Exam Project

Group 33

Andreas de Reus (s183952)
Bryndís Rósa Sigurpalsdóttir (s212423)
Hanna Reß (s212220)
Anton Schousen (s175159)
Paweł Zieliński (s213231)
Eydís Sjöfn Kjærbo (s212943)
Christopher Bonnevie Henriksen (s183938)

Technical University of Denmark
Course 02291 - System Integration

May 4th, 2022

Contents

1 Introduction

author: Bryndís and Eydís
In this project, we will use the knowledge we have gained throughout the course to model a toll system. The toll system consists of many stations connected to a motorway. At each station you can both enter and exit the motorway, but to do so you need to pay tolls. The price of these tolls varies depending on the vehicle type: truck, car, or motorcycle. Our solution allows users to purchase a single ticket with a credit card at entry or a toll tag through a website for wireless check-in and check-out. Each station has a cashier who deals with circumstances where our technology fails or human mistakes arise. Each station has a station manager that can access station reports displaying various information regarding that particular station. The enterprise has an enterprise manager that can access all station reports and change the toll rates.

The model's various components will be documented in the report. These components are: requirements, acceptance tests, design, and validation.

1.1 Methodology Used

author: Anton and Andreas
As a first part of the project, the glossary, domain model, business activity diagram and use cases was developed as a group. Afterwards, people were given specific use cases and tasked with writing the detailed use cases and related fit tests. These have afterwards been reviewed and modified by the group. Lastly, the group has also made class diagrams, OCL constraints, LSM's and sequence diagrams, together, though each person has been the main author for some scenarios. For tools used by the group, a Miro board was primarily used, as it enabled us to all work in parallel while also giving us the flexibility and freedom to make the different types of diagrams on this same platform. It also ensured that we always had a good overview over the whole project, and what needed to be modified/reviewed.

Lastly, everyone have contributed equally to the project, but each section has two main authors, as to comply with the DTU guidelines.

2 Requirements

2.1 Domain analysis

author: Paweł and Bryndís
For the domain analysis, we defined a glossary which contains all the relevant terms to understand the toll system that will be modeled in this report. After that, we modeled the toll system in a class diagram. In the end, we provide activity diagrams containing the basic workflows of the toll system.

2.1.1 Glossary

author: Christopher and Hanna
The glossary describes every important definition of the toll system. It is divided into objects and actions. The objects are nouns describing a specific part of the system. The actions explain use cases that are provided to customers or employees of the toll system.

Objects

- **Customer**: A person who owns one or more single tickets and/or toll tags. The person also has one or more vehicles, as well as credit cards.
- **Credit Card**: Owned by a person, and used to pay for single tickets in a card reader when entering the motorway using a normal lane, or used online to order a toll tag for a specific vehicle, owned by the same customer.
- **Vehicle**: Either a truck, car or a motorbike owned by a customer. Can be associated with one or more toll tags.
- **Truck**: A type of vehicle.
- **Car**: A type of vehicle.
- **Motorbike**: A type of vehicle.
- **License plate**: A unique identification number mounted on a vehicle.
- **Single ticket**: A ticket that has a fixed price depending on vehicle type, regardless of distance travelled. Can be bought using a credit card with a credit card reader and touch screen at normal check-in lanes. Has to be returned when exiting the motorway at a normal check-out lane. A Single Ticket is associated with a single vehicle and is only valid for one trip (max. 24h).
- **Toll tag**: A toll tag is an wireless device, which can be ordered through the enterprise website by the customer. It is mounted to a vehicle, and is thus associated to a single customer, credit card and vehicle. The toll tag is used at express check-in and check-out lanes, where a antenna registers entering and exiting the motorway. The price paid depends on the distance travelled. At the end of the month, the customer has to pay a monthly bill, whose price is based on the distance travelled with the toll tag on the motorway in the previous month. Note that if the customer fails to pay, the toll tag will be invalidated and will not be reactivated until the debt is paid. If the toll tag is identified as invalid upon exiting the highway, the customer can still exit and the trip expenses are added to his account.
- **Bill**: The payments are deducted from the customer's bank account using the credit card number that the customer provided. In a case that the payment fails the customer will be notified that he will need to pay the outstanding bill in order to use his tag/s again. In that scenario, the consumer must view the outstanding bill via the internet application, where he may pay the open bills by providing a credit card number.

- **Toll Lane**: A toll lane is a lane used by vehicles to either enter (check-in) or exit (check-out) of the motorway. They can either by normal lanes, where one can buy/use single tickets, or express lanes, where one can use toll tags. A toll lane consists of a toll lane computer, which is always connected to the barrier. In a normal check-in lane, the computer is also connected to a credit card reader, touch screen and a single ticket printer. In a normal check-out lane, the computer is connected to a single ticket reader. In express lanes, the computer is connected to a antenna. In all cases, the toll lane computer is also connected to the toll-stations station server. A toll lane is associated with a single toll station.
- **Normal Lane**: A normal lane is a lane where a single ticket can be used. Single tickets can be purchased in a normal check-in lane with a credit card reader and touch screen, printed with a printer, and returned in a normal check-out lane with a single ticket reader.
- **Express Lane**: An express lane allows for the use of a toll tag to wirelessly and automatically check-in and check-out using an antenna at the express lanes.
- **Check-in Lane**: A lane to check-in to the motorway. A normal check-in lane allows the user to purchase a single ticket, while a toll tag is used at express check-in lanes.
- **Check-out Lane**: A lane to check-out from the motorway. A normal check-out lane requires the user to return a single ticket, while an express check-out lane reads the vehicle's toll tag.
- **Toll Lane Computer**: A toll lane computer is associated with a single toll lane and the toll station's station server, and also is connected to various other devices, depending on the type of toll lane: If it is a normal check-in lane, it is connected to a credit card reader, touch screen and printer. If it is a normal check-out lane, it is connected to a single ticket reader. If it is either an express check-in or check-out lane, it is connected to an antenna. In all cases, it is also connected to a barrier.
- **Barrier**: A barrier is placed at each toll lane, and is connected to the toll lane computer. When a customer successfully checks-in or checks-out, the barrier is raised to allow the vehicle to pass. Barriers have a sensor that detects if a vehicle has passed the barrier
- **Antenna**: An antenna is placed at each express lane (both check-in and check-out), and is connected to the toll lane computer. It is used to read the toll-tags on passing vehicles, as to note the distance the vehicle has travelled on the motorway.
- **Single Ticket Reader**: A device to read single tickets when exiting the motorway using a normal check-out lane. Is connected to the toll lane computer.
- **Printer**: A device used to print a single ticket when a customer has paid for it. Is placed when entering the motorway using a normal check-in lane. Is connected to the toll lane computer.
- **Touch Screen**: A device used by the customer to select vehicle type for their single ticket when entering the motorway. Is placed at normal check-in lanes. Is connected to the toll lane computer.
- **Credit Card Reader**: A device used by the customer to pay for their single ticket when entering the motorway. Is placed at normal check-in lanes, and is connected to the toll lane computer.
- **Toll Station**: A toll station is a collection of toll lanes, which are either check-in or check-out, but can be a mix between normal and express lanes. Each toll station has a station server, station client, station manager and a cashier. Each station is linked to an enterprise server, which is linked to an enterprise client.
- **Station Server**: A station server holds data for all toll lanes at the toll station, and is accessed via a station client. The station server is connected to all the toll lane computers, as well as the single enterprise server.

- **Station Client**: A station client is placed at a toll station, and connected to a station server. The station client allows the station manager to print a station report, containing information about the toll station and related toll lanes. The manager can view reports and access and modify customer data using the station client.
- **Station Manager**: A station manager is a person employed to manage a single toll station. He can use the station client to generate a station report.
- **Station Report**: A report generated for a toll station by a station manager using the station client. Contains information about the specific toll station, specifically how many and which types of vehicles perform check-ins and check-outs in freely definable time period. Also, the report should describe how much money was earned, as well as how many single tickets and toll tags were used in the time period.
- **Cashier**: A single cashier is associated with each toll station. The cashier is there to handle cases where a single ticket is invalid at check-out, and find a way for the customer to check-out.
- **Enterprise Server**: A single server which the enterprise has. It is connected to the website and bank, and is used to check the validity of toll tags. An enterprise manager can access the enterprise server through the enterprise client to generate an enterprise report. A Web server connects the enterprise server to the Internet.
- **Enterprise Client**: A computer client that is connected to the enterprise server. Allows the enterprise manager to generate a enterprise server.
- **Enterprise Manager**: A single enterprise manager is employed in the enterprise. The enterprise manager can use the enterprise client to generate enterprise reports, change the toll rates for single tickets and toll tags, and send notifications to customers.
- **Enterprise Report**: A report that contains the same categories of information as in the station report, but can do this for the whole enterprise, e.g. for all stations. Is generated by the enterprise manager using the enterprise client.
- **Bank**: The bank received information from the enterprise server, as to deduct money from customers bank accounts.
- **Website**: Is used by the customers to order toll tags and pay monthly bills for the toll tags.

Actions

- **Buy toll tag**: A customer uses the Website application to order one or more toll tags for his vehicles. To do so, he must provide personal information such as his name, credit card number, license plate, and vehicle type. The toll tag is then sent by mail home to the customer.
- **Check-in with toll tag**: A customer drives past an antenna at an express check-in lane while in their vehicle.
- **Check-out with toll tag**: A customer drives past an antenna at an express check-out lane while in their vehicle.
- **Check-in with single ticket**: The customer drives up to the touch screen at a normal check-in lane. He chooses his vehicle type on a touch screen and uses his credit card at the credit card reader. The printer prints out a single ticket, which the customer takes.
- **Check-out with single ticket**: The customer drives up to a single ticket reader and gives it his single ticket.
- **Enter motorway**: After check-in is done correctly, either with toll tag or using a single ticket, the barrier is lifted and the vehicle can enter the motorway.

- **Exit motorway**: After check-out is done correctly, either with toll tag or using a single ticket, the barrier is lifted and the vehicle can exit the motorway.
- **Send monthly bill**: At the end of each month, the enterprise server sends a bill to each customer who owns at least one toll. This is accomplished by withdrawing money from the customer's bank account using the credit card number given by the consumer
- **Pay overdue bill**: The customer uses the website to approve the payment of the monthly bill for his toll tags, given he has enough money on his credit card that he has to fill out. .
- **Pay overdue bill**: The customer must utilize a Web application to view his outstanding bills and pay by giving a credit card number to pay with. If the payment fails, the consumer might try again later with a different credit card number.
- **Notify customers**: The enterprise manager sends out a notification to the customers via email using the enterprise client.
- **Generate station reports**: A station manager uses a station client to generate a station statistical report for the specific stations and its associated toll lanes.
- **Generate enterprise reports**: The enterprise manager uses an enterprise client to generate an enterprise report for the enterprise, containing data from all stations.
- **Change toll rate**: The enterprise manager uses the enterprise client to change the toll rates of the toll tags and/or the single tickets. As a result, all stations' prices change at the same time.

2.1.2 Domain Model

author: Eydís and Andreas
Below is our model of the domain. It includes the classes (nouns) identified from the assignment description, as has been described in the glossary. All the classes in the domain are connected, but note that one can identify are primary collection of classes related to the customer, while another collection of classes are related to the enterprise. Lastly, decisions have been made to not include associations between classes, that are only used in actions, e.g. between credit card and card reader, or between enterprise client and (sending) notifications. This is due to these classes not sharing a direct relation of ownership, e.g. a notification not "being" owned by a enterprise client. The class diagram can be found in figure 1.

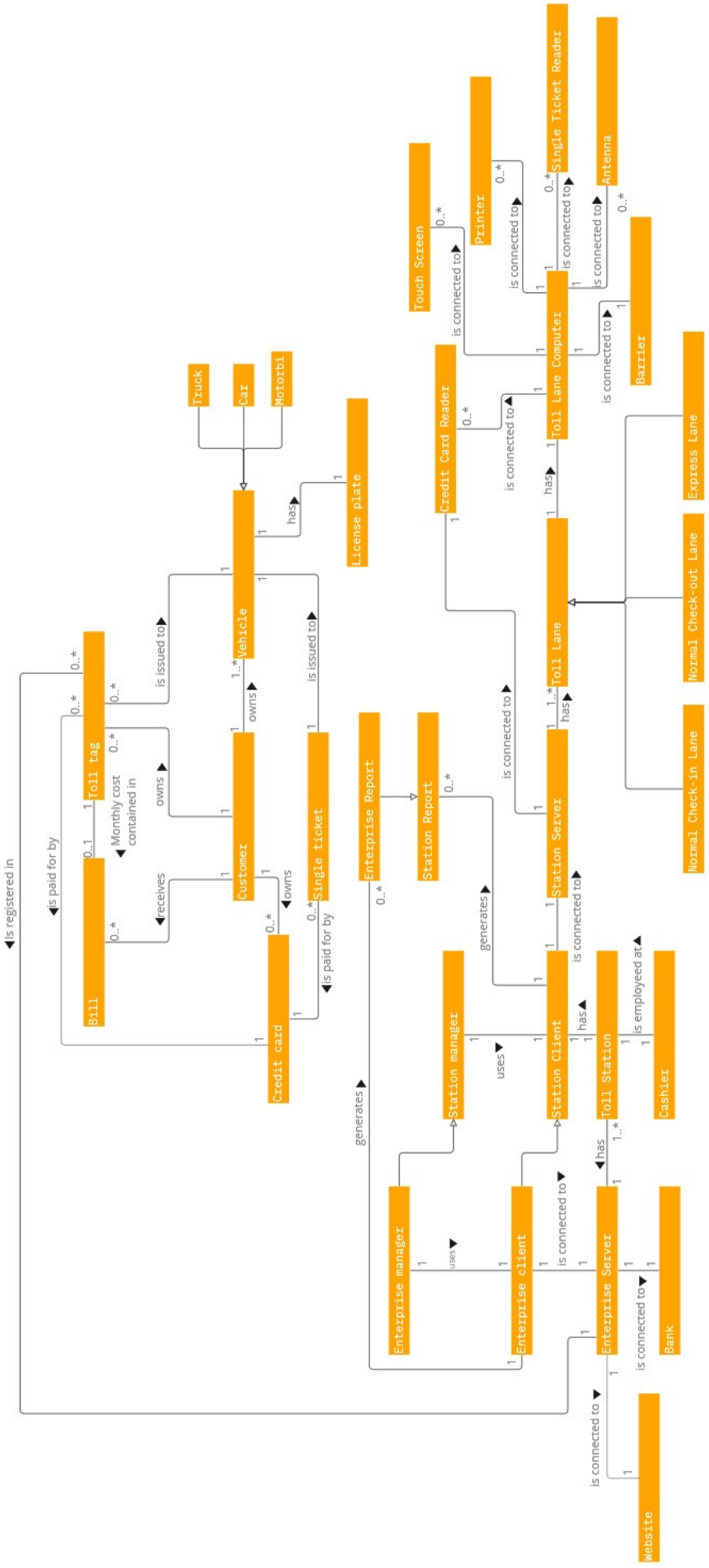
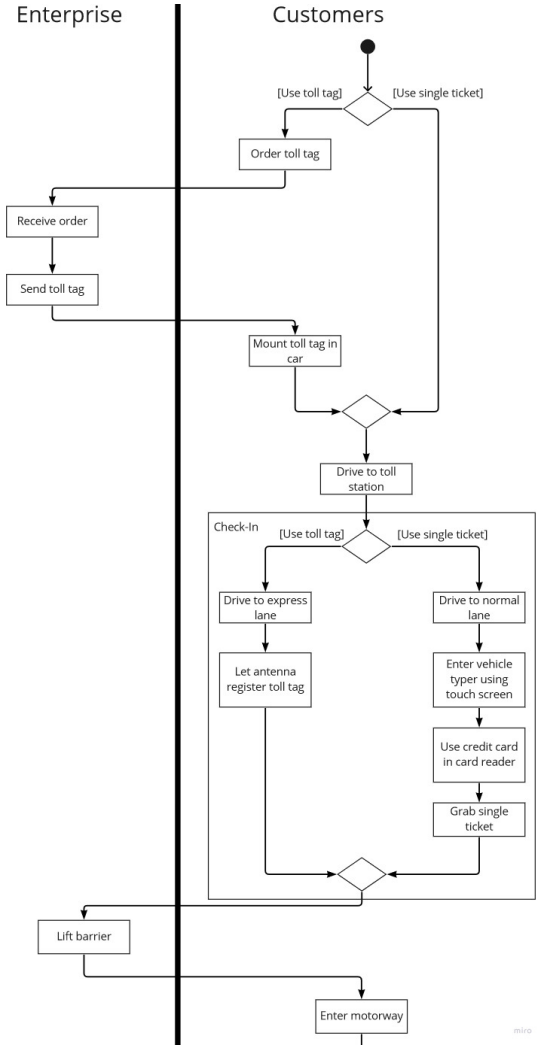


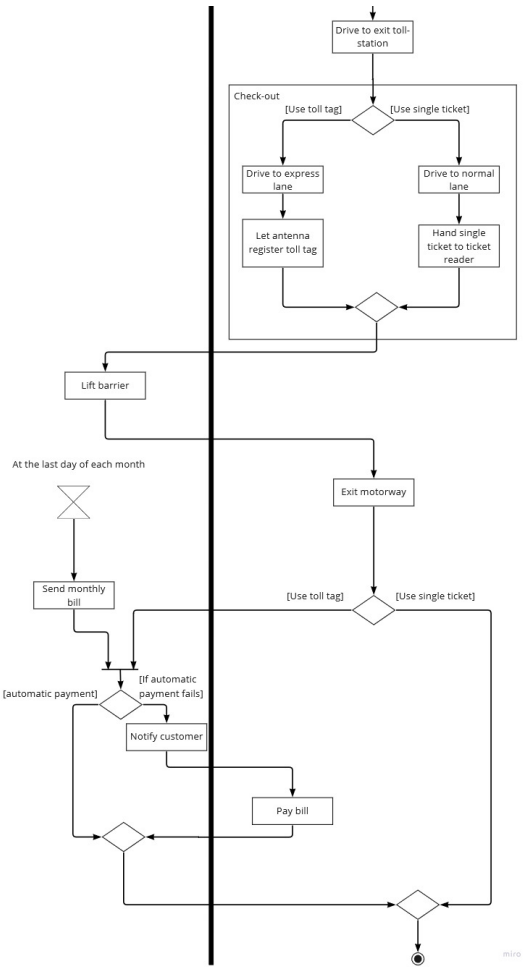
Figure 1: Domain model of the toll system

2.1.3 Activity Diagram

author: Anton and Christopher
For the activities that can be performed in the toll system, we have combined them into a single activity diagram, which shows how a customer interacts with the system and the relevant activities on the enterprise side. In this case, the basic workflow is considered the customer buying a toll tag, entering the motorway (using either single tickets or toll tag), leaving it again and possible afterwards paying overdue bills. This activity diagram can be seen below. The activity diagram can be found in figure 2.



(a) Upper part of the activity diagram



(b) Lower part of the activity diagram

Figure 2: Activity diagram for basic workflow of the toll system

2.2 Functional requirements

2.2.1 Use Case Diagram Diagram

author: Paweł and Bryndís

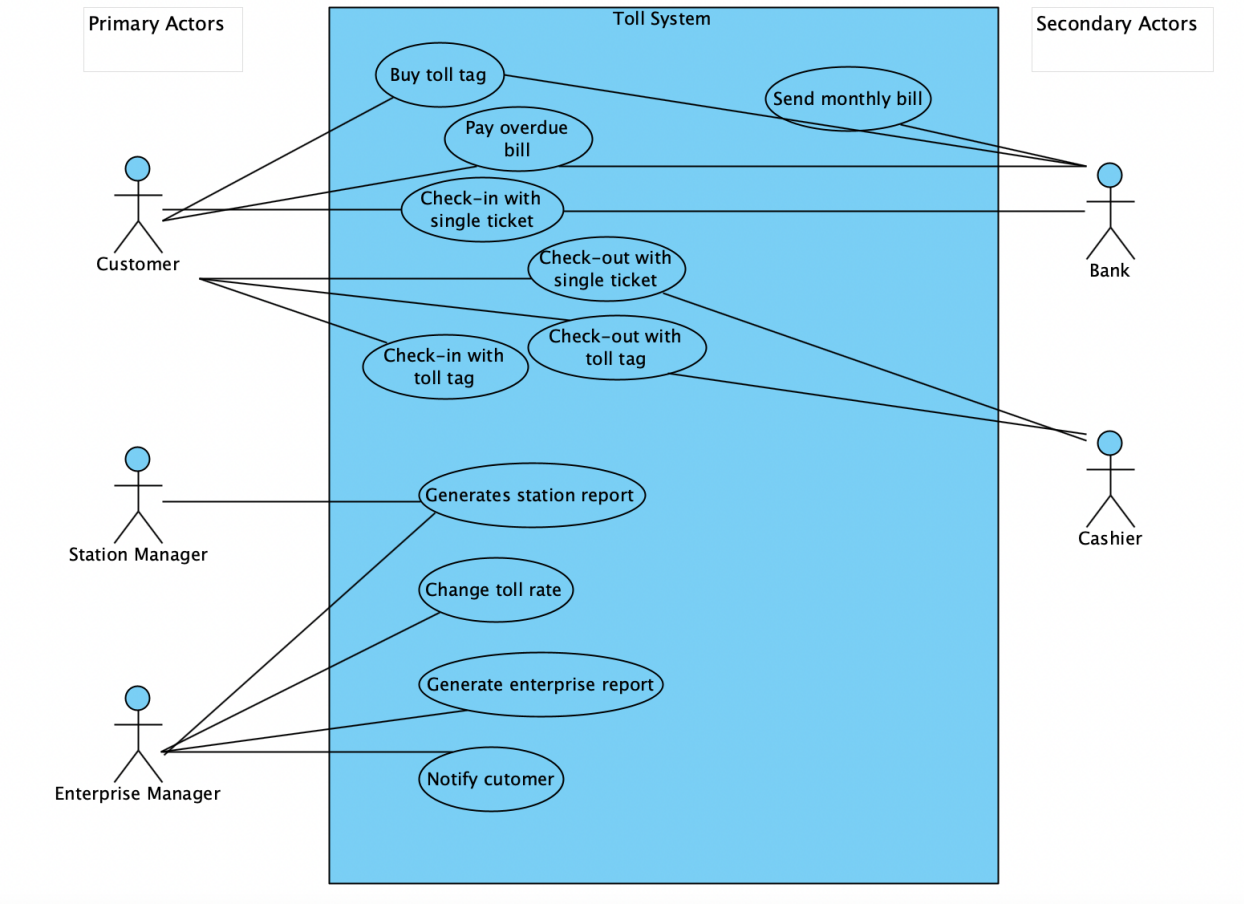


Figure 3: Use case diagram of the toll system

2.2.2 Use Cases

We selected seven use cases from the use case diagram that we thought were most essential to our customer. The ones we selected are 'Buy toll tag', 'Pay overdue bill', 'Check-in with single ticket', 'Check-out with single ticket', 'Check-in with toll tag', 'Check-out with toll tag' and 'Generate station report'. In the next pages, we'll go through these use cases in further depth.

Use case 1:

- **Use Case Name:** Buy toll tag
- **Summary:** A customer registers and purchase a toll tag online.
- **Actors:** The customer
- **Preconditions:** Web application functionates properly, the user has an account, and he or she is logged in. There's a working connection with bank.
- **Basic course of events:**

1. Customer fills the personal data attached to the purchase (name, vehicle type, license plate, credit card).
2. Customer clicks "Order toll tag" button.
3. Data is processed by a system.
4. Action is continued through payment website.
5. Transaction is made.
6. Toll system receives confirmation from the bank.
7. Toll system creates new order of the toll tag and sends it to production.
8. Customer is informed that the toll tag has been ordered.

- **Alternative paths:** Transaction is rejected by a bank

5a. Transaction is rejected by a bank (e.g. not enough resources).
6a. Message pop up "Transaction failed".
7a. System returns to second case.

- **Author:** Paweł and Andreas

Use case 2:

- **Use Case Name:** Pay overdue bill
- **Summary:** A customer owning a toll tag is notified of an overdue bill.
- **Actors:** The customer
- **Preconditions:** Automatic billing failed and customer is notified of an overdue bill
- **Basic course of events:**

1. The customer logins to the web application
2. Customer inserts new credit card information
3. The overdue bill is paid
4. Toll tag is valid again

- **Alternative paths:** The payments fails

3a. The payments fails (e.g. not enough resources)
4a. Message pop up "Payment failed try again later"

- **Author:** Andreas and Hanna

Use case 3:

- **Use Case Name:** Check-in with single ticket
- **Summary:** The customer checks in at an normal check-in toll lane using a credit card and receives a single ticket.
- **Actors:** The customer
- **Preconditions:** The customer owns a credit card. The vehicle is placed in a normal check-in lane.
- **Basic course of events:**

1. The customers enters his vehicle type through a touch screen.
2. The system notifies the costumer of the fee he must pay.

3. The customer pays for the single ticket with his credit card through a Credit card reader
4. The system checks if the payment was approved.
5. The customer receives a single ticket printed by the system.
6. The barrier is lifted

- **Alternative paths:** Credit card payment failed

4a. Credit card payment failed

5a. The system notifies the customer that the payment failed and tells him he can try again.

- **Postconditions:** The customer enters the motorway and the barrier is lowered after the car has passed. The touch screen resets.
- **Author:** Bryndis And Eydís

Use case 4:

- **Use Case Name:** Check-out with single ticket
- **Summary:** The customer checks out at a normal check-out lane using an already paid single ticket.
- **Actors:** Customer
- **Preconditions:** The customer bought and checked in using a single ticket. The vehicle is placed in a normal check-out lane.
- **Basic course of events:**

1. The customer places the ticket into the single ticket card reader.
2. The system checks if the ticket is valid.
3. The system lifts the barrier.

- **Alternative paths:** The customer has lost his ticket.

1a. The customer has lost his ticket.

2a. The cashier decides what to do i.e. charges the customer correctly and lifts the barrier.

- **Alternative paths:** The reader can't read the ticket.

2b. The reader can't read the ticket.

3b. The system notifies the customer.

4b. The system notifies the cashier.

5b. The cashier decides what to do i.e. charges the customer correctly and lifts the barrier.

- **Alternative paths:** The ticket is invalid.

2c. The ticket is invalid.

3c. The system notifies the customer.

4c. The system notifies the cashier.

5c. The cashier decides what to do i.e. charges the customer correctly and lifts the barrier.

- **Postconditions:** The customer exits the motorway and the barrier is lowered after the car has passed.
- **Notes:** Valid ticket is a ticket that has not been used and is less than 24 hours old.
- **Author:** Eydís and Christopher

Use case 5:

- **Use Case Name:** Check-in with toll tag
- **Summary:** The customer checks in at an express check-in toll lane using his toll tag mounted on his vehicle
- **Actors:** Customer
- **Preconditions:** The customer has a valid toll tag
- **Basic course of events:**
  1. The customer drives up to the barrier at an express check-in toll lane
  2. The antenna at the toll lane detects the toll tag
  3. The antenna verifies the toll tag as valid and opens lifts the barrier
  4. The customer enters the motorway in their vehicle
  5. The barrier is lowered
- **Alternative paths:** The customer's toll tag is invalid
  - 3a. The system is noted as invalid in the system
  - 4a. The cashier is informed of this incident
  - 5a. The cashier redirects the customer to a normal check-in toll lane
- **Alternative paths:** The antenna does not recognize the toll tag
  - 2b. The antenna does not recognize the customer's toll tag
  - 3b. The customer pushes a button to inform the cashier
  - 4b. The cashier is informed of this incident
  - 5b. The cashier redirects the customer to a normal check-in toll lane
- **Postconditions:** The customer has entered the motorway. An additional check-in has been noted in the statistical database of the system.
- **Author:** Christopher and Anton

Use case 6:

- **Use Case Name:** Check-out with toll tag
- **Summary:** The customer wants to exit the motorway by using the toll tag.
- **Actors:** Customer
- **Preconditions:** The customer has checked-in with the toll tag and entered the motorway.
- **Basic course of events:**
  1. A customer drives up to an antenna at an express check-out lane.
  2. The toll tag is recognized by the antenna.
  3. The price is calculated and added to the bill.
  4. The barrier opens, the customer exits the motorway and the barrier closes.
- **Alternative paths:** The antenna does not recognize the toll tag.
  - 2a. The toll tag is not recognized by the antenna.
  - 3a. The cashier is notified and decides how to check-out the vehicle.

4a. The customer pays the required amount with a credit card.

5a. The barrier opens, the customer exits the motorway and the barrier closes.

- **Postconditions:** The customer has exited the motorway.
- **Notes:** Checking for a valid toll tag is not necessary since it does not change anything for the customer.
- **Author:** Hanna and Bryndís

Use case 7:

- **Use Case Name:** Generate station report
- **Summary:** A station manager uses a station client to generate a station report for the specific station and its associated toll lanes.
- **Actors:** Station manager
- **Preconditions:** The station client has connection to the station server.
- **Basic course of events:**

  1. The station manager accesses the station client.
  2. The station manager request the system to generate the station report.
  3. The system generates the station report automatically.
  4. System saves the report and displays message: "Report Generated".

- **Alternative paths:** Generationg the report fails.

  3a. The system fails to generate the station report.

  4a. System displays message: "generation of report failed".

- **Author:** Anton and Paweł

2.3 Non-Functional requirements

author: Anton and Hanna
Non-functional requirements of the planned toll system evaluate the software system based on responsiveness, usability, security, portability, and other non-functional criteria that are crucial to the software system's success. Furthermore, the nonfunctional criteria for our system ensure that the software toll system complies with legal and regulatory standards.

Non-Functional requirements

- The credit card reader should only be disconnected from the bank for a maximum 1 min per day.
- The toll lane computer can at most be disconnected from the station server at a maximum of 1 min per day.
- The Station server can at most be disconnected from the enterprise server at a maximum of 1 min per day. At least once a day to get toll rate.
- Customer data stored should always be compliant with the GDPR laws.
- The toll lane system should have the capacity to handle all lanes without the system getting overloaded.
- The toll lane system should be available 24h with a downtime of at most 1 min a day.
- It is essential that the software be portable. So switching from one operating system to another is not an issue.

3 Acceptance Tests

Here we will present a Fit test for each of the seven use case we defined in section 2.1.

3.1 Buy toll tag

Author: Pawel and Andreas

3.1.1 Main scenario

start	WebServerFixture	
check	Toll tags	111a23-1, 111a23-2, 132wec-1
enter	Name, license plate number, credit card number	Jane Doe, 123EES, 1234-5678
press	Vehicle type	Car
press	Order Toll Tag button	
check	Credit card transaction is approved	True
check	Toll tags in Order	123EES-1
check	Toll tags	111a23-1, 111a23-2, 132wec-1, 123EES-1
check	123EES-1 information	Jane Doe, 123EES, Car, 1234-5678
check	Message	"Toll tag has been ordered"

Table 1

3.1.2 Alternative scenario: Transaction fails

start	WebServerFixture	
check	Toll tags	111a23-1, 111a23-2, 132wec-1
enter	Name, license plate number, credit card number	Jane Doe, 123EES, 1234-5678
press	Vehicle type	Car
press	Order Toll Tag button	
check	Credit card transaction is approved	False
check	Toll tags in Order	
check	Toll tags	111a23-1, 111a23-2, 132wec-1
check	Message	"Transaction Failed""

Table 2

3.2 Pay overdue bill

Author: Andreas Hanna

3.2.1 Main scenario

start	PayOverDueBillFixture	
enter	customer login	Jane, 1234
enter	customer logged in	true
press	”Overdue bills” button	
check	message	”overdue bill of 300 kr”
enter	creditcard Number	1234
check	credit card transaction is approved	true
check	toll tags validity	true
check	message	”Payment completed”

Table 3

3.2.2 Alternative scenario: Payment of overdue bill fails

start	PayOverDueBillFixture	
enter	customer login	Jane, 1234
enter	customer logged in	true
press	”Overdue bills” button	
check	message	”overdue bill of 300 kr”
enter	creditcard Number	1234
check	credit card transaction is approved	false
check	toll tags validity	false
check	message	”payment failed, try again later”

Table 4

3.3 Check-in with single ticket

Author: Bryndís and Eydís

3.3.1 Main scenario

start	SingleTicketReaderTouchScreenFixture	
check	Valid tickets	null
check	Barrier closed	True
press	Vehicle Type	Car
check	Message	”The fee for this vehicle is 50kr please insert card”
enter	card info	1234
check	credit card transaction is approved	true
check	dispensed	single ticket
press	Valid tickets	123456789abc
check	Barrier lifted	True
press	Enter motorway in vehicle	
check	Car has passed	True
check	Barrier closed	True
check	Touch screen reset	True

Table 5

3.3.2 Alternative scenario: Payment fails

start	SingleTicketReaderTouchScreenFixture	
check	Valid tickets	null
check	Barrier closed	True
press	Vehicle Type	Car
check	Message	”The fee for this vehicle is 50kr please insert card”
enter	card info	1234
check	credit card transaction is approved	false
check	dispensed	nothing
check	Valid tickets	null
check	Barrier lifted	False
check	Message	”Try again credit card payment failed”
check	Touch screen reset	True

Table 6

3.4 Check-out with single ticket

Author: Eydís and Christopher

3.4.1 Main scenario

start	SingleTicketReaderFixture	
check	Valid tickets	123456789abc
check	Barrier closed	True
enter	Ticket/input	123456789abc
check	Valid tickets	null
check	Barrier lifted	True
press	Enter motorway in vehicle	
check	Car has passed	True
check	Barrier closed	True

Table 7

3.4.2 Alternative scenario: The customer has lost his ticket

This is a scenario that might easily occur as a result of human mistake. Nothing happens since nothing is entered into the system. The customer must make contact with the cashier.

3.4.3 Alternative scenario: The reader can't read the ticket

start	SingleTicketReaderFixture	
check	Valid tickets	123456789abc
check	Barrier closed	True
enter	Ticket/input	null
check	Message	"Can't read ticket, please wait for the cashier"
check	Cashier informed	True

Table 8

3.4.4 Alternative scenario: The ticket is invalid

start	SingleTicketReaderFixture	
check	Valid tickets	null
check	Barrier closed	True
enter	Ticket/input	123456789abc
check	Message	"Invalid ticket, please wait for the cashier"
check	Cashier informed	True

Table 9

3.5 Check-in with toll tag

Author: Christopher and Anton

3.5.1 Main scenario

start	TollTagCheckInFixture	
enter	move to barrier in vehicle with specific toll tag ID	123
check	toll tag detected by antenna	”123”
check	toll tag valid	true
check	barrier lifted	true
press	enter motorway in vehicle	
check	car has passed	true
check	barrier closed	true

Table 10

3.5.2 Alternative scenario: Toll tag is invalid

start	TollTagCheckInFixture	
enter	move to barrier in vehicle with specific toll tag ID	123
check	toll tag detected by antenna	”123”
check	toll tag valid	false
check	cashier is informed	true
check	cashier redirects customer to normal lane	true
check	barrier closed	true

Table 11

3.5.3 Alternative scenario: Toll tag is not detected by antenna

start	TollTagCheckInFixture	
enter	move to barrier in vehicle with specific toll tag ID	123
check	toll tag not detected by antenna	true
press	button to inform cashier	
check	cashier is informed	true
check	cashier redirects customer to normal lane	true
check	barrier closed	true

Table 12

3.6 Check-out with toll tag

Author: Hanna and Bryndís

3.6.1 Main scenario

start	TollTagCheckOutFixture	
enter	car enters express lane with toll tag	123
check	toll tag detected by antenna	”123”
press	price added	
check	barrier lifted	true
press	car through barrier	
check	barrier closed	true

Table 13

3.6.2 Alternative scenario A: Toll tag not recognized

start	TollTagCheckOutFixture	
check	toll tag not detected by antenna	true
press	button to inform cashier	
check	cashier informed	true
check	barrier closed	true
press	cashier decides price	
press	pay with credit card	
check	barrier lifted	true
press	car through barrier	
check	barrier closed	true

Table 14

3.7 Generate station report

Author: Anton And Paweł

3.7.1 Main scenario

start	Station Client	
enter	dates (start,end)	01-12-2021 , 01-01-2022
Press	Generate Report	
check	correct dates	true
Check	StationReport is generated	Success
check	message	”report generated”

Table 15

3.7.2 Generating station report failed

start	Station Client	
enter	dates (start,end)	01-01-2022 , 01-12-2021
Press	Generate Report	
check	correct dates	false
Check	StationReport is generated	Failed
check	message	”generation of report failed”

Table 16

4 Design

4.1 Design decisions and assumptions

author: Paweł and Eydís
For the cases where the antenna does not recognize the toll tag, a simple button component has been added for the customer to interact with, as to call on the assistance from the cashier. This is done, as we assume that the normal workflow of the toll system is automatic, e.g. checking in and out with single tickets and toll tag. Thus we assume that a cashier is not assigned to a toll lane, but rather the station itself, so they can be flexible and help out at the toll stations where assistance is needed. For this case, we also assume that the station server can contact a cashier regarding assistance needed at a toll lane, but we do not add a new component for this, as to keep the design simple. We only add interfaces between the cashier and the station server. Additionally, when a manual check-out is needed, the cashier can also open the barrier by interacting with the relevant check-out computer. This has also been modelled as interfaces.

4.2 Component Design

author: Christopher and Andreas
The component diagram gives an overview of all components in our system. It can be found in figure 4. After that we provide the ports and their interfaces for each connection.

4.2.1 Component Diagram with interfaces: Web Server - Enterprise Server

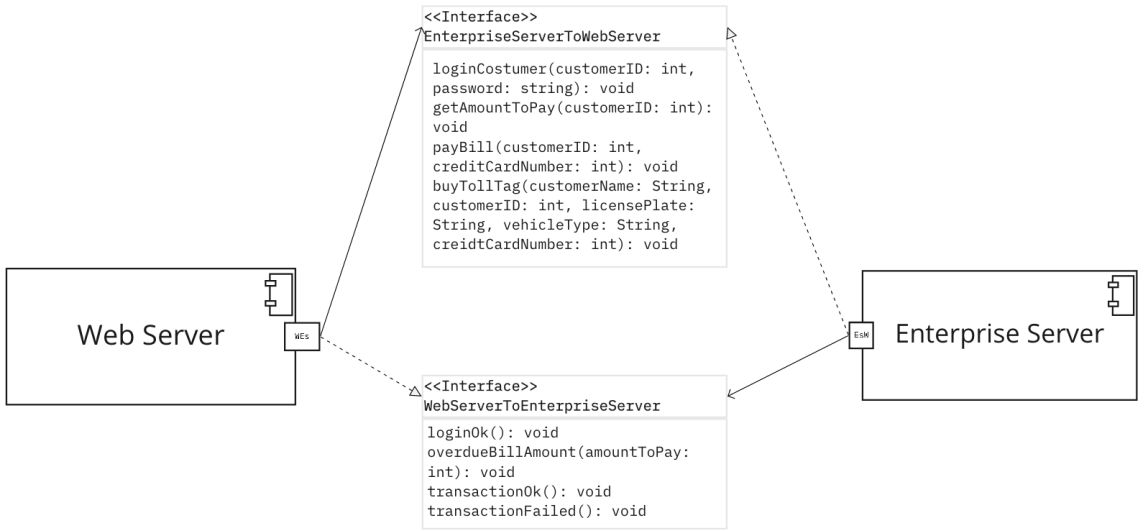


Figure 5: Component Diagram with interfaces: Web Server - Enterprise Server

4.2.2 Component Diagram with interfaces: Bank - Enterprise Server

author: Bryndís and Eydís

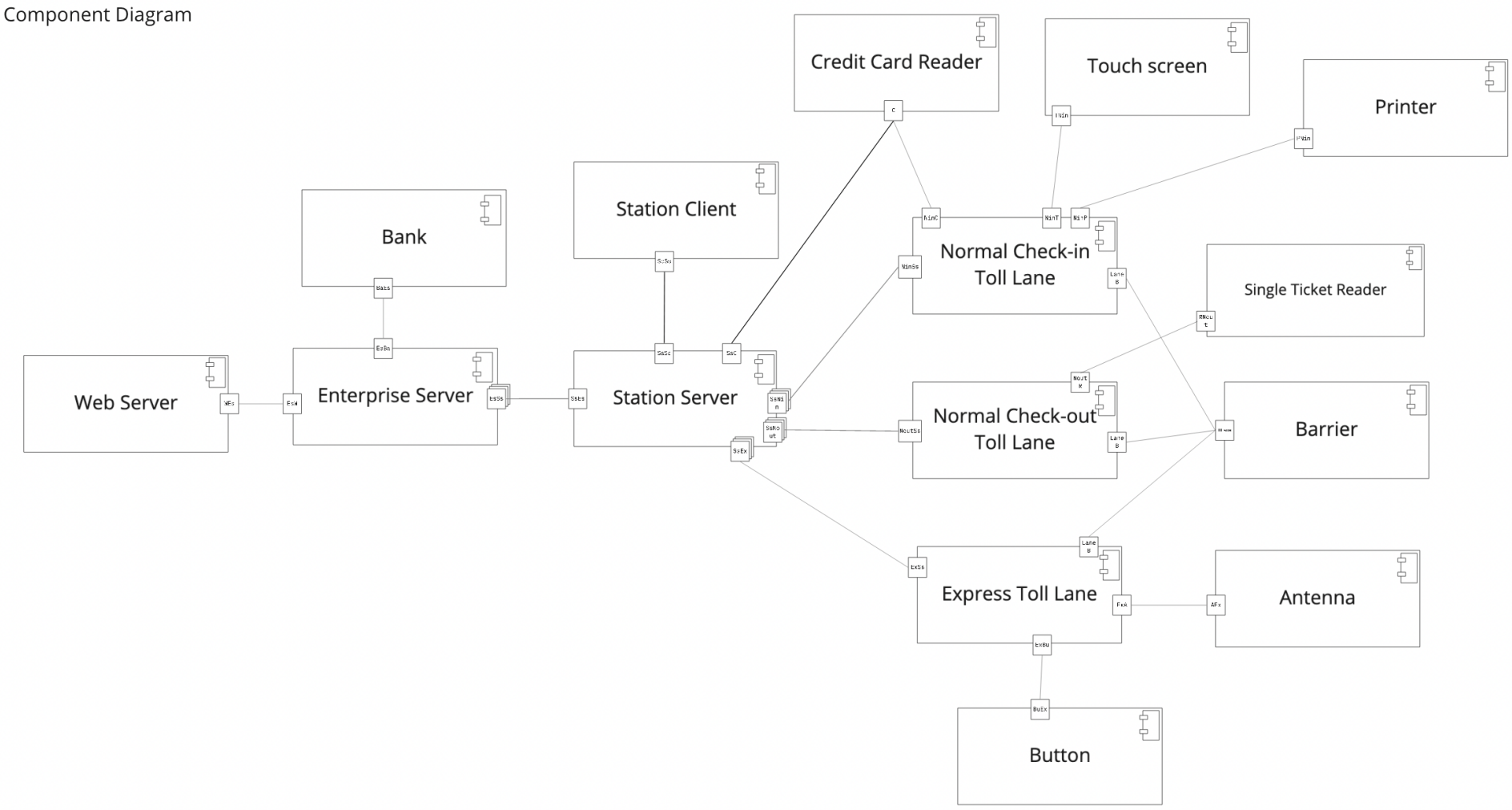


Figure 4: Component diagram of the toll system

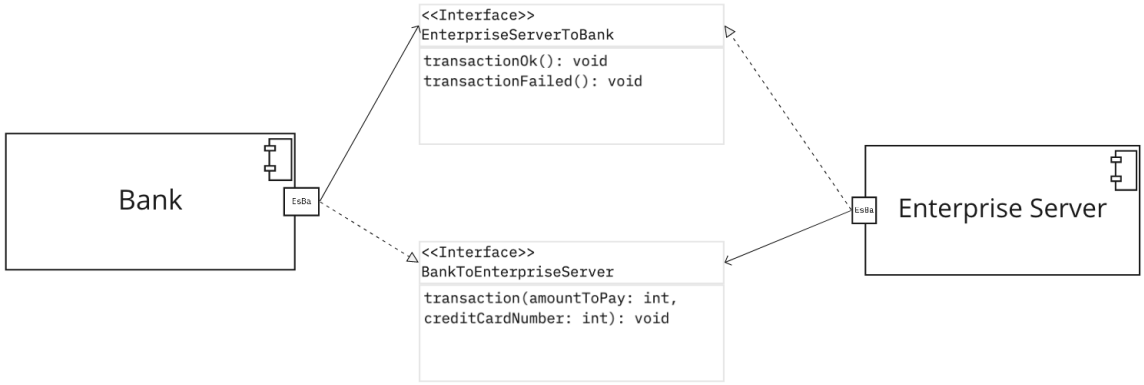


Figure 6: Component Diagram with interfaces: Bank - Enterprise Server

4.2.3 Component Diagram with interfaces: Station Server - Enterprise Server

author: Anton and Andreas

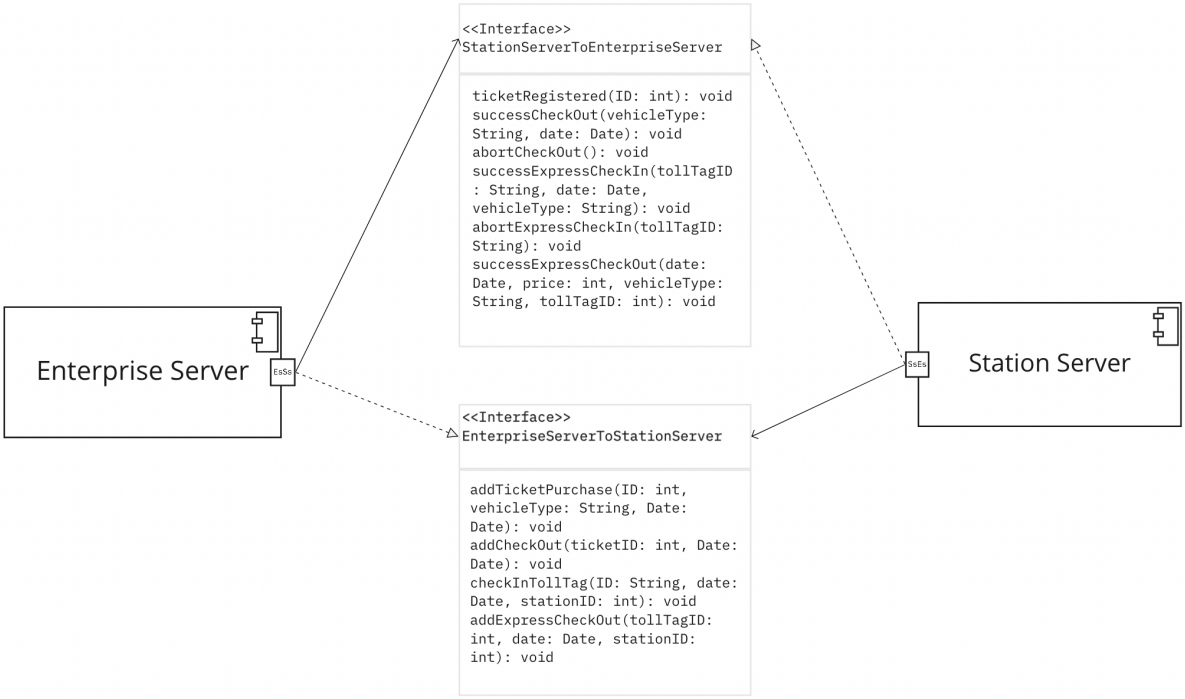


Figure 7: Component Diagram with interfaces: Station Server - Enterprise Server

4.2.4 Component Diagram with interfaces: Station Server - Station Client

author: Christopher and Hanna

Figure 8: Component Diagram with interfaces: Station Server - Station Client

4.2.5 Component Diagram with interfaces: Station Server - Credit Card Reader

author: Bryndís and Anton

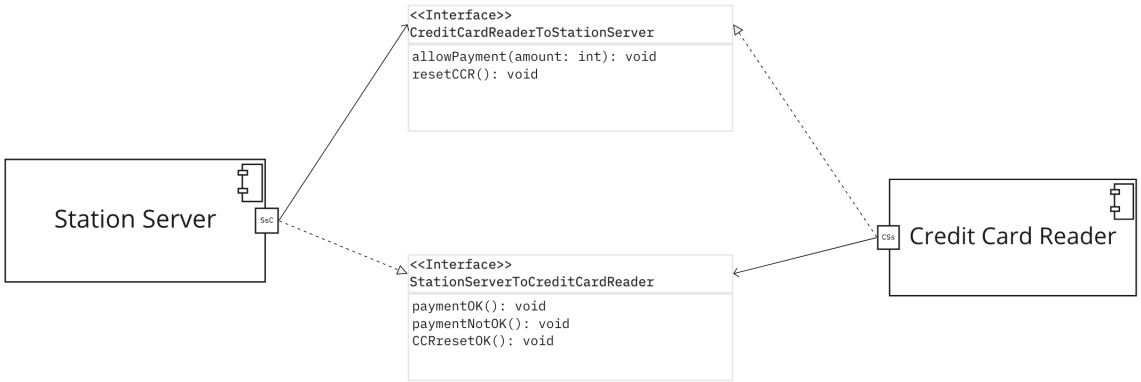


Figure 9: Component Diagram with interfaces: Station Server - Credit Card Reader

4.2.6 Component Diagram with interfaces: Normal Check-in Toll Lane - Station Server

author: Christopher and Hanna

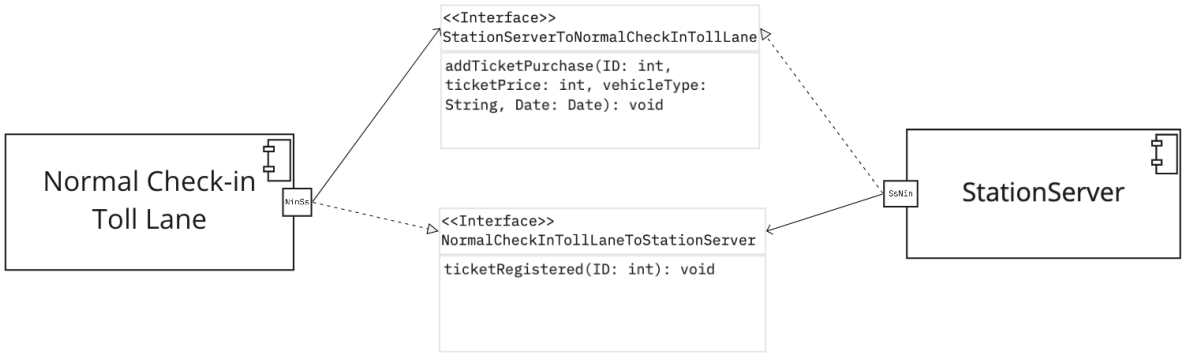


Figure 10: Component Diagram with interfaces: Normal Check-in Toll Lane - Station Server

author: Paweł and Christopher

4.2.7 Component Diagram with interfaces: Normal Check-in Toll Lane - Credit Card Reader

author: Andreas and Eydís

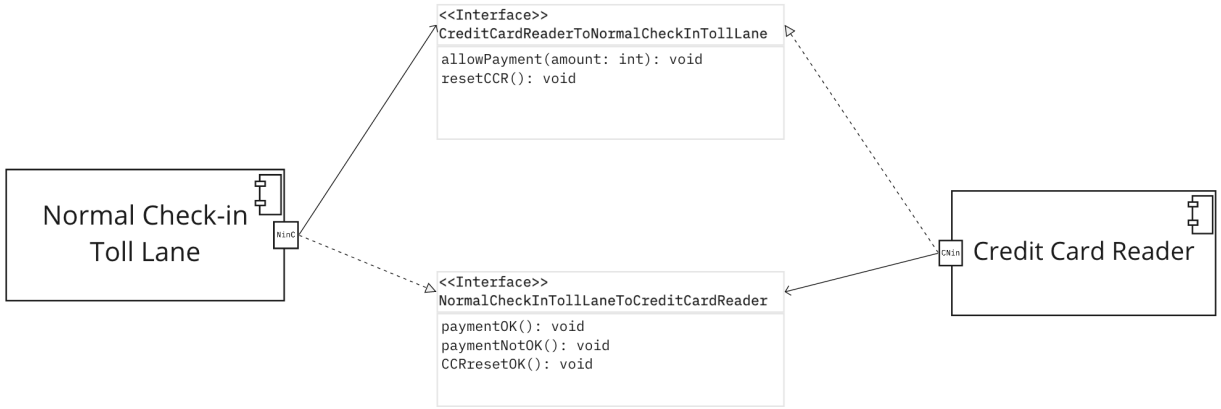


Figure 11: Component Diagram with interfaces: Normal Check-in Toll Lane - Credit Card Reader

author: Bryndís and Paweł

4.2.8 Component Diagram with interfaces: Normal Check-in Toll Lane - Touch Screen

author: Christopher and Paweł

Figure 12: Component Diagram with interfaces: Normal Check-in Toll Lane - Touch Screen

4.2.9 Component Diagram with interfaces: Normal Check-in Toll Lane - Printer

author: Paweł and Christopher

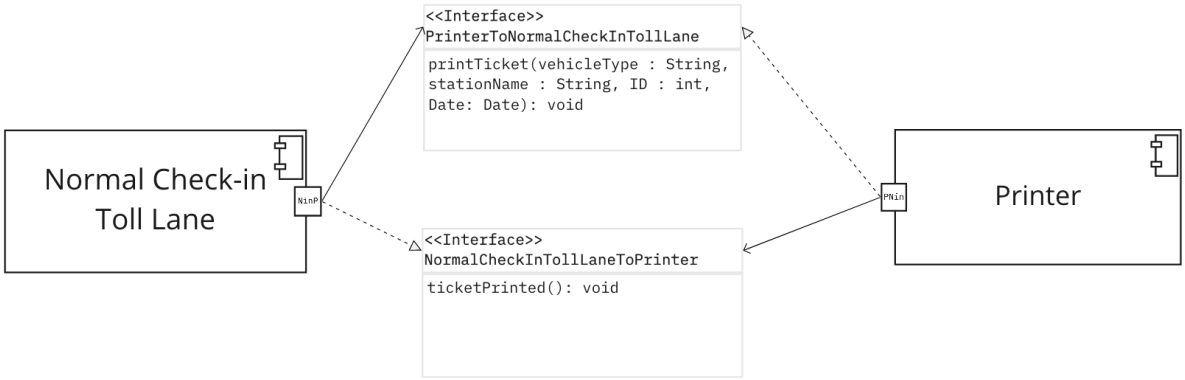


Figure 13: Component Diagram with interfaces: Normal Check-in Toll Lane - Printer

4.2.10 Component Diagram with interfaces: Normal Check-out Toll Lane - Station Server

Author: Hanna and Andreas

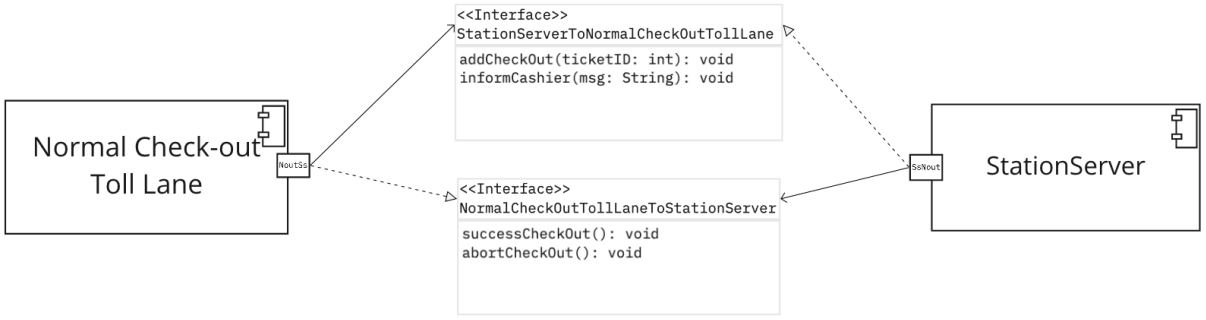


Figure 14: Component Diagram with interfaces: Normal Check-out Toll Lane - Station Server

4.2.11 Component Diagram with interfaces: Normal Check-out Toll Lane - Single Ticket Reader

Author: Eydís and Bryndís

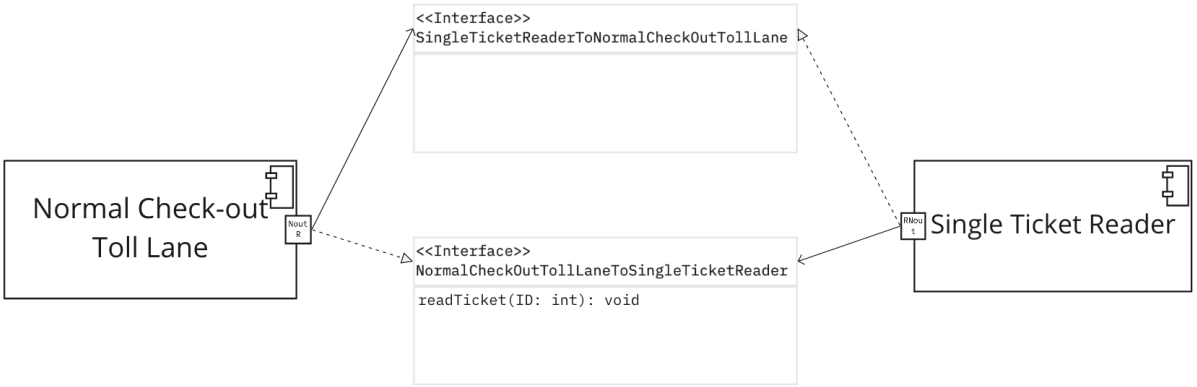


Figure 15: Component Diagram with interfaces: Normal Check-out Toll Lane - Single Ticket Reader

4.2.12 Component Diagram with interfaces: Express Toll Lane - Station Server

Author: Anton and Paweł

Figure 16: Component Diagram with interfaces: Express Toll Lane - Station Server

4.2.13 Component Diagram with interfaces: Express Toll Lane - Antenna

Author: Christopher and Hanna

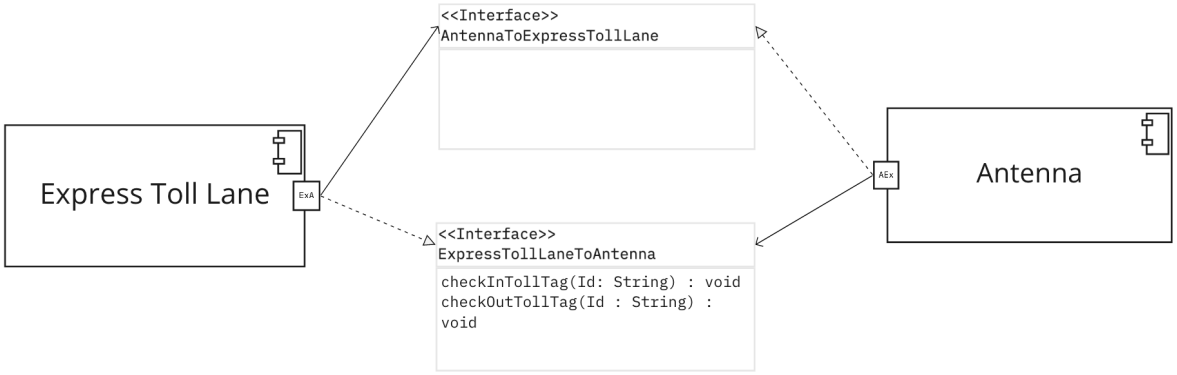


Figure 17: Component Diagram with interfaces: Express Toll Lane - Antenna

4.2.14 Component Diagram with interfaces: Express Toll Lane - Button

Author: Andreas and Eydís

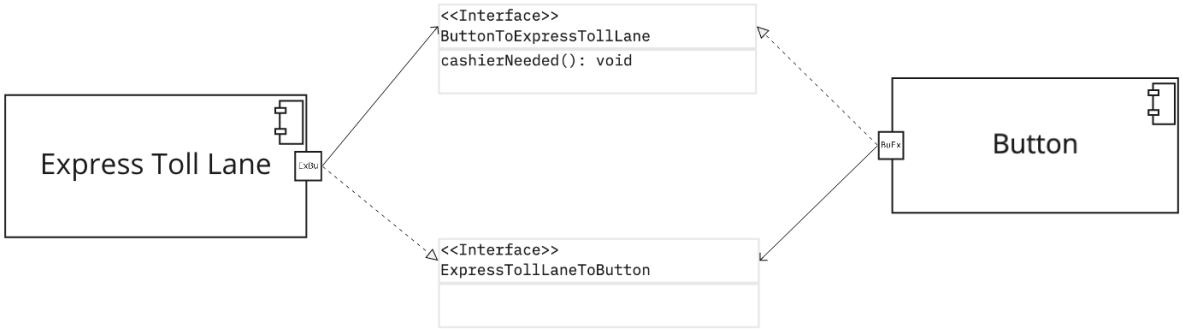


Figure 18: Component Diagram with interfaces: Express Toll Lane - Button

4.2.15 Component Diagram with interfaces: Lane - Barrier

Author: Bryndís and Anton

Figure 19: Component Diagram with interfaces: Lane - Barrier

4.3 Class Design

This section shows the different class diagrams for each component including the interfaces. Also shown are the relevant OCL constraints for specific functions in the classes.

4.3.1 Class Diagram: Enterprise Server

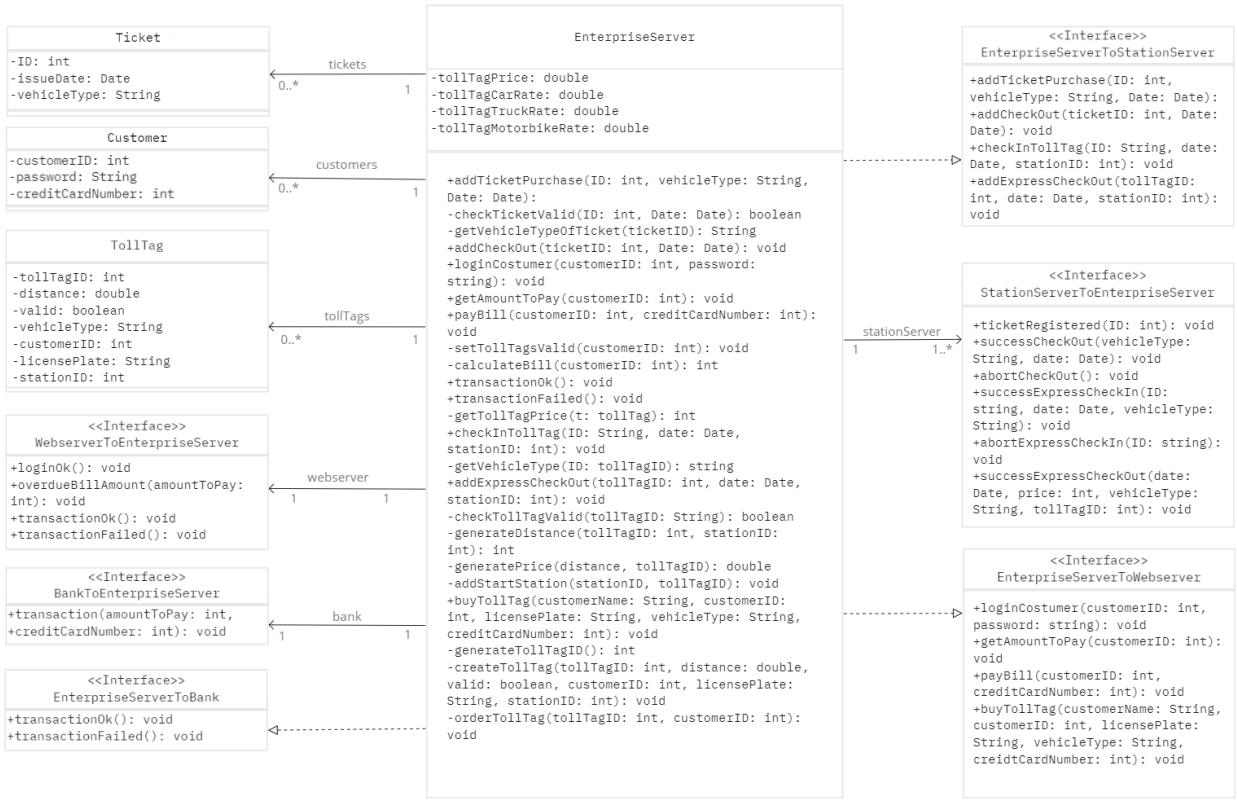


Figure 20: Class Diagram for enterpriseServer

Author: Andreas and Anton
The enterprise server stores all the main data for the whole toll lane system. It allows station servers to request validity checks of toll tags and tickets, as well as creation of both kinds of objects. The enterprise server is also involved when ordering a toll tag, as well as in paying overdue bills.

4.3.2 OCL Constraints for Enterprise Server

```
{ context EnterpriseServer :: addStartStation(stationID: int,
tollTagID int): void

pre: (tollTags -> exists(x | x.tollTagID = tollTagID))->size() = 1

post: tollTags -> exists(x | x.stationID = stationID & x.tollTagID
= tollTagID)->size() = 1
 }
```

Figure 21: OCL Constraint for addStartStation in Enterprise Server. This OCL constraint ensures that when a toll tag is checked-in, this toll tag will have its stationID-variable, denoting the starting station of the trip, set to that specific stationID, as to calculate the distance between the check-in station and the check-out station.

Author: Christopjer and Bryndís

```
{ context EnterpriseServer :: addTicketPurchase(ID: int, vehicleType:
String, Date: Date):

pre: (ID >= 0) and ((vehicleType = "Car") or (vehicleType = "Truck") or
(vehicleType = "Motorbike"))

post: t.oclIsNew() and t.oclIsTypeOf(ticket) and tickets -> includes(t)}
```

Figure 22: OCL Constraint for addTicketPurchase in Enterprise Server. Ensures that a the vehicleType of a ticket can only be one of the three possible vehicleTypes, and that a ticket then will be created and added to ticket database/collection.

Author: Christopher and Anton

```
{ context EnterpriseServer :: calculateBill(customerID: int): int

pre: customers -> exists(x | x.customerID = customerID) and
tollTags -> exists(x | x.customerID = customerID)

post: result = ((tollTags -> select(x | x.customerID =
customerID)) -> collect(x | x.distance *
getTollTagPrice(x)))->sum()
 }
```

Figure 23: OCL Constraint for calculateBill in Enterprise Server. If there exists a customer with that customerID and has a toll registered, then the function will output the price needed to be paid for all those toll tags.

Author: Christopher and Eydís

```
{ context EnterpriseServer :: checkTicketValid(ID: int, Date: Date): boolean

pre: (tickets -> exists( x | x.ID = ID))

post:(tickets -> exists( x | (x.ID = ID) and (x.valid = false))) }
```

Figure 24: OCL Constraint for checkTicketValid in Enterprise Server. If there exists a ticket with that ticketID, it is set to false. This is because this function is only called at a normal check-out case. If the ticket is already invalid, nothing is changed.

Author: Eydís and Andreas

```
{ context EnterpriseServer :: createTollTag(tollTagID: int, distance:
double, valid: boolean, vehicleType: String, customerID: int,
licensePlate: String, stationID: int ): void

pre: (tollTags -> forAll( x | x.tollTagID <> tollTagID)) and distance =
0.0 and valid = true
and ((vehicleType = 'Car') or (vehicleType = 'motorBike') or (vehicleType
= 'Truck'))
and (customers -> exists( x | x.customerID = customerID))
and (licensePlate <> null) and (stationID = 0)

post: tollTags -> exists( x | x.customerID = customerID and x.tollTagID =
tollTagID) }
```

Figure 25: OCL Constraint for createTollTag in Enterprise Server. Creates a tolltag with those parameters and assigns it to the customer by settig its customerID-variable.

Author: Christopher and Hanna

```
{ context EnterpriseServer :: generateDistance(tollTagID int,
stationID: int): int

pre: tollTags -> exists(x | x.stationID <> 0 and x.stationID <>
stationID and x.tollTagID = tollTagID)

post: tollTags -> exists(x | x.stationID = 0 and x.tollTagID =
tollTagID) and result > 0 }
```

Figure 26: OCL Constraint for generateDistance in Enterprise Server. If the given tolltag with that tolltagID has a its stationID set, then the distance between that stationID and the stationID parameter is calculated. The stationID variable is afterwards reset.

Author: Andreas and Hanna

```
{ context EnterpriseServer :: generatePrice(distance int,tollTagID
int): int

pre: tollTags -> exists(x | x.tollTagID = tollTagID)

post: result = distance * getTollTagPrice(tollTagID)
 }
```

Figure 27: OCL Constraint for generatePrice in Enterprise Server. When given a valid toll tag, calculates the price to be paid for that toll tag based on the inputted distance

Author: Hanna and Christopher

```
{ context EnterpriseServer :: generateTollTagID(): int

pre: true

post: tollTags -> forAll( x | x.tollTagID <> result) }
```

Figure 28: OCL Constraint for generateTollTagID in Enterprise Server. Generates a unique tollTagID that is not already used by a registered toll tag.

Author: Anton and Bryndís

```
{ context EnterpriseServer :: getTollTagPrice(t: tollTag): int

pre: (t.vehicleType = "Car") or (t.vehicleType = "Motorbike") or
(t.vehicleType = "Truck")

derive: if t.vehicleType = "Car" then carTollTagRate
     else if t.vehicleTupe = "Truck" then truckTollTagRate
     else motorBikeTollTagRate endif
     endif

post: result > 0 }
```

Figure 29: OCL Constraint for getTollTagPrice in Enterprise Server. Given a toll tag with a correct vehicleType, generate the price of that toll tag based on the vehicleType and the rate for that vehicleType.

Author: Eydís and Paweł

```
{ context EnterpriseServer :: getVehicleType(tollTagID: int): String

pre: (tollTags -> exists( x | x.tollTagID = tollTagID))->sum() = 1

post: (tollTags -> exists( x | x.tollTagID = tollTagID and x.vehicleType =
result)) }
```

Figure 30: OCL Constraint for getVehicleType in Enterprise Server. Given an existing tollTag, returns the vehicleType of that toll tag

Author: Andreas and Christopher

```
{ context EnterpriseServer :: setTollTagsValid(customerID: int): void

pre: ((customers -> exists( x | x.customerID = customerID))->sum() = 1) and
((tollTags -> exists( x | x.customerID = customerID))->sum() > 0)

post:((tollTags -> forall( x | x.customerID)) -> forAll( x | x.
}
```

Figure 31: OCL Constraint for setTollTagValid in Enterprise Server. Given a customerID that exists and is associated with at least one toll tag, set all toll tags of that customer as valid. Is used after the customer has paid his/her overdue bill.

Author: Hanna and Bryndís

4.3.3 Class Diagram: Station Server

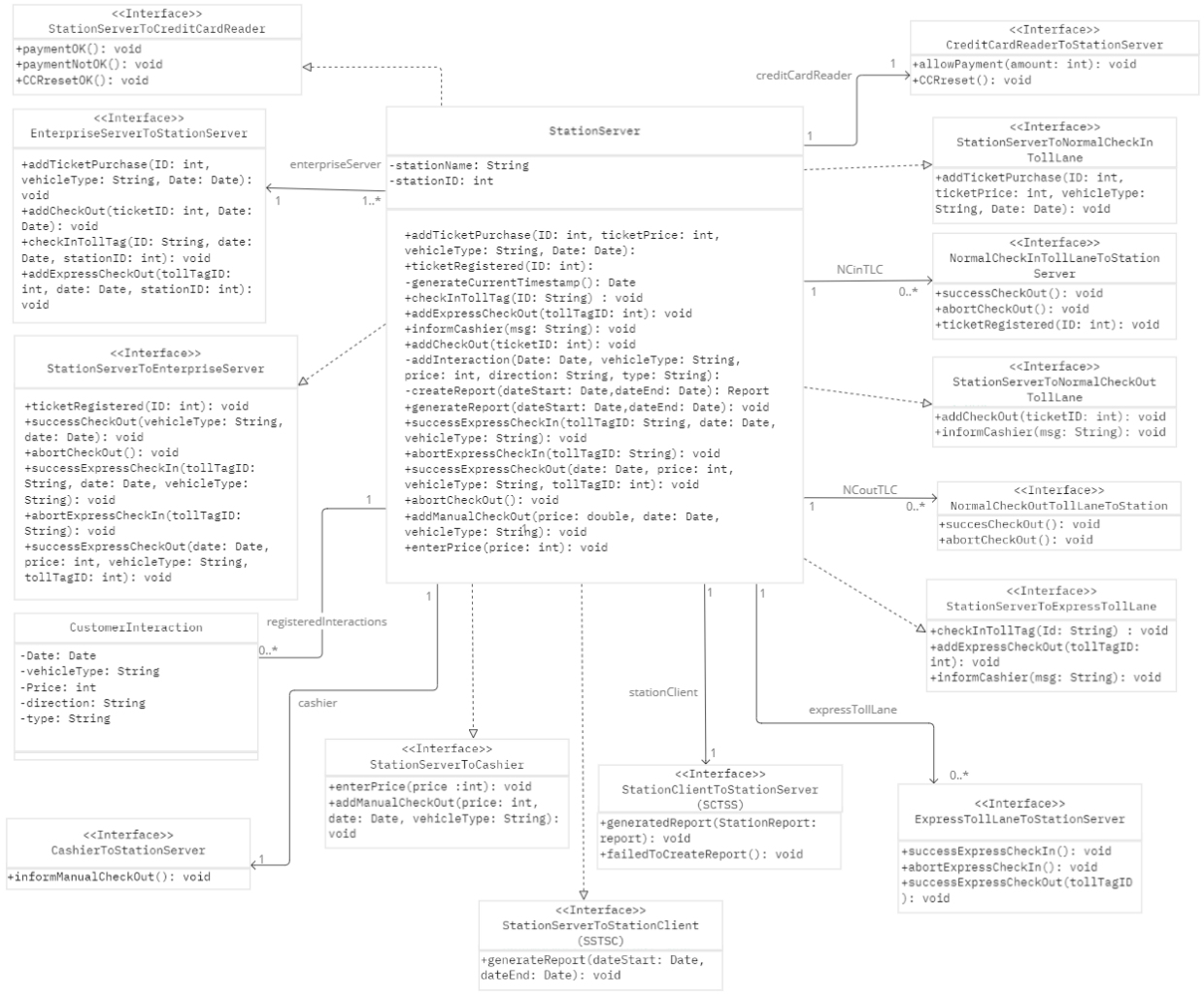


Figure 32: Class Diagram for stationServer

Author: Christopher and Eydís

The station server relays information between the toll lane computers and the enterprise server. It notes down statistical data for the reports, and can also be used to perform manual check-outs via the cashier.

4.3.4 OCL Constraints for Station Server functions

```
{ context StationServer
:: addInteraction(Date: Date, vehicleType: String, price: int, direction:
String, type: String):

pre: ((vehicleType = "truck") or (vehicleType = "car") or (vehicleType =
"motorbike"))  and
((direction = "check-in") and (type = "singleTicket") and (price>0)) or
((direction = "check-out") and (type = "singleTicket") and (price = 0))
or ((direction = "check-in") and (type = "tollTag") and (price = 0))
or ((direction = "check-out") and (type = "tollTag") and (price >= 0))

post: interaction.oclIsNew() and interaction.oclIsTypeOf(customerInteraction)
and registeredInteractions -> includes(interaction) }
```

Figure 33: OCL Constraint for addInteraction in stationServer. Given an correct combination of parameters, adds that customerInteraction to the statistical database. Note that a ticket check-out always has a price of 0 as it is paid on check in, while the toll tag check-in always has a price 0 as it is "paid"(=registered) at check-out.

Author: Anton and Paweł

```
{ context stationServer :: createReport(dateStart: Date, dateEnd: Date): Report

pre: (startDate < endDate) and (registeredInteractions.notEmpty())

post: result = report and report.oclIsNew() and report.oclIsOfType(Report)

and report.moneyGenerated = (customerInteractions -> select( x | x.Price>0) and
(startDate<x.Date<endDate)))->sum()

and report.carCheckIn = (customerInteractions -> select( x | (x.vehicleType =
"car") and (x.direction = "check-in")and (startDate<x.Date<endDate)))->sum()

and report.motorbikeCheckIn = (customerInteractions -> select( x | (x.vehicleType
= "motorbike") and (x.direction = "check-in")and
(startDate<x.Date<endDate)))->sum())

and report.truckCheckIn = (customerInteractions -> select( x | (x.vehicleType =
"truck") and (x.direction = "check-in")and (startDate<x.Date<endDate)))->sum()

and report.carCheckOut = (customerInteractions -> select( x | (x.vehicleType =
"car") and (x.direction = "check-out")and (startDate<x.Date<endDate)))->sum()

and report.motorbikeCheckOut = (customerInteractions -> select( x | (x.vehicleType
= "motorbike") and (x.direction = "check-out")and
(startDate<x.Date<endDate)))->sum()

and report.truckCheckOut = (customerInteractions -> select( x | (x.vehicleType =
"truck") and (x.direction = "check-out")and (startDate<x.Date<endDate)))->sum()

and report.singleTicketCheckIn = (customerInteractions -> select( x | (x.type =
"singleTicket") and (x.direction="check-in")and
(startDate<x.Date<endDate)))->sum()

and report.tollTagCheckIn = (customerInteractions -> select( x | (x.type =
"tollTag") and (x.direction="check-in")and (startDate<x.Date<endDate)))->sum()

and report.singleTicketCheckOut = (customerInteractions -> select( x | (x.type =
"singleTicket") and (x.direction="check-out")and
(startDate<x.Date<endDate)))->sum()

and report.tollTagCheckOut = (customerInteractions -> select( x | (x.type =
"tollTag") and (x.direction="check-out")and (startDate<x.Date<endDate)))->sum()

and report.manualCheckOut = (customerInteractions -> select( x | (x.type =
"Manual") and (x.direction="check-out")and (startDate<x.Date<endDate)))->sum() }
```

Figure 34: OCL Constraint for createReport in stationServer. Given the startdate is before the endDate, and there is data registered, then the report will be generated with the correct data within that time frame.

Author: Bryndís and Eydís

4.3.5 Class Diagram: Station Client

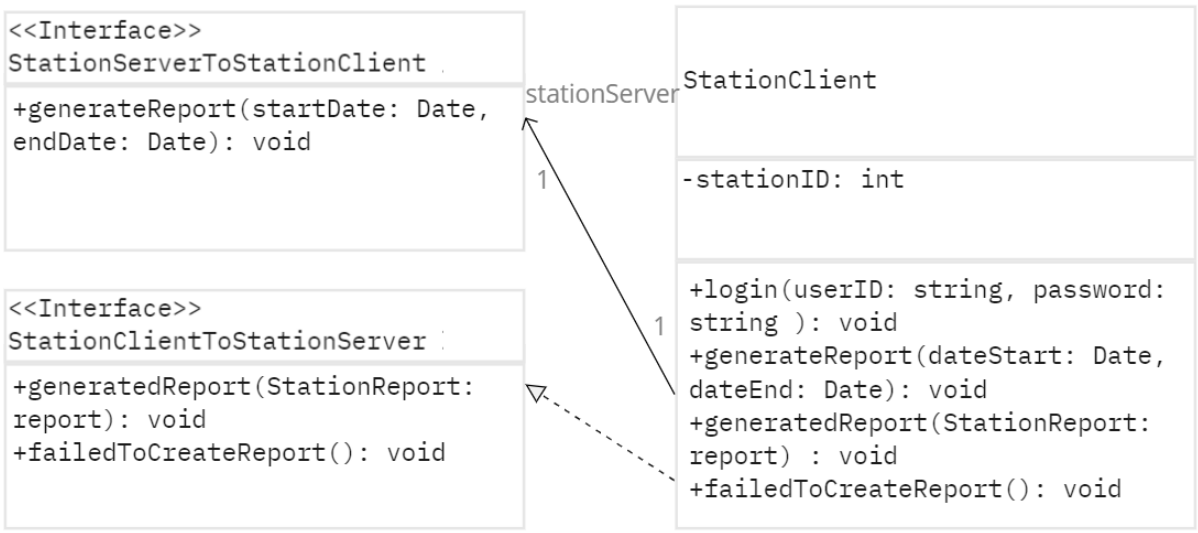


Figure 35: Class Diagram for stationClient

Author: Anton and Christopher

The station client is only used to relay request to generate reports from the station manager to the station server and receiving those reports.

4.3.6 Class Diagram: Normal Check-in Toll Lane

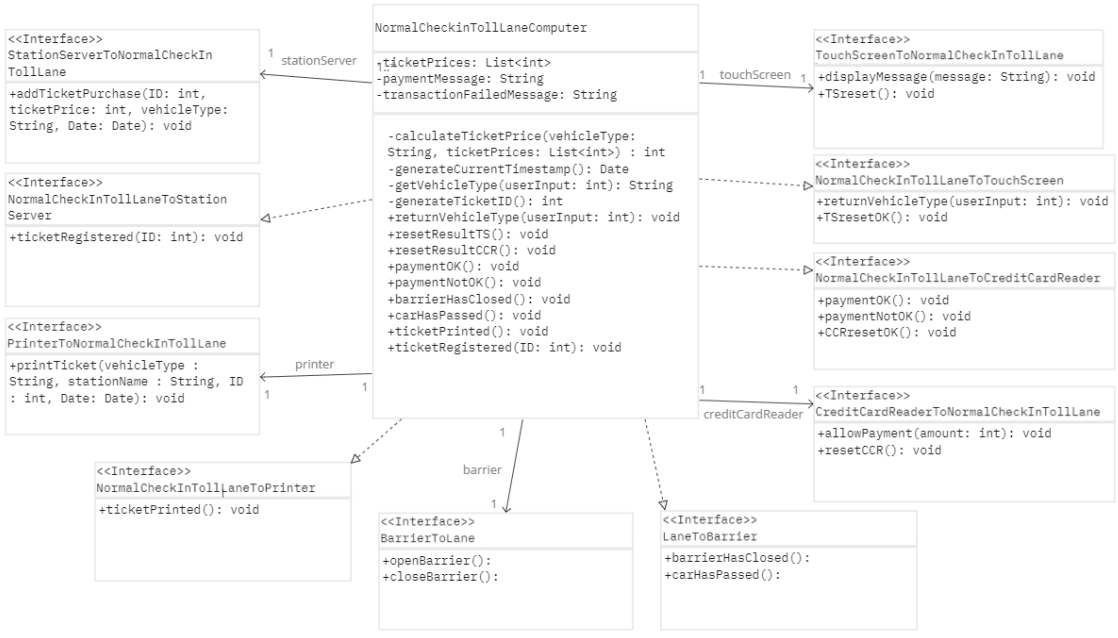


Figure 36: Class Diagram for Normal check in lane

Author: Hanna and Bryndís

The normal check-in toll lane is used to handle check-ins using single tickets.

4.3.7 OCL Constraints for Normal Check-in Toll Lane

```
{ context NormalCheckinTollLaneComputer :: calculateTicketPrice(vehicleType:
String, ticketPrices: List<int>) : int

pre: (collection{ticketPrices} -> forAll(x | x> 0)) and ((vehicleType="car")
or (vehicleType="truck") or (vehicleType="mortorbike"))

post: result > 0 }
```

Figure 37: OCL Constraint for calculateTicketPrice in Normal Check-in Toll Lane. When given a list of valid ticket prices and a valid vehicleType, return the correct ticketprice for that vehicletType

Author: Andreas and Hanna

4.3.8 Class Diagram: Normal Check-out Toll Lane

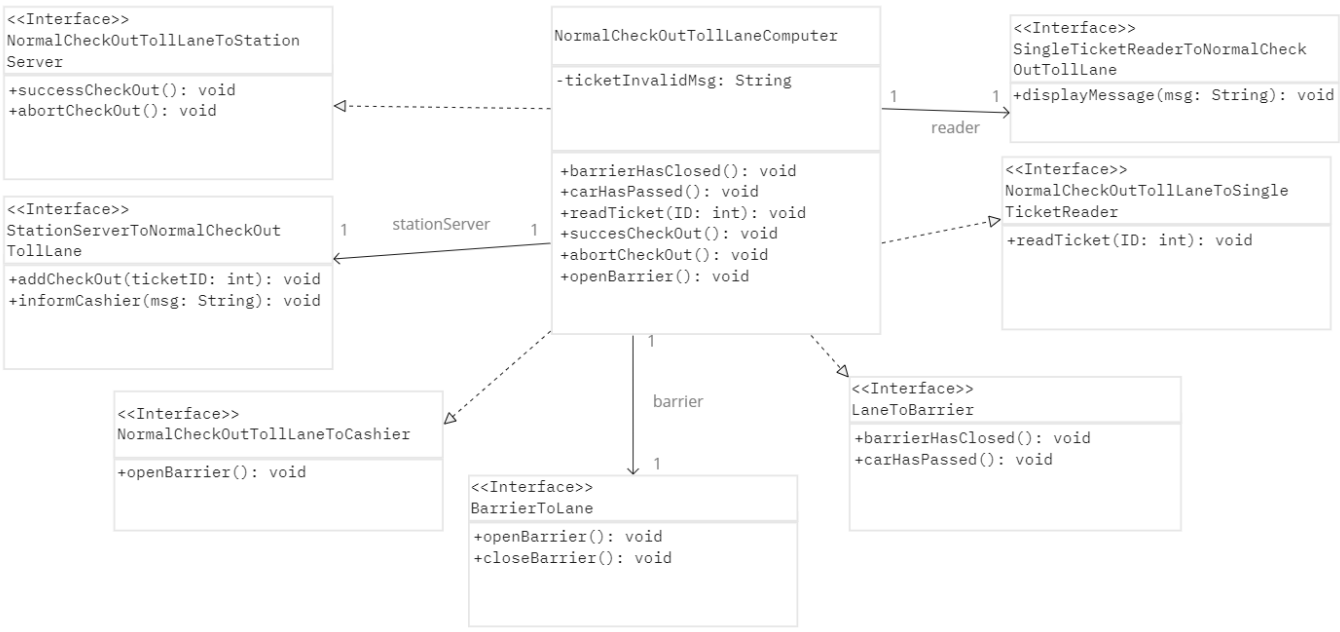


Figure 38: Class Diagram for Normal check out lane

Author: Hanna and Andreas

The normal check-out toll lane is used to handles check-out using single tickets, including checking if their are valid.

4.3.9 Class Diagram: Express Toll Lane

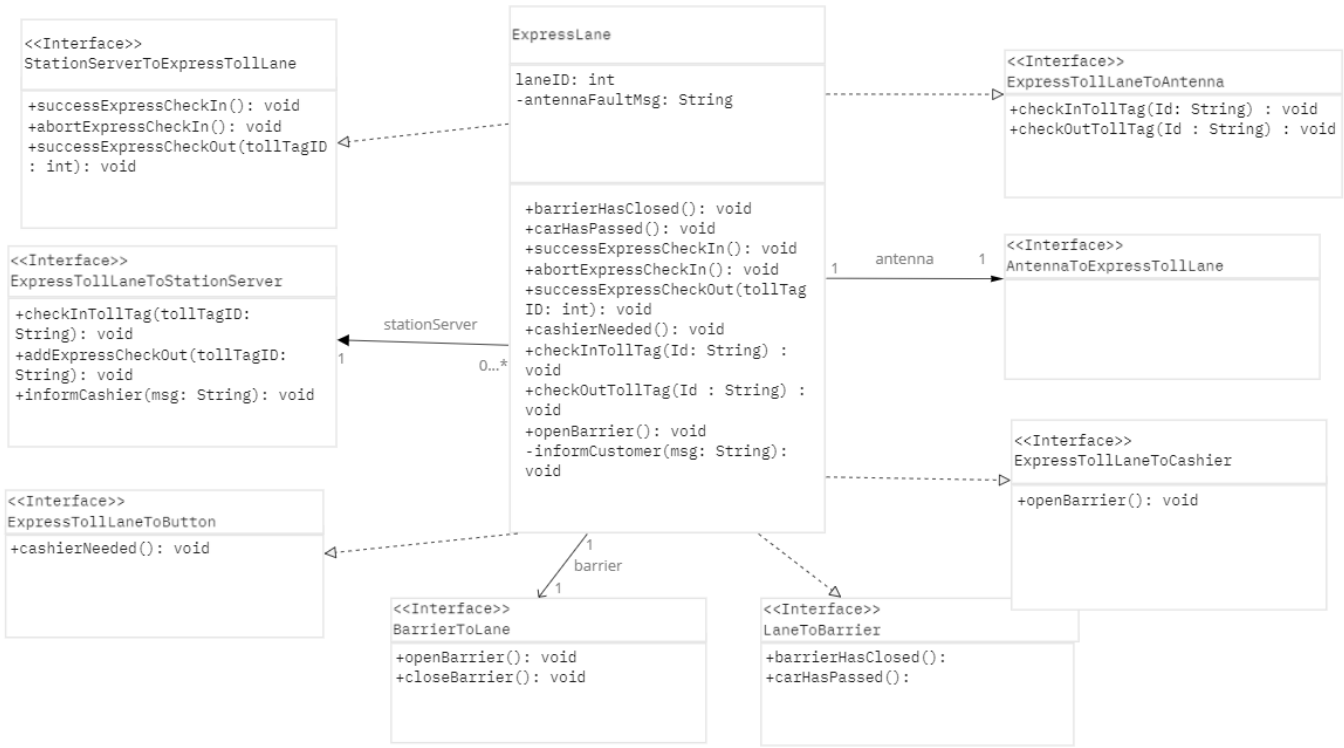


Figure 39: Class Diagram for Express TollLane

Author: Eydís and Paweł

Handles both check-ins and check-outs using toll tags.

4.3.10 Additional Class Diagrams

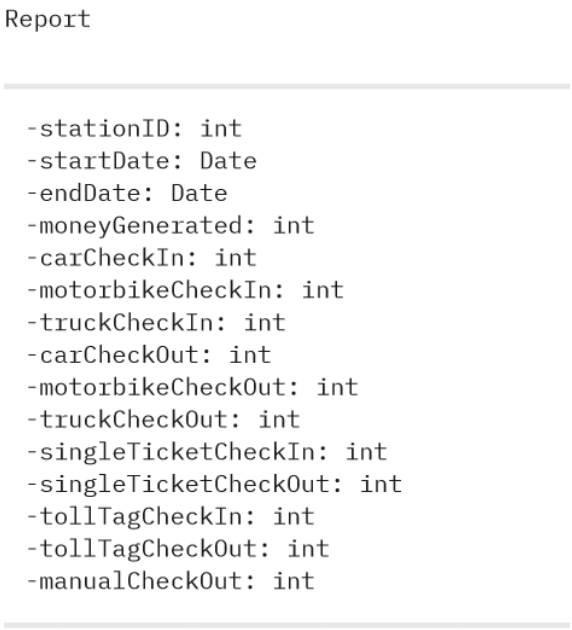


Figure 40: Class Diagram for Reports

Author: Bryndís and Paweł

Objects of the report class is not stored/associated with any other class, but is passed from the station server to the station client.

4.4 Behaviour Design

In this section, we provide the Life Cycle State Machines for each component we defined earlier in this chapter.

4.4.1 LSM: Enterprise Server

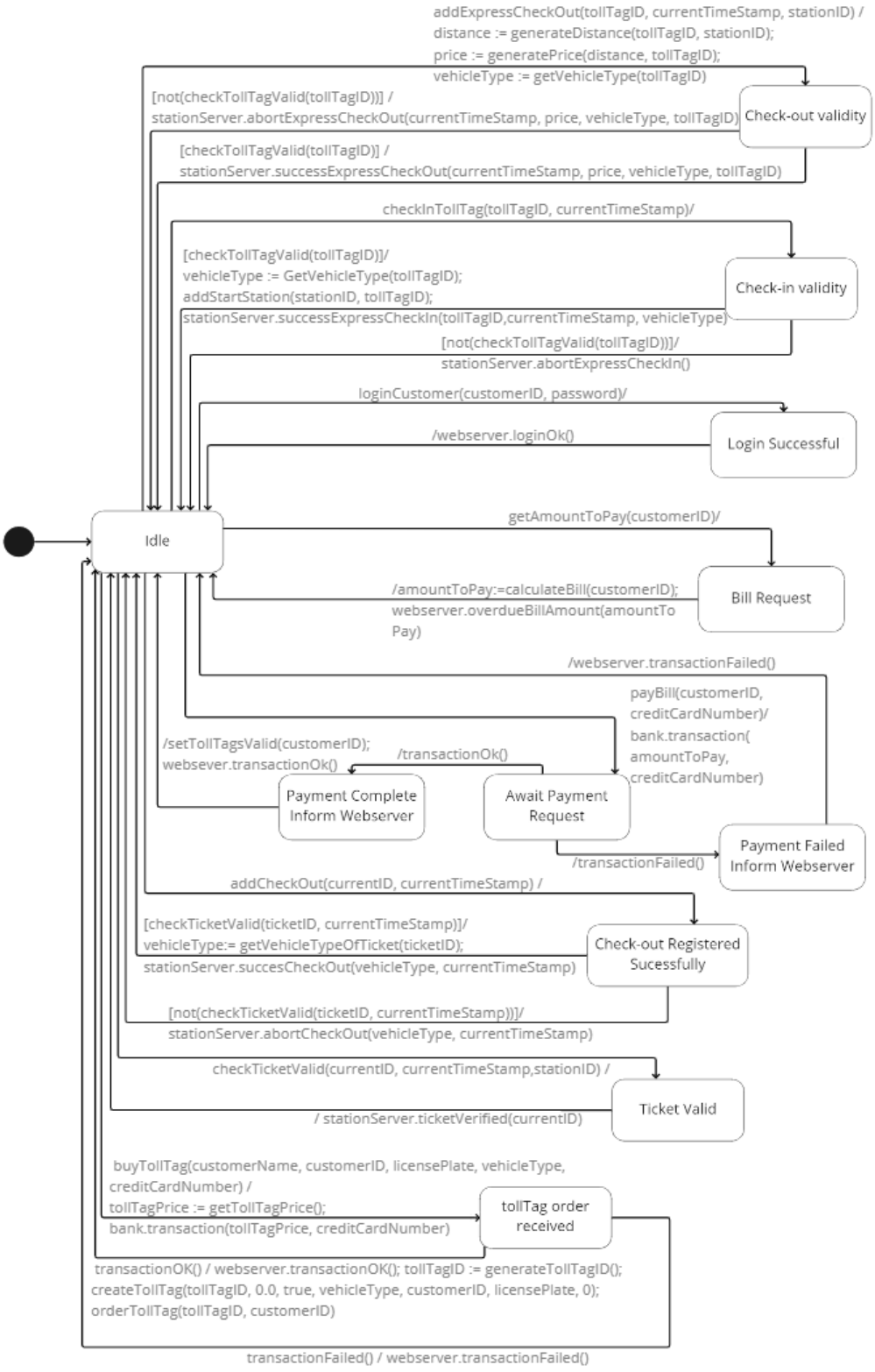


Figure 41: LSM for enterpriseServer

Author: Andreas And Anton

4.4.2 LSM: Station Server

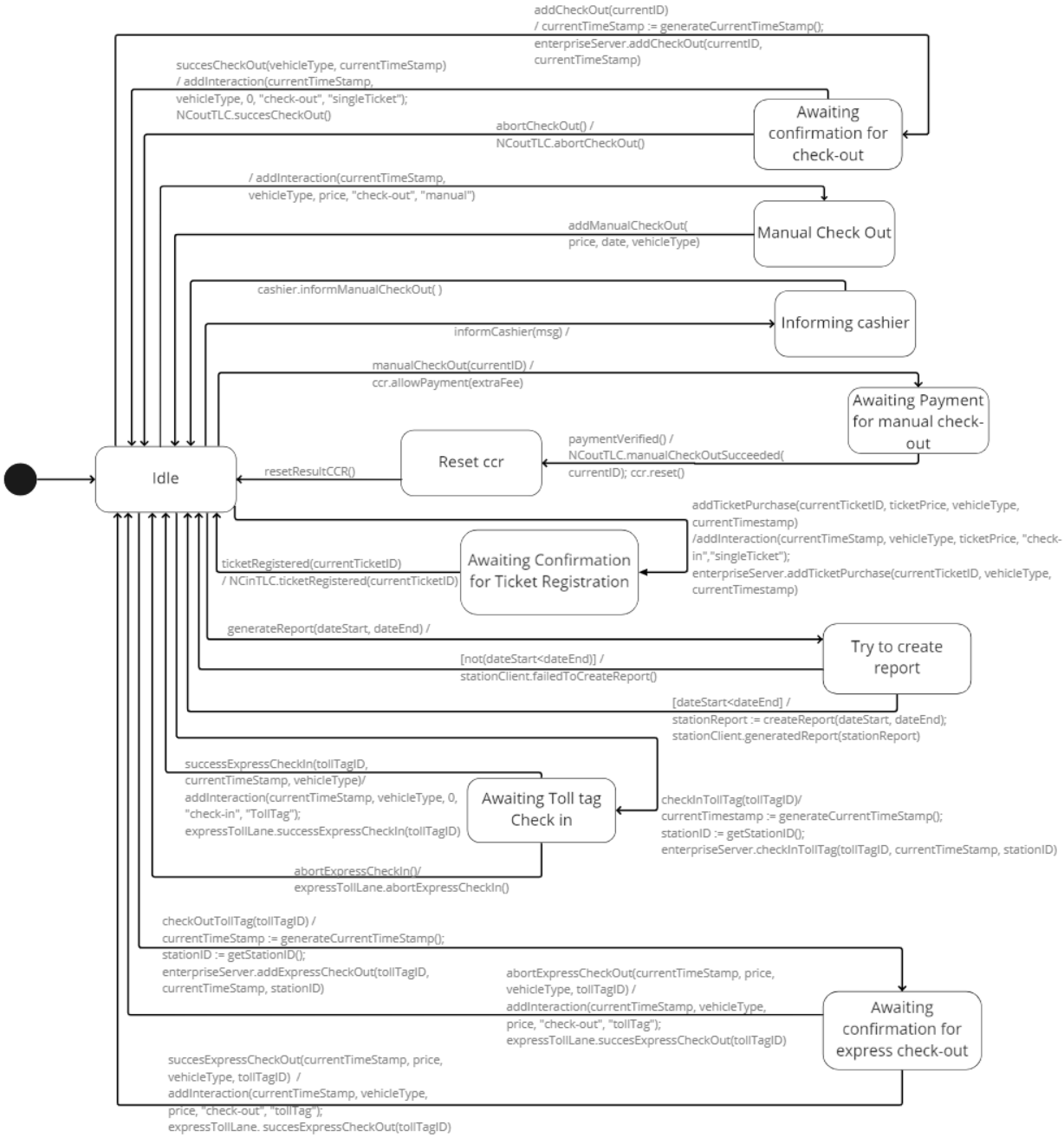


Figure 42: LSM for stationServer

Author: Christopher and Hanna

4.4.3 LSM: Station Client

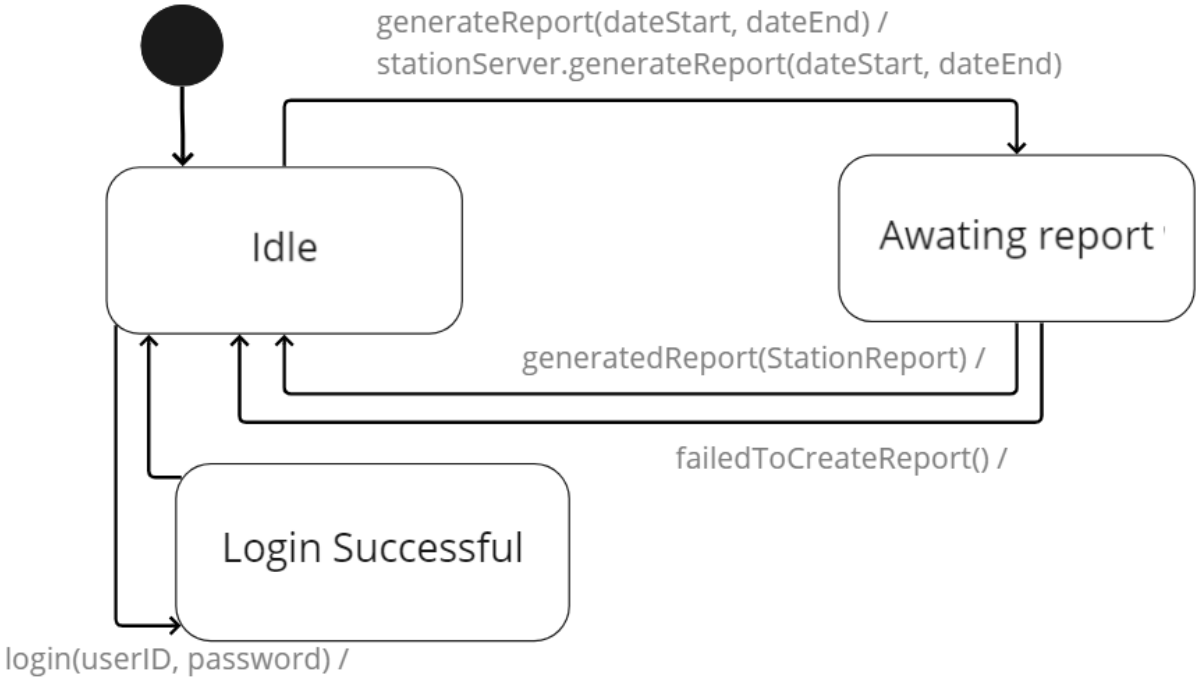


Figure 43: LSM for stationClient

Author: Hanna and Paweł

4.4.4 LSM: Normal Check-in Toll Lane

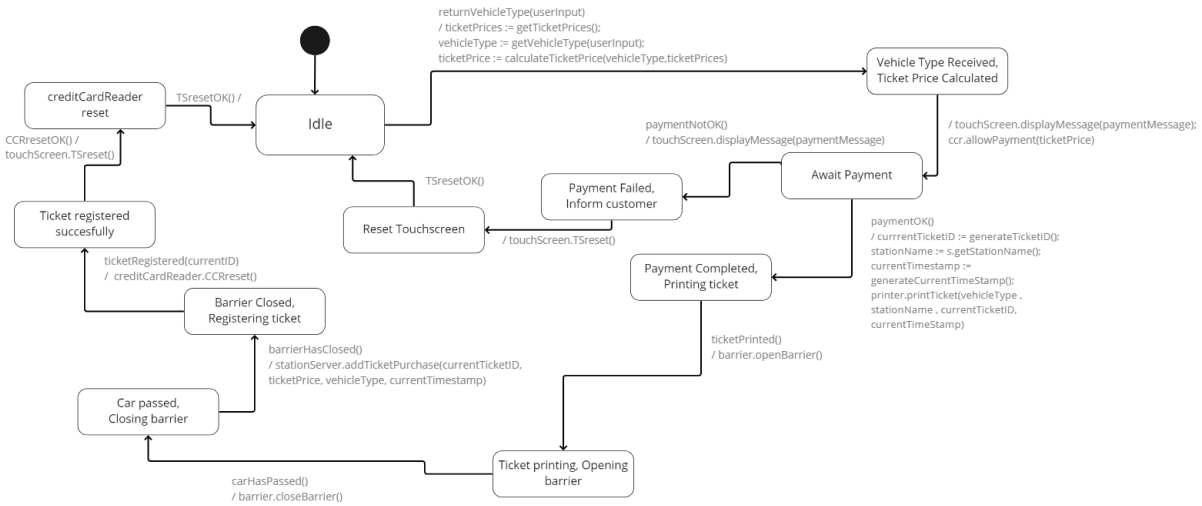


Figure 44: LSM for Normal Check-In Toll Lane

Author: Christopher and Andreas

4.4.5 LSM: Normal Check-out Toll Lane

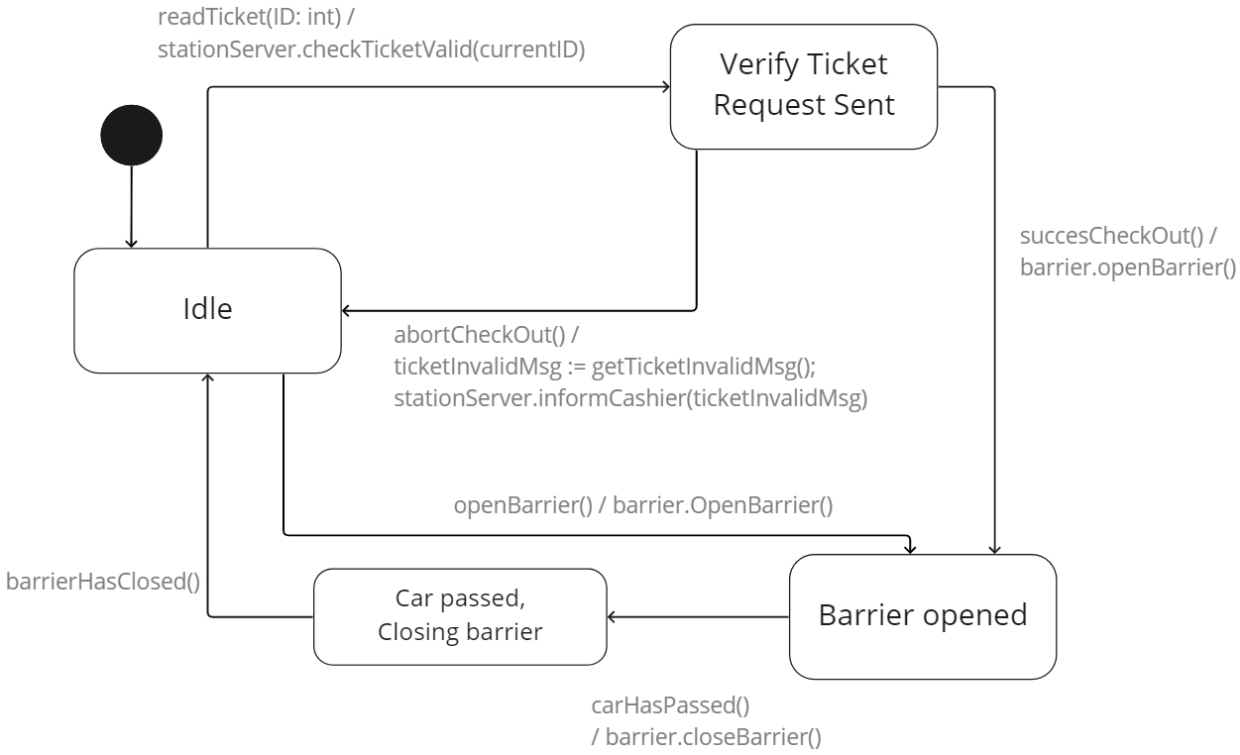


Figure 45: LSM for Normal Check-out Toll Lane

Author: Bryndís and Eydís

4.4.6 LSM: Express Toll Lane

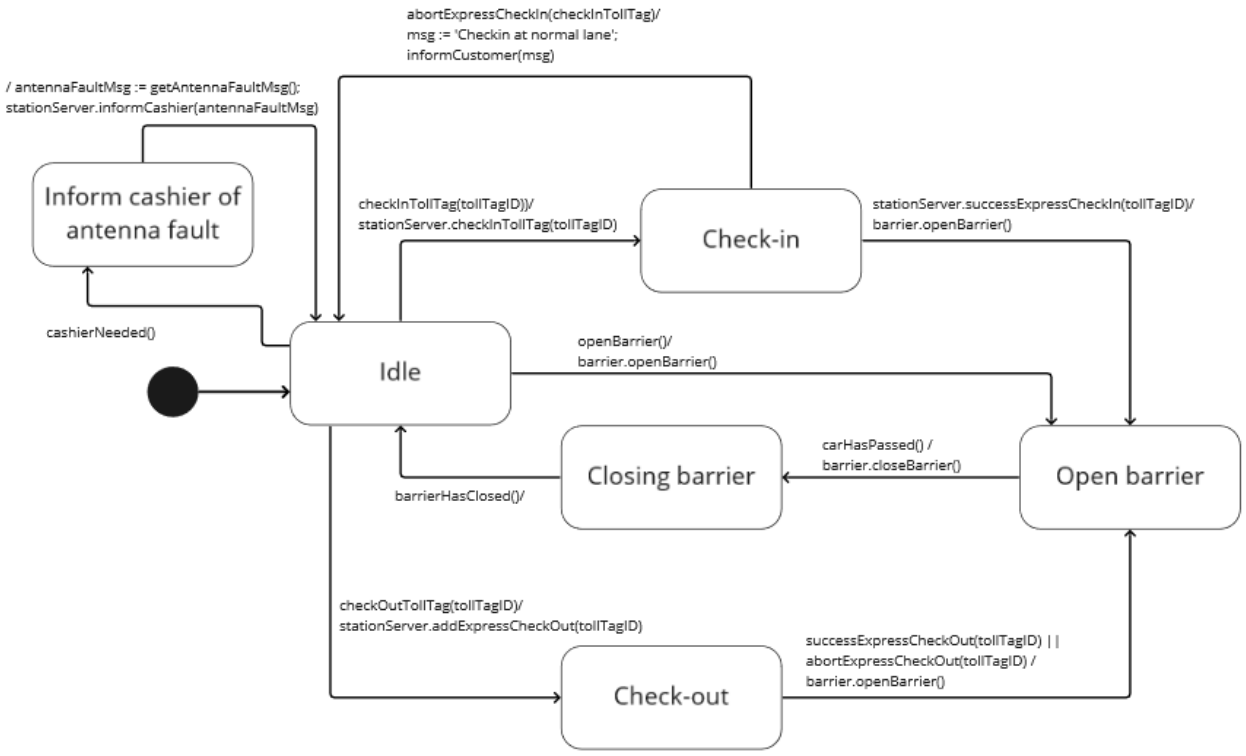


Figure 46: LSM for expressTollLane

Author: Eydís and Anton

5 Validation

5.1 Use Case Realisation

We decided to do asynchronous sequence diagrams since every station has only one station server but needs to communicate with all toll lane computer at the same time. In a synchronous way, two different customers would not be able to check out at the same time since the station server is occupied by one check out already.

5.1.1 Sequence Diagrams: Buy toll tag

Author: Christopher and Anton
The main scenario can be found in figure 47 and the alternative scenario in figure 48.

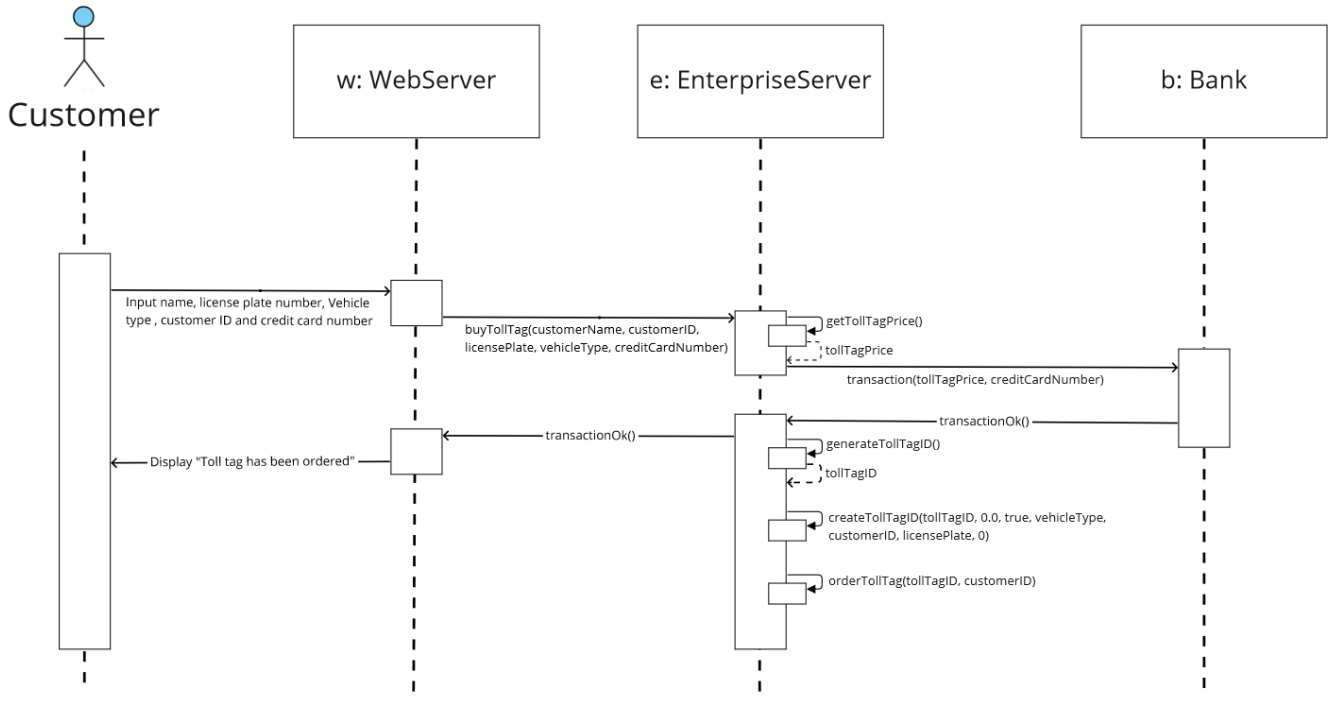


Figure 47: Sequence Diagram - Buy Toll Tag - Main Scenario

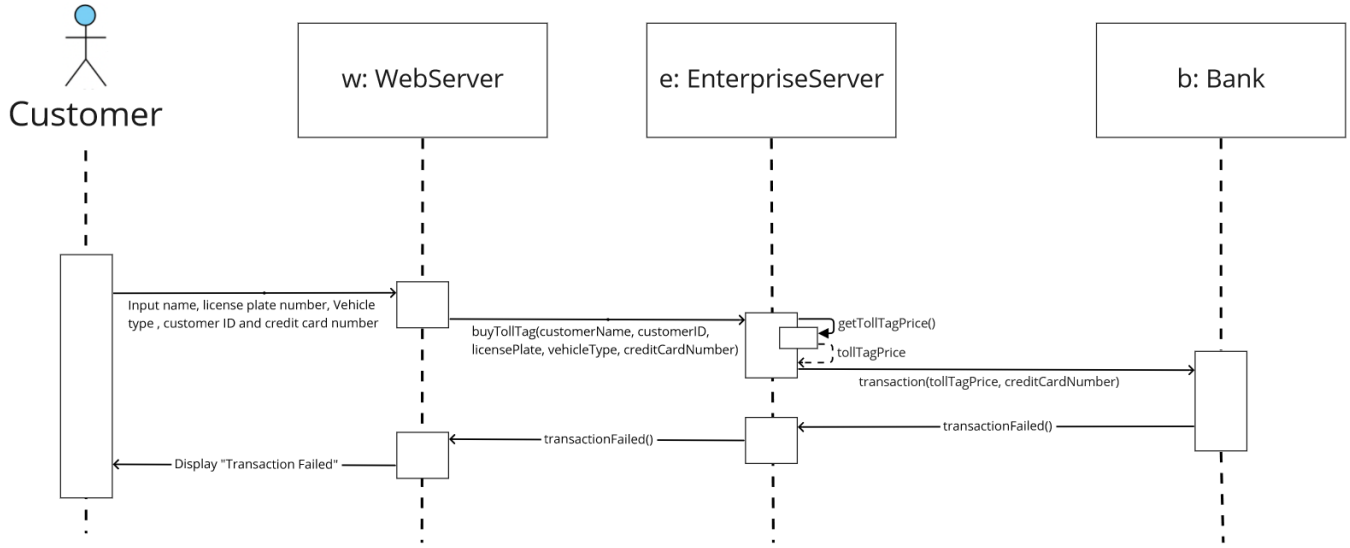


Figure 48: Sequence Diagram - Buy Toll Tag - Alternative (Fail) Scenario

5.1.2 Sequence Diagrams: Pay overdue bill

Author: Andreas and Hanna
The main scenario can be found in figure 49 and the alternative scenario in figure 50.

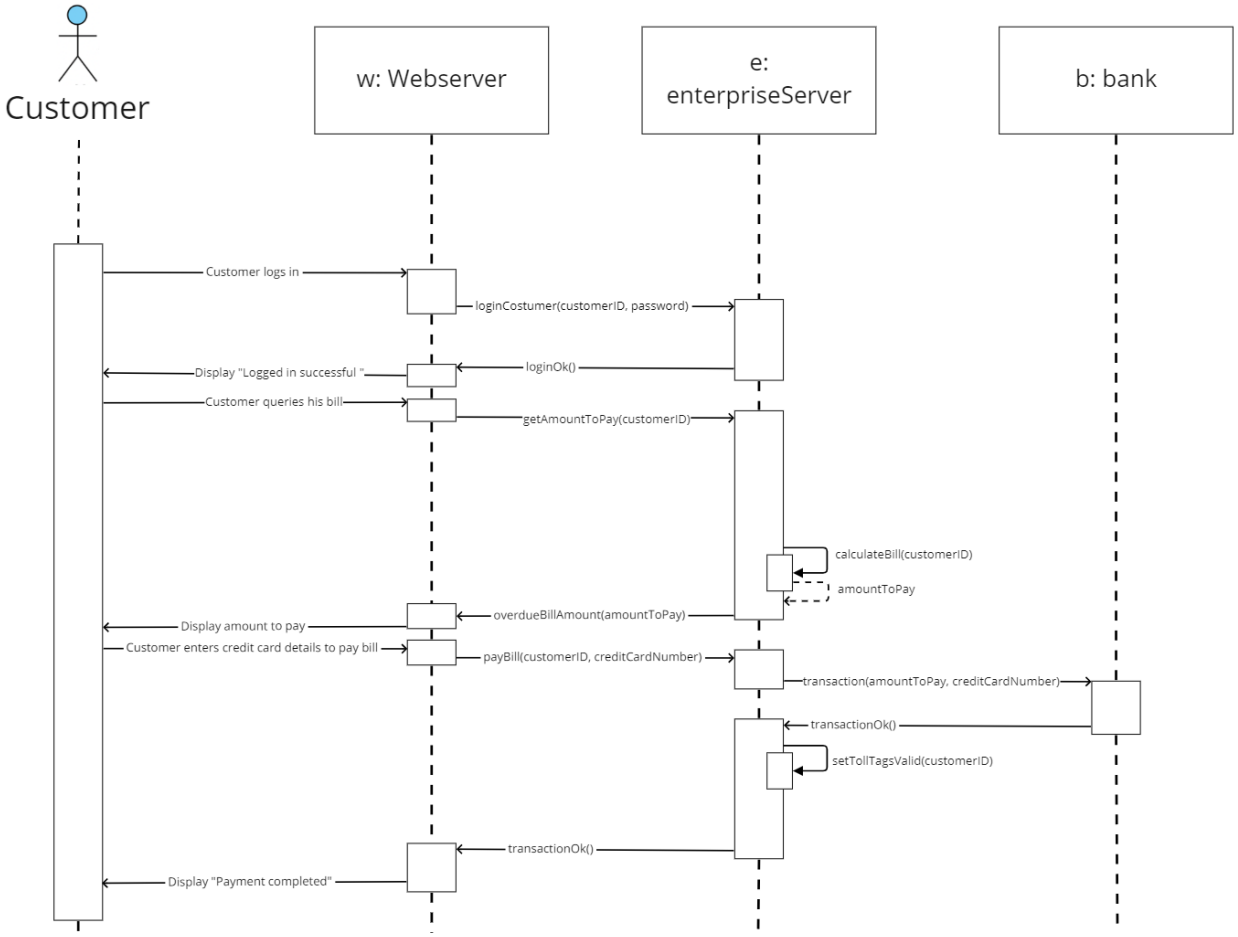


Figure 49: Sequence Diagram - Pay overdue bill - Main Scenario

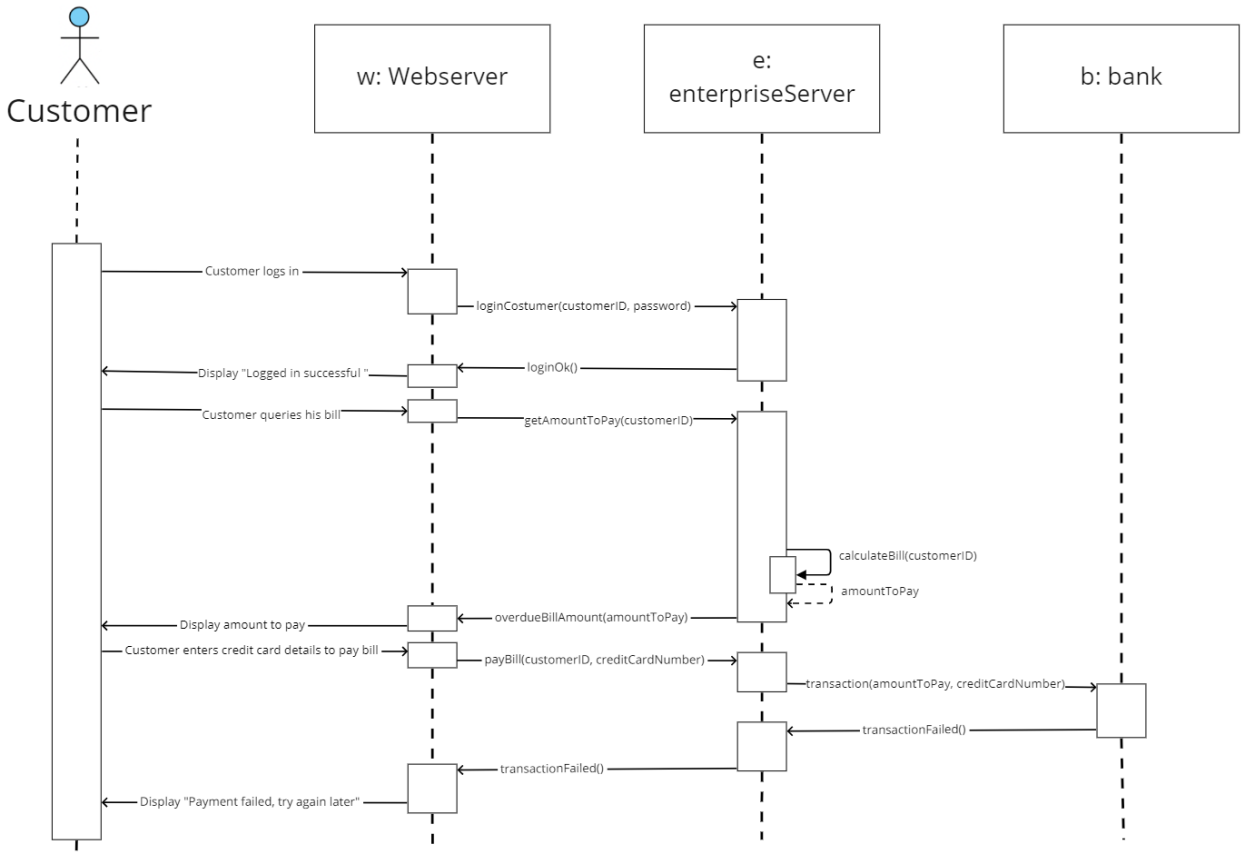


Figure 50: Sequence Diagram - Pay overdue bill - Alternative (Fail) Scenario

5.1.3 Sequence Diagrams: Check-in with single ticket

Author: Christopher and Andreas

The main scenario can be found in figure 51 and the alternative scenario in figure 52.

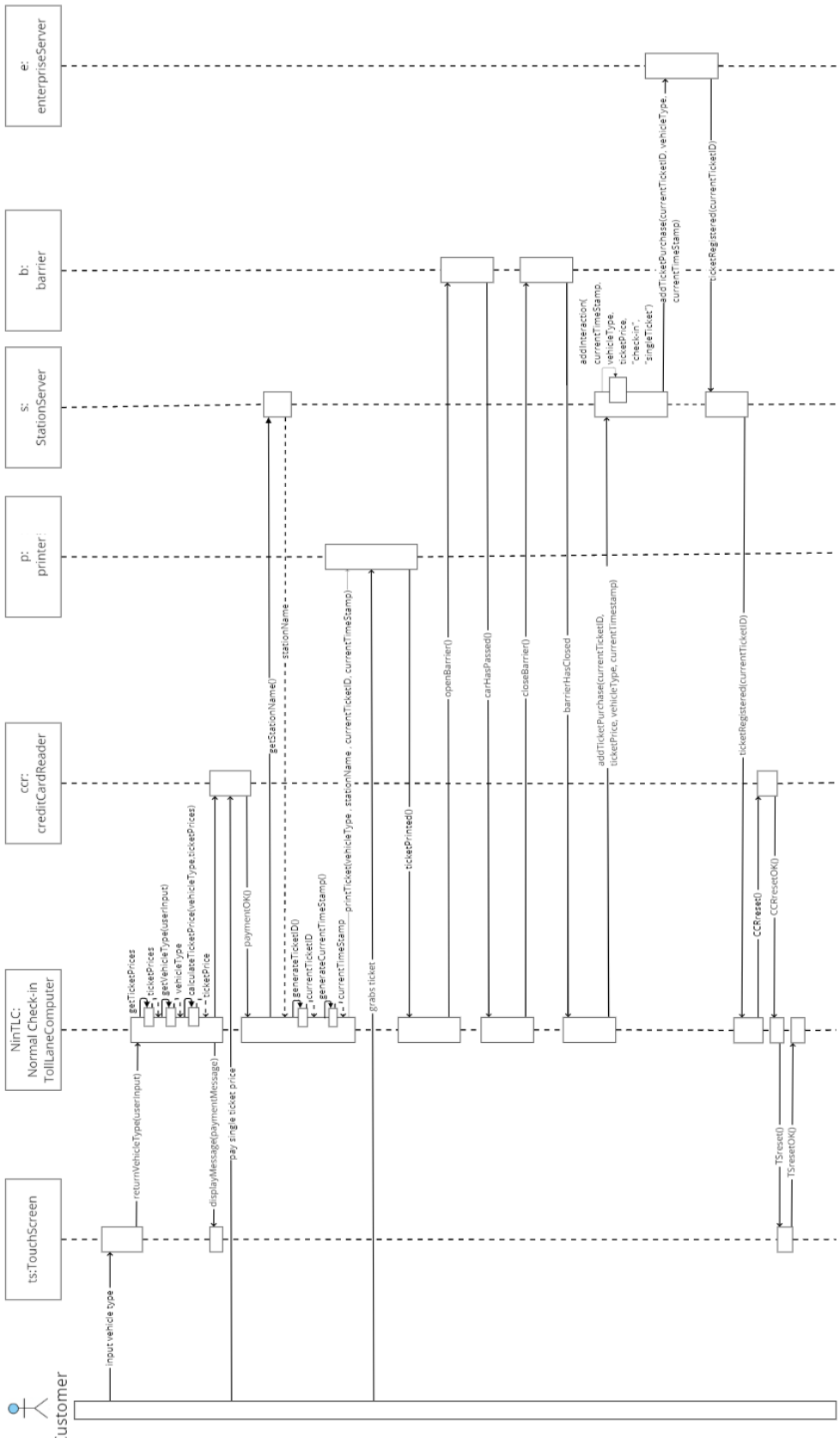


Figure 51: Sequence Diagram - Normal Check-in - Main Scenario

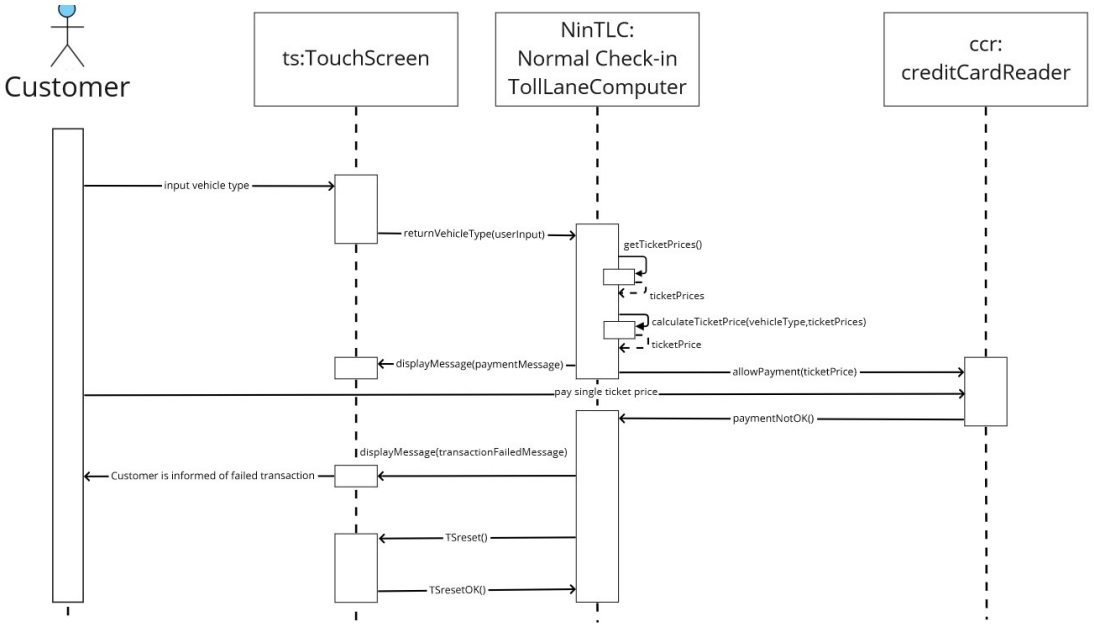


Figure 52: Sequence Diagram - Normal Check-in - Alternative (Fail) Scenario

5.1.4 Sequence Diagrams: Check-out with single ticket

Author: Bryndís and Eydís
The main scenario can be found in figure 53, second alternative scenario in figure 54 and the third alternative scenario in figure 55. The first alternative scenario was not modeled as a sequence diagram due to no direct input of an actor starting the action.

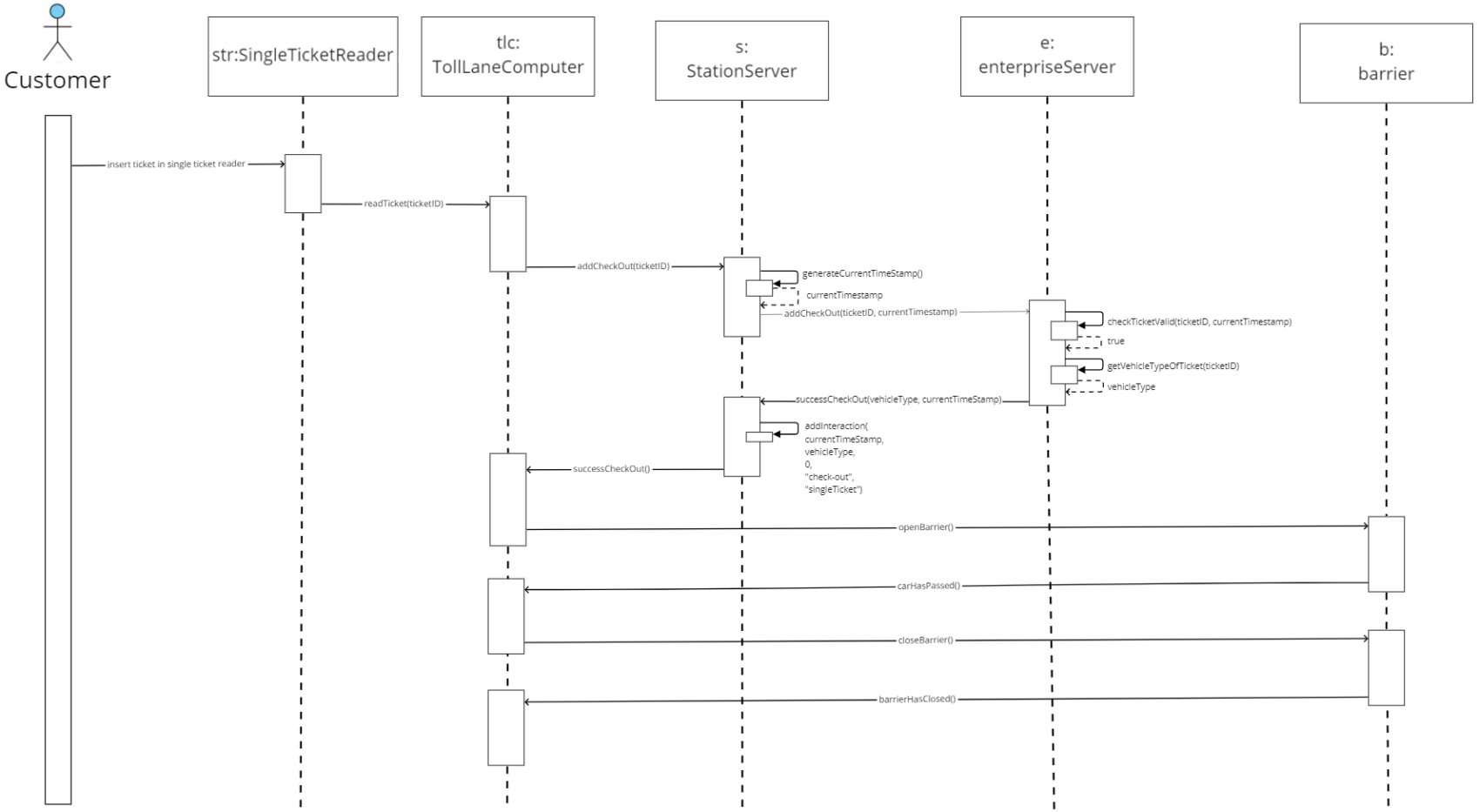


Figure 53: Sequence Diagram - Normal Check-out - Main Scenario

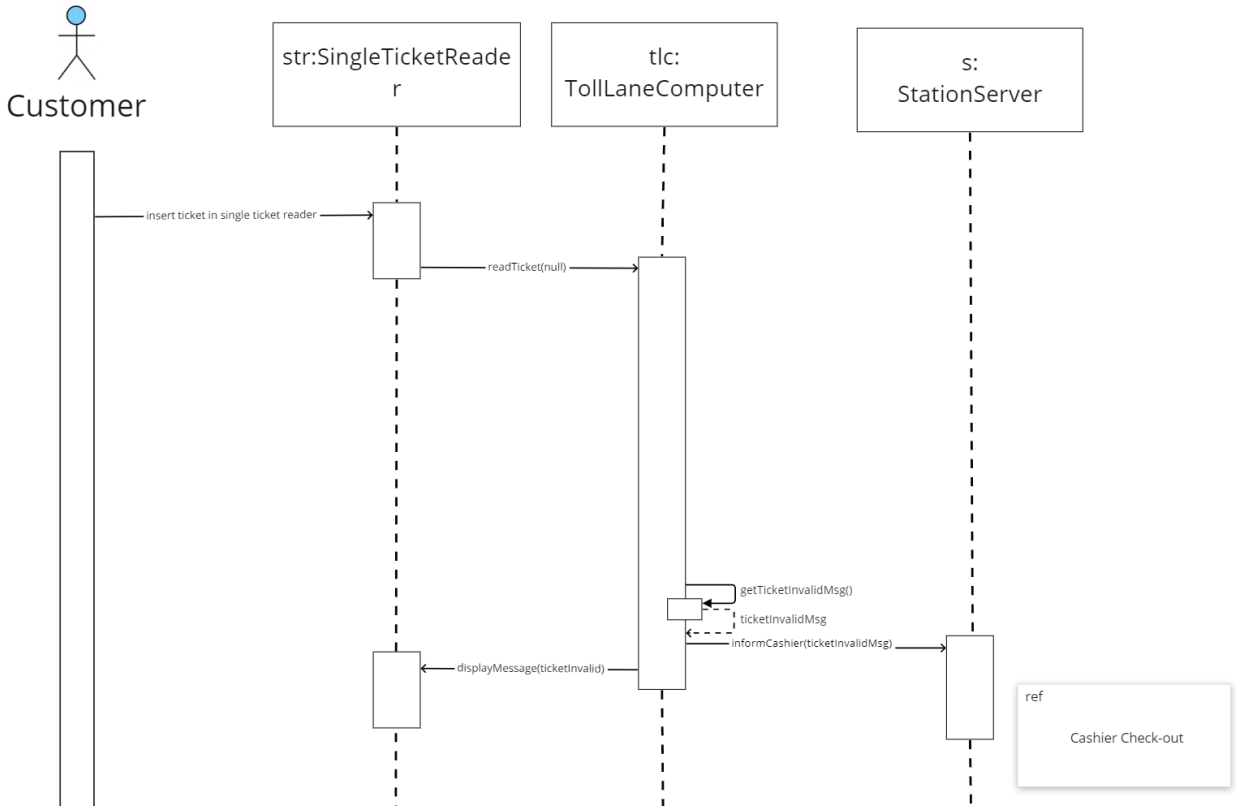


Figure 54: Sequence Diagram - Normal Check-out - Alternative Scenario 1 (Ticket cannot be read)

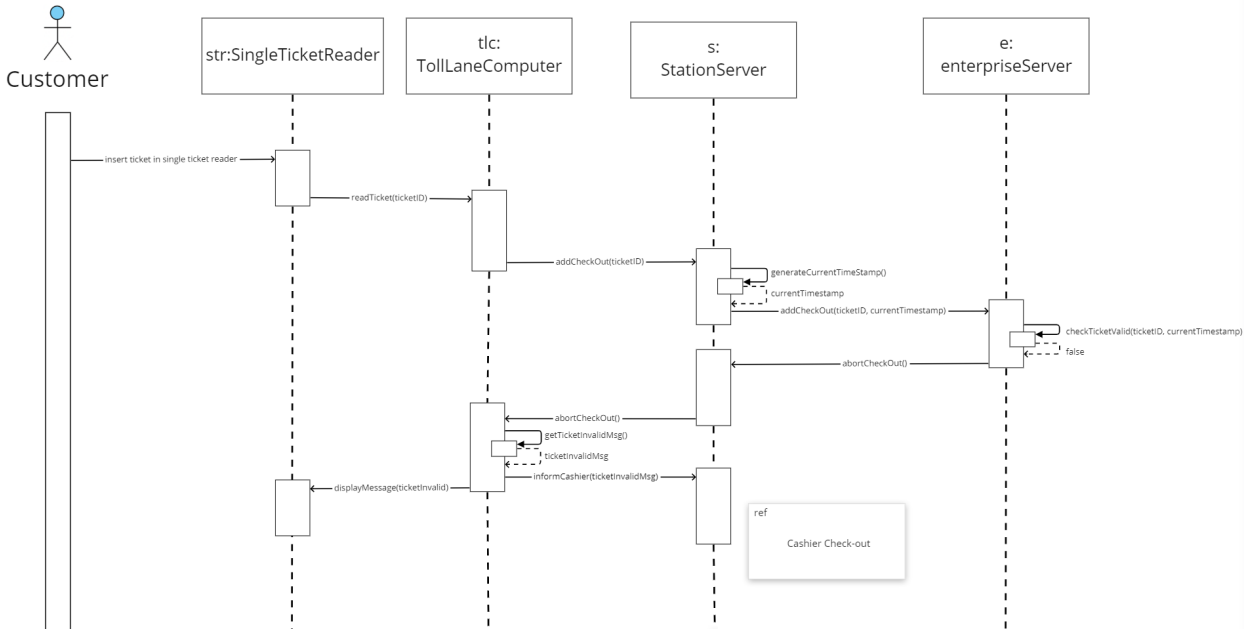


Figure 55: Sequence Diagram - Normal Check-out - Alternative Scenario 2 (Ticket is invalid)

5.1.5 Sequence Diagrams: Check-in with toll tag

Author: Anton and Bryndís
The main scenario can be found in figure 56, alternative scenario in figure 57

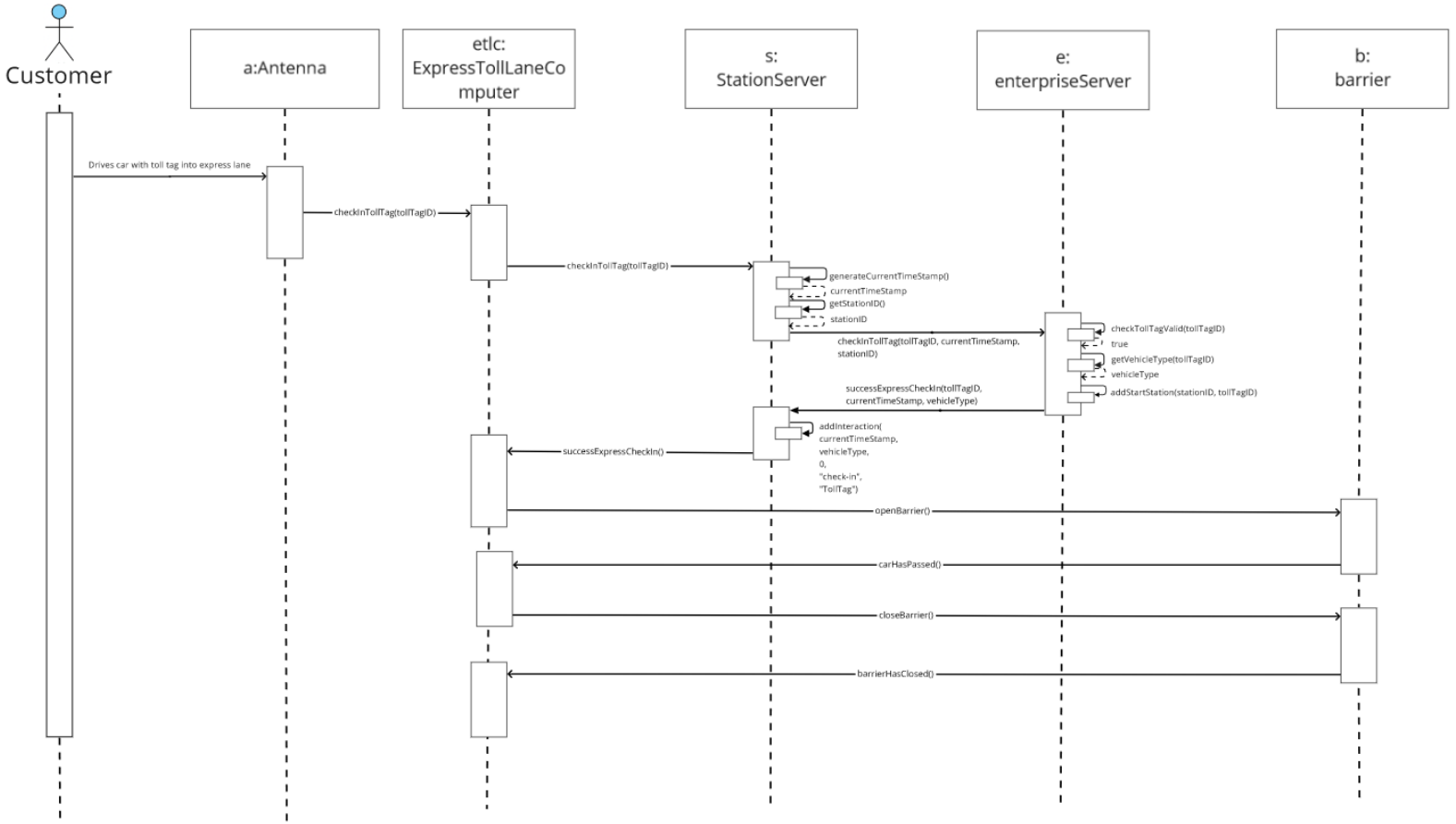


Figure 56: Sequence Diagram - Express Check-in - Main scenario

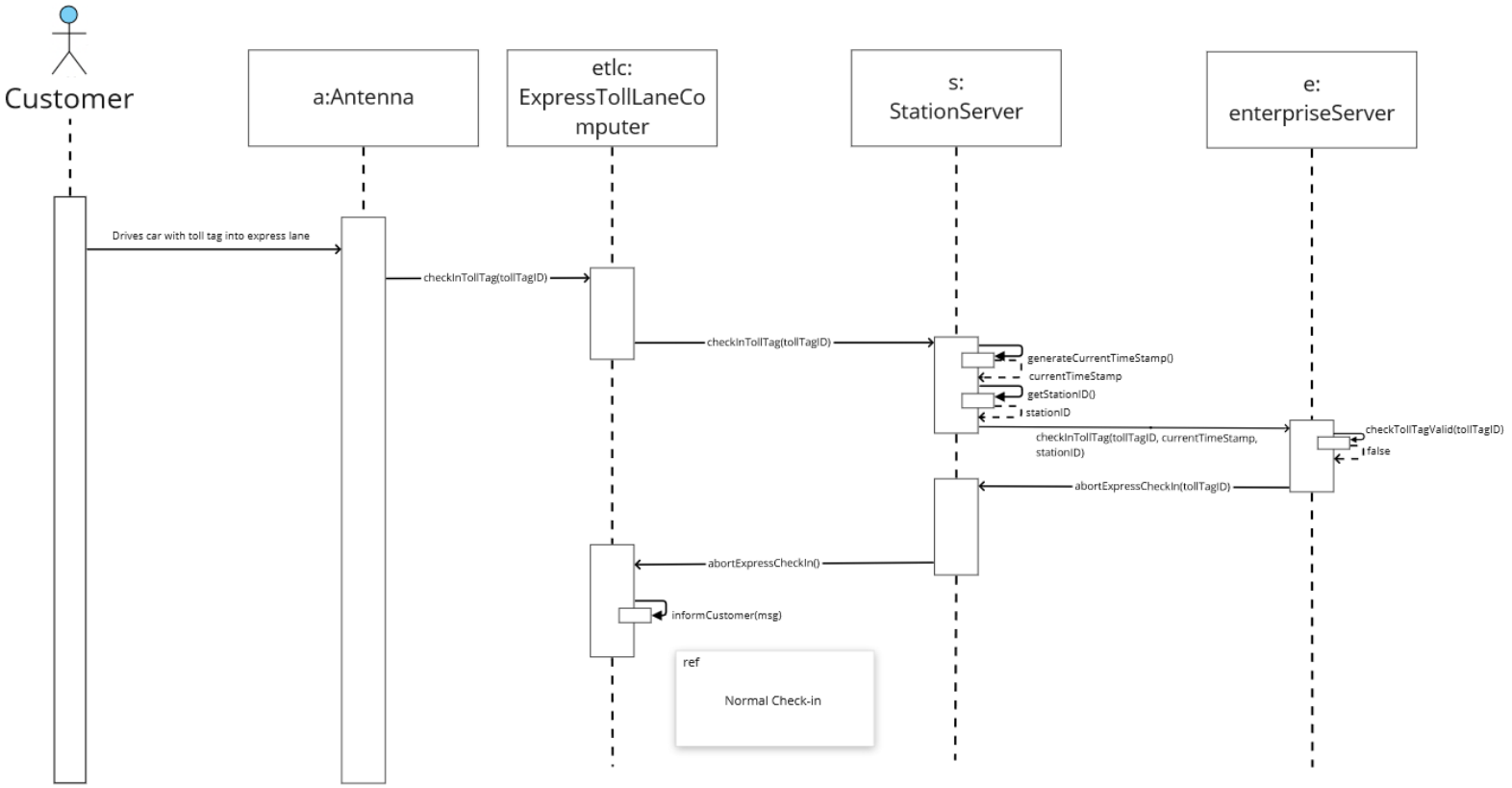


Figure 57: Sequence Diagram - Express Check-in - Alternative Scenario A (Toll tag invalid)

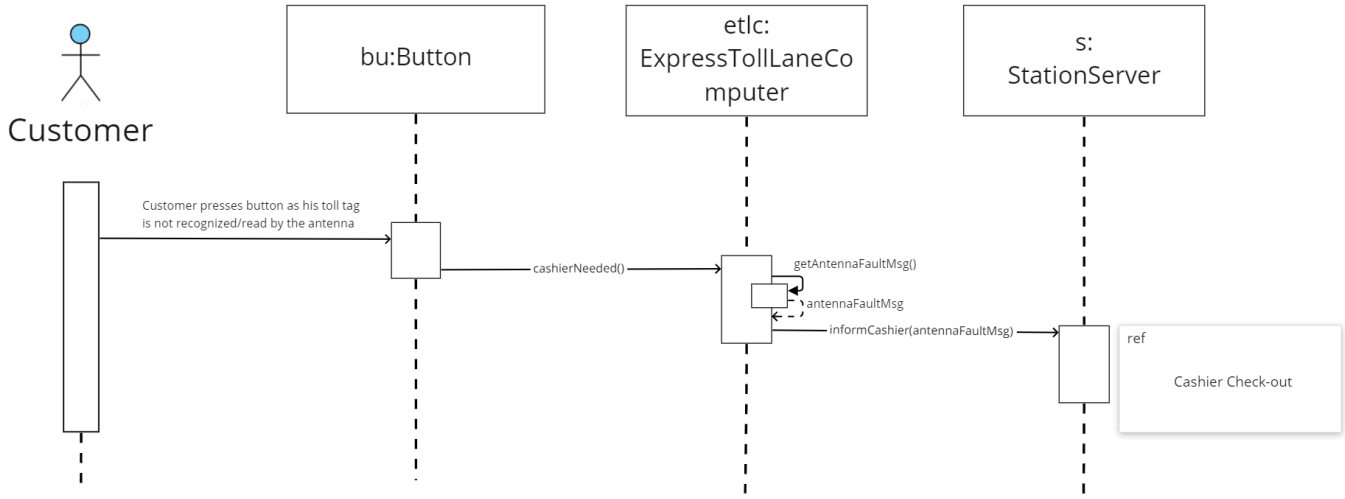


Figure 58: Sequence Diagram - Express Check-in - Alternative Scenario B (Antenna can't read toll tag

5.1.6 Sequence Diagrams: Check-out with toll tag

Author: Hanna and Eydís

The main scenario can be found in figure 59, the alternative scenario in figure 60.

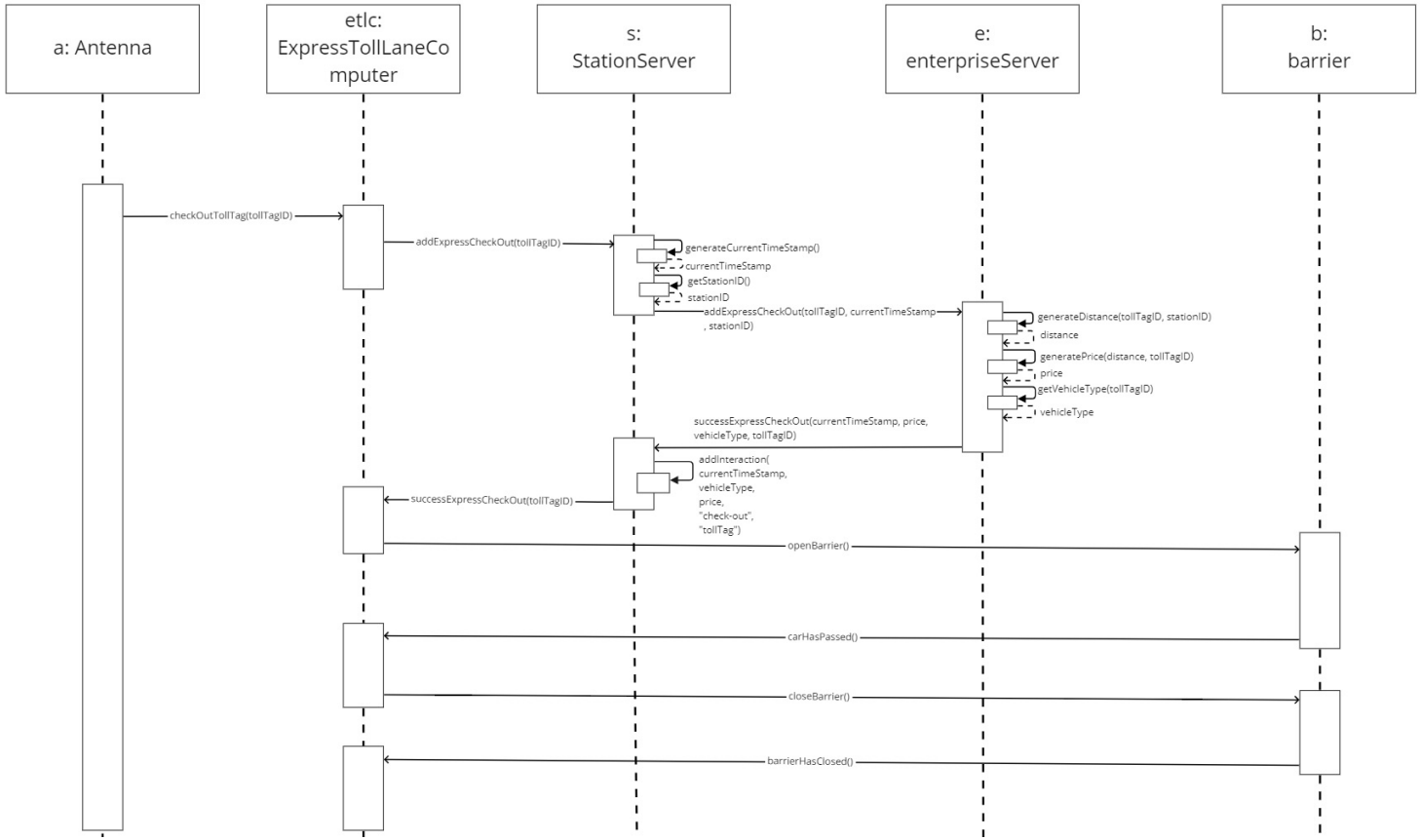


Figure 59: Sequence Diagram - Check-out Toll Tag - Main Scenario

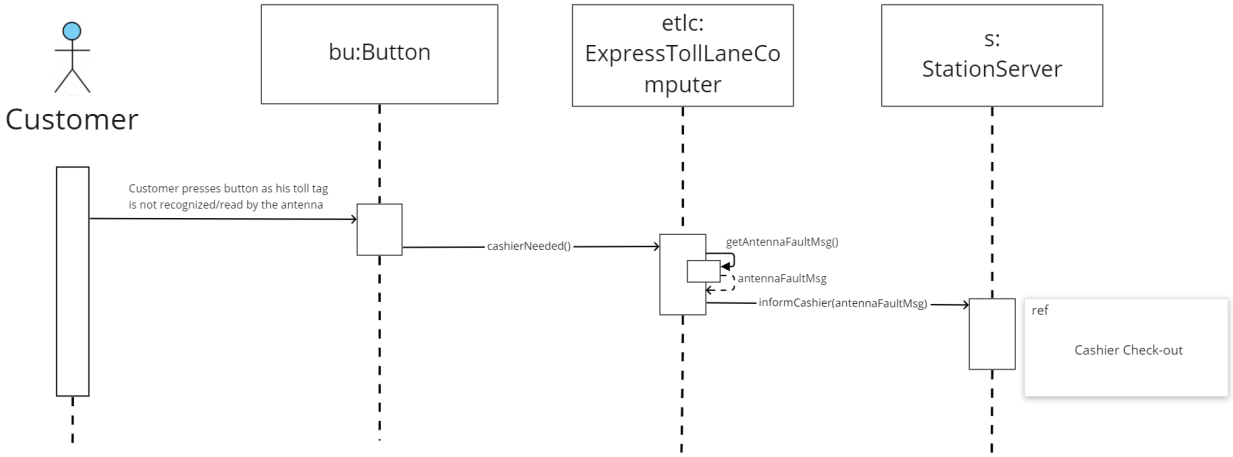


Figure 60: Sequence Diagram - Check-out Toll Tag - Alternative Scenario (Toll Tag cannot be read)

5.1.7 Sequence Diagrams: Generate station report

Author: Paweł And Christopher
The main scenario can be found in figure 61, the alternative scenario in figure 62.

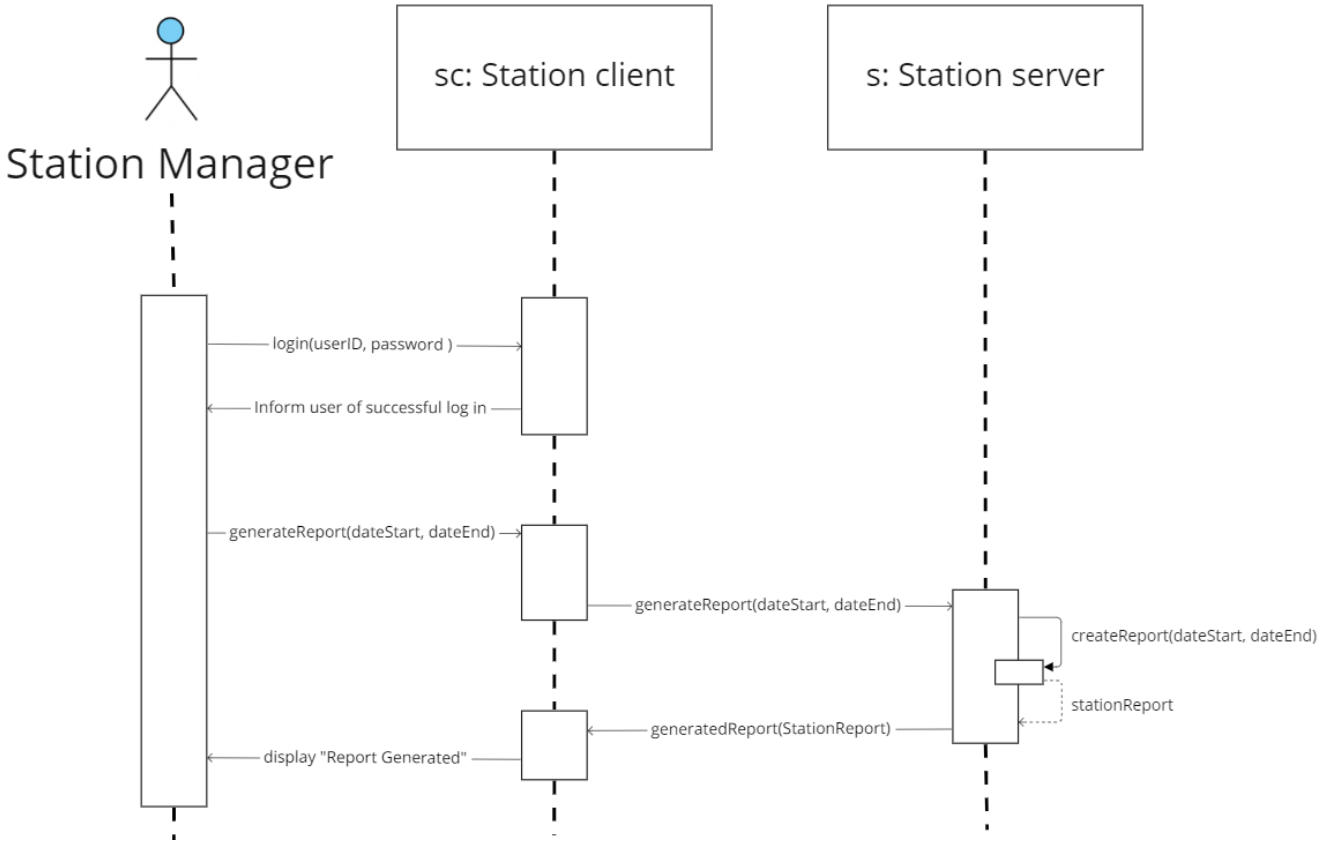


Figure 61: Sequence Diagram - Main Scenario

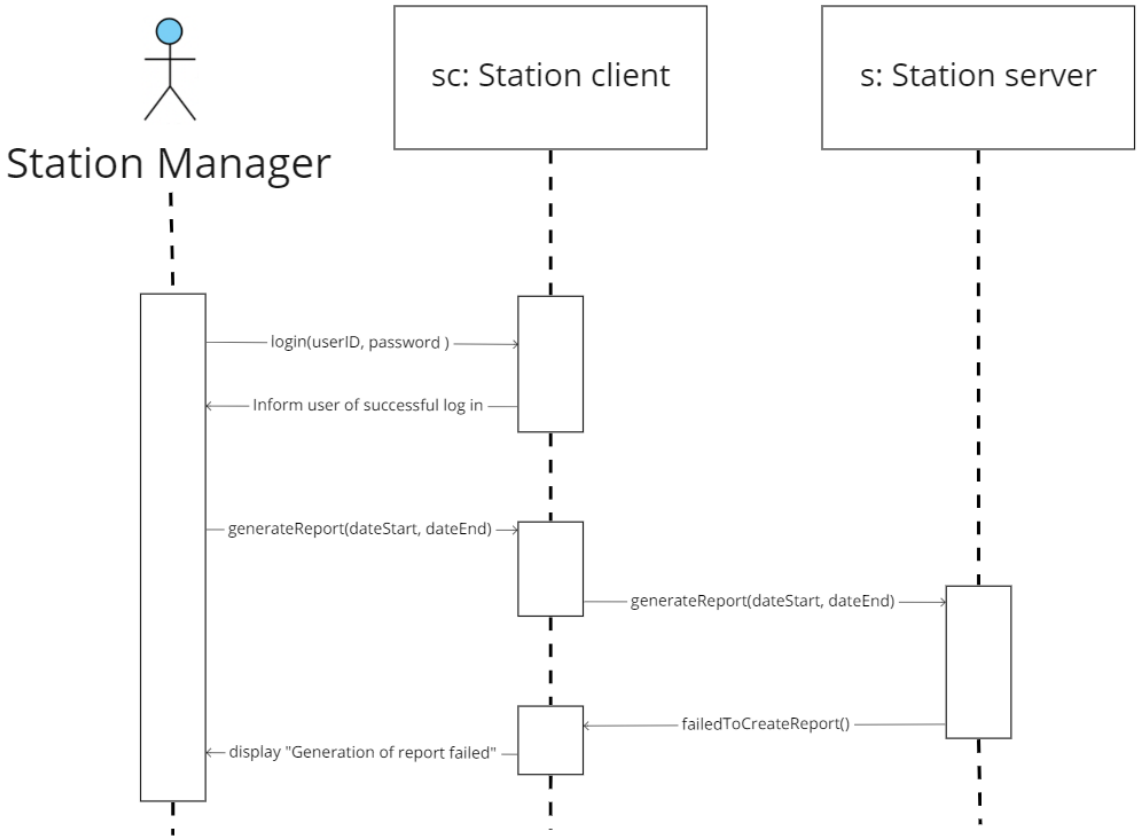


Figure 62: Sequence Diagram - Alternative Scenario

5.1.8 Sequence Diagrams: Cashier Checkout Helper

Author: Paweł And Hanna

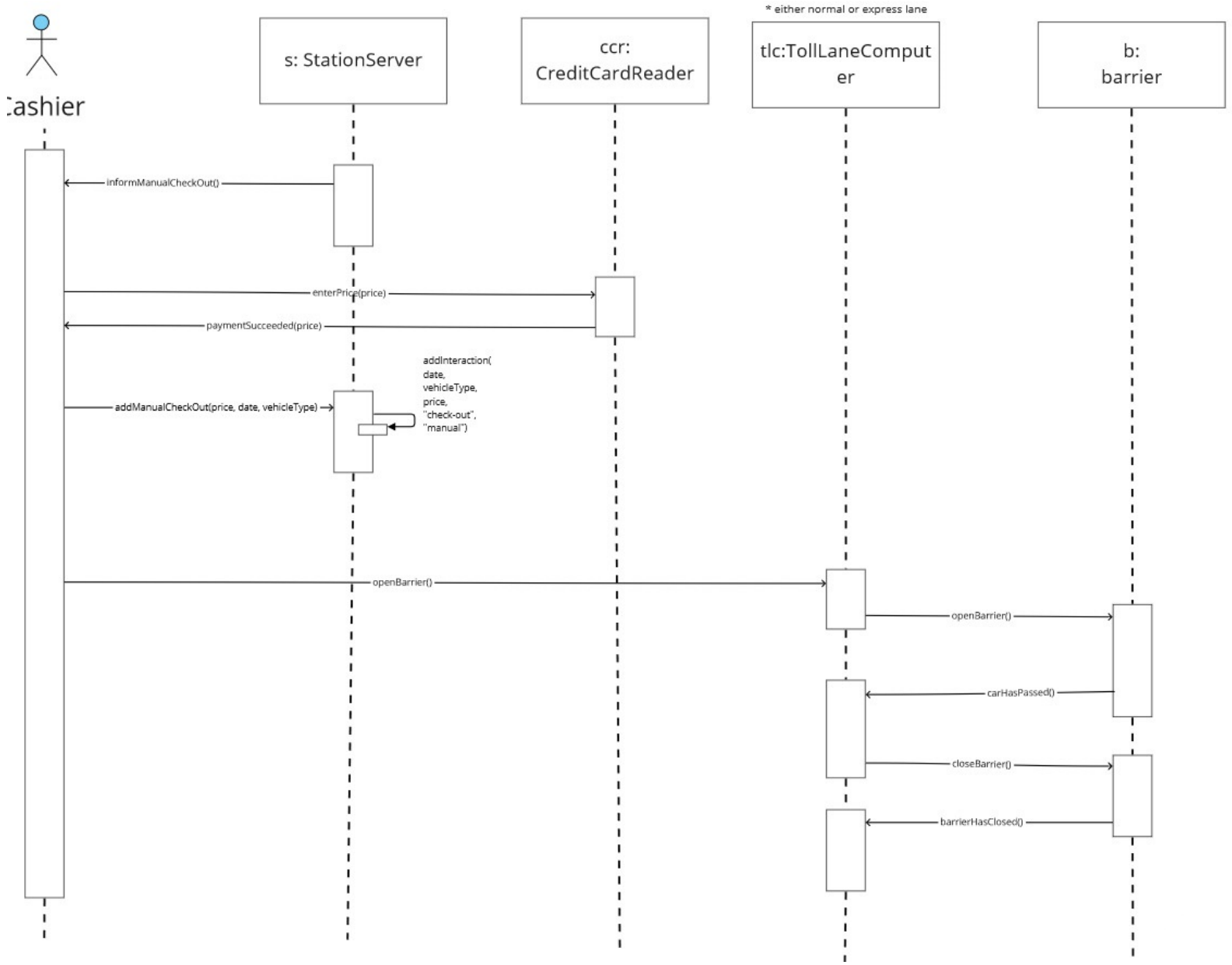


Figure 63: Sequence Diagram - Cashier Checkout

6 Conclusion

6.1 Experience with the project

We all had a conceptual understanding of the foundations of system integration before to the start of this project, owing to the study material provided in this course and previous courses. This project, on the other hand, allowed us to go a step further and design a toll system, which helped to improve our expertise of what is involved in system integration. While designing the whole system for the toll system, which was made up of multiple hardware component systems, we realised how much each individual component required a thorough examination in order for everything to work well together.

We also learned that working in groups can be more efficient since it allows two sets of eyes to look at the same problem at the same time. This improved the design of the components.