

OOAD Project Documentation

*Super Shine Carwash*

**DAT17xx**

**Project Participants:**

*Rasmus H*

*Rasmus W*

*Nikolaj*

*Tycho*

*Martin N*

Document Version: 00.01.00

Contents

[History 3](#_Toc499798920)

[Vision 4](#_Toc499798921)

[“Optional” - Additional informative Artifacts before Requirements? 5](#_Toc499798922)

[Requirements 6](#_Toc499798923)

[Use Cases 6](#_Toc499798924)

[Supplementary Specification 7](#_Toc499798925)

[FURPS+ 7](#_Toc499798926)

[Functional Requirements 8](#_Toc499798927)

[Non-functional requirements 8](#_Toc499798928)

[“Optional” – Other Information? 9](#_Toc499798929)

[Appendix 10](#_Toc499798930)

[Glossary 10](#_Toc499798931)

# Vision

{ TO BE REMOVED…

**The GOAL!**It defines the stakeholders view of the product/service to be developed, specified in terms of the stakeholders key needs and features. Containing an outline of the envisioned core requirements.

**Summary of System Features**

Definition  
Features are behavioral functions a system can do. They should pass this linguistic test:  
The system does <feature X>.

For example:  
The system does payment authorization.

The major features include:  
• POS services  
• Inventory management  
• Web-based shopping  
• ...

**Other Requirements in the Vision?**In the Vision, system features briefly summarize functional requirements often detailed in the use cases. Likewise, the Vision can summarize other requirements (for example, reliability and usability) that are detailed in the Supplementary Specification. But be careful to avoid going down the path of repeating yourself.

**See Larman for more information:**  
Chapter 7.6. NextGen Example: (Partial) Vision  
Chapter 7.7. Commentary: Vision

}

# Requirements

**Requirements for user:**

1. User must have a WashCard
2. User must be able to see current amount on WashCard
3. User must be able to recharge money to WashCard, with Creditcard
4. User must be able to get receipt from WashCard
5. User must be able to choose between 3 different Wash types.
   1. Economy
   2. Standard
   3. Deluxe
   4. Supershine 🡪 20% discount on weekdays, before 14:00, except Deluxe.

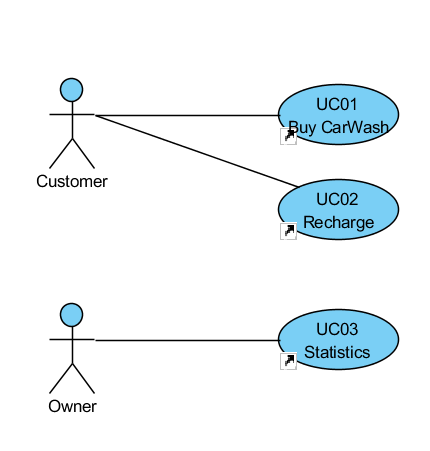
**Requirements for owner:**

1. Owner must be able to receive statistics from user purchases. Including the different wash types and frequencies.

## Use Cases

We have defined 3 use cases,

1. How a customer buys a carwash (Use Case: Buy CarWash)
2. How a customer recharges he’s wash card (Use Case: Recharge)
3. How the owner interacts with the statistic system (Use Case: Statistics)



## Buy Carwash (Brief)

|  |
| --- |
| 1. SYSTEM Insert WashCard |
| 2. Customer Inserts WashCard |
| 3. SYSTEM Shows available wash types |
| 4. Customer Selects wash type |
| 5. SYSTEM Initiates wash, and ejects WashCard |
| 6. SYSTEM Shows info |

## Buy Carwash (Fully dressed)

|  |
| --- |
| 1. SYSTEM Insert WashCard |
| 2. Customer Inserts WashCard |
| 3. SYSTEM Verifies WashCard |
| 3.a. |
| 1. SYSTEM CarWash card verification fails |
| 2. SYSTEM Ejects CarWash card |
| 3. jump to 1. SYSTEM Insert WashCard |
| 4. SYSTEM Creates session timestamp |
| 5. if EarlyBird special |
| 5.1. SYSTEM Show EarlyBird menu |
| 5.1.a. |
| 1. if Customer select Abort Purchase |
| 1.1. Eject Carwash Card |
| 1.2. Runs Selfdiagnostic |
| 1.3. Resets |
| 1.4. jump to 1. SYSTEM Insert WashCard |
| end if |
| 5.1.1. EarlyBird Economy |
| 5.1.2. EarlyBird Standard |
| 5.1.3. Deluxe |
| 6. else |
| 6.1. SYSTEM Shows normal menu |
| 6.1.a. |
| 1. if Customer select Abort Purchase |
| 1.1. Eject Carwash Card |
| 1.2. Runs Selfdiagnostic |
| 1.3. Resets |
| 1.4. jump to 1. SYSTEM Insert WashCard |
| end if |
| 6.1.1. Economy |
| 6.1.2. Standard |
| 6.1.3. Deluxe |
| end if |
| 7. Customer Selects CarWash |
| 8. SYSTEM Checks for sufficient balance |
| 8.a. |
| 1. if CarWash price is bigger than balance |
| 1.1. SYSTEM Goes to Recharge |
| end if |
| 9. SYSTEM Withdraws money from WashCard |
| 10. SYSTEM Store purchase/data |
| 11. SYSTEM Initiates wash |
| 12. SYSTEM Ejects WashCard |
| 13. SYSTEM Prompts Customer for receipt |
| 14. if Customer wants receipt |
| 14.1. SYSTEM Prints receipt |
| end if |
| 15. SYSTEM Shows Instructionscreen |
| 16. SYSTEM Shows Waitscreen |
| 17. SYSTEM Runs Selfdiagnostic |
| 17.a. |
| 1. if Selfdiagnostic fails |
| 1.1. SYSTEM Show Errorscreen |
| end if |
| 18. SYSTEM Resets |
| 18.a. |
| 1. SYSTEM Clears session |
| 2. jump to 1. SYSTEM Insert WashCard |

## Recharge (Brief)

|  |
| --- |
| 1. SYSTEM Insert WashCard |
| 2. Customer Inserts WashCard |
| 3. SYSTEM Shows menu |
| 4. Customer Choose recharge |
| 5. SYSTEM Choose amount |
| 6. Customer Inserts creditcard |
| 7. SYSTEM Replies with confirmation |
| 8. SYSTEM Sends customer back to wash menu |

## Recharge (Casual)

|  |
| --- |
| 1. SYSTEM Insert WashCard |
| 2. Customer Inserts WashCard |
| 2.1. SYSTEM Verifies WashCard |
| 3. SYSTEM Shows menu |
| 4. Customer Choose recharge |
| 5. SYSTEM Choose amount |
| 5.a. |
| 1. if Customer abouts recharge |
| 1.1. jump to 3. SYSTEM Shows menu |
| end if |
| 5.1. Option 1: 200 DKK |
| 5.2. Option 2: 500 DKK |
| 5.3. Option 3: 1000 DKK |
| 6. Customer Chooses option |
| 7. SYSTEM Insert creditcard |
| 8. Customer Inserts creditcard |
| 9. SYSTEM Verifies creditcard |
| 10. SYSTEM Transfers money to WashCard |
| 11. SYSTEM Ejects creditcard |
| 12. SYSTEM Sends Customer back to Wash menu |

## Statistics (Brief)

|  |
| --- |
| 1. Owner Logs into SuperShine webpage |
| 2. SYSTEM Choose statistics |
| 3. Owner Is shown statistics |

## Statistics (Casual)

|  |
| --- |
| 1. Owner Logs into SuperShine webpage |
| 2. SYSTEM Shows statistics menu |
| 2.1. Total revenue |
| 2.2. Amount of washtypes (5 types) |
| 2.3. Time of purchases |
| 3. Owner Chooses type of statistics |
| 4. SYSTEM Shows statistic |

## Supplementary Specification

{ TO BE REMOVED…

**Use Either FURPS+ or Functional/Non-Functional**basically, everything not in the use cases. This artifact is primarily for all non-functional requirements, such as performance or licensing. It is also the place to record functional features not expressed (or expressible) as use cases; for example, a report generation.

**See Larman for more information:**  
Chapter 5.4. What are the Types and Categories of Requirements?  
Chapter 5.5. How are Requirements Organized in UP Artifacts?

}

### FURPS+

Functional

Security is setup by using hardware (firewalls/secure tunnels), so it has nothing to do with our program.

Usability

The program must be easy to use, with an easy interface, that is usable by humans that have issues seeing etc. and can use a touch screen.

Reliability

Very important, in case of any errors, the user must be notified about this.

Performance

There are no performance demands, except that the system is working within the timeframe you would normally expect when you use your creditcard (in case of recharge of the washcard). And that the user can choose the option to wash their car, with out having to wait on the system.

Supportability

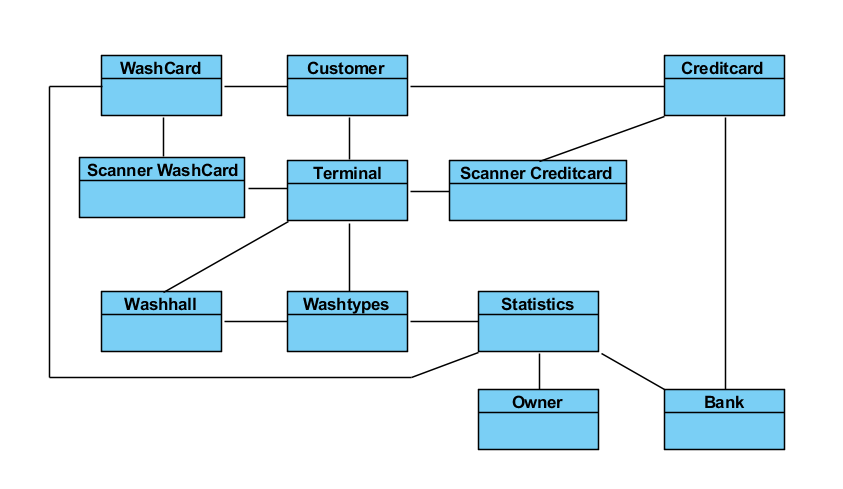
There is currently only one washhall, but the system must be prepared for adding more washhalls.

+  
Implementation

We use specific hardware terminals and the washhall also have a interface we can interact with, it must be possible from the program to change the terminals or washhall with out having to completely create a new program.

# 

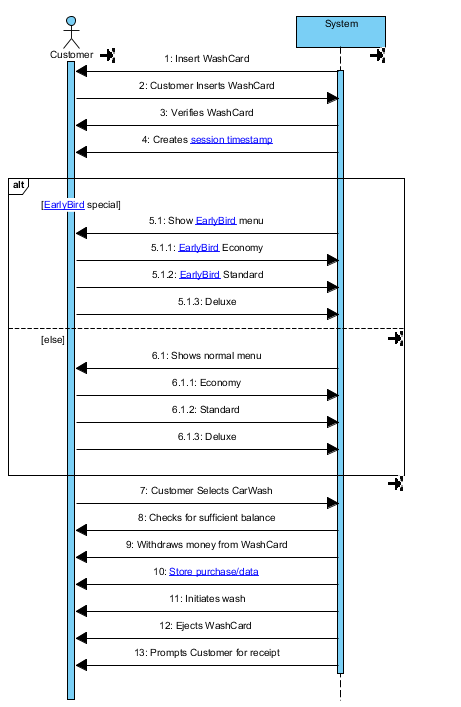
# Domain model



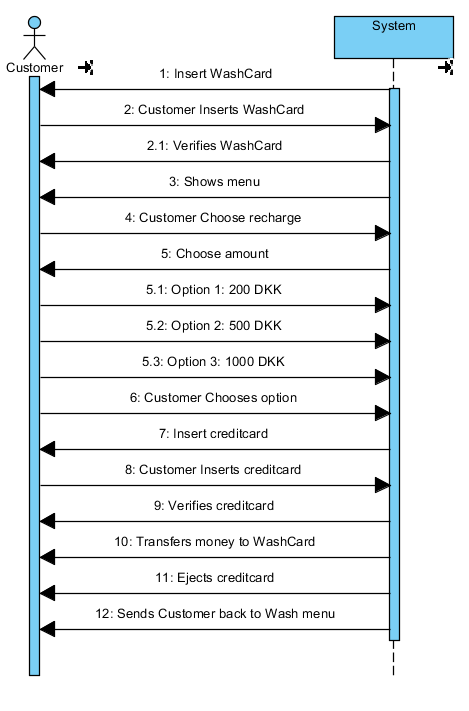
The terminal in our domain model, is the terminal the user is using to create a purchase or recharge of the card. To get access to the terminal they use their WashCard. We are creating a program for our terminal that gives them this option.

# System sequence diagram

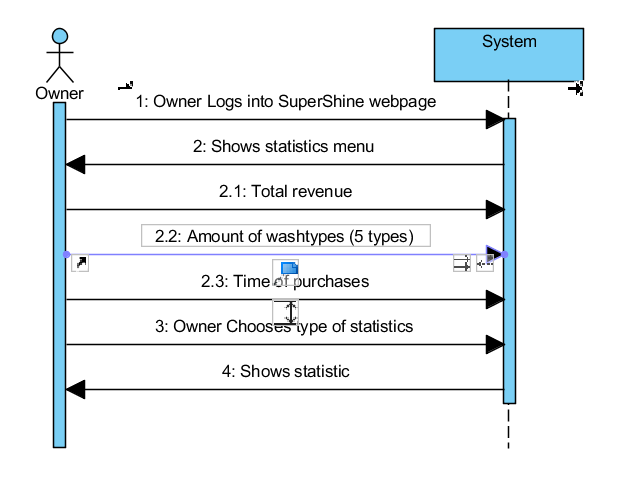
## Buy Carwash (Fully dressed main scenario)



## Recharge (Casual)



## Statistic (Casual)



# Operation contracts

## Buy CarWash: Customer Inserts WashCard

**cross reference**: buy carwash

**preconditions**:

system holds no current account data and shows the “Insert card screen”

**postconditions:**

currentAccount in CarWashManager is set to a new Account instance

currentAccount.id becomes the card id

currentAccount.credit became the amount of credit associated with the id in the accounts database

depending on the time and date the display is changed to either early bird menu or normal menu

## Buy CarWash: Customer Selects CarWash

**cross reference:** buy carwash

**preconditions:**

customers account data is loaded

correct carwash menu (early bird or normal) is shown

**postconditions:**

credit is withdrawn from account, washcard is ejected and receipt menu is shown

# Class Diagrams

We divided our program into multiple classes.

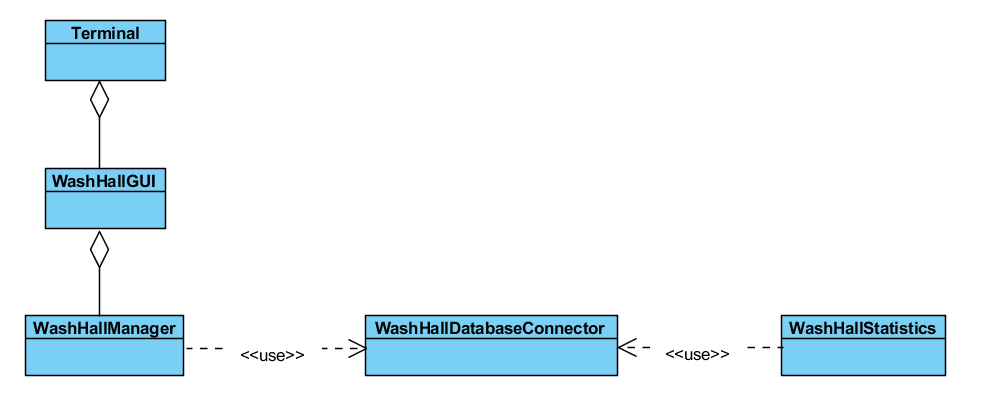
WashHallGUI: Is the GUI (graphical User Interface) the user sees, and with out any logic. That interacts with WashHallManager.

WashHallManager: Is the business logic, that interacts with the WashHallDatabaseInterface. And has no dependencies to any specific GUI.

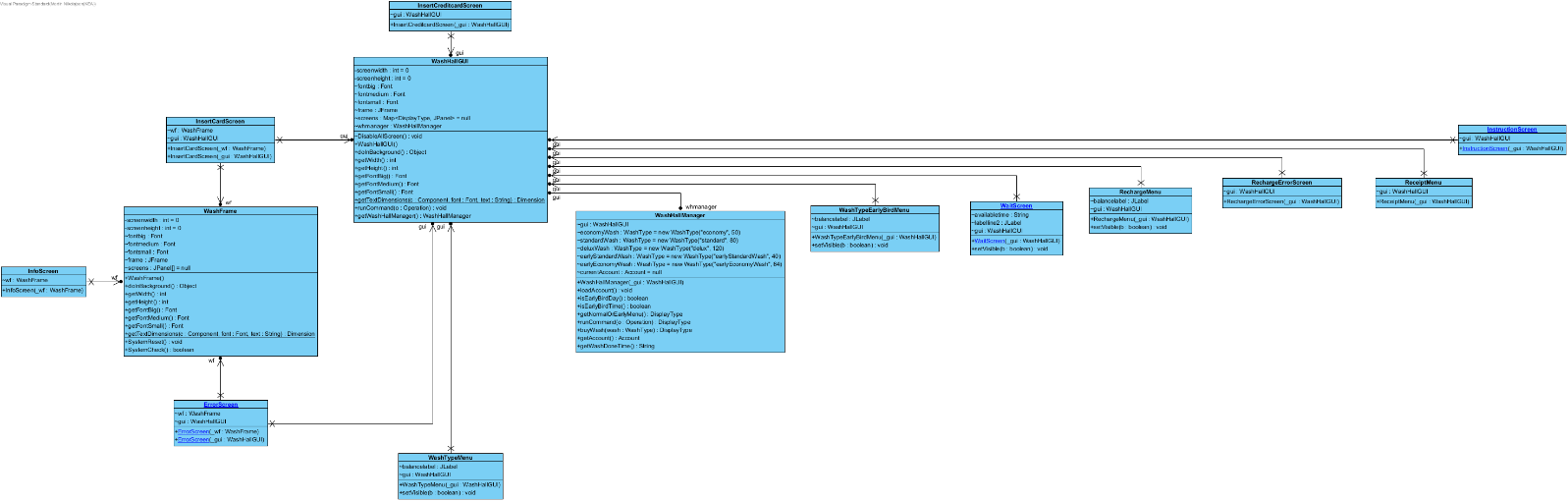
WashHallDatabaseInterface: Is the interface to the (SQLite) database where we store user information, purchases.

WashHallStatistics: Is the interface to the owner, so he can see purchases from customers. Interacts with WashHallDatabaseInterface.

The different parts of the program interacts with each other.

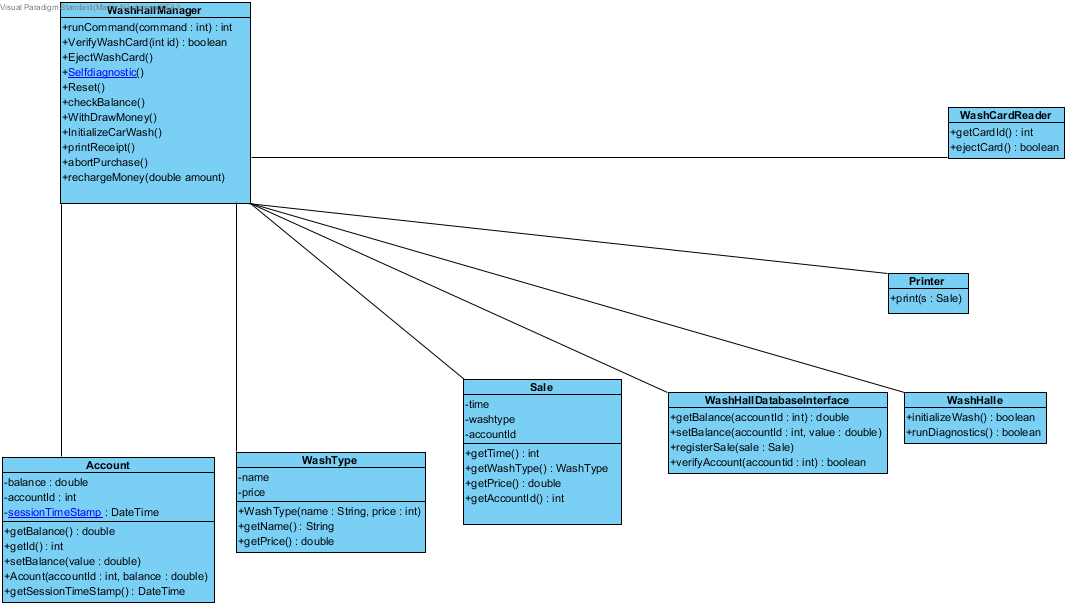


## WashHallGUI



# 

## WashHallManager



## WashHallDatabaseInterface

Kun I forsøgs stadie.

## WashHallStatistics

Kun I forsøgs stadie.

//TODO

# “Optional” – Other Information?

{ TO BE REMOVED…

Other informative text/diagram to include?

}

# Appendix

{ TO BE REMOVED…

“Optional” - Additional informative Artifacts after Requirements?

* Domain Model
* System Sequence Diagram
* Sequence Diagram
* Activity Diagram
* Class Diagram
* …

}

Glossary

{ TO BE REMOVED…

**Use either 1 Glossary with 2 section or 2 Glossaries:**  
One for Domain/Business  
One for Technical/Data dictionary

**See Larman for more information:**Chapter 7.8. NextGen Example: A (Partial) Glossary for more information  
Chapter 7.9. Commentary: Glossary (Data Dictionary)

**}**