

**Super Shine Carwash**

**DAT17xx**

**Project Participants:**

Rasmus H

Rasmus W

Nikolaj

Tycho

Martin N

# 

**Indholdsfortegnelse**

[**Vision** 3](#_Toc528341571)

[**Mission** 3](#_Toc528341572)

[**Business Model Canvas** 3](#_Toc528341573)

[**Requirements** 4](#_Toc528341574)

[Requirements for user: 4](#_Toc528341575)

[Requirements for owner: 4](#_Toc528341576)

[**Use Cases** 5](#_Toc528341577)

[Buy Carwash (Brief) 5](#_Toc528341578)

[Recharge (Brief) 5](#_Toc528341579)

[Statistics (Brief) 6](#_Toc528341580)

[Recharge (Casual) 6](#_Toc528341581)

[Statistics (Casual) 6](#_Toc528341582)

[Buy Carwash (Fully dressed) 7](#_Toc528341583)

[**FURPS** 8](#_Toc528341584)

[Functional 8](#_Toc528341585)

[Usability 8](#_Toc528341586)

[Reliability 8](#_Toc528341587)

[Performance 9](#_Toc528341588)

[Supportability 9](#_Toc528341589)

[**Domain model** 9](#_Toc528341590)

[**System sequence diagram** 10](#_Toc528341591)

[Buy Carwash (Fully dressed main scenario) 10](#_Toc528341592)

[Recharge (Casual) 11](#_Toc528341593)

[Statistic (Casual) 12](#_Toc528341594)

[**Sequence Diagram** 12](#_Toc528341595)

[**Operation contracts** 13](#_Toc528341596)

[Buy CarWash: Customer Inserts WashCard 13](#_Toc528341597)

[Buy CarWash: Customer Selects CarWash 13](#_Toc528341598)

[**Class Diagrams** 14](#_Toc528341599)

[**Glossary** 17](#_Toc528341600)

[**How the program works** 18](#_Toc528341601)

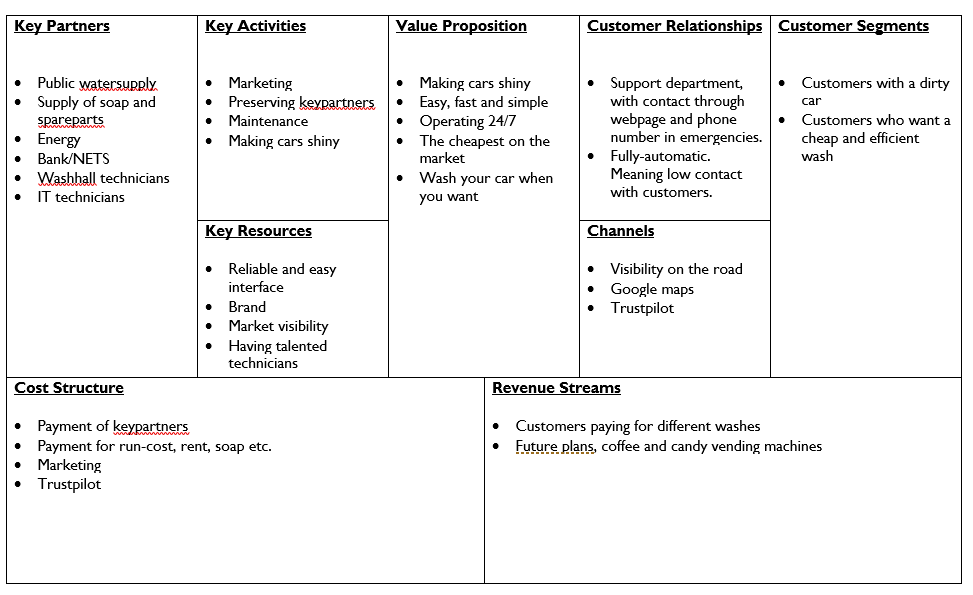
# **Vision**

Starting with one car washhall in central Copenhagen, Supershine’s vision is to expand to more cities in Denmark and to be the leading and preferred car washhall in Denmark.

# **Mission**

Our mission is to provide our customers with a fast and easy-going carwash service. At SuperShine we developed a fully automatic system, which gives our customers the option, to wash their car, at any time of the day, anytime of the week, without the need for human assistance, and technical inconveniences.

# **Business Model Canvas**



# 

# **Requirements**

## **Requirements for user:**

1. User must have a WashCard, bought from SuperShine
2. User must be able to see current balance on WashCard, after inserting WashCard into the WashCard reader.
3. User must be able to recharge money to WashCard, with a creditcard, directly from the Terminal.
4. User must be able to print receipt after a purchase.
5. User must be able to choose between 5 different Wash types, depending on the time and day of the week.
   1. Economy
   2. Standard
   3. Deluxe
   4. On weekdays before 14.00 pm, user will get a 20% discount, on Economy and Standard wash:
      1. EarlyBird Economy
      2. EarlyBird Standard

## **Requirements for owner:**

1. Owner must be able to receive statistics from user purchases. Including the different wash types and frequencies. Owner will log into a webpage, to see the statistics.

# **Use Cases**

In this section, we have made a total of 3 different use-cases as shown on figure 1. The first 3 are brief, and shows a success flow of, a customer buying a carwash, a customer recharging a WashCard, and how the owner, is shown statistics of customers purchases.

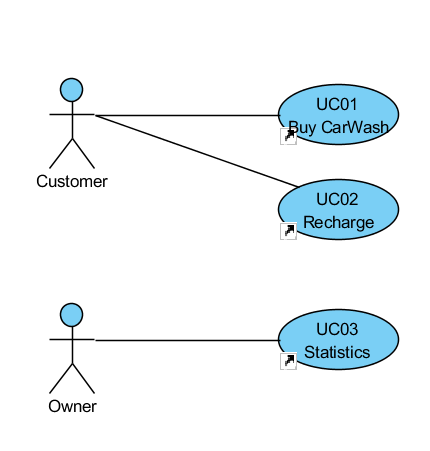


Figure 1

## **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 |
| 1. SYSTEM Shows info  **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 | | 1. SYSTEM Sends customer back to wash menu  **Statistics (Brief)**  |  | | --- | | 1. Owner Logs into SuperShine webpage | | 2. SYSTEM Choose statistics | | 3. Owner Is shown statistics |  **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 (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 | | |

## **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 |

# **FURPS**

FURPS is a technique to validate the prioritised requirements after an understanding with client’s needs and necessities. In the requirements section, we received the owner’s ideas, on how the system should work. The functionality requirements and ideas, is then made into use-cases, as shown above. In the section below, we have put a few more words, into the different arrays of 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 who have issues seeing, 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, without having to wait on the system.

## **Supportability**

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

# 

# **Domain model**

Our domain model shows the conceptual ideas of how we wanted our program to interact with different classes. 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.

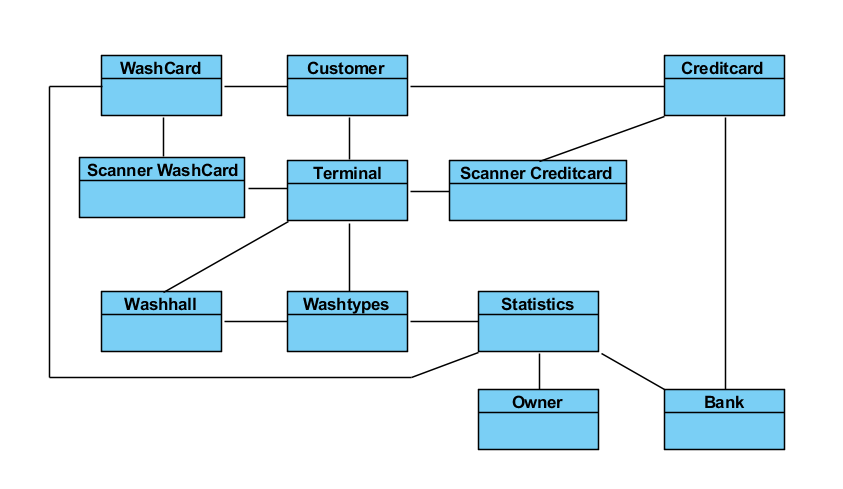


Figure 2

# **System sequence diagram**

For this part, we have created 3 different system-sequence-diagrams. 1 fully dressed, which shows how the customer interacts with the system, when buying a carwash. 2 casual use-cases, where the first shows the flow, when a customer wants to recharge an amount to a WashCard. And the last one illustrates how the owner, logs into a webpage, and is thereafter shown statistics of customers purchases.

## **Buy Carwash (Fully dressed main scenario)**

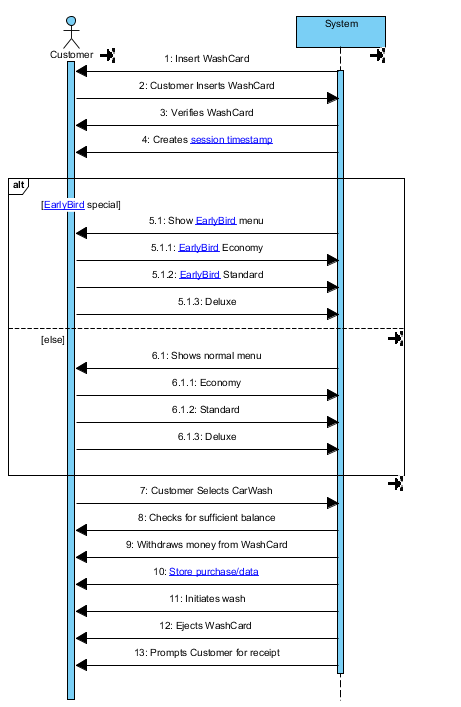


Figure 3

## **Recharge (Casual)**

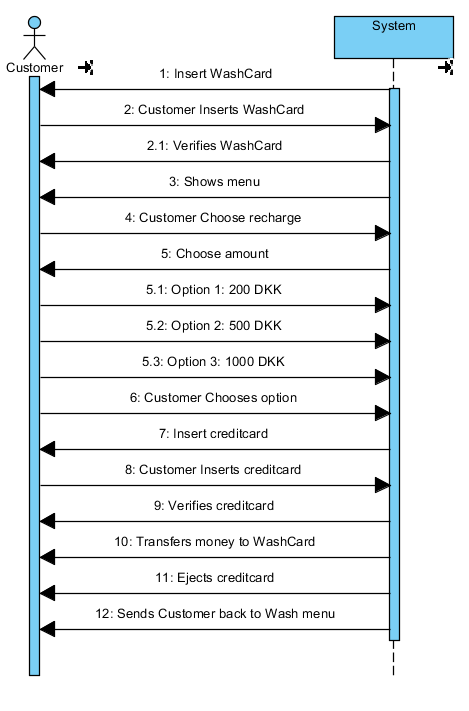


Figure 4

## **Statistic (Casual)**

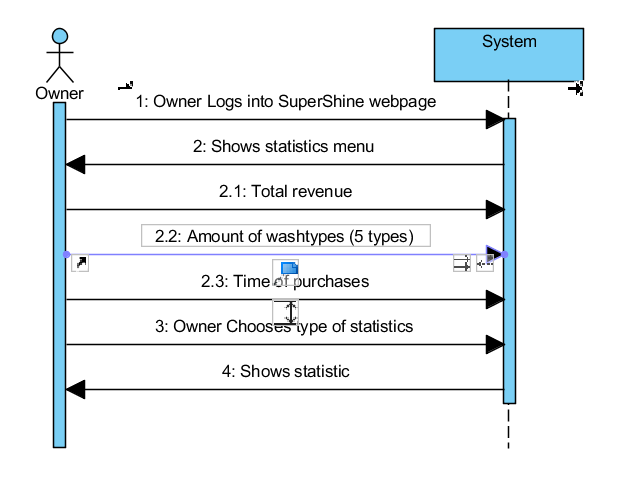


Figure 5

# **Sequence Diagram**

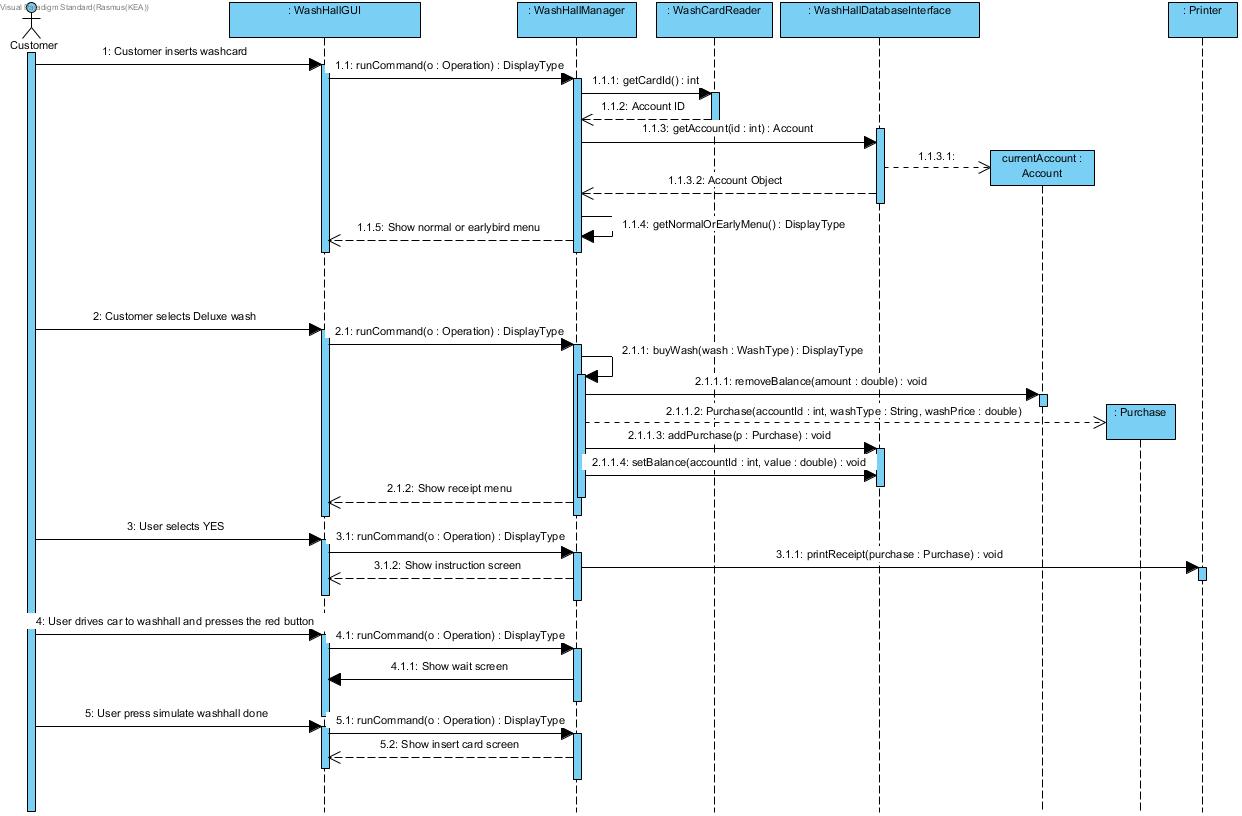


Figure 6

# **Operation contracts**

Operation Contracts are defined in terms of system operations. They describe how the internal state of the concepts in the domain model may change, even though we didn’t include contracts in our diagram. Furthermore, Operation Contracts are described in terms of preconditions and postconditions. Below is shown 2 examples of Operations Contracts in our system.

## **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**

Our program became a lot bigger, than first anticipated. Therefore, we made a decision to split up our class diagram into smaller parts, and into different iterations. The final Class Diagram, can be found in the folder “Aflevering”, named classdiagram2.jpg. The whole diagram is so comprehensive, that it would be unreadable, if presented as a picture in this document.

The first diagram as shown on figure 7, is an overview on how the WashHallGUI which interacts with WashHallManager. The GUI (Graphical User Interface) is what the user sees and without any logic.

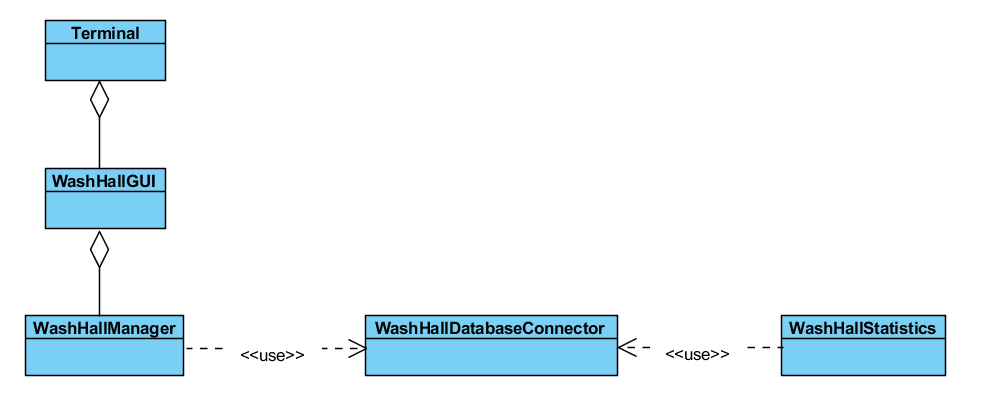


Figure 7

On figure 8, is shown the first iteration of how our WashHallManager interacts with the WashHallDatabaseInterface. It includes the business logic but has no dependencies to the WashHallGUI. In the folder “Aflevering”, a picture file called WashHallManager2.jpg, shows the 2nd iteration.

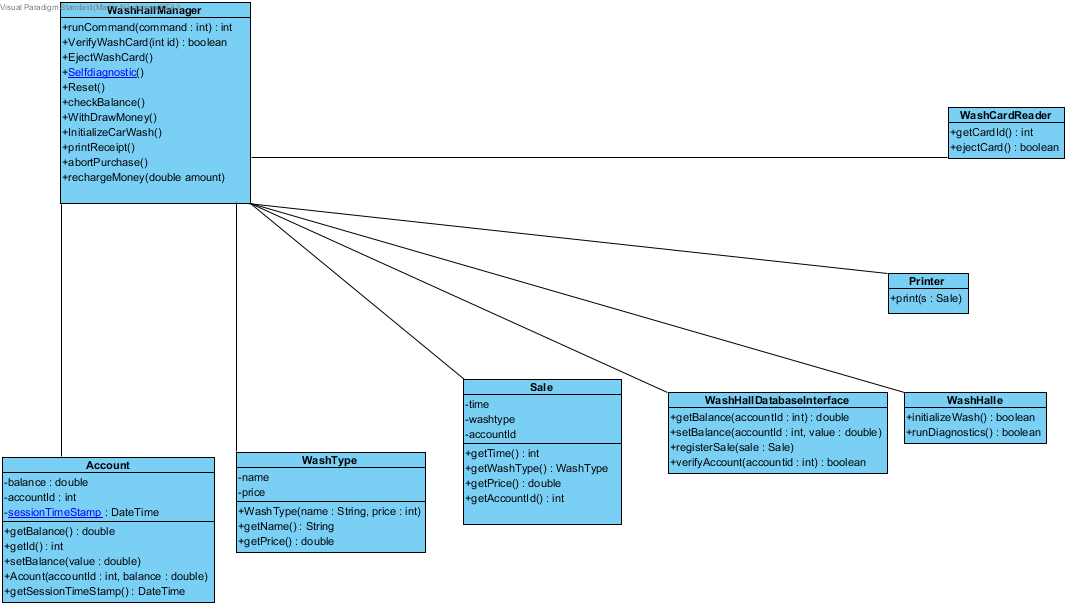


Figure 8

Figure 9 shows the WashHallDatabaseInterface, which is the interface to the SQLite database where we store user information, purchases etc.

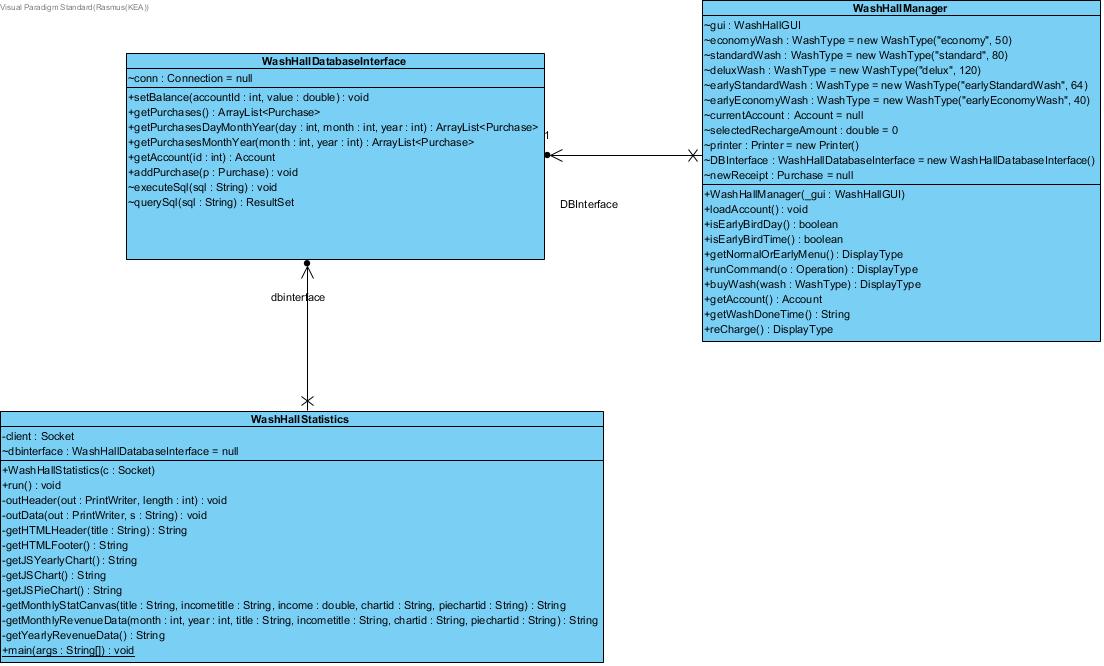


Figure 9

Figure 10 below shows the WashHallStatistics, which interacts with WashHallDatabaseInterface. WashHallStatistics is specific to the owner, and not the customer. The owner can see purchase statistics from customers.

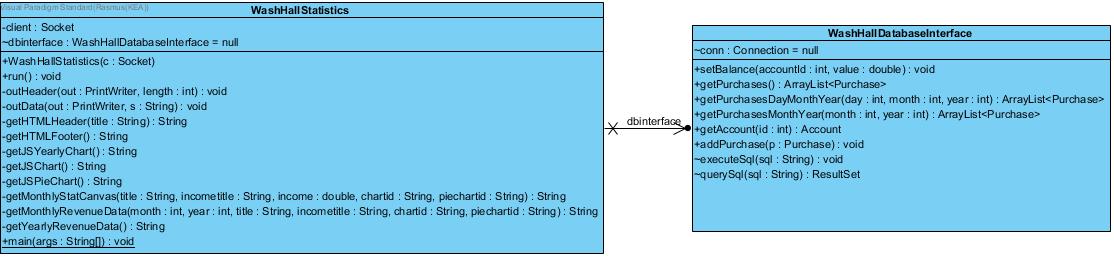


Figure 10

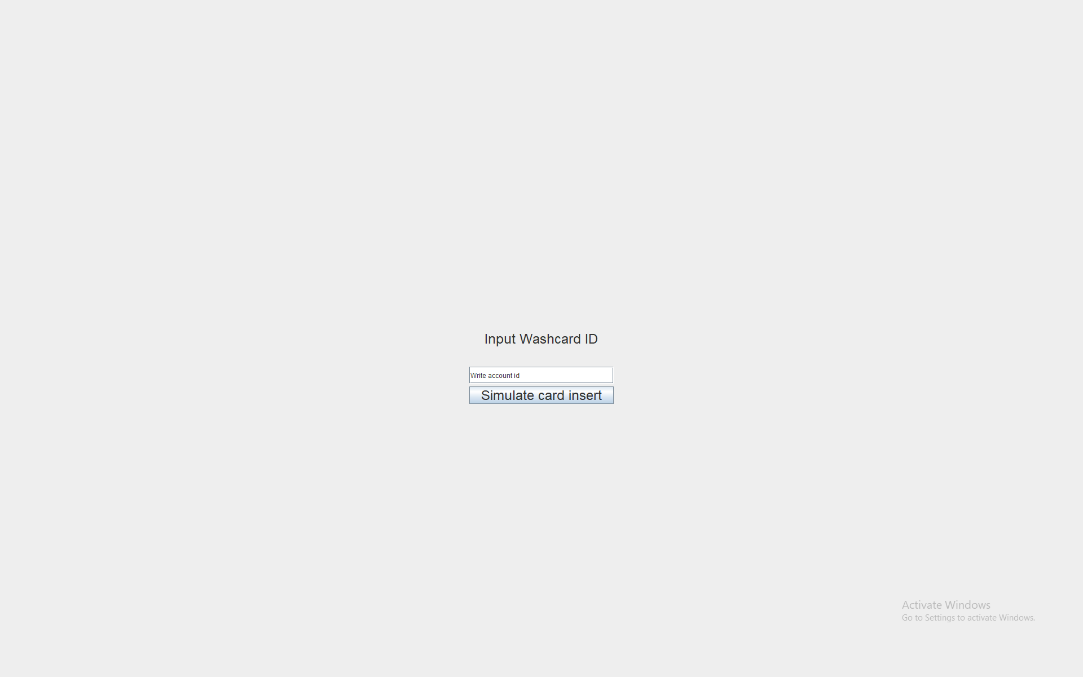
# **Glossary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Session timestamp** |  |  |  | | When user inserts WashCard, the Systems logs the start session timestamp, which is used to calculate Earlybird deals. |
| |  |  |  |  | | --- | --- | --- | --- | | **Errorscreen** |  |  |  | | If the system has mechanical errors, the system shows Errorscreen |
| |  |  |  |  | | --- | --- | --- | --- | | **Selfdiagnostic** |  |  |  | | System checks for mechanical errors, such as soap, water etc. |
| |  |  |  |  | | --- | --- | --- | --- | | **Store purchase/data** |  |  |  | | Logs data of user purchases, such as what time of purchase, card ID, type of wash , so owner has access to statistics. |
| |  |  |  | | --- | --- | --- | | **Waitscreen** |  |  | | When user initiates wash, the Waitscreen, indicates that the system is busy. Also shows time remaining of current wash. |
| |  |  |  | | --- | --- | --- | | **EarlyBird** |  |  | | Special price, which is available on weekdays, from 00:01 am to 02:00 pm. |
| |  |  |  |  | | --- | --- | --- | --- | | **Instructionscreen** |  |  |  | | Shows the customer how to initiate the car wash |

# **How the program works**

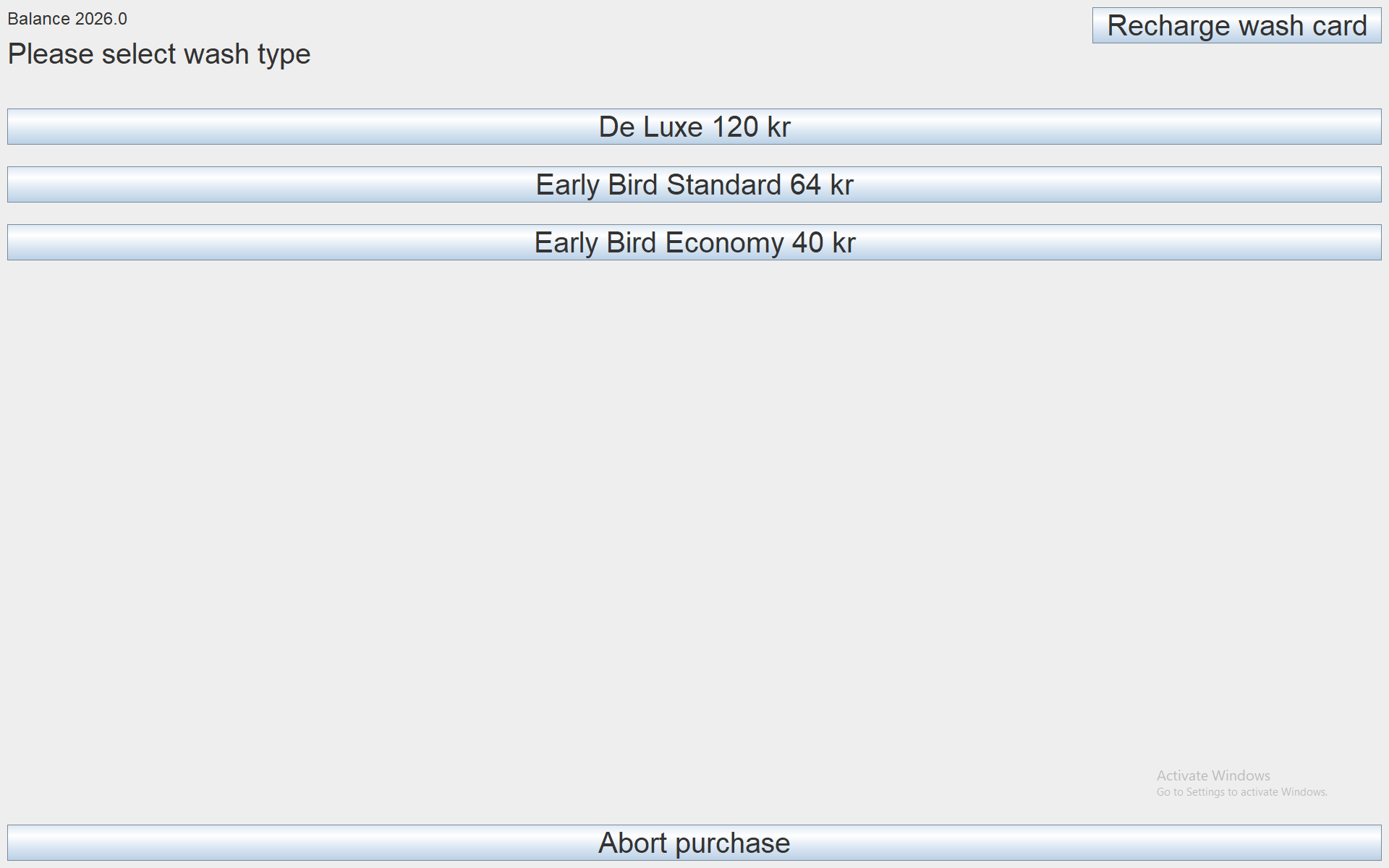
**1.**

To run the program, open the folder “Program”, and execute the file named CarwashTerminal.jar. The program will now launch, and the user will be presented with the first screen. In the “Write account Id” field, the user can insert 3 premade Id’s, either 1, 2 or 3. After entering Id, press the “Simulate card insert” button.

****

**2.**

After entering the Id, the user is presented with a menu, where its possible to buy 3 different car washes. Depending on the time and day, the program includes 2 different menus. If it is a weekday, and the time is before 13:59, an Earlybird menu is shown, where the user will get a 20% discount, on Economy and Standard wash. Furthermore, the WashCard Id’s balance, is shown in the top left corner, and the possibility to recharge the WashCard is shown to the top right.



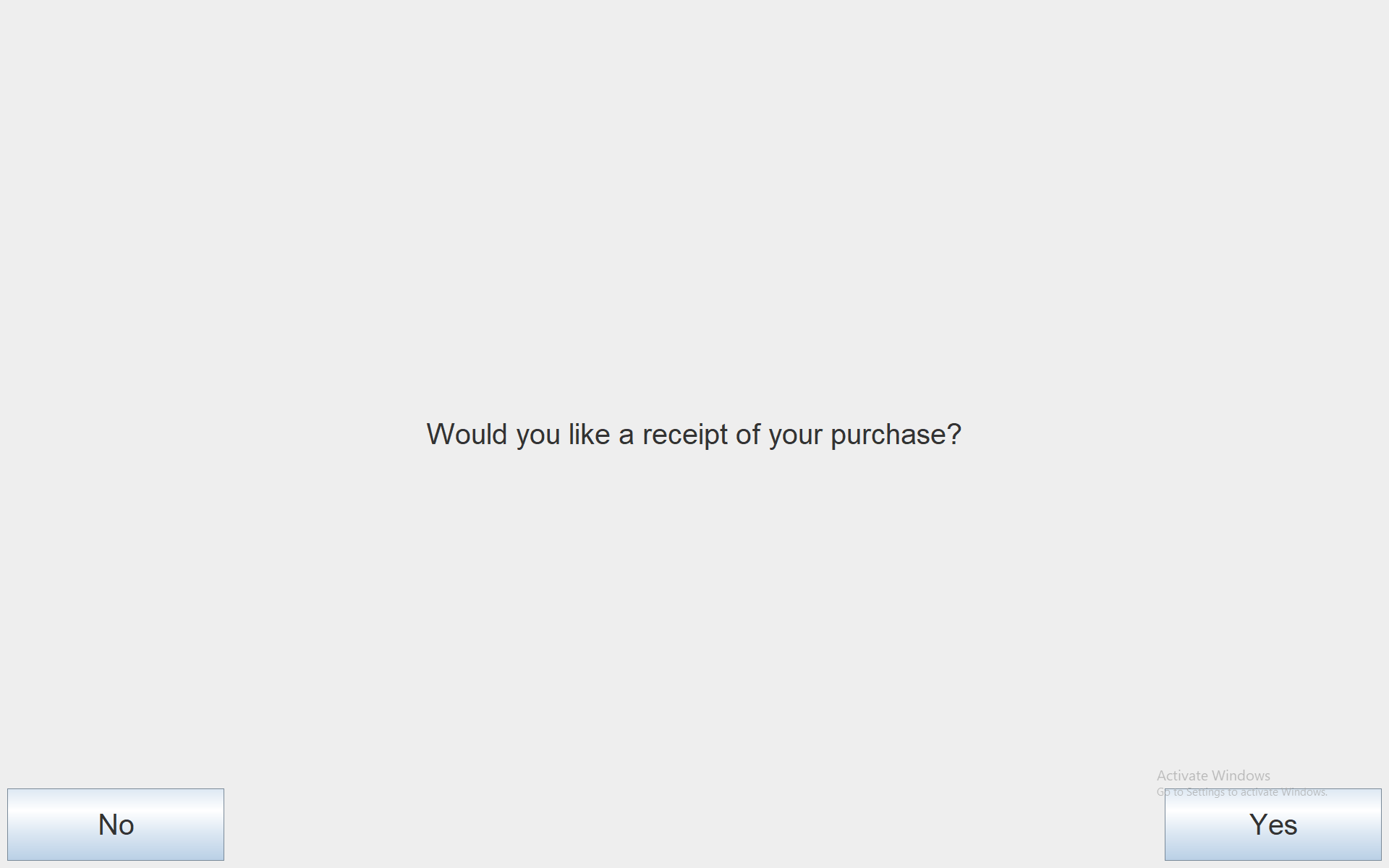
**3.**

If the user is launching the program, at all other times, the following menu is presented.



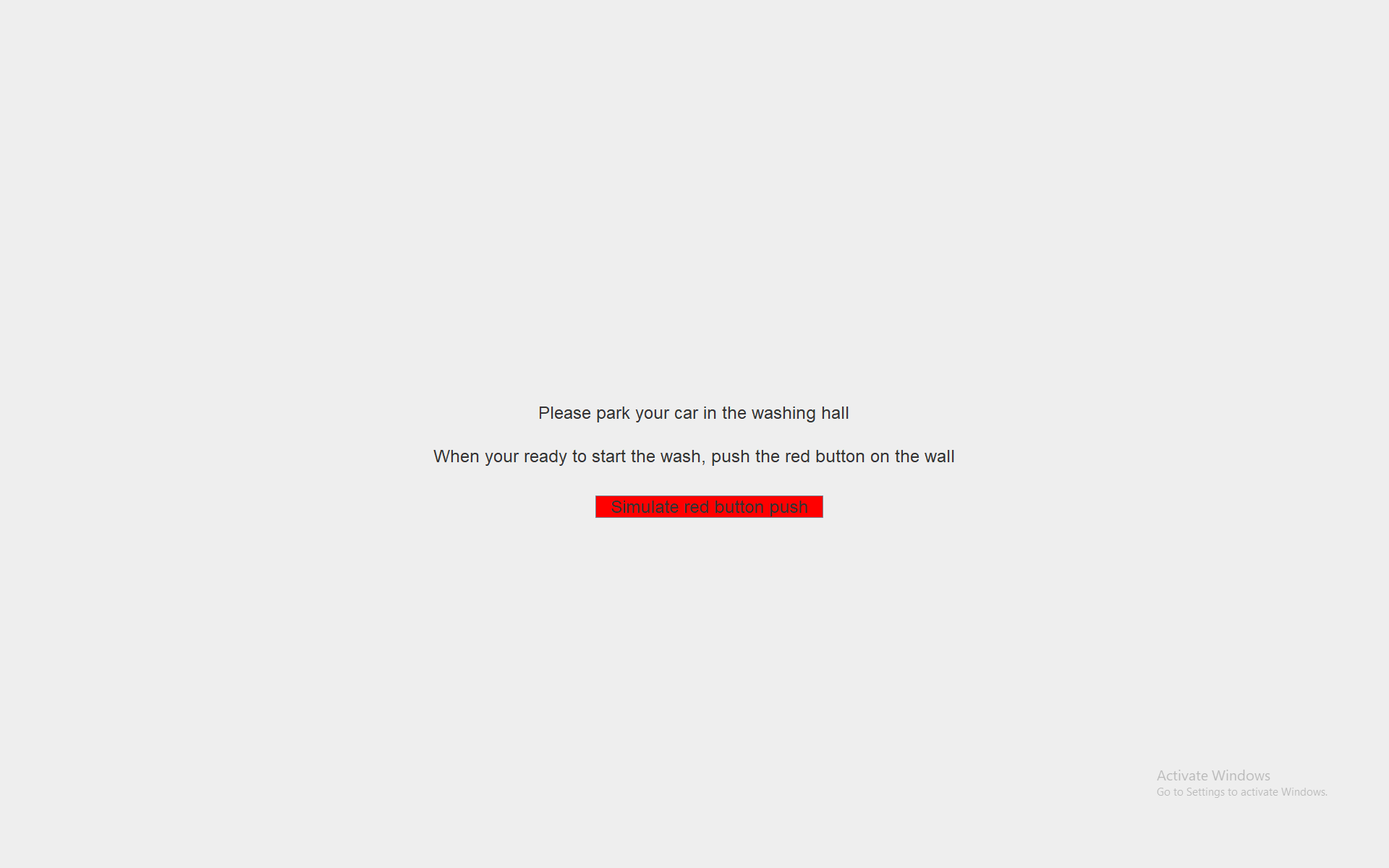
**4.**

If the user chooses to buy a wash, the following screen, will prompt and ask if the user wants a receipt, with an option of “Yes” and “No”. Choosing “Yes”, will print the receipt to the console, with time and day, WashCard Id and WashCard balance.



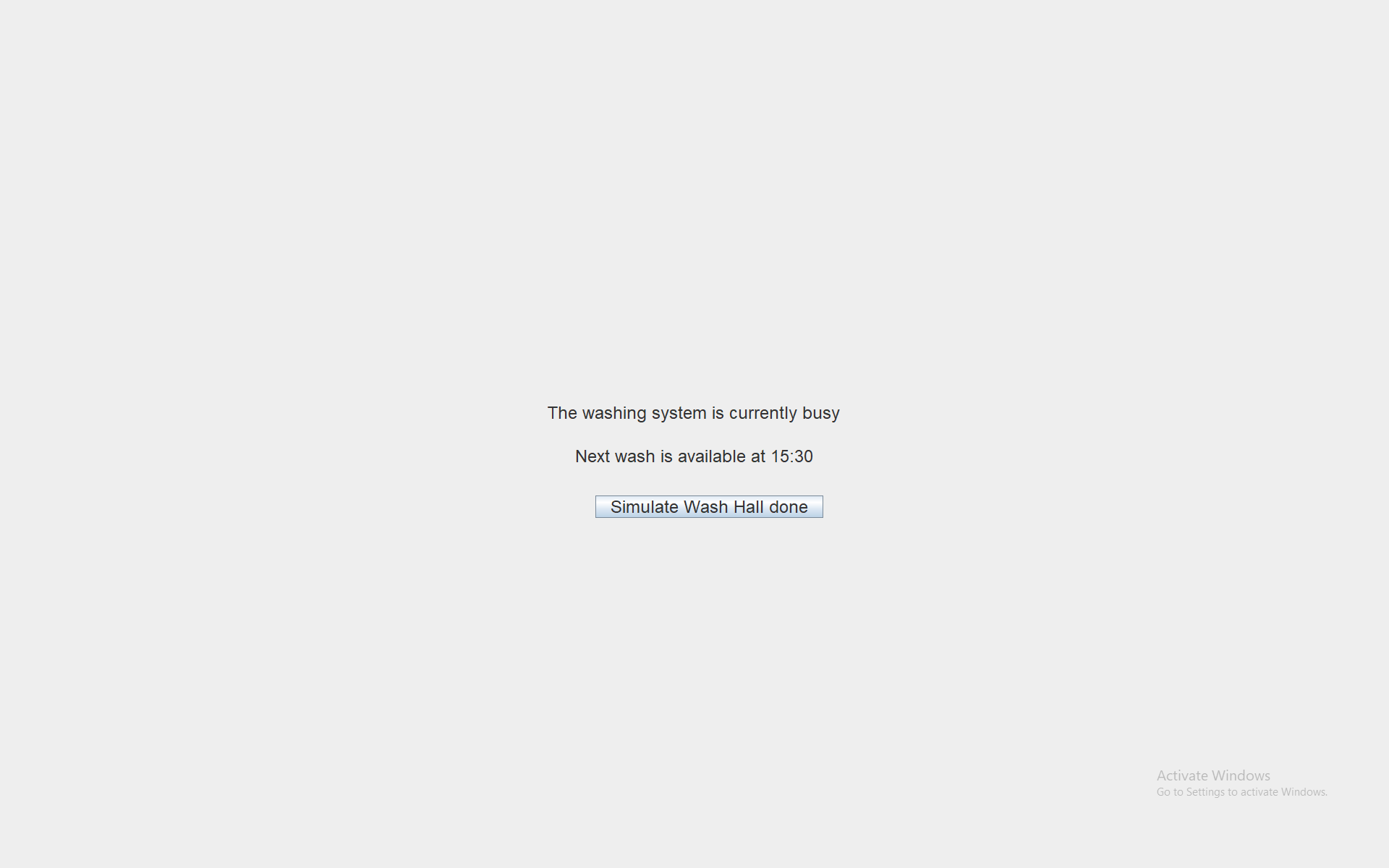
**4.**

The user is now presented with the instructions screen, which simulates, that the user drives their car into the washhall. When the car is parked at the correct spot, the user would now press the “Simulate red button push” button.



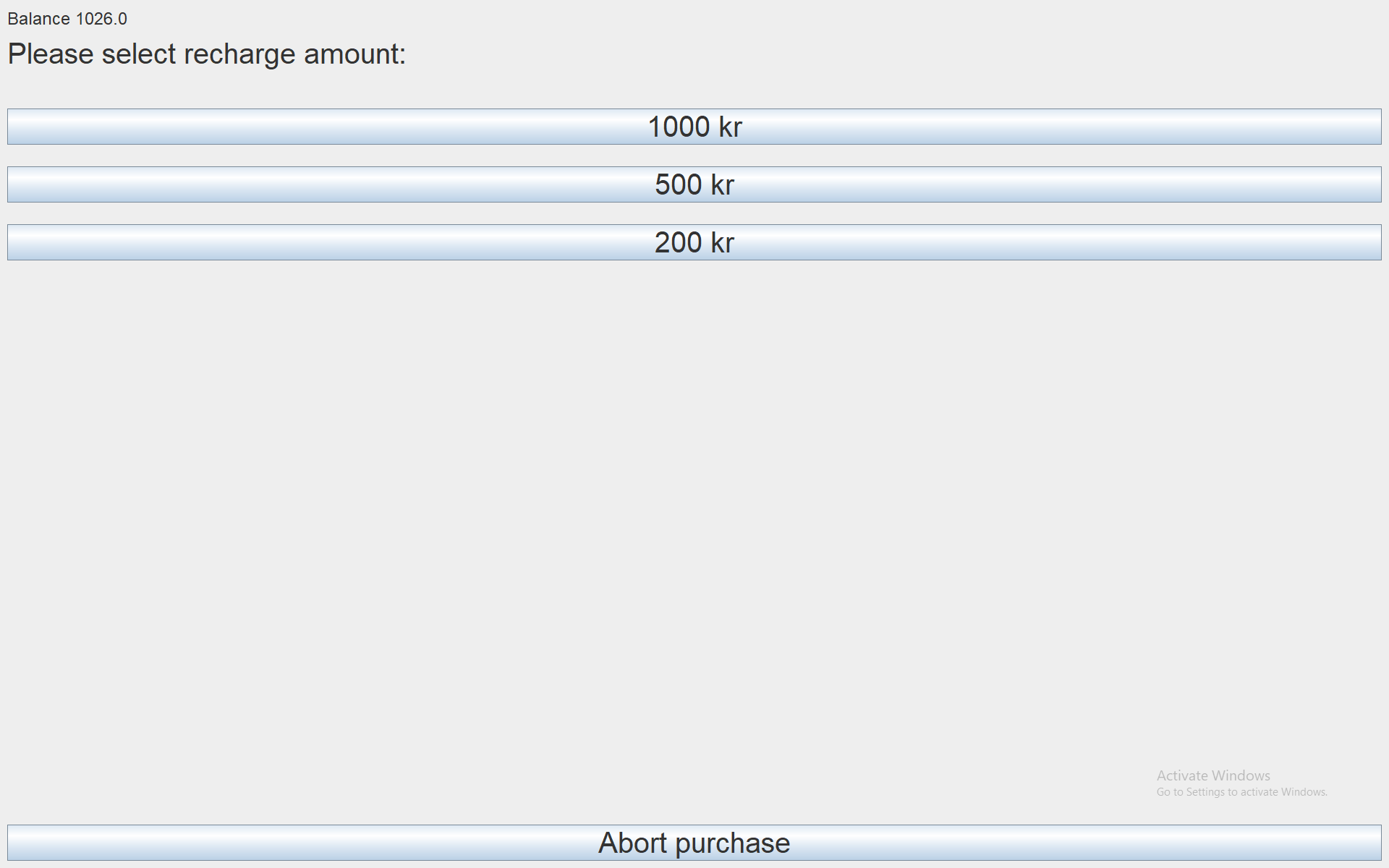
**5.**

The wait screen is now presented, and when the “wash” is done, the user should press the “Simulate Wash Hall done” button, which will restart the program.



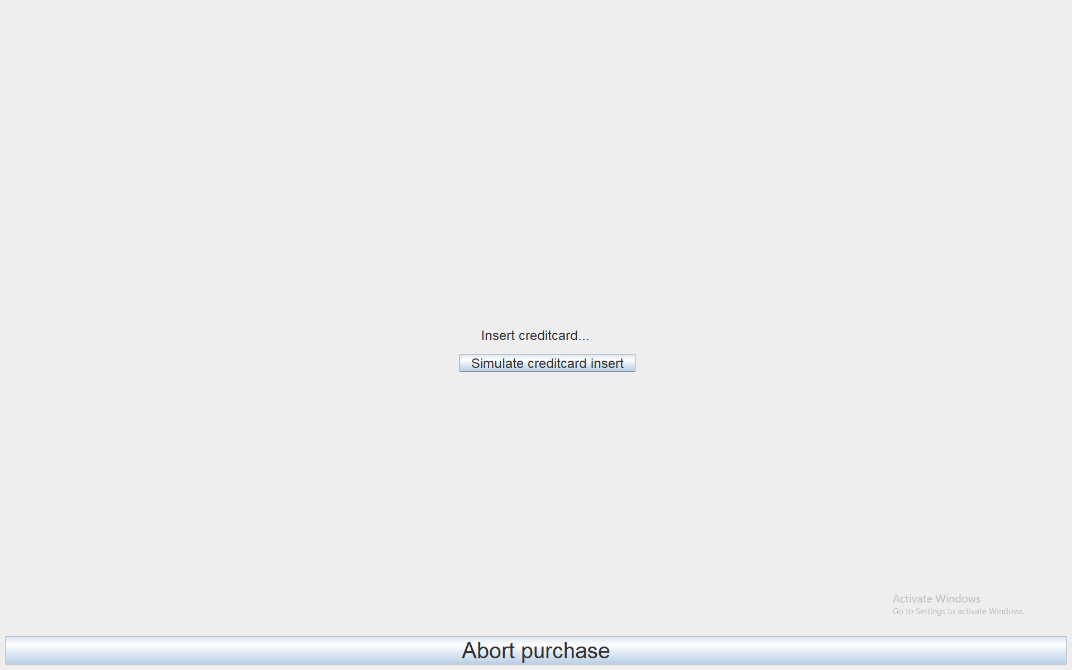
**6.**

Going back to point 2 or 3. The user also have an option to recharge their current WashCard. Entering the recharge menu, will show the user the following screen. The program offers 3 different amounts, 1000 DKK, 500 DKK or 200 DKK.



**7.**

When the user chooses one of the 3 options, a screen, with a simulation of entering a creditcard, is shown. When pressing the “Simulate creditcard insert” button is pushed. The user is transferred back to point 2 or 3, and the new balance is shown in the top right corner.



**8.**

To watch owner statistics, once again navigate to the folder “Program”, and execute the file named CarwashStatistics.jar. After launching the program, open your favourite browser (Firefox, IE, Chrome etc.), and enter <http://localhost/> in the address field. It will show the user, the following screen.

