UNIVERSITÄT DUISBURG ESSEN

Open-Minded

Lab for Software Engineering
Winter term 2022/2023

Lab for Software Engineering

Cinema Management Application

Ifrat Jahan (3098878)
Jennifer Maxisch (3106694)
Georgios Adamos (3093306)
Thomas Klimek (3067855)
Melvin van der Linde (3106762)

December 19, 2022

University of Duisburg-Essen · Faculty of Engineering Working group Software Engineering – Prof. Dr. M. Heisel
Date: Do. 12-14- Supervisor: Marcel Schweikert

Contents

List of Figures

1 Analysis

1.1 A1

1.1.1 Requirements & Domain-Knowledge

Requirements

R1 Customers can create an account by providing an e-mail address and a password. If an e-mail address which is already associated with an account is provided, account creation fails.

R2 Customers can log in by providing their e-mail address and their password.

R3 A logged in customer can log out.

R4 A customer can browse available showings, ascendingly sorted by date.

R5 A logged in customer can book tickets by selecting the showing from the browsing list and selecting the desired seats. A showing can only be booked up to 15 minutes before it starts.

R6 Staff can add new showings to the database by providing the required data.

R7 Once a showing starts it is marked as "archived".

R8 Archived showings are visible to staff, but not to customers.

R9 Staff can cancel showings. When a show is cancelled all customers who booked tickets for it are notified via e-mail and the showing is then deleted.

R10 Showings which took place a year ago or longer are automatically removed from the database.

R11 When a showing is deleted its associated bookings are also deleted.

Facts

F1 A showing consists of the title of the movie, its duration, the date date, the hall number and unique ID.

F2 A hall consists of a number of rows, a number of seats per row and a unique hall number.

F3 Only one person at a time can sit in a seat.

Assumptions

A1 A web application is a good choice for implementing the desired functionality and all customers are able to use it.

A2 Customers only provide e-mail addresses they can access.

A3 Customers will stay up to date with the list of available showings.

A4 Every booking is paid via an external service.

A5 Staff will only add showings which take place in the future.

1.1.2 Contextdiagram

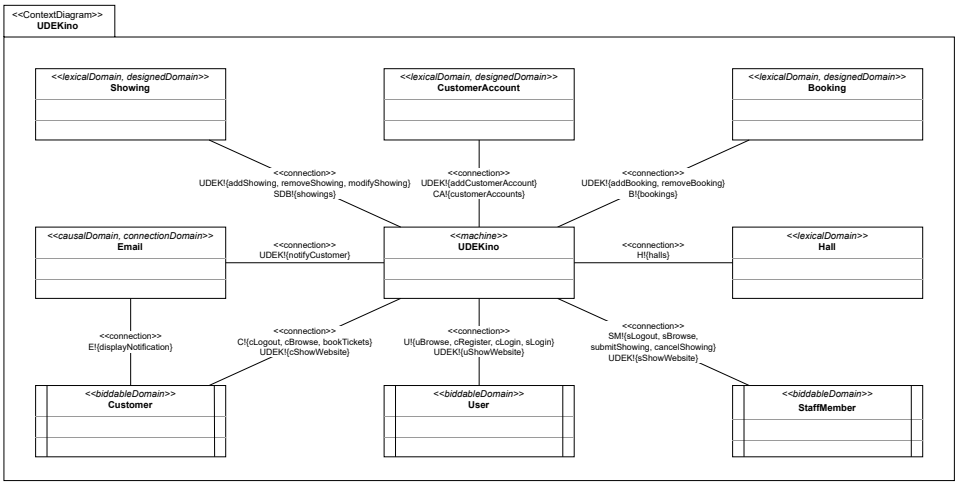

Figure 1.1: Contextdiagram

1.2 A2

We can derive the following problem diagrams

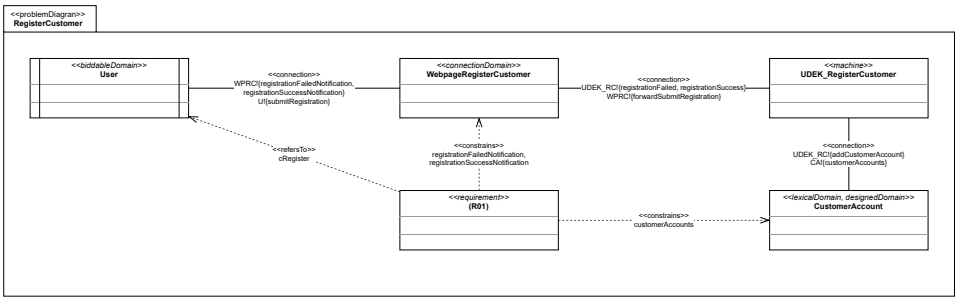

Figure 1.2: Problem diagram for R1

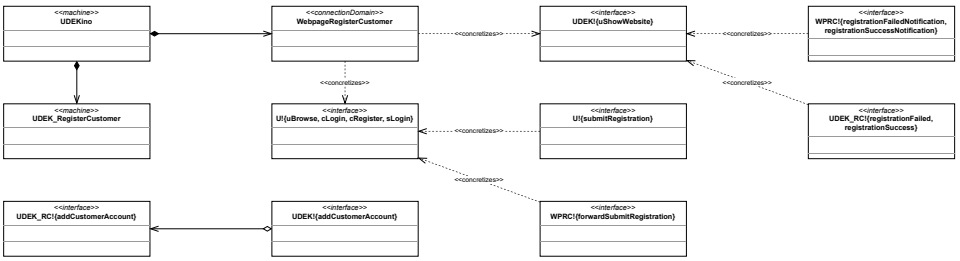

Figure 1.3: Mapping diagram for R1

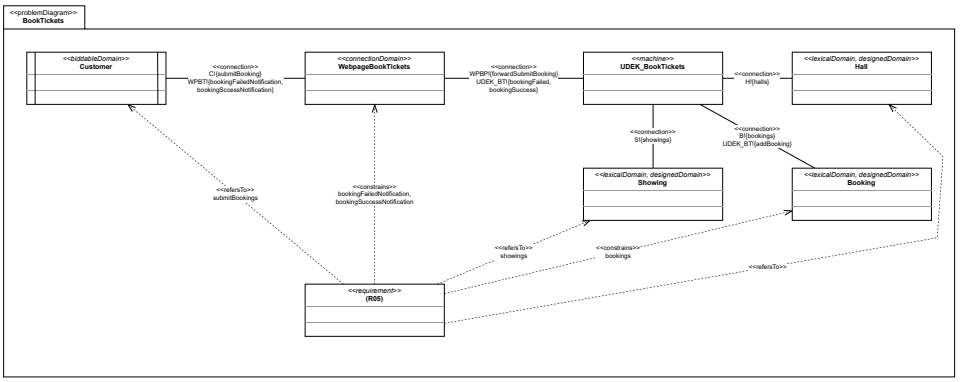

Figure 1.4: Problem diagram for R5

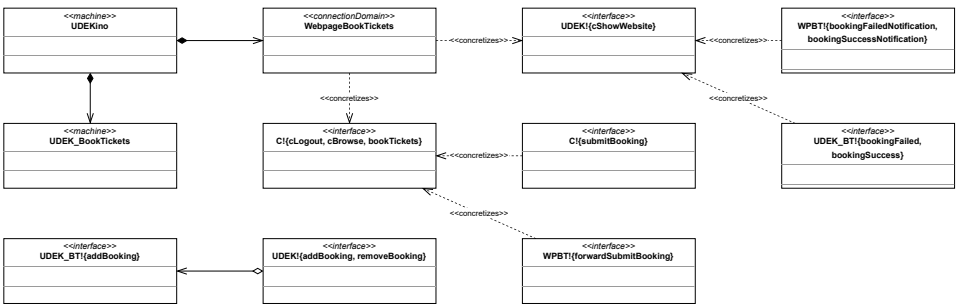

Figure 1.5: Mapping diagram for R5

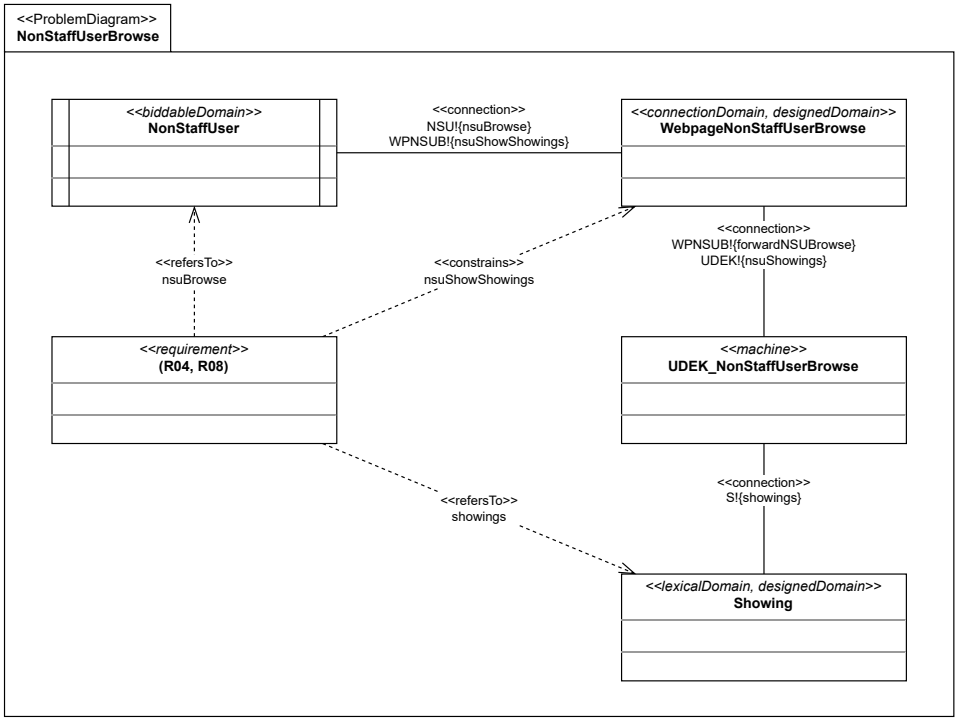

Figure 1.6: Problem diagram for R4 / R8

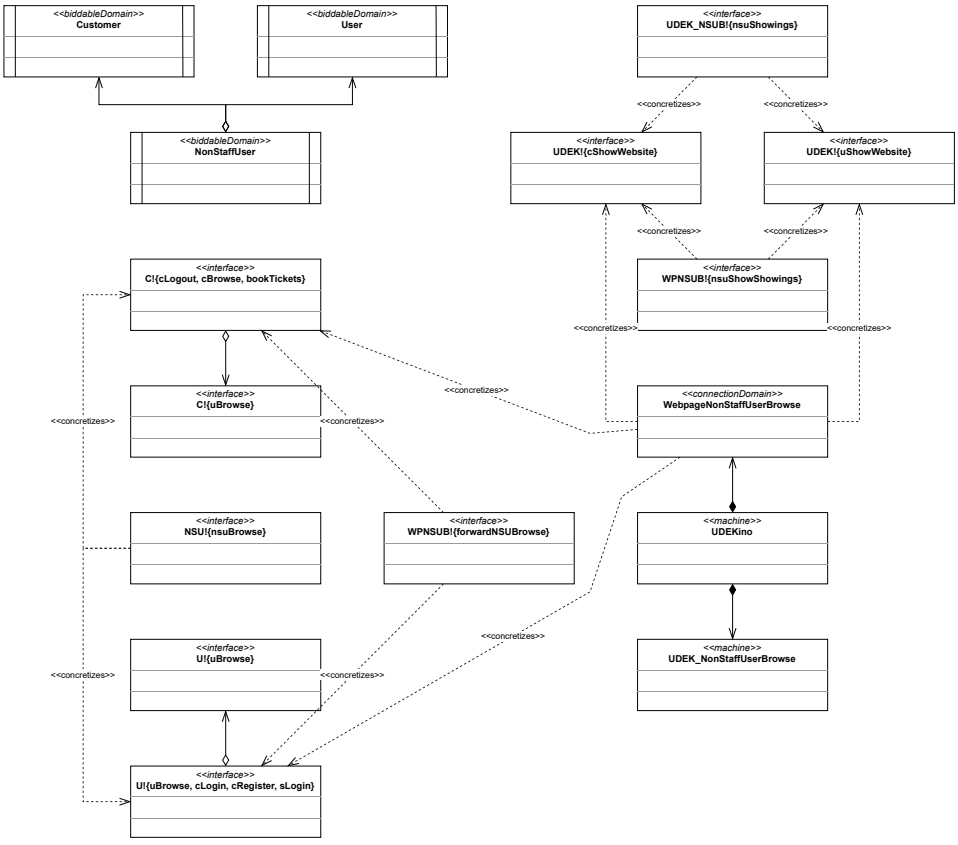

Figure 1.7: Mapping diagram for R4 / R8

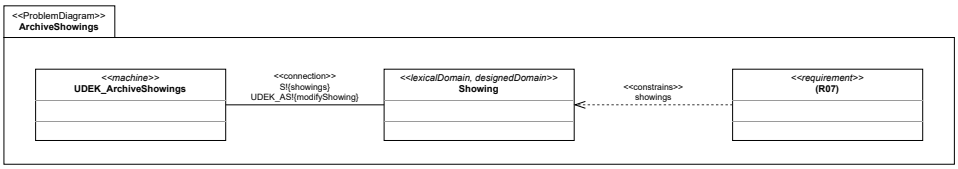

Figure 1.8: Problem diagram for R7

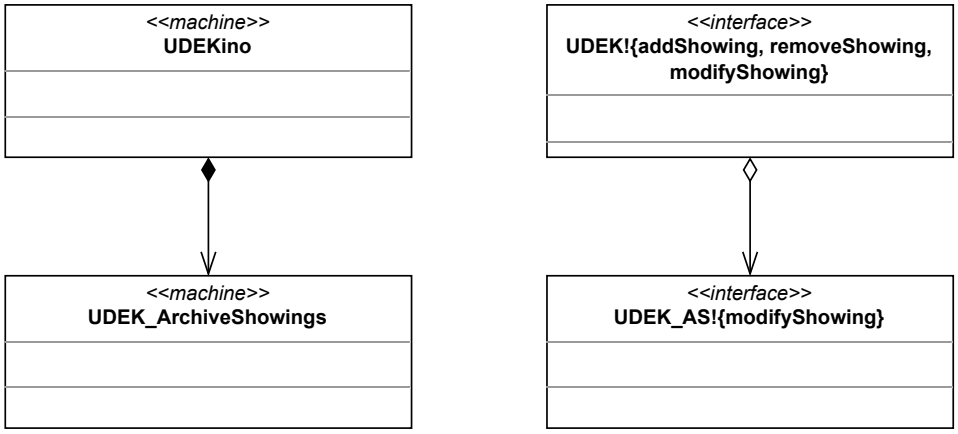

Figure 1.9: Mapping diagram for R7

Frames

- R1 fits to update 2
- R5 fits to update 2
- R4 / R8 fits to query 2
- R7 fits to simple transformation

1.3 A3

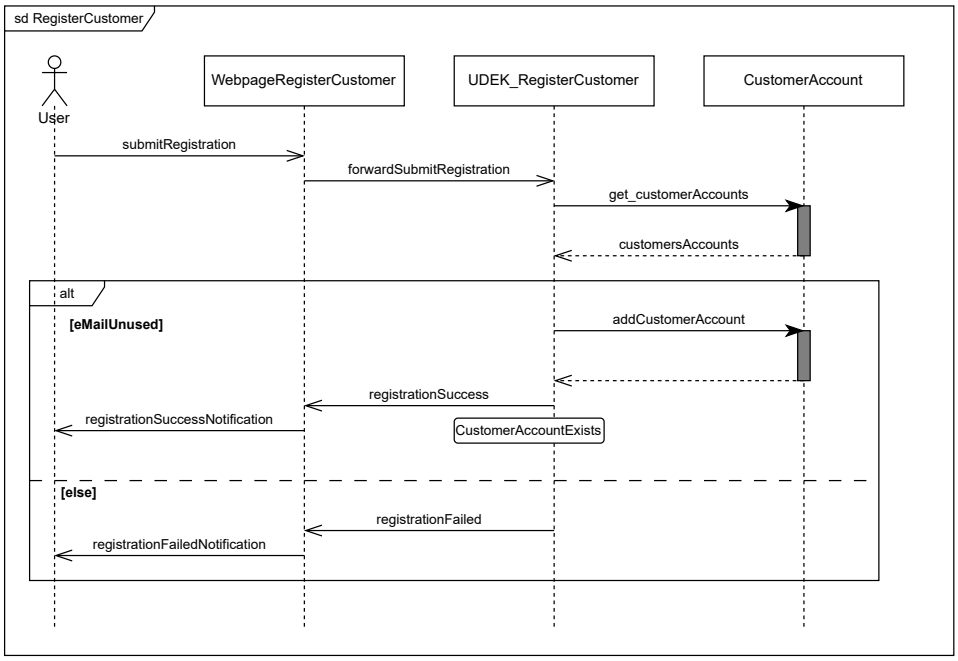

Figure 1.10: Sequence diagram for R1

S1a **WebpageRegisterCustomer**
When the WebpageRegisterCustomer recieves the command "submitRegistration", the command is forwarded to machine with "forwardSubmitRegistration". Results are recieved via commands "registrationFailed" or "registrationSuccess" and displayed to the User via "registrationFailedNotification" / "registrationSuccessNotification".

S1b **UDEK_RegisterCustomer**
When the machine receives the command "fowardSubmitRegistration" the availability of the e-mail address is checked against existing Customer accounts in the CustomerAccount database via "get_customerAccounts". If the e-mail address is available, a new Customer account is created with the data from the forwarded request and added to the CustomerAccount database via "addCustomerAccount" and a confirmation is sent to the WebpageRegisterCustomer via "registrationSuccess". If the e-mail address is not available, account creation fails and a failure notification is sent to the WebpageRegisterCustomer via "registrationFailed".

S1c **CustomerAccount**
When the database receives the command "get_customerAccounts", all Customer accounts are returned as the data "customerAccounts". When the database receives the command "addCustomerAccount", the Customer account is added.

$$(A2) \wedge (S1a) \wedge (S1b) \wedge (S1c) \implies (R1)$$

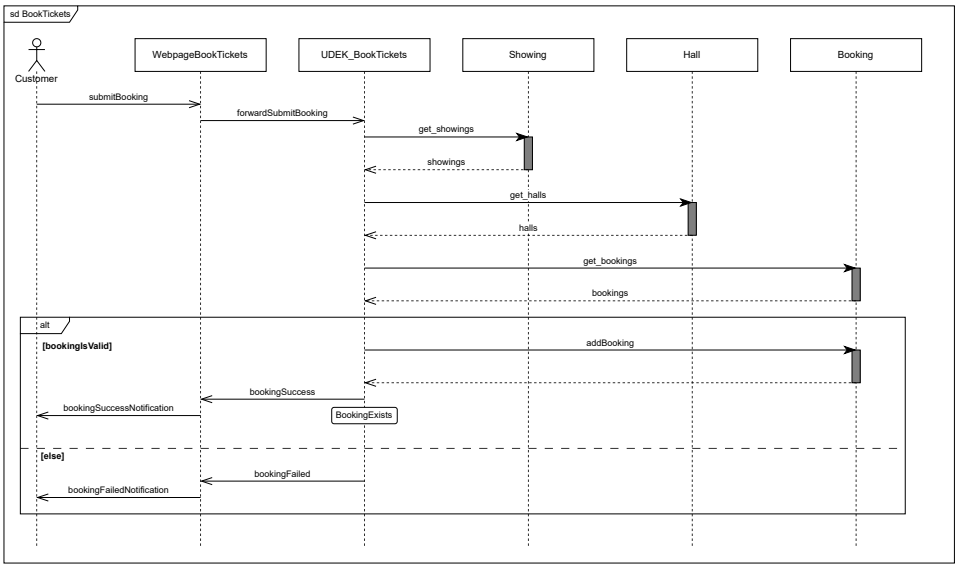

Figure 1.11: Sequence diagram for R5

S2a **WebpageBookTickets**
When the Webpage receives the command "submitBooking", the command is forwarded to the machine with the command "forwardSubmitBooking". Results are received via "bookingFailed" or "bookingSuccess" and displayed the the Customer via "bookingFailedNotification" / "bookingSuccessNotification"

S2b **UDEK_BookTickets** When the machine receives the command "forwardSubmitBooking", the machine checks the availability of the desired showing and seats against the Showing database, Hall database and Booking database via "get_showings", "get_halls" and "get_bookings". If the desired showing and seats exist, the showing begins in more than 15 minutes and the seats are not already booked, the booking is added to the Booking database via "addBooking" and a success notification is sent to the WebpageBookTickets via "bookingSuccess". Otherwise the booking fails and the Webpage is notified of the failure via "bookingFailed".

S2c **Showing** When the database receives the command "get_showings", all showings are returned as the data "showings".

S2d **Hall** When the database receives the command "get_halls", all halls are returned as the data "halls".

S2c **Booking** When the database receives the command "get_bookings", all bookings are returned as the data "bookings". When the database receives the command "addBooking", the booking is added.

$$(F3) \wedge (S2a) \wedge (S2b) \wedge (S2c) \wedge (S2d) \wedge (S2e) \implies (R5)$$

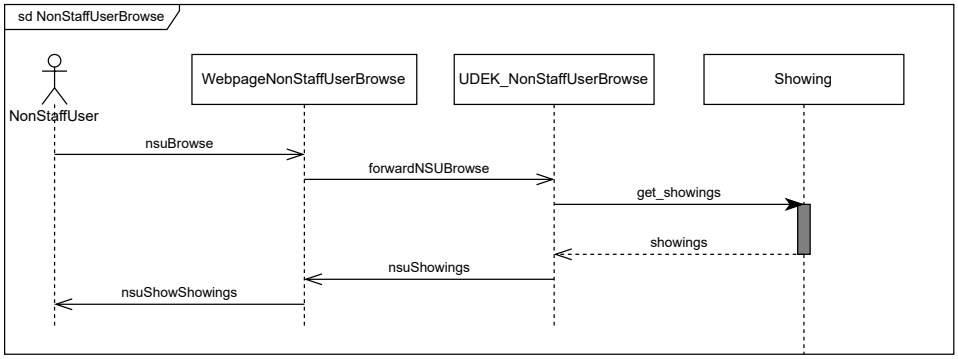

Figure 1.12: Sequence diagram for R4/R8

S3a **WebpageNonStaffUserBrowse** When the Webpage receives the command "nsuBrowse", the command is forwarded to the machine with the command "forwardNSUBrowse". Results are received via "nsuShowings" and displayed to NonStaffUser via "nsuShowShowings".

S3b **UDEK_NonStaffUserBrowse** When the machine receives the command "forwardNSUBrowse", the machine gets all showings from the Showing database via "get_showings". All non-archived showings are send/transfered to the Webpage via "nsuShowings".

S3c **Showing** When the database receives the command "get_showings", all showings are returned data as "showings"

$$(S3a) \wedge (S3b) \wedge (S3c) \implies (R4) \wedge (R8)$$

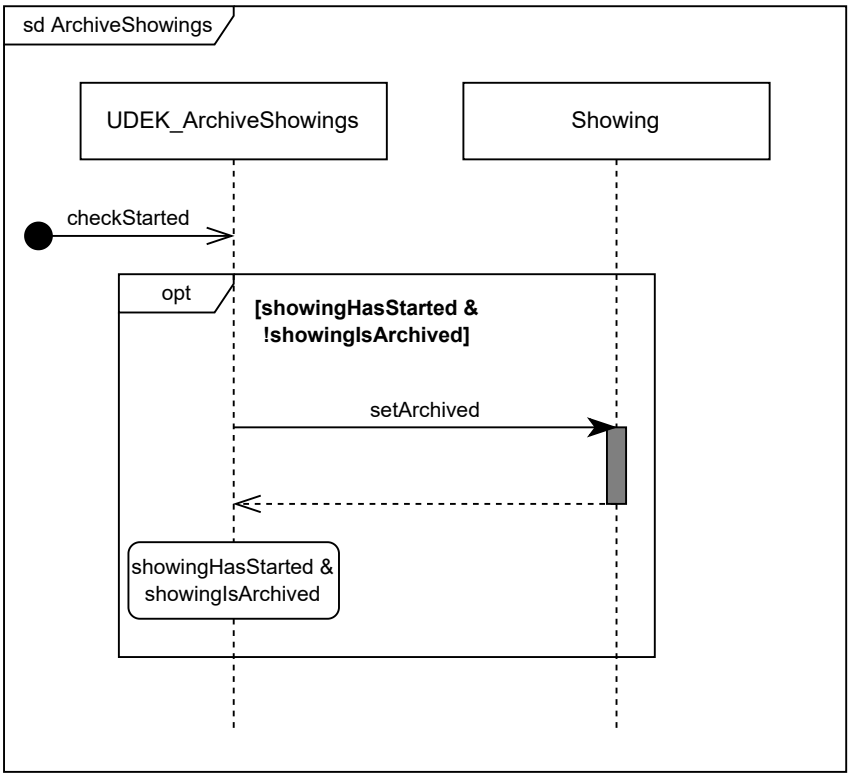

Figure 1.13: Sequence diagram for R7

S4a **UDEK_ArchiveShowings** When receiving the command "checkStarted", all showings which have already started, and are not yet marked as archived, are marked as archived using the command "setArchived".

S4b **Showing** When receiving the command "setArchived", all showings which have already started, and are not yet marked as archived, are marked as archived.

$(S4a) \wedge (S4b) \implies (R7)$

1.4 A4

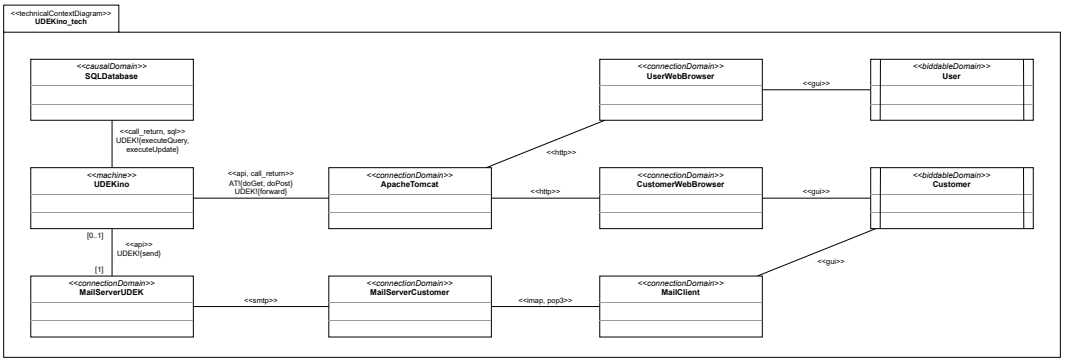

Figure 1.14: Technical Context Diagram

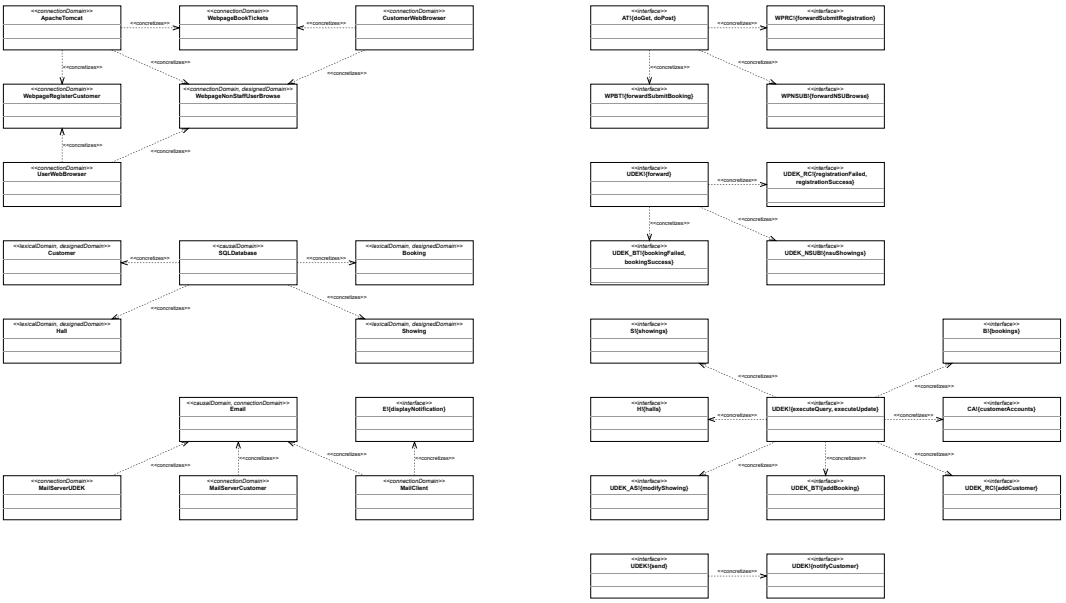

Figure 1.15: Mapping Diagram of the TCD

1.5 A5

A short OCL example:

1.5.1 RegisterCustomer

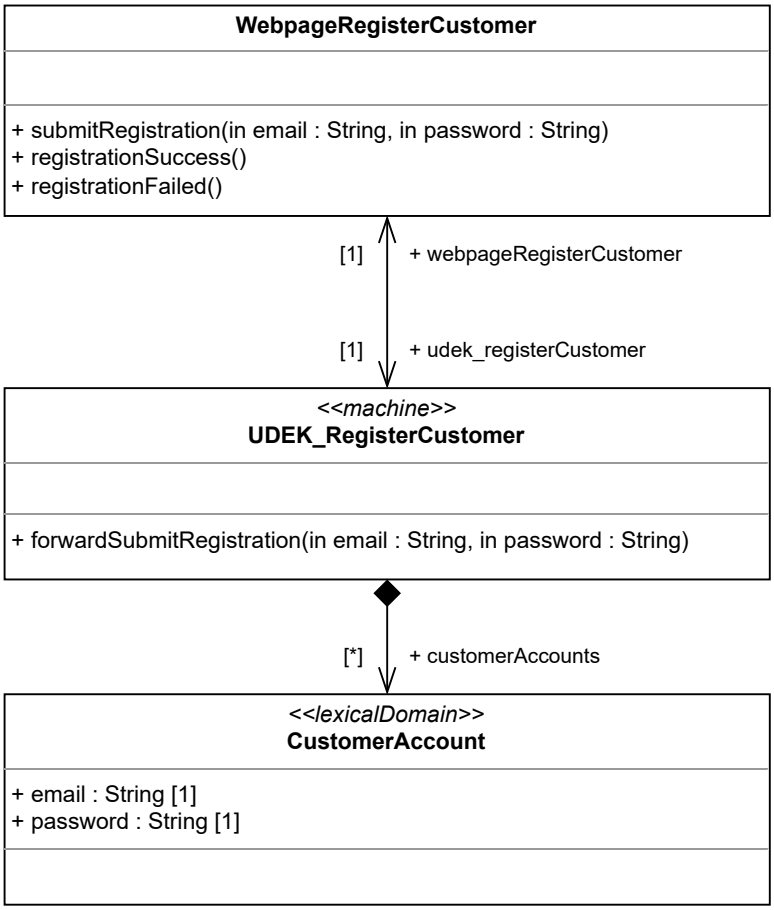

Figure 1.16: Class model of the operation RegisterCustomer

Name: forwardSubmitRegistration
Description: Creates a new Customer Account with the supplied e-mail address and password and adds it to the database and then sends a success notification to the webpage, or sends a failure notification to the webpage
OCL constraint:

```
context UDEK_RegisterCustomer
inv: customerAccounts->isUnique(email)

context UDEK_RegisterCustomer::forwardSubmitRegistration(email
    : String, password : String)
pre: true
post:
    let eMailUnused : Boolean =
```

```
        customerAccounts@pre->one(
            c : CustomerAccount
                | c.email = email
        )
    in let CustomerAccountExists : Boolean =
        customerAccounts->one(
            c : CustomerAccount
                | c.email = email and c.password = password
        )
    in let CustomerAccountsSizeCheck : Boolean =
        costomerAccounts->size() = customerAccounts@pre->size()
            + 1
    in
        if eMailUnused
        then CustomerAccountExists and
            webpageRegisterCustomer^registrationSuccess()
        else webpageRegisterCustomer^registrationFailed()
```

1.5.2 NonStaffUserBrowse

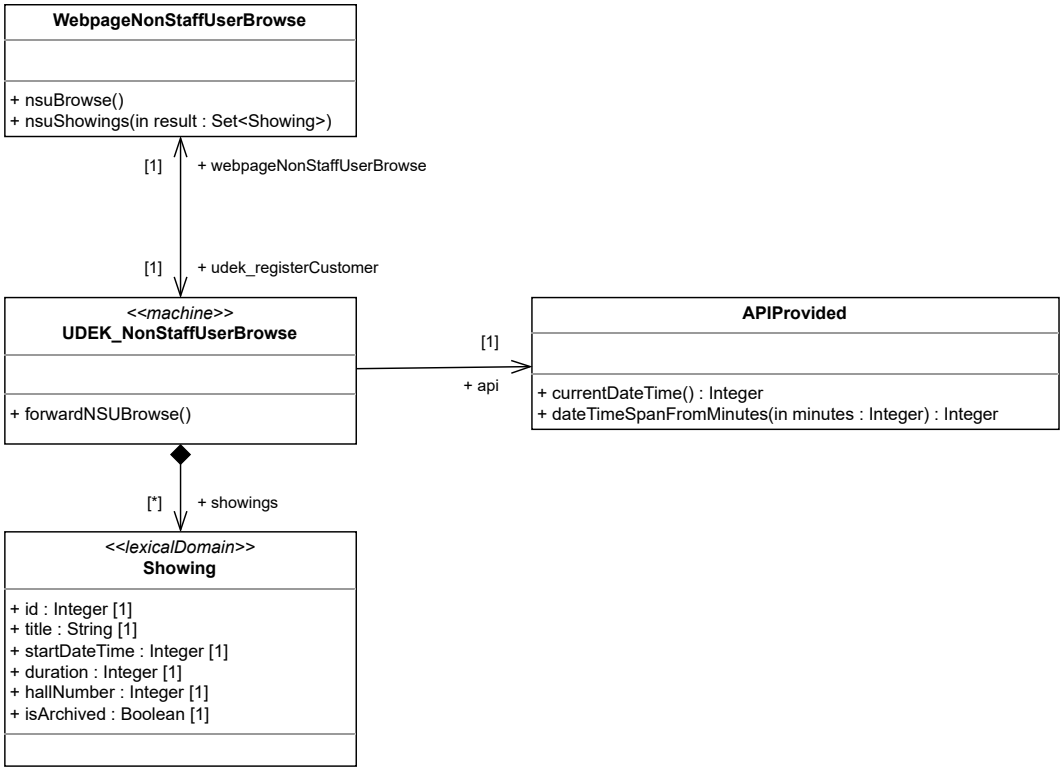

Figure 1.17: Class model of the operation NonStaffUserBrowse

Name: forwardNSUBrowse
Description: sends a set containing all showings which are not archived to the webpage
OCL constraint:

```
context UDEK_NonStaffUserBrowse
inv:
    showings->isUnique(id)
    and showings->forAll(
        s : Showing
            | s.isArchived = s.startDateTime >
                api.currentDateTime() -
                api.dateTimeSpanFromMinutes(15))

context UDEK_NonStaffUserBrowse::forwardNSUBrowse()
pre: true
post: webpageNonStaffUserBrowse^nsuShowings(showings->select(s
    : Showing | not s.isArchived))
```

8.4

G: 9/10

title is missing.

1.5.3 BookTickets

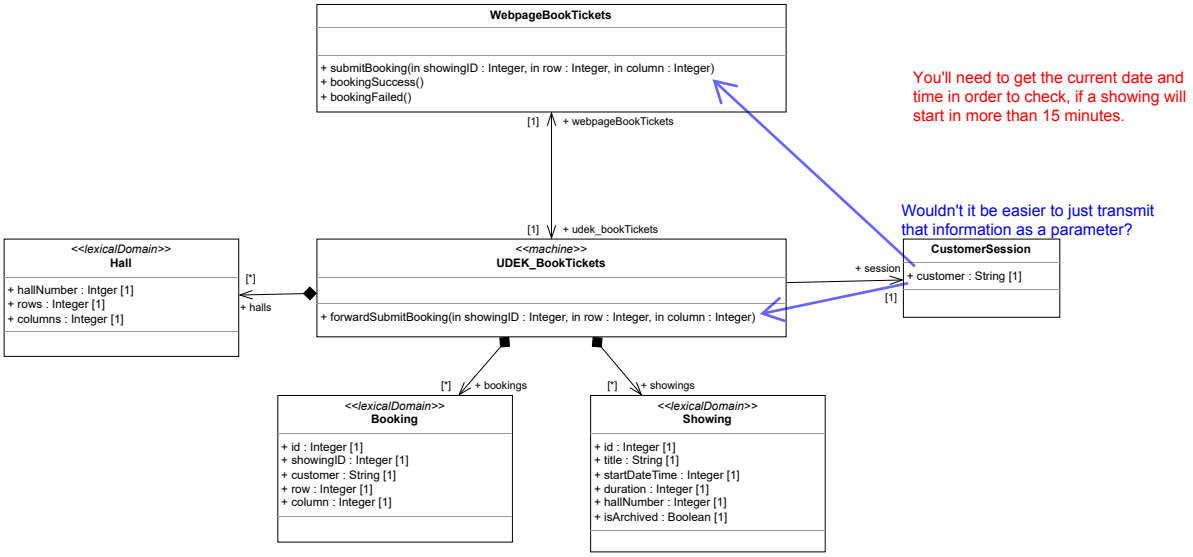

Figure 1.18: Class model of the operation BookTickets

Invariants that show that the ids of showings and bookings are unique would have been nice.
And before you complain, that you specified it before, let me remind you that UDEK_NonStaffUserBrowse is not in this diagram.

Name: forwardBookTickets

Description: tries to book the requested seat and sends a notification whether the booking succeeded to the webpage.

OCL constraint:

```
context UDEK_BookTickets::forwardSubmitBooking(showingID :
    Integer, row : Integer, column : Integer)
pre: true
post:
    let BookingExists : Boolean =
        bookings->one(
            b : Booking
```

Customers should be able to book more than one seat at a time.

That's what we wanna see after a successful booking.

```
                | b.showingID = showingID
                    and b.customer = session.customer
                    and b.row = row
                    and b.column = column
        )
    in let bookingIsValid : Boolean =
        1 <= row and 1 <= column
        and showings->one(
            s : Showing
                | s.showingID = showingID
                    and s.row <= s.rows
                    and s.column <= s.columns
        ) and not bookings@pre->exists(
            b : Booking
                | b.showingID = showingID
                    and b.row = row
                    and b.column = column
        )
    in let BookingsSizeCheck : Boolean =
        bookings->size() = bookings@pre->size() + 1
    in
        if bookingIsValid
        then BookingExists and BookingsSizeCheck and
            webpageBookTickets^bookingSuccess()
        else webpageBookTickets^bookingFailed()
```

You even check that the row and column values don't go below 1. Cool.

The Showing to be booked needs to exist.

Showing doesn't have any of these attributes.

You still need to check, that the showing starts in 15 minutes or later.

You make sure that the seat isn't booked yet.

Really interesting use of let-in expressions. Makes this part so easy to read.

Remarks: `CustomerSession` contains session data of the logged in customer who sent the request.

1.5.4 ArchiveShowings

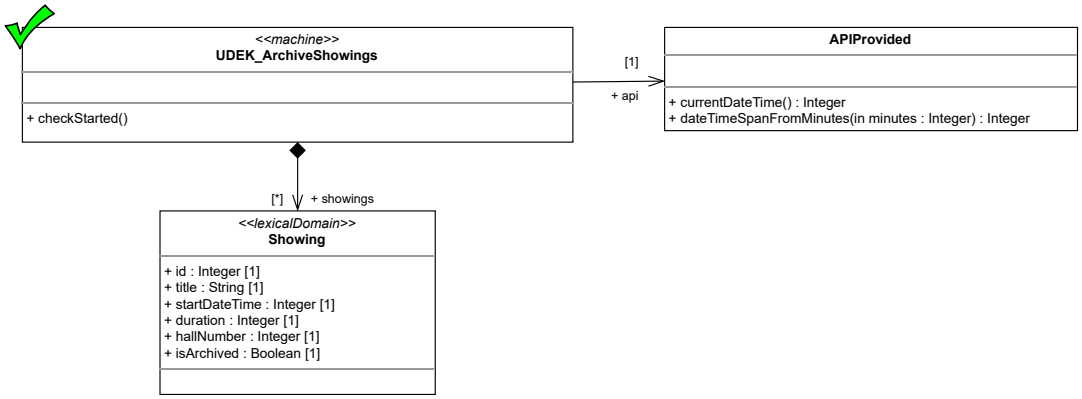

Figure 1.19: Class model of the operation ArchiveShowings

Name: checkStarted
Description: sets all showings which start in 15 minutes or less (or already have started) as archived.
OCL constraint:

Only the ones that have started!
The ones that start in less than 15 minutes should still be visible, but not bookable anymore.

```
1 context UDEK_ArchiveShowings::checkStarted()
2 pre: true
3 post:
4     showings->forAll(
5         s : Showing
6             | s.isArchived = s.startDateTime >
                  api.currentDateTime() -
                  api.dateTimeSpanFromMinutes(15)
7     )
```

Ok, if I read that correctly, you subtract 15 minutes from the current time and archive everything that starts later than that.
So all showings that started 15 minutes ago and all future showings get archived. At the same time, all showings that started more than 15 minutes ago are not archived anymore.
s.isArchived = s.startDateTime <= api.currentDateTime() is what you really want here.

1.6 A6

Examples of a life-cycle using the math-environment:

$LC_{User} = (RegisterCustomer|NonStaffuserBrowse)^*$

$LC_{Customer} = (Browse^+;[Book])^*$ Booking is optional.

$LC_{NonStaffUser} = (NonStaffuserBrowse)^*$

$LC_{UDEKino} = (||_{i=1}^{n} LC_{User_i})||(||_{j=1}^{m} LC_{Customer_j})||(||_{k=1}^{l} LC_{Customer_k})||ArchiveShowings^*$

I'm gonna take a wild guess and say that's supposed to be the NonStaffUser.

2 Design

2.1 D1

2.2 D2

2.3 D3

2.4 D4

State diagrams with tikZ:

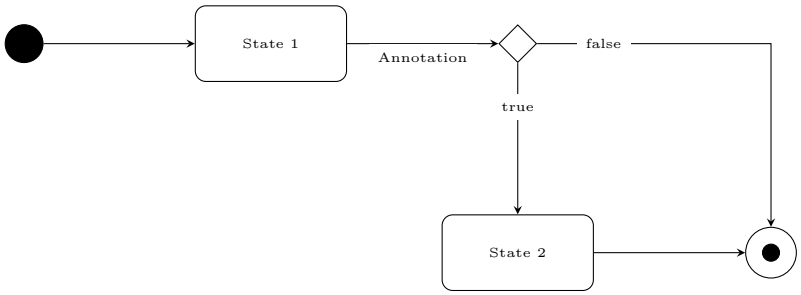

Figure 2.1: Zustandsdiagramm Person 1

3 Implementation & Testing

3.1 I

3.2 T1

3.3 T2

3.4 T3

4 Glossary

Table 4.1: Glossary

Name	Type	Description	Source
A			
addBooking	phenomenon	the machine adds a new booking to the bookings database	CD
addBooking	message	contains a showing ID and seats	SD R5
addCustomerAccount	phenomenon	the machine adds a new customer to the customer accounts database	CD
addCustomerAccount	message	contains an e-mail address and a password	SD R1
addShowing	phenomenon	the machine adds a new showing to the customer accounts database	CD
APIProvided	class	a class containing various auxiliary functions provided by the runtime environment	Class Model
api	class call name	an instance of the APIProvided class	Class Model
ApacheTomcat	connection domain	An Open Source JSP and Servlet Container from the Apache Foundation.	TCD
B			
Booking	lexical domain, designed domain	a database containing the bookings made by customers	CD
Booking	object	the database containing all bookings	SD R5
Booking	class	a record representing a booking of a seat for a showing	Class Model
BookingExists	state predicate	given booking exists within the Booking database	SD R5
bookingFailed	phenomenon	the machine notifies the webpage that a booking has failed	PD R5
bookingFailed	message	informs the WebpageBookTickets that the booking failed	SD R5
bookingFailed()	method	displays a notification to the customer that the booking failed	Class Model
bookingFailedNotification	phenomenon	the webpage displays a notification to the customer that a booking has failed	PD R5
bookingFailedNotification	message	informs the user that the booking failed	SD R5

Table 4.1: Glossary

Name	Type	Description	Source
bookingIsValid	guard	showing with ID contained in request exists and starts in more than 15 minutes and the seats contained in the request exist in the showing's hall and are not already booked	SD R5
bookings	phenomenon	the bookings database provides the bookings data to the machine	CD
bookings	class call name	the database of bookings	Class Model
bookings	message	all bookings in the Booking database	PD R5
bookingSuccess	phenomenon	the machine notifies the webpage that a booking has succeeded	PD R5
bookingSuccess	message	informs the WebpageBookTickets that the booking was successful	SD R5
bookingSuccess()	method	displays a notification to the customer that the booking succeeded	Class Model
bookingSuccessNotification	phenomenon	the webpage displays a notification to the customer that a booking has succeeded	PD R5
bookingSuccessNotification	message	informs the Customer that the booking was successful	SD R5
bookTickets	phenomenon	a customer books tickets for a showing	CD
C			
cBrowse	phenomenon	a customer browses available showings	CD
checkStarted	found message	a prompt for the UDEK_ArchiveShowings machine to mark all showings which have already started and are not marked as archived, as archived	SD R7
checkStarted()	methods	archives all showings that start in 15 minutes or less (or already have started)	Class Model
cLogin	phenomenon	a user attempts to log into a customer account	CD
cLogout	phenomenon	a customer attempts to log out	CD
column	attribute	the column of the booked seat	Class Model
column	parameter	the column of the seat that is to be booked	Class Model
columns	attribute	the number of columns of seats the cinema hall contains	Class Model
cRegister	phenomenon	a user attempts to create customer account on UDEKino	CD
cShowWebsite	phenomenon	the machine shows a website to the customer	CD

Table 4.1: Glossary

Name	Type	Description	Source
currentDateTime()	auxiliary function	returns the current time in unix epoch time	Class Model
Customer	biddable domain	a customer of UDEKino; a user who has logged into a customer account	CD. TCD
Customer	actor	a customer who wishes to book tickets	SD R5
customer	attribute	the e-mail address of the customer who made the booking	Class Model
customer	attribute	the e-mail of the session's customer	Class Model
CustomerAccountExists	state predicate	the customer account with the given e-mail address and password exists within the CustomerAccount database	SD R1
customerAccounts	phenomenon	the customerAccounts database provides the customerAccounts data to the machine	CD
customerAccounts	message	all customer accounts in the CustomerAccount database	SD R1
CustomerAccount	lexical domain, designed domain	a database containing customer accounts	CD
CustomerAccount	class	a record representing a Customer account	Class Model
customerAccounts	class call name	the database of CustomerAccounts	Class Model
CustomerSession	class	an auxiliary class containing auxiliary functions and data of a logged in Customer's session	Class Model
CustomerWebBrowser	connection domain	Web browser used by a logged in customer, e.g. Mozilla Firefox.	TCD
CustomerAccount	object	the database of customer accounts	SD R1
D			
displayNotification	phenomenon	the customer's e-mail client displays a notification e-mail to the customer	CD
dateTimeSpanFromMinutes (in minutes : Integer)	auxiliary function	returns the parameter minutes as unix epoch time	Class Model
doGet	technical phenomenon	A procedure called by the Jakarta Servlet container in which the machine can handle an incoming HTTP GET request. (See forward.)	TCD
doPost	technical phenomenon	A procedure called by the Jakarta Servlet container in which the machine can handle an incoming HTTP POST request. (See forward.)	TCD

Table 4.1: Glossary

Name	Type	Description	Source
duration	attribute	the duration of the movie that is to be shown	Class Model
E			
Email	causal domain, connection domain	an e-mail service offering to deliver e-mails	CD
eMailUnused	guard	the e-mail contained in the registration request is not contained in customerAccounts	SD R1
executeQuery	technical phenomenon	A procedure the machine can call to query the contents of a SQL database.	TCD
executeUpdate	technical phenomenon	A procedure the machine can call to manipulate a SQL database.	TCD
F			
forward	technical phenomenon	An assortment of procedures and manipulable resources the machine can use to prepare HTTP responses which are then sent by the Jakarta Servlet container.	TCD
forwardNSUBrowse	phenomenon	the website sends a request for a list of upcoming showings to the machine	PD R4 / R8
forwardNSUBrowse()	method	the machine handles the browse request	Class Model
forwardNSUBrowse	message	a request for the machine to send a list of available, i.e., non-archived, showings	SD R4/8
forwardSubmitBooking	phenomenon	the webpage forwards a request to book tickets to the machine	PD R5
forwardSubmitRegistration()	method	the machine handles the registration request: it creates a new account if possible and sends a status notification to the webpage	Class Model
forwardSubmitBooking	message	contains the showing ID and the desired seats	SD R5
forwardSubmitBooking	method	tries to book the given seat for the given showing and informs the webpage of the success or failure afterwards	Class Model
forwardSubmitRegistration	phenomenon	the webpage forwards a request to register a customer account to the machine	PD R1
forwardSubmitRegistration	message	a request from the WebpageRegisterCustomer to register a new customer account, containing an e-mail address and a password	SD R1
G			
get_bookings	message	contains all messages in the Booking database	SD R5

Table 4.1: Glossary

Name	Type	Description	Source
get_customerAccounts	message	returns all customer accounts in the CustomerAccount database	SD R1
get_halls	message	returns all halls in the Hall database	SD R5
get_showings	message	returns all showings in the Showing database	SD R5, 4/8, 7
gui	technical phenomenon	The web browser renders a webpage.	TCD
H			
Hall	object	the database containing the cinema halls	SD R5
Hall	class	a record representing a cinema hall	Class Model
hallNumber	attribute	the number of the hall the showing will take place in	Class Model
halls	phenomenon	the halls database provides the halls data to the machine	CD
halls	message	all halls in the Hall database	SD R5
halls	class call name	the database of cinema halls	Class Model
Hall	lexical domain	a database containing the cinema halls, provided by the cinema operator	CD
http	technical phenomenon	The Hypertext Transfer Protocol. A client-server protocol for requesting and providing data, like webpages, over the internet.	TCD
I			
id	attribute	the unique id of the showing	Class Model
id	attribute	the unique ID of the booking	Class Model
imap	technical phenomenon	Internet Message Access Protocol	TCD
isArchived	attribute	indicates whether the showing is archived	Class Model
J			
K			
L			
LC_User	life-cycle	Life-cycle for one user	LC
LC_Customer	life-cycle	Life-cycle for one logged in customer	LC
LC_NonStaffUser	life-cycle	Life-cycle for one user who is not logged in as staff	LC
LC_UDEKino	life-cycle	Combined life-cycle (all users, customers and internal operations)	LC
M			
MailClient	connection domain	the Customer's E-Mail client	TCD
MailServerCustomer	connection domain	the customer's E-Mail server	TCD
MailServerUDEK	connection domain	the system's E-Mail server	TCD

Table 4.1: Glossary

Name	Type	Description	Source
minutes	parameter	the minutes to be converted to unix epoch time	Class Model
modifyShowing	phenomenon	the machine modifies a showing in the showings database	CD
N			
NonStaffUser	biddable domain	either of Customer or User	PD R4 / R8
NonStaffUser	actor	a user who is not logged in as staff and wishes to browse available showings	SD R4/8
notifyCustomer	phenomenon	the machine notifies the customer via e-mail	CD
nsuBrowse	phenomenon	either of cBrowse or uBrowse	PD R4 / R8
nsuBrowse	message	a request for the WebpageNonStaffUserBrowse to display available showings	SD R4/8
nsuBrowse()	method	the user requests a list of available showings on the webpage	Class Model
nsuShowings	phenomenon	the machine sends a list of upcoming showings to be displayed by the website	PD R4 / R8
nsuShowings	message	contains available, i.e., non-archived, showings	SD R4/8
nsuShowings()	method	the machine sends a set of available showings to the webpage	Class Model
nsuShowShowings	phenomenon	the website displays a list of upcoming showings to the user	PD R4 / R8
nsuShowShowings	message	a rendition of available, i.e., non-archived, showings	SD R4/8
O			
P			
pop3	technical phenomenon	Post Office Protocol - Version 3	
Q			
R			
registrationFailed	phenomenon	the machine notifies the webpage that the registration has failed	PD R1
registrationFailed	message	informs the WebpageRegisterCustomer that account creation has failed	SD R1
registrationFailed()	method	the webpage is notified that the registration was successful	Class Model
registrationFailedNotification	phenomenon	the webpage displays a to the user that the registration has failed	PD R1
registrationFailedNotification	message	informs the user that account creation has succeeded	SD R1

Table 4.1: Glossary

Name	Type	Description	Source
registrationSuccess	phenomenon	the machine notifies the webpage that the registration has succeeded	PD R1
registrationSuccess	message	informs the WebpageRegisterCustomer that account registration has succeeded	SD R1
registrationSuccess()	method	the webpage is notified that the registration was unsuccessful	Class Model
registrationSuccessNotification	phenomenon	the webpage displays a notification to the user that the registration has succeeded	PD R1
registrationSuccessNotification	message	informs the User that account creation has succeeded	SD R1
removeBooking	phenomenon	the machine removes a booking from the bookings database	CD
removeCustomer	phenomenon	the machine removes a customer from the customers database	CD
removeShowing	phenomenon	the machine removes a showing from the showings database	CD
row	attribute	the row of the booked seat	Class Model
row	parameter	the row of the seat that is to be booked	Class Model
rows	attribute	the number of rows of seats the cinema hall contains	Class Model
result	parameter	the set of available showings	Class Model
S			
sBrowse	phenomenon	a staff member browses available showings	CD
sCancelShowing	phenomenon	a staff member attempts to cancel a showing	CD
send	technical phenomenon	the machine sends an e-mail	TCD
session	class call name	the request's session	Class Model
setArchived	message	contains the ID of the showing which is to be marked as archived	SD R7
Showing	lexical domain, designed domain	a database containing the cinema showings	CD
Showing	object	the database containing the showings	SD R5, 4/8, 7
Showing	class	a record representing a showing	Class Model
ShowingHasStarted	guard / state predicate	whether the showing in question has already started, i.e., its starting date and time lies in the past	SD R7
showingID	attribute	the ID of the showing of the booking	Class Model
showingID	parameter	the ID of the showing that is to be booked	Class Model
ShowingIsArchived	guard / state predicate	whether the showing in question is marked as archived"	SD R7

Table 4.1: Glossary

Name	Type	Description	Source
showings	phenomenon	the showings database provides the showings data to the machine	CD
showings	message	contains all showings in the Showing database	SD R5, 4/8, 7
showings sLogin	class call name phenomenon	the database of Showings a user attempts to log in as a staff member	Class Model CD
sLogout	phenomenon	a staff member attempts to log out	CD
SMTP	technical phenomenon	Simple Mail Transfer Protocol	TCD
sShowWebsite	phenomenon	the machine shows a website to the staff member	CD
StaffMember	biddable domain	a member of cinema staff; a user who has logged in as staff	CD
startDateTime	attribute	the date and time the showing will start at in unix epoch time	Class Model
submitBooking	phenomenon	the customer selects the tickets they wish to book and hits the submit button	PD R5
submitBooking	message	contains the showing ID and desired seats	SD R5
submitBooking(in showingID : Integer, in row : Integer, in column : Integer)	method	forwards the booking request to the machine	Class Model
submitRegistration	phenomenon	the user submits a request to register a new customer account, containing an e-mail address and a password	PD R1
submitRegistration	message	a request from the user to register a new customer account, containing an e-mail address and a password	SD R1
submitRegistration(in email : String, in password : String)	method	the method with which the user submits the registration form	Class Model
submitShowing	phenomenon	a staff member submits a new showing to the machine for entry into the database	CD
T			
U			
uBrowse	phenomenon	a user browses available showings	CD
UDEKino	machine	the machine to be developed	CD, TCD
UDEK_ArchiveShowings	machine	the sub-machine responsible for automatically archiving showings once they have begun	PD R7

Table 4.1: Glossary

Name	Type	Description	Source
UDEK_ArchiveShowings	object	the sub-machine responsible for archiving showings which have already started	SD R7
UDEK_ArchiveShowings	class	the machine class	Class Model
UDEK_BookTickets	machine	the sub-machine responsible for customer booking tickets	PD R5
UDEK_BookTickets	object	the machine responsible for the booking of tickets	SD R5
UDEK_BookTickets	class	the machine class	Class Model
udek_bookTickets	class call name	the machine class intance	Class Model
UDEK_NonStaffUserBrowse	machine	the sub-machine responsible for registered and non-registered customers browsing upcoming showings	PD R4 / R8
UDEK_NonStaffUserBrowse	class	the machine class	Class Model
udek_NonStaffUserBrowse	class call name	the instance of the machine class the webpage belongs to	Class Model
UDEK_RegisterCustomer	machine	the sub-machine responsible for customer account registration	PD R1
UDEK_RegisterCustomer	object	the machine responsible for customer account registration	SD R1
UDEK_RegisterCustomer	class	the machine class	Class Model
udek_registerCustomer	class call name	the instance of the machine class the webpage belongs to	Class Model
User	biddable domain	a user of the application who is not logged in	CD, TCD
UserWebBrowser	connection domain	Web browser used by a user who is not logged in, e.g. Mozilla Firefox.	TCD
User	actor	a user of who wishes to register a new customer account	SD R**??**
uShowWebsite	phenomenon	the machine shows a website to the user	CD
V			
W			
WebpageBookTickets	connection domain, designed domain	a webpage via which a customer can book tickets	PD R5
WebpageBookTickets	object	the webpage for booking tickets	SD R5
WebpageBookTickets	class	the class of the webpage for the booking of tickets	Class Model
webpageBookTickets	class call name	the webpage via which the request was sent	Class Model
WebpageNonStaffUserBrowse	connection domain, designed domain	a webpage via which a user can browse upcoming showings	PD R4 / R8
WebpageNonStaffUserBrowse	object	the webpage for NonStaffUsers to browse available showings	SD R4/8
WebpageNonStaffUserBrowse	class	the class representing the webpage for browsing showings	Class Model

Table 4.1: Glossary

Name	Type	Description	Source
webpageNonStaffUserBrowse	class call name	the webpage instance whose request is currently being handled	Class Model
WebpageRegisterCustomer	connection domain, designed domain	a webpage via which a user can register a new customer account	PD R1
WebpageRegisterCustomer	object	the webpage for registering a new customer account	SD R1
WebpageRegisterCustomer	class	the class of the webpage for customer registration	Class Model
webpageRegisterCustomer	class call name	the instance of the registration webpage class whose request is currently being handled	Class Model
X			
Y			
Z			