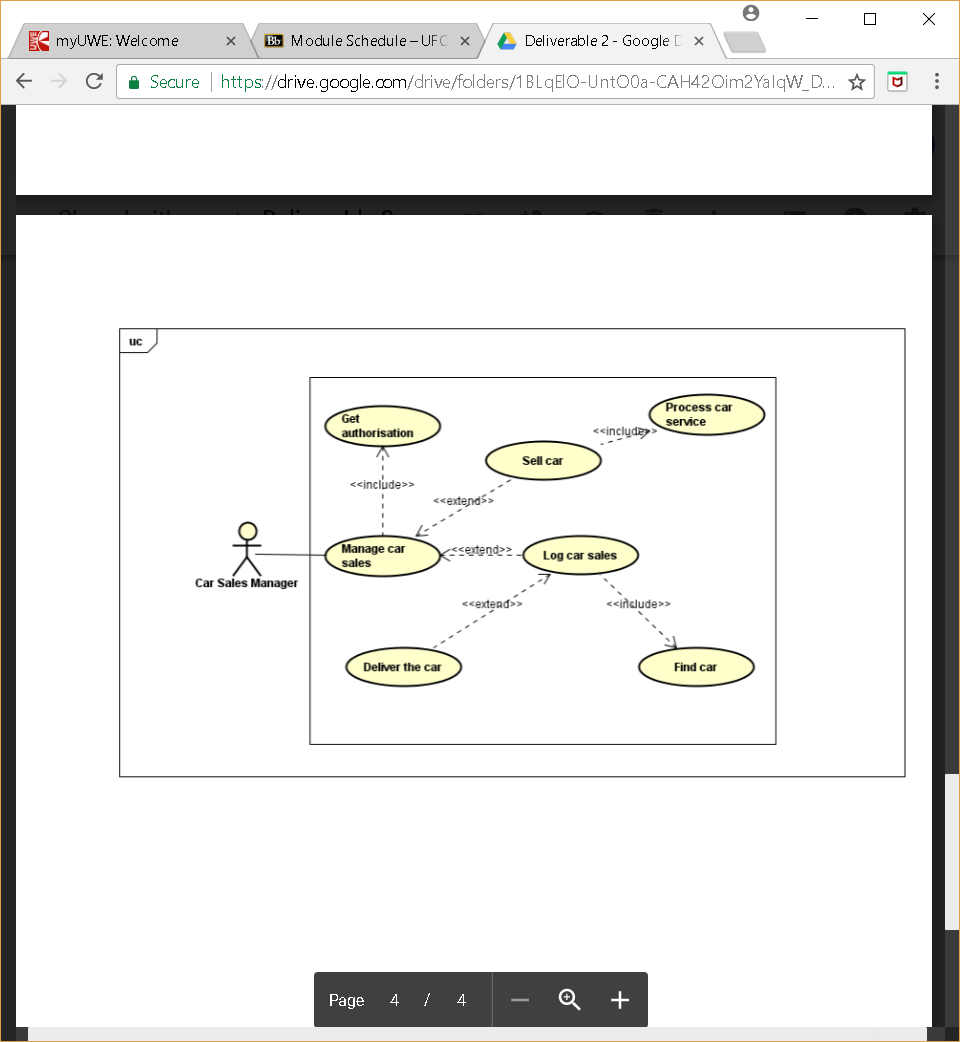
Part sales are marked as complete in all reports

Once the part was sold has been completed the service will have to be marked as complete in all of the necessary reports.

All databases have been updated

After the part was sold, the system must be updated the quality of remaining product in that branch

### Zheenyar Karim *– Car Sales Mammogrid*



**Description:**

This use case shows the flow of tasks in which the car sales manager actor would undertake to manage car sales and the logging/delivering that would come with this. Whenever there is a car sale process, the car sales manager will handle not only the sale of cars but also the car services and the delivering that would come with this.

**Actors:**

Car Sales Manager:

*The car seller’s job is to log in the selling of a car once the keys have exchanged hands. When the customer has finished the necessary paper work and the car is officially sold then the Car Seller logs the car into the system and marks the car as sold. The car seller must know the necessary information on the car ensuring that before being sold the car has passed all the tests, that it’s road ready and what dealership the car is in.*

**Flow of events:**

**Basic flow:**

* + ‘Car Sales Manager’ logs into the system and checks car sales
  + - This car sales manager begins by authorising a car sale
  + - The cars service is then processed
  + - The car sales manager then logs the car sale
  + - The manager views the report, assessing the required information

**Alternative flows:**

* Authorisation is declined
  + The sales person requests authorisation while managing a car sale
    - The authorisation is denied
    - He contacts the ‘System Administrator’ about the authorisation
    - The ‘System Administrator’ investigates the sale
    - The sales person receives the reason i.e. car already sold
    - With the car sales manager, the issue is resolved
* Sales person enters the wrong details for the car sale (exception not alternative flow)
  + - Sales person enters the wrong car log details upon a car sale
    - The car is sale cannot go ahead without the correct logging
    - The sale must be relogged with the correct details or the car sale cannot go through

**Non-functional:**

All non-functional requirements mentioned in the previous section of the project is still needed here

**Pre-conditions:**

A Car is needed to be serviced ahead of a car sale

There is a car sale and the car therefore needs to be serviced. Without the necessary car sale, there would be no need for this use case to be run.

The ‘Part Sales Manager’ can log into the system

The ‘Car Service Manager’ must be able to log into the system. We are assuming that the ‘Car Sales Manager’ can access the relevant documents so that he can begin doing his jobs.

**Post conditions**

The sales person has successfully sold a car

The car has been successfully sold and the sales person has been able to log the system.

A car is serviced

Upon sale, the car is serviced. Once the servicing has been completed the service will have to be marked as complete in all the necessary reports.

The car is delivered

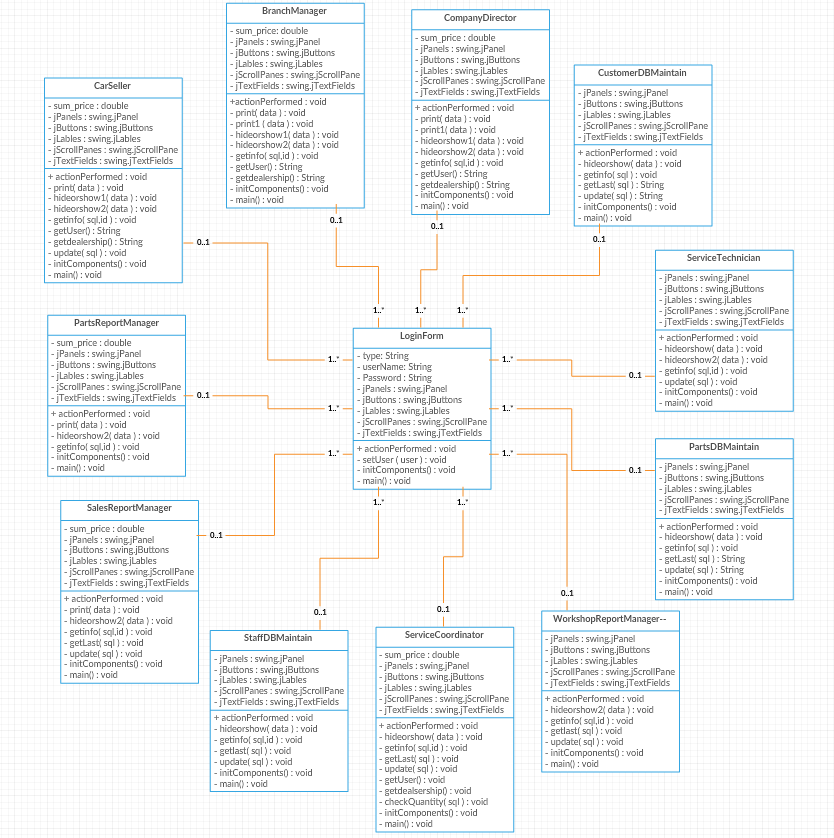
Upon sale and service, the car is delivered to the chosen branch and can be picked up by the customer.

Databases have all been updated

After sale, the car must be taken down from advertisement and logged. All databases must be updated of these changes so that the business can track its sales and performance.

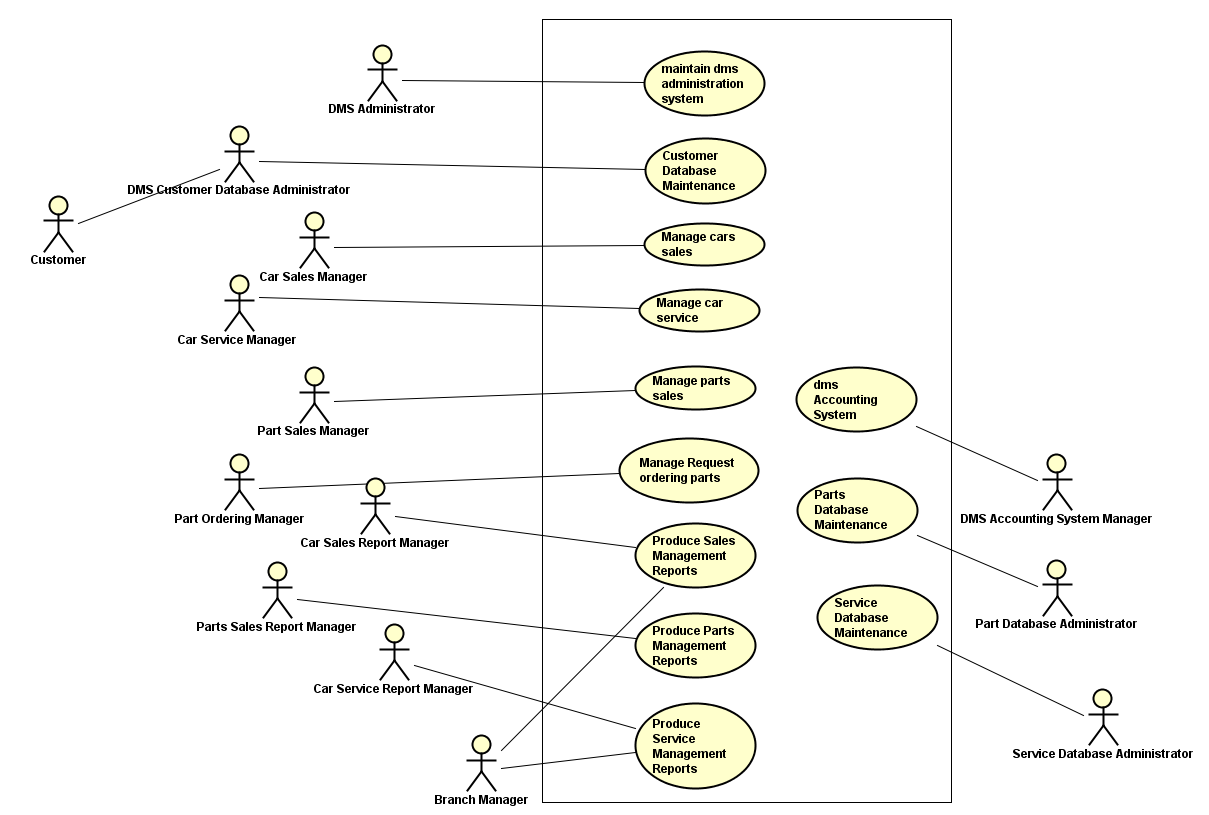
## **System models:**

### *Class diagram*

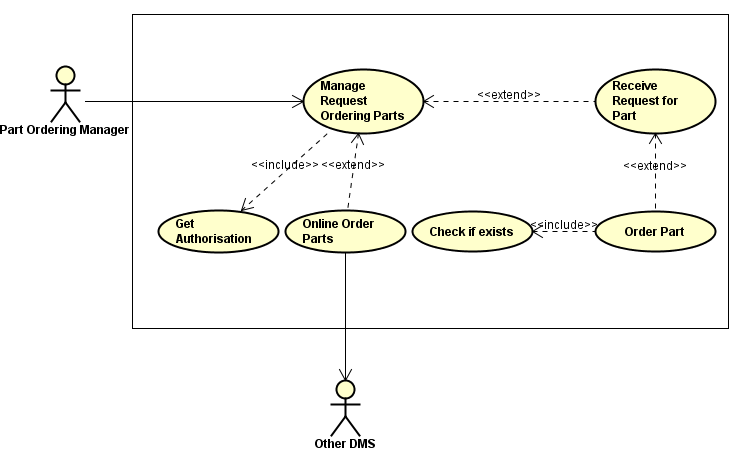
******

### *Use-case views*

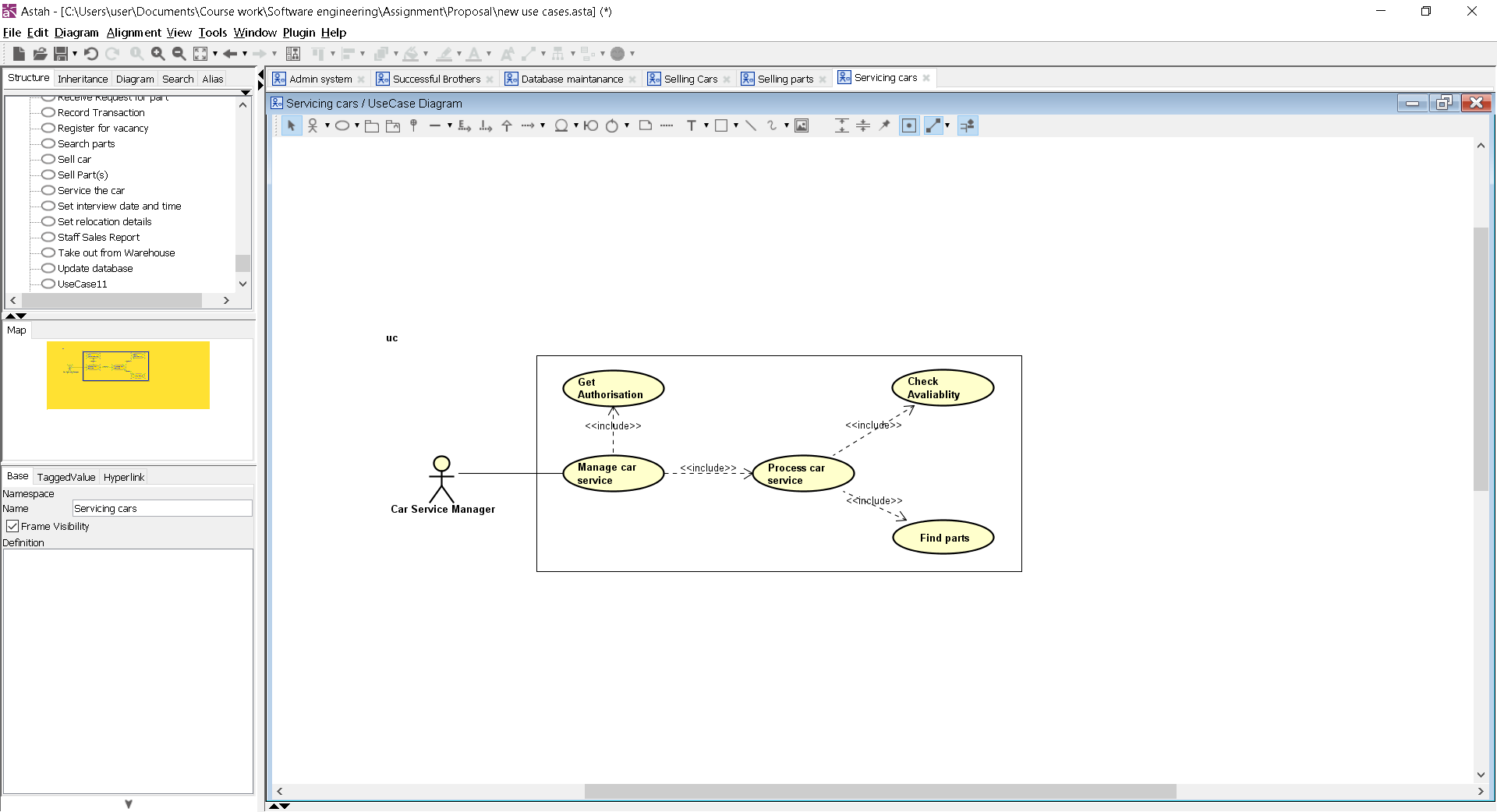
#### High Level Use Case:

**

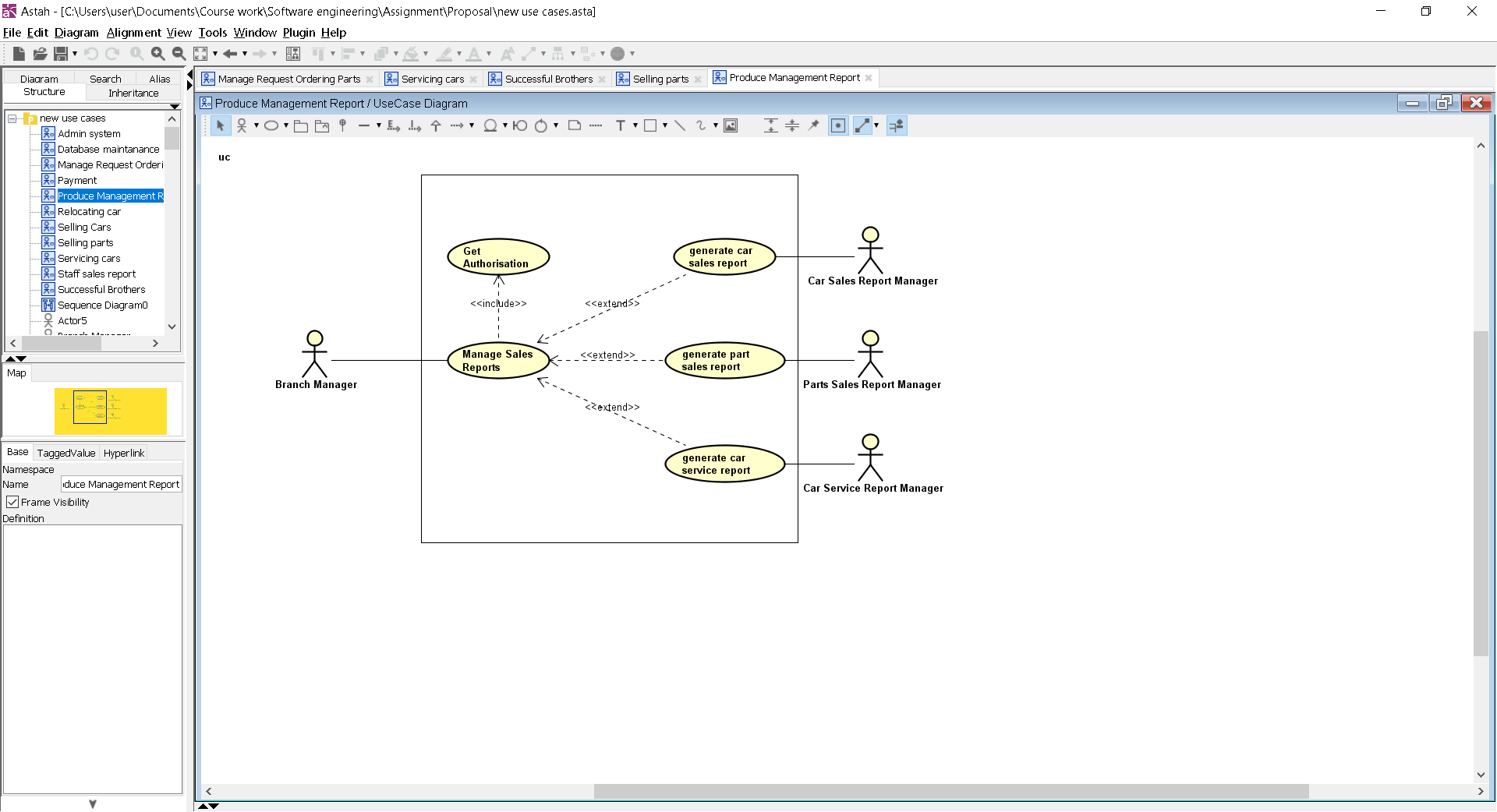
#### Isaac Rickwood - Part Ordering Use Case:

**

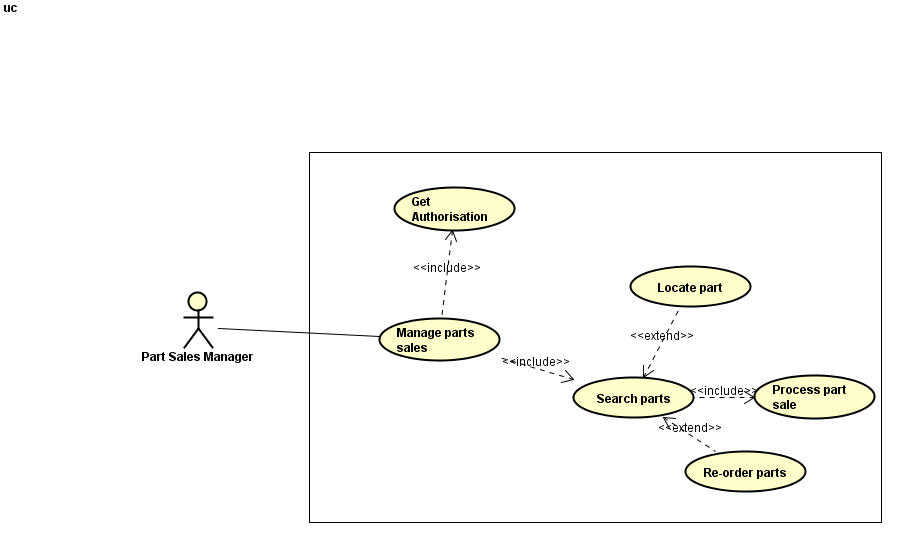
#### Fintan Kennedy – Car Service Use Case:



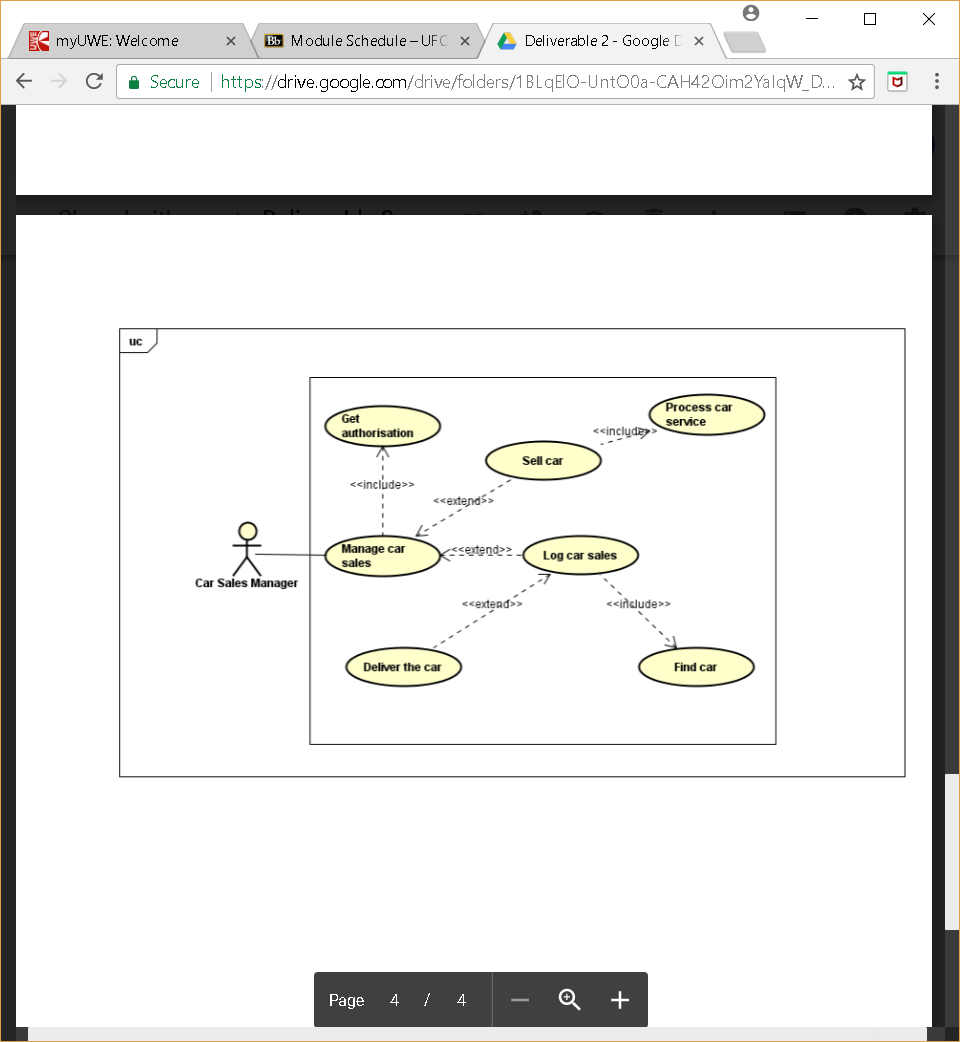
#### Spas Spasov – Car Sales Report Use Case



#### Vuong Khac Duy – Part Sales Use Case

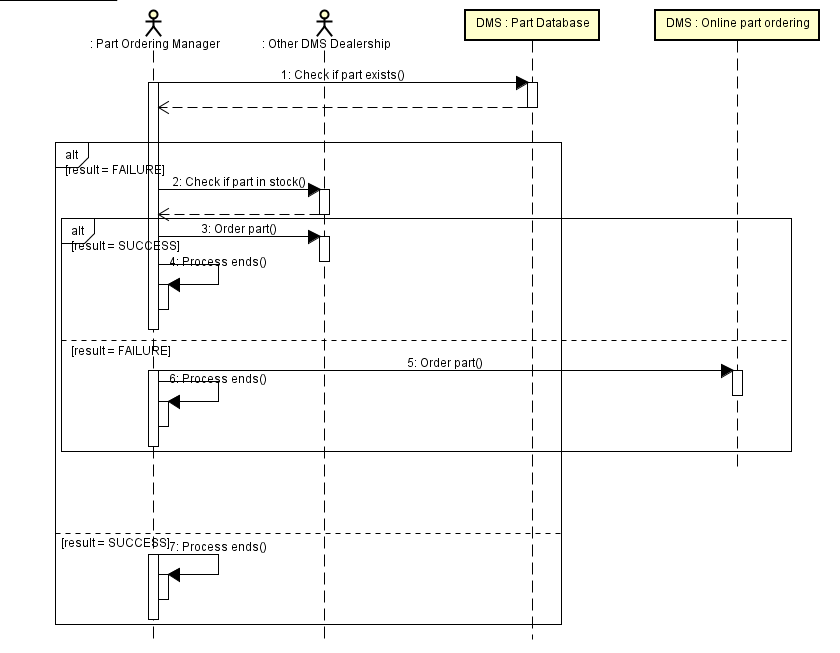


#### Zheenyar Karim – Car Sales Use Case

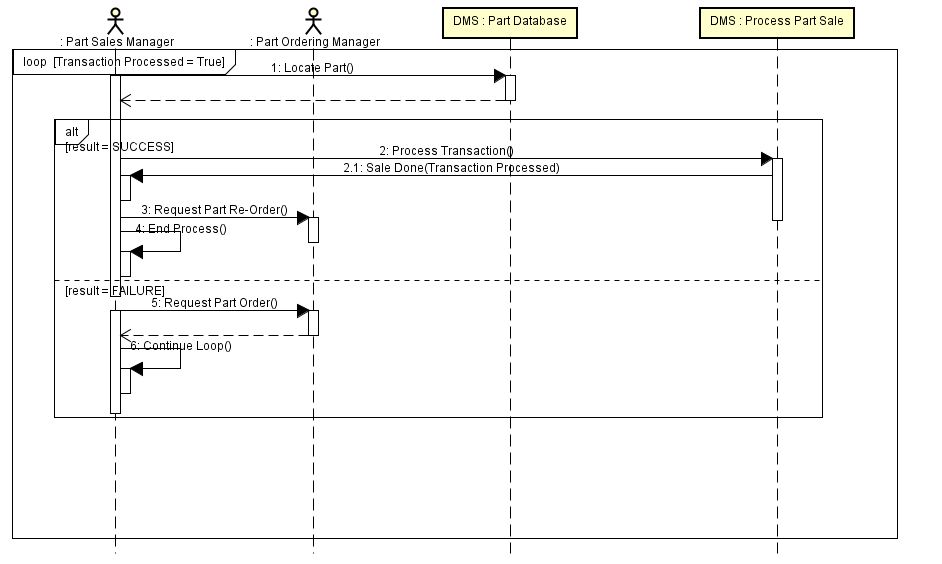


### *Sequence diagrams*

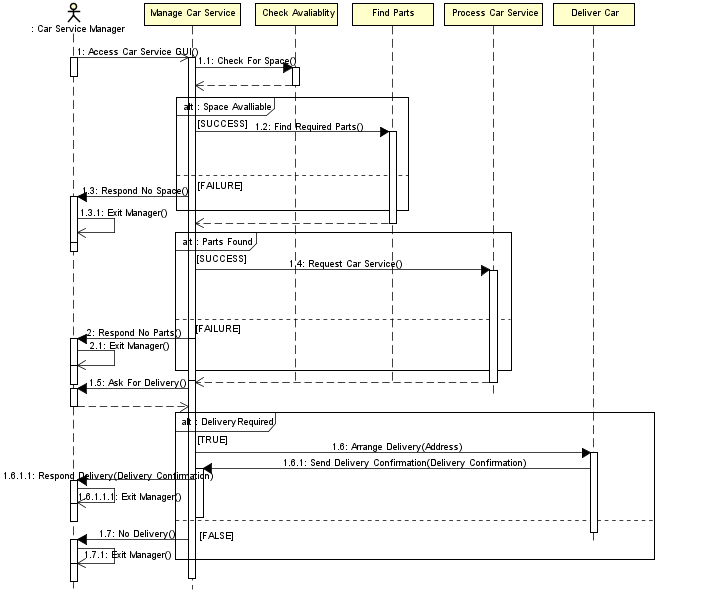
#### Isaac Rickwood – Part Ordering Sequence Diagram



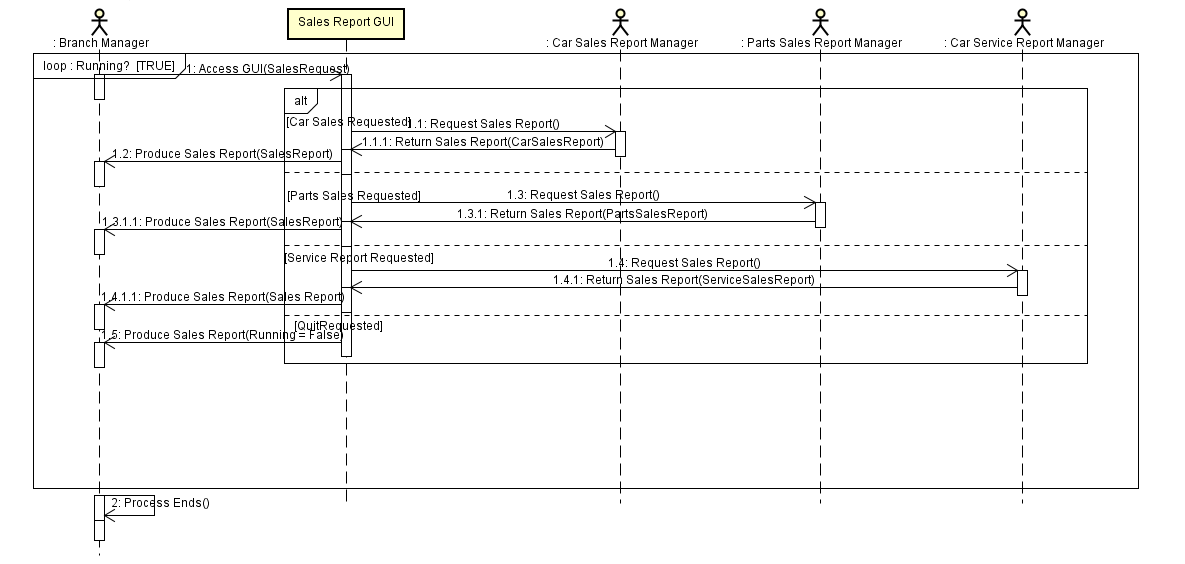
#### Vuong Khac Duy - Part Sales Sequence Diagram



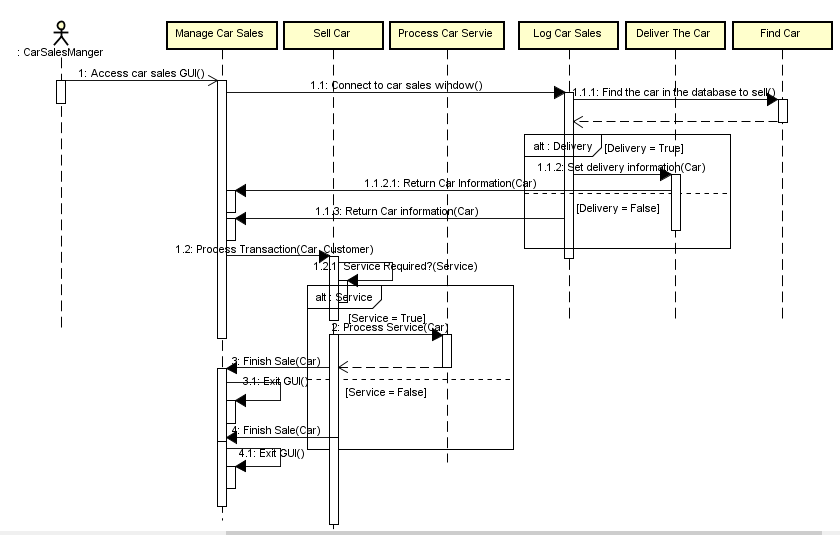
#### Fintan Kennedy – Car Service Sequence Diagram

****

#### Spas Spasov – Car Sales Report Sequence Diagram

****

#### Zheenyar Karim – Car Sales Sequence Diagram



## **Non-functional requirements:**

### *Product non-functional requirements*

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Expectations** | **Priority**  *Red = high Orange = medium Green = low* | **Dependencies** |
| 1. Ease of use | Our system needs to be designed so that it is quick to learn and won’t require much training. Each individual element of the system must be created with ergonomics in mind and be suited towards employees who don’t have much IT experience. |  |  |
| 2. Reliability | The system must be reliable and have minimum down time. This will be to avoid delays within the system such as delays to orders, services and payments which can cost the company lots of money. |  | 1 |
| 2.1 Maintainability | The system should be easy and cheap to maintain. This is because the system should be able to be maintained by very few admins as the system should be very self-sufficient. General maintenance should only include hardware upgrades and database management. |  | 2 |
| 2.2 Serviceability | The system should be simple to service by internal employees. Due to the system being easy to maintain services should be infrequent and should only be performed in the event of a system error or a similar event. |  | 2.1 |
| 2.4 Error handling | All errors should be handled by the system efficiently and shouldn’t lead to an entire system crash. If an error occurs the system should manage the error and alert the user of the error and prompt the user on the appropriate steps they need to take to solve the issue. |  | 1, 2 |
| 3. Range of viable OSs | The system should be able to run on all types of popular operating systems to avoid unnecessary expenses of replacing existing machines. The system should be able to run on Windows, MAC and popular variations of Linux (such as Ubuntu) so that the company doesn’t have to change all computers in their workplaces to accommodate the new system. |  |  |
| 3.1 Range of compatible machines | The system should be able to run comfortably on different machines of differing specifications such as computers, laptops and some tablets. This is important so that workplaces that have limited access to computers can use other devices in order to still use the system. |  | 3 |
| 4. Legal actions | The created system should comply with all legal aspects of a system of this variety. It must hold to the data protection act and other laws of a similar standard so that customers and employees alike are comfortable that any data stored/used within the system will be only used for lawful practice. |  |  |
| 4.1 Security | The system should have adequate security so that it passes any possible security laws revolving around the processing and possible storage of customer data. The system should force all users to use strong passwords to keep their data secure. |  | 4 |
| 5. Employee ergonomics | All employees expected to use the system on a regular basis should be provided the necessary tools to use the system. This should include appropriate seating, computing components and anything else the employee may require. |  | 5 |
| 6. Customer accessibility | The system should be easy for the customer to see what data is stored about them as well as easily request for the employees to book them services, order parts and purchase items without much hassle. |  | 1, 4.1 |
| 7. Expandability | The system should maintain the ability to be expanded to include extra dealerships if required. Despite the system only having 5 dealerships at the moment the system should hold onto the possibility of creating new dealerships in the future of the company. |  | 2, 2.1, 2.2, 3, 3.1 |

### *Organisational non-functional requirements*

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Expectations** | **Priority**  *Red = high Orange = medium Green = low* | **Dependencies** |
| 1. Ease of use | The software used to create the system should be easy to use and understand to avoid any delays within the creation of the system. This will enable the programmers to easily understand the software being used as well as not needing to spend a long time learning the software which would waste a lot of time. |  |  |
| 2. Reliability | The software used must be reliable and have minimum down time. This will be to avoid delays within the system creation such as delays, programming schedules and software errors which can cost the company lots of money. |  | 1 |
| 2.1 Maintainability | The software that is used to create the program should be easy to maintain with consistent updates to keep the software up to date. The software should also have a minimal impact on the system allowing the developers to work unhindered. |  | 2 |
| 2.2 Serviceability | The software should allow the system to be simple to service by internal employees. Due to the system being easy to maintain services should be infrequent and should only be performed in the event of a system error or a similar event. |  | 2.1 |
| 2.4 Error handling | All errors should be handled by the system efficiently and shouldn’t lead to an entire system crash. If an error occurs the system should manage the error and alert the user of the error and prompt the user on the appropriate steps they need to take to solve the issue. The software used to create the system should therefore enable the programmers to create proper error handling to keep the system in working order. |  | 1, 2 |
| 3. Range of viable OSs | The system should be able to run on all types of popular operating systems to avoid unnecessary expenses of replacing existing machines. The system should be able to run on Windows, MAC and popular variations of Linux (such as Ubuntu) so that the company doesn’t have to change all computers in their workplaces to accommodate the new system. To achieve this the software used should be able to create systems that run on all types of OSs instead of just focusing on a single one. |  |  |
| 3.1 Range of compatible machines | The system should be able to run comfortably on different machines of differing specifications such as computers, laptops and some tablets. This is important so that workplaces that have limited access to computers can use other devices in order to still use the system. To achieve this the software used should be able to create systems that run on all types of machines instead of just focusing on a single one. |  | 3 |
| 4. Legal actions | The software used to create the system should be a legal piece of software that has been purchased through legitimate methods from the company that created it. On top of this, the software should not allow access to any unethical programming practices such as illegal data storage etc. |  |  |
| 4.1 Security | The software that is used to create the system should be secure and verified. The software will need to be checked before it is installed onto the working PCs to check it contains no viruses or malware. On top of this the software should enable the programmers to create a system that is secure from external assault and can keep user data safe. |  | 4 |
| 5. Employee ergonomics | All employees that are expected to use the programming software should be trained to a necessary level to use it. This may require training days to be organised if some employees are unfamiliar with the software chosen. |  | 5 |
| 6. Expandability | The software should enable the programmers to create a system that can be extended upon once it has been created. This is important because it is very likely that the system will need further development upon completion. |  | 2, 2.1, 2.2, 3, 3.1 |

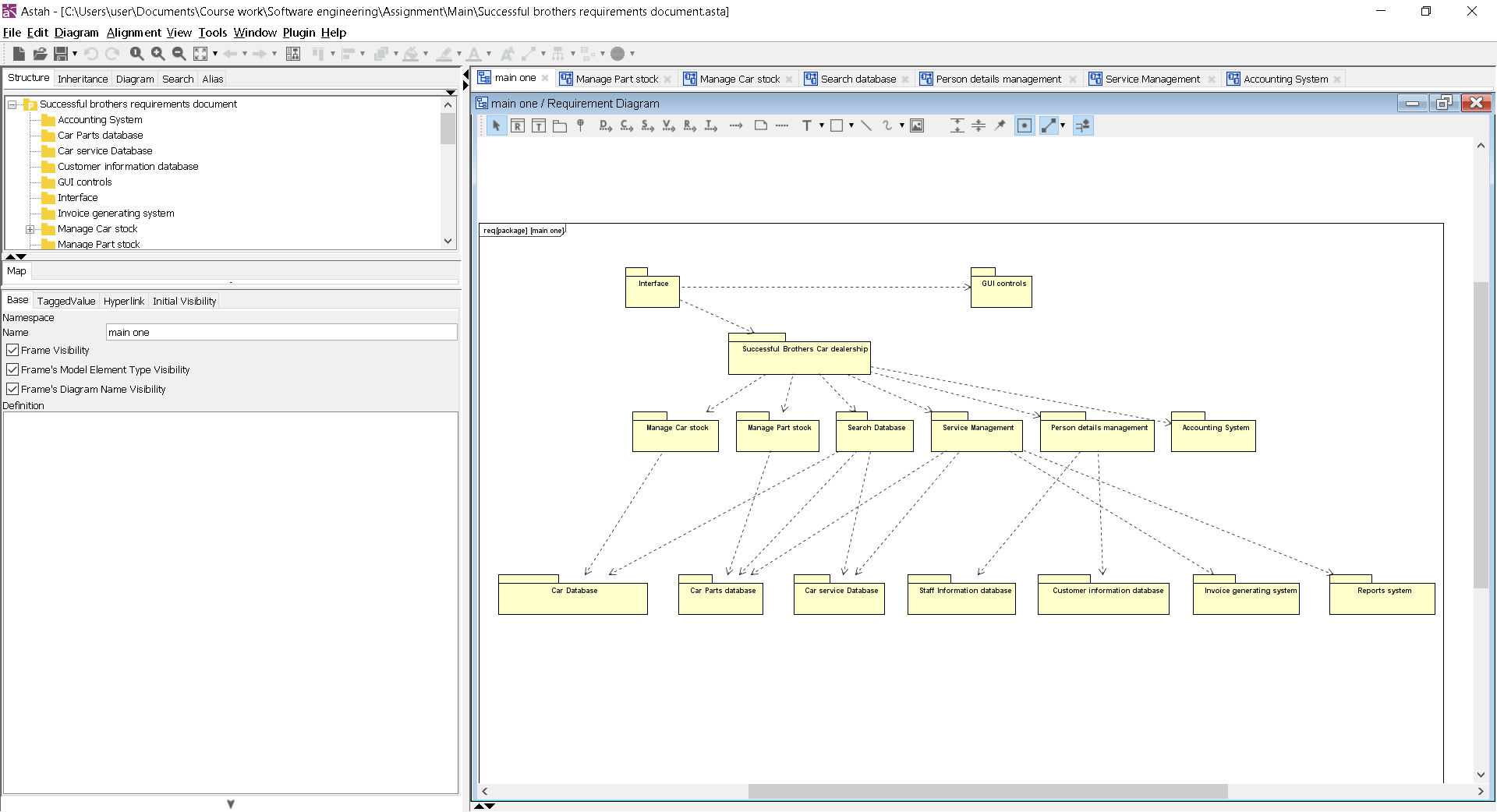
### *External non-functional requirements*

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Expectations** | **Priority**  *Red = high Orange = medium Green = low* | **Dependencies** |
| 1. Ease of use | The system that is created should be easy to use for external entities. This includes shareholders that may want to access the system, customers that want to access the data that they have stored on the database and new employees who join the company in the future. |  |  |
| 2. Reliability | The system must be reliable and have minimum down time. This is important because the company will never know when an external entity may have need to use the system and the last thing that the company wants is to keep people such as shareholders waiting. |  | 1 |
| 3. Shareholder input | The system should be created within the approved boundaries of both the owning company and the shareholders within the project. This is important because if these requirements aren’t adhered to then the shareholders may withdraw from the project if they feel the system is no longer what they want. |  | 1 |
| 3.1 Requirement discussions | The creators of the system will need to have constant discussions with both the shareholders and the owning company to address any problems that arise during the development process. This is important because if a problem occurs with the requirements an alternative solution will need to be created to avoid a dead-end occurring. |  | 3 |
| 4. Legal actions | The created system should comply with all legal aspects of a system of this variety. It must hold to the data protection act and other laws of a similar standard so that customers and employees alike are comfortable that any data stored/used within the system will be only used for lawful practice. |  |  |
| 4.1 Security | The system should have adequate security so that it passes any possible security laws revolving around the processing and possible storage of customer data. The system should force all users to use strong passwords to keep their data secure. |  | 4 |
| 5. Customer accessibility | The system should be easy for the customer to see what data is stored about them as well as easily request for the employees to book them services, order parts and purchase items without much hassle. |  | 1, 4.1 |
| 7. Expandability | The system should maintain the ability to be expanded to include extra dealerships if required. Despite the system only having 5 dealerships at the moment the system should hold onto the possibility of creating new dealerships in the future of the company. This expandability will need to adhere to the external entity’s needs and have space to expand in any way the company or shareholders require. |  | 2 |

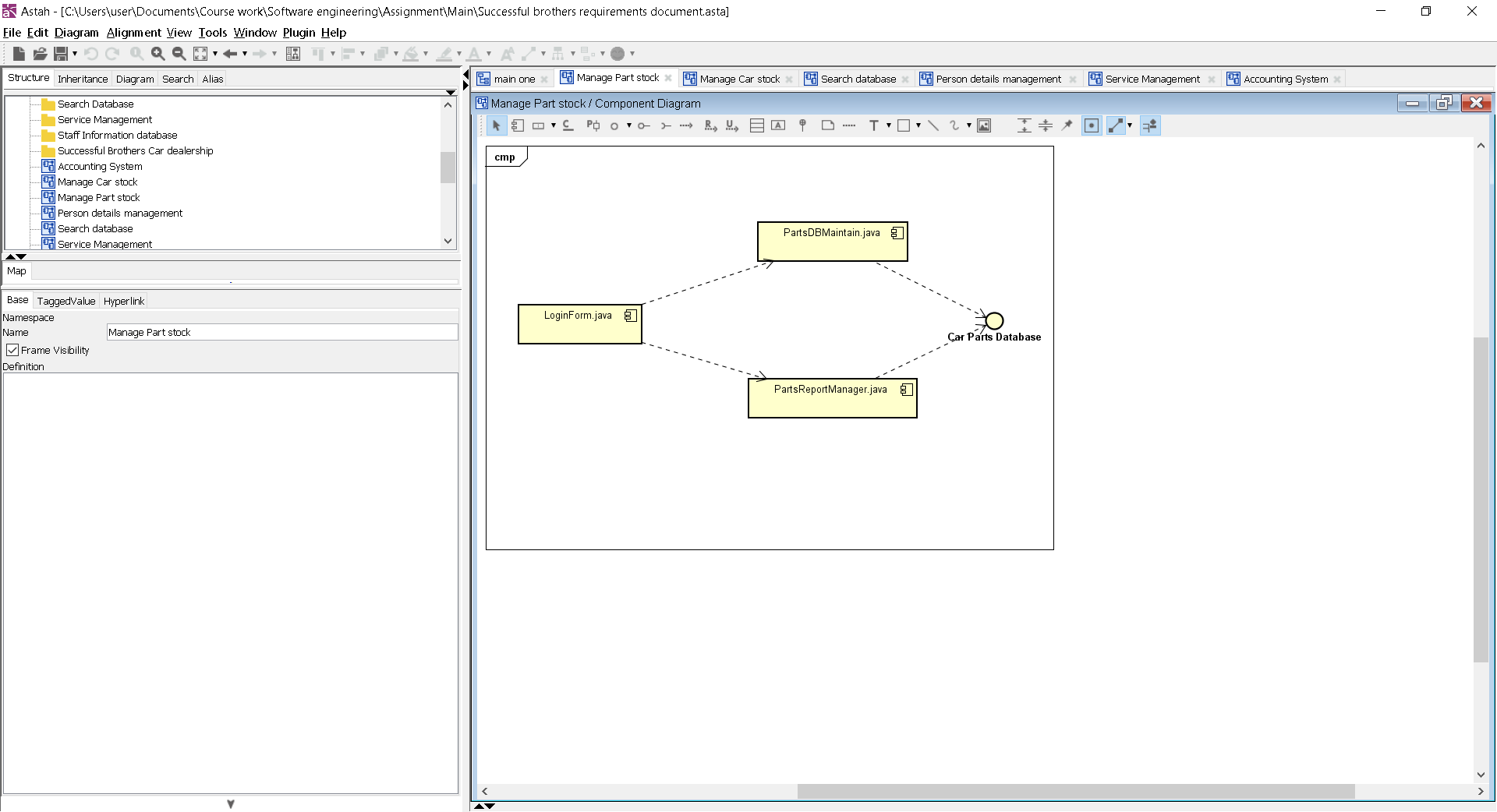
# **Solution Architecture:**

Implementation view, 4+1 model. This view of architecture concerns itself with the actual software module organization within the development environment. The implementation view of architecture takes into account derived requirements related to ease of development, software management, reuse, and constraints imposed by programming languages and development tools. A package in this view of architecture represents a physical partitioning of the system. The packages are organized in a hierarchy of layers where each layer has a well-defined interface. The typical layers of packages are user interface, application specific packages, reusable business packages, key mechanisms, and hardware and operating system packages. Within our own project we used the implementation view of the 4+1 model extensively.

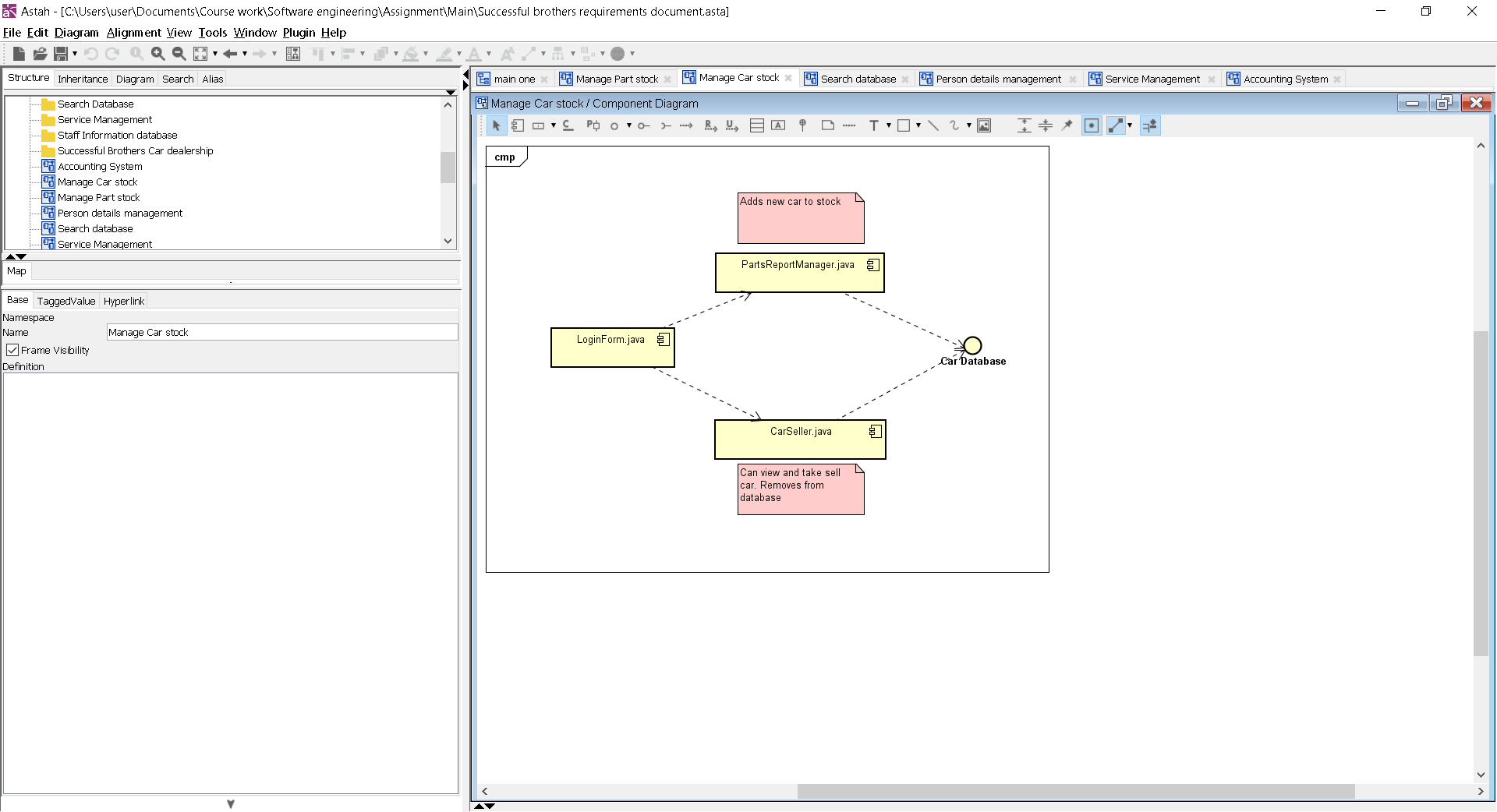
The diagrams below feature the implementation view that we identified and agreed upon as a team



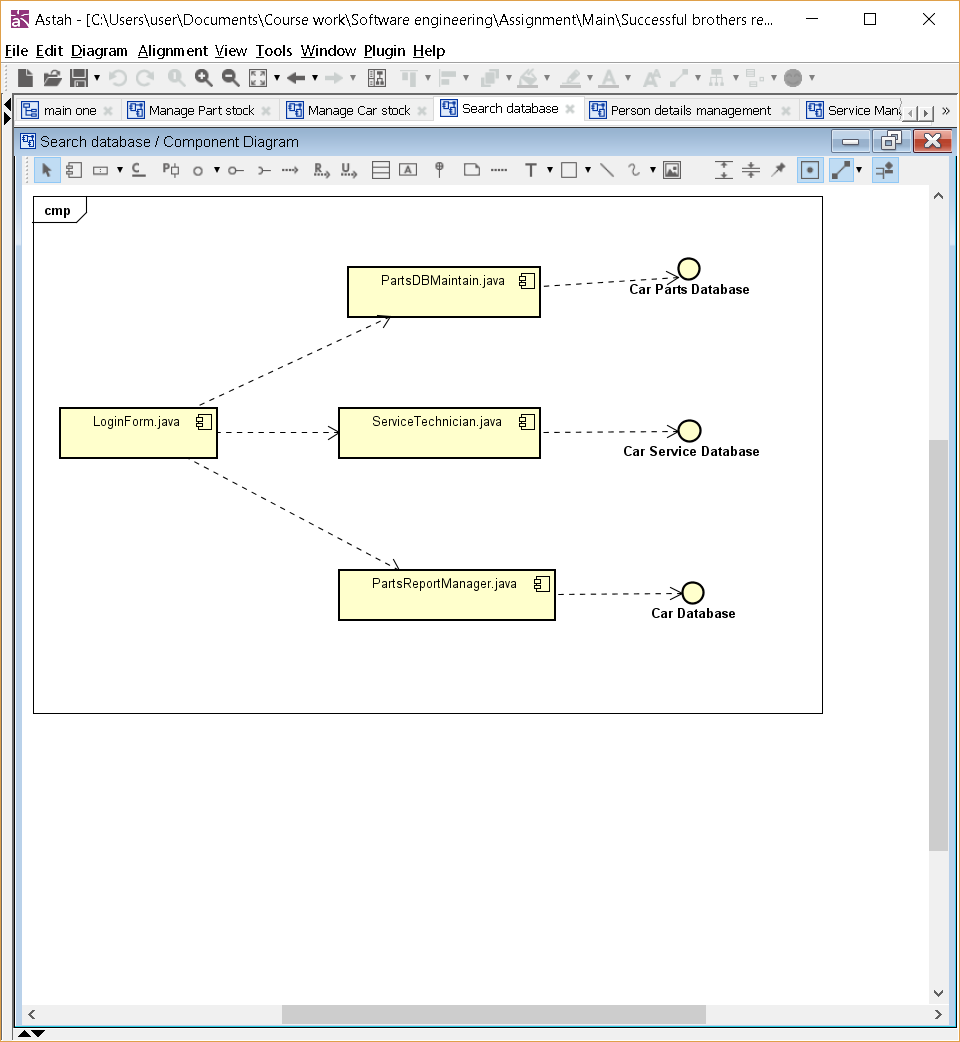
## **Manage part Stock:**



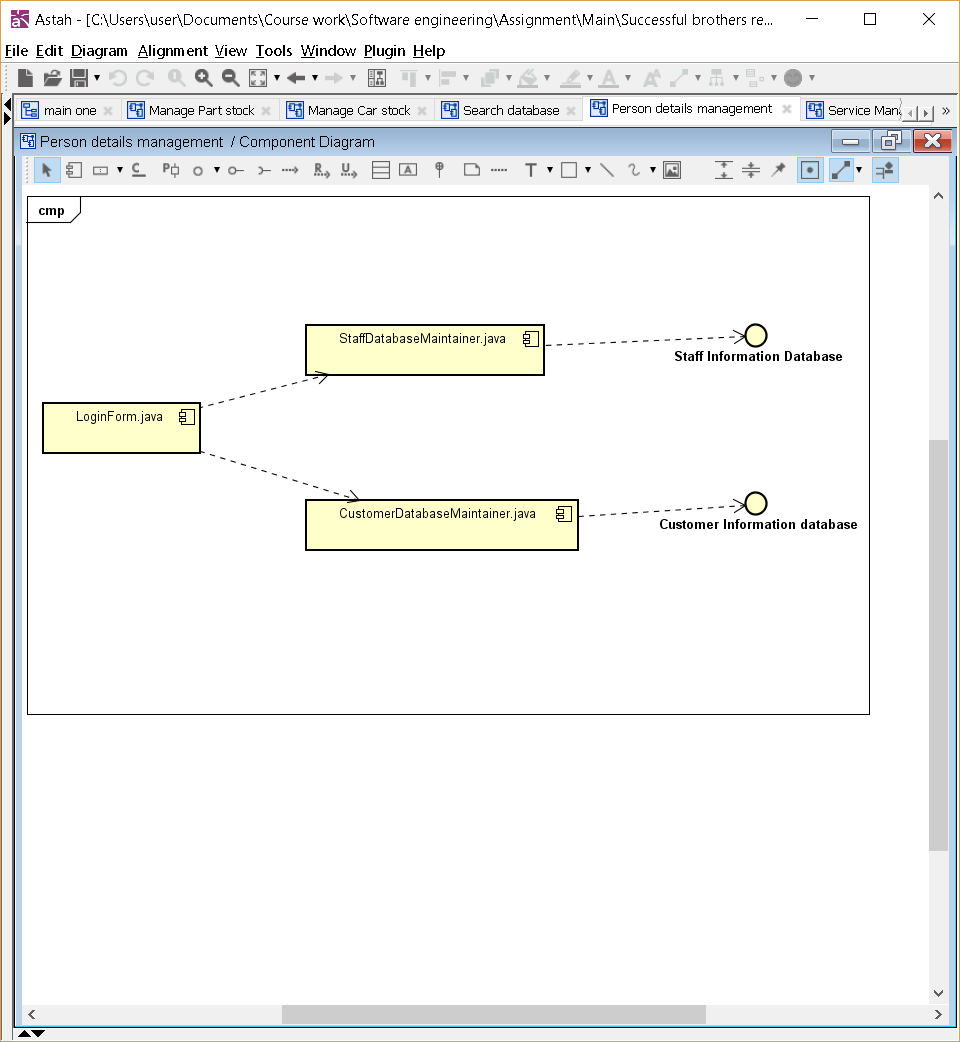
## **Manage Car Stock:**



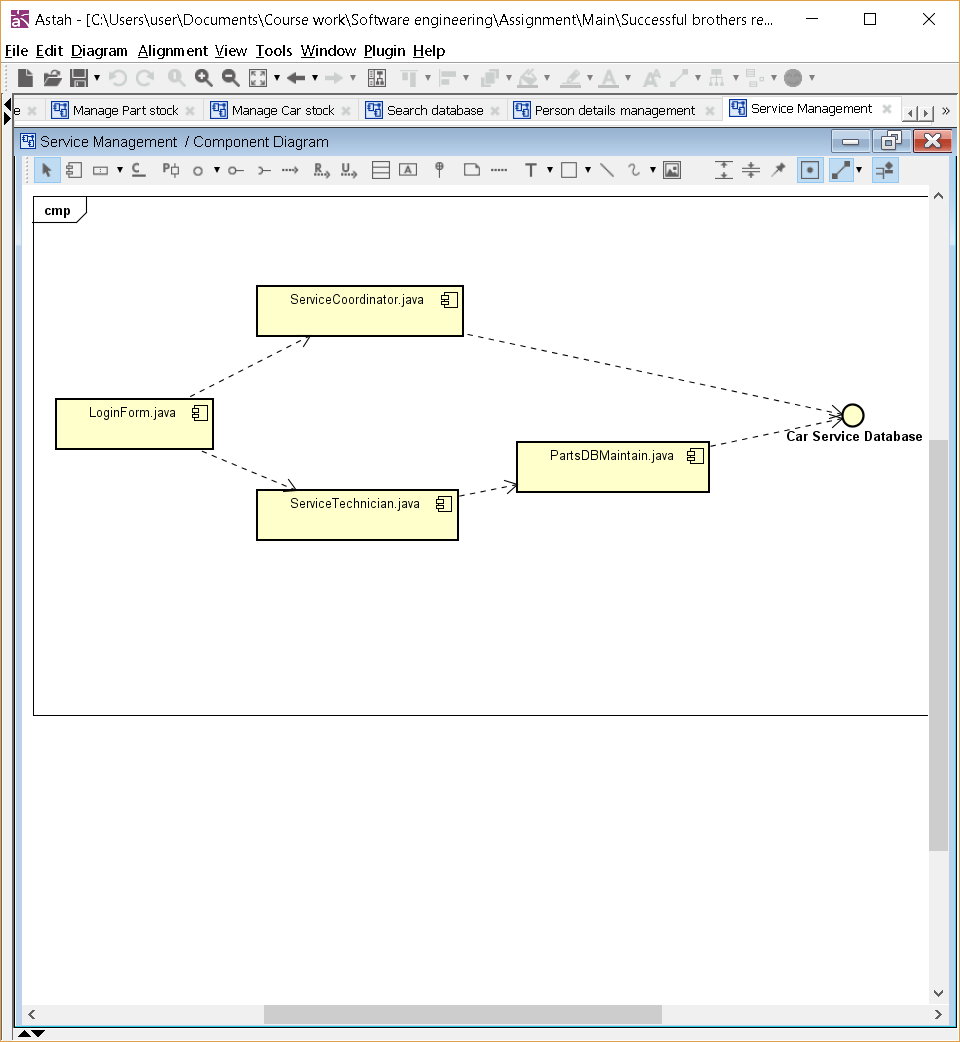
## **Search database:**



## **Person details management:**



## **Service management:**



# **Software Prototype:**

The final prototype is submitted alongside this document in the zip file.

# **Software Test Specification:**

## **Part Ordering Testing:**

To test that this part of the software is functioning as intended we will need to test this part of the system using a test strategies. The way in which will intend to test this part of the system is through using Junit testing as this is an inbuilt function within java which allows us to work out if a piece of code is working as intended. The software that we would need for this is NetBeans as this is the same software that was used to create this part of the system and it also allows us to implement Junit testing. On top of this we would need a decent PC to run the tests on because on a lower specification computer these tests could take a very long time which would waste a lot of time.

The functional requirements that would be tested within this test would be the following:

* Search stock
* A GUI to access the software through
* Online ordering
* Part order
  + Manual
  + Automated
* DMS database
  + Parts Database

The non-functional requirements that would be tested within this test would be the following:

* Ease of use
* Reliability
* Error handling

The way in which we will record the data coming from the tests would be to produce a series of tables showing the test and how the test performed. This would be shown through having ‘expected result’, ‘actual result’ and screenshots to show a successful or failed test.

## **Part Sales Testing:**

To test that this part of the software is functioning as intended we will need to test this part of the system using a test strategies. The way in which will intend to test this part of the system is through using Junit testing as this is an inbuilt function within java which allows us to work out if a piece of code is working as intended. The software that we would need for this is NetBeans as this is the same software that was used to create this part of the system and it also allows us to implement Junit testing. On top of this we would need a decent PC to run the tests on because on a lower specification computer these tests could take a very long time which would waste a lot of time.

The functional requirements that would be tested within this test would be the following:

* Search stock
* A GUI to access the software through
* Produce invoices
* Process cash payments
* Auto transactions sent to account
* Part sales report
* DMS database
  + Parts Database
  + Customer database
* Accounting system

The non-functional requirements that would be tested within this test would be the following:

* Ease of use
* Reliability
* Error handling
* Customer accessibility
* Security

The way in which we will record the data coming from the tests would be to produce a series of tables showing the test and how the test performed. This would be shown through having ‘expected result’, ‘actual result’ and screenshots to show a successful or failed test.

## **Car Sales Testing:**

To test that this part of the software is functioning as intended we will need to test this part of the system using a test strategies. The way in which will intend to test this part of the system is through using Junit testing as this is an inbuilt function within java which allows us to work out if a piece of code is working as intended. The software that we would need for this is NetBeans as this is the same software that was used to create this part of the system and it also allows us to implement Junit testing. On top of this we would need a decent PC to run the tests on because on a lower specification computer these tests could take a very long time which would waste a lot of time.

The functional requirements that would be tested within this test would be the following:

* Search stock
* A GUI to access the software through
* Auto transactions sent to account
* Process cash payments
* Produce invoices
* Relocate cars
* Car sales reports
* DMS database
  + Cars for sale database
  + Customer database

The non-functional requirements that would be tested within this test would be the following:

* Ease of use
* Reliability
* Error handling
* Security
* Customer accessibility

The way in which we will record the data coming from the tests would be to produce a series of tables showing the test and how the test performed. This would be shown through having ‘expected result’, ‘actual result’ and screenshots to show a successful or failed test.

## **Car Service Testing:**

To test that this part of the software is functioning as intended we will need to test this part of the system using a test strategies. The way in which will intend to test this part of the system is through using Junit testing as this is an inbuilt function within java which allows us to work out if a piece of code is working as intended. The software that we would need for this is NetBeans as this is the same software that was used to create this part of the system and it also allows us to implement Junit testing. On top of this we would need a decent PC to run the tests on because on a lower specification computer these tests could take a very long time which would waste a lot of time.

The functional requirements that would be tested within this test would be the following:

* A GUI to access the software through
* Logging services
* Produce invoice
* Process cash payments
* Auto transactions sent to account
* Car service report
* DMS database
  + Customer Database
  + Service database
* Accounting system

The non-functional requirements that would be tested within this test would be the following:

* Ease of use
* Reliability
* Error handling
* Customer accessibility
* Security

The way in which we will record the data coming from the tests would be to produce a series of tables showing the test and how the test performed. This would be shown through having ‘expected result’, ‘actual result’ and screenshots to show a successful or failed test.

## **Car Sales Report Testing:**

To test that this part of the software is functioning as intended we will need to test this part of the system using a test strategies. The way in which will intend to test this part of the system is through using Junit testing as this is an inbuilt function within java which allows us to work out if a piece of code is working as intended. The software that we would need for this is NetBeans as this is the same software that was used to create this part of the system and it also allows us to implement Junit testing. On top of this we would need a decent PC to run the tests on because on a lower specification computer these tests could take a very long time which would waste a lot of time.

The functional requirements that would be tested within this test would be the following:

* A GUI to access the software through
* Produce reports:
  + Car sales reports
  + Staff sales reports
* DMS database
  + Cars for sale database

The non-functional requirements that would be tested within this test would be the following:

* Ease of use
* Reliability
* Error handling

The way in which we will record the data coming from the tests would be to produce a series of tables showing the test and how the test performed. This would be shown through having ‘expected result’, ‘actual result’ and screenshots to show a successful or failed test.

## **Example tests:**

**Log in**

|  |  |  |
| --- | --- | --- |
| Log in with wrong details | Should notify user that wrong details was provided | Notify user that wrong details was provided |
|  | | Check in database if the provided user exist and password is correct |

|  |  |  |
| --- | --- | --- |
| Log in with correct details | Should redirect user to the appropriate window | Redirect user to the appropriate window |
|  | | Check in database if the provided user exist and password is correct |

**Workshop Manager Window**

|  |  |  |
| --- | --- | --- |
| List all finished services | Should list all finished services | Lists all finished services |
|  | | Check in database for all finished services |

|  |  |  |
| --- | --- | --- |
| Search finished service and review it | User should be able to search for finished services  and review them | User can search for finished service and can review it |
|  | | Search in database for finished service that user is looking for and pull data |

**Service Technician Window**

|  |  |  |
| --- | --- | --- |
| Select car to work on, review details and mark as fixed | User should be able to select a car, view all the details of it and mark it as fixed | User can select a car, view details and mark it as fixed |
|  | | Search in database for no finished service, list them, on user select a car display all details for car and Fixed button to mark car as fixed |

**Staff Maintainer Window**

|  |  |  |
| --- | --- | --- |
| Search for user, check user details, edit/delete user and add a new user | User should be able to search for users, check user details, edit/delete user and add a new user | User can search for user, check user details, edit/delete user and add a new user |
|  | | Search in database for user, if found display info about it, Text fields are provided for add, edit details of user and delete it. |

**Service Coordinator Window**

|  |  |  |
| --- | --- | --- |
| Search for part, check part details, edit/delete part and add a new part | User should be able to search for parts, check part details, edit/delete part and add a new part | User can search for part, check part details, edit/delete part and add a new part |
|  | | Search in database for parts, if found display info about it, Text fields are provided for add, edit details of part and delete it. |

**Sales Report Manager Window**

|  |  |  |
| --- | --- | --- |
| List all sold cars, search for sold car, check car details, add a new car | User should be able to view all sold cars, search for sold car, check car details, add a new car | User can view all sold cars, search for sold car, check car details, add a new car |
|  | | Search in database for sold cars, list them, user can search for car and view all details, can add new car |

**Parts Report Manager Window**

|  |  |  |
| --- | --- | --- |
| List all sold parts, search for sold part, check part details | User should be able to view all sold parts, search for car part, check part details | User can view all sold parts, search for car part, check part details |
|  | | Search in database for sold parts, list them, user can search for parts and view all details |