Daniel Piro 311434526

Bar Rozenman 207172552

Nitay Vitkin 205660699

Omry Arviv 316519131

ID	Module	Category	Description	Priority	Risk	Functional	implemented
1	TP	Logistics	The system will support transportation registration	MH	Low	V	V
2	TP	Administrative	Every truck that leaves must register in the truck management database	MH	Low	V	V
3	TP	Data security	The system must verify that the source of delivery is a supplier	MH	Low	V	V
4	TP	Administrative	The system will provide lists of receipt of products to each destination on a numbered document	MH	Low	V	V
5	TP	Data security	The system will ensure that no driver can be assigned to any truck without a suitable license	MH	Low	V	V
6	TP	Transportation	The delivery locations will be divided to sections for more efficient distribution	MH	Low	V	V
7	TP	Logistics	Whenever is deficit of supplies the system support send a truck for completion of goods	NTH	Low	V	We still don't know what the suppliers support
8	TP	Logistics	In case there is a deviation from the maximum weight the	MH	Low	V	V

			system will redesign the distribution route or reduce products accordingly				
9	TP	Administrative	Destination can be a supplier or company branch address	MH	Low	-	V
10	TP	Administrative	The system will save the phone number , address and name of contact, of the destination	MH	Low	-	V
11	TP	Administrative	The system will save the number, model , net weight, maxweight of the trucks	MH	Low	-	V
12	TP	Administrative	The system will keep a log history of every delivery and receive goods ever made	NTH	Low	V	We need to check if there's a need for it
13	TP	Administrative	The system will keep a log of each document that was taken by driver	NTH	Low	V	We need to check if the client interested in this feature
14	TP	Administrative	The system will alert when there is a weight abnormality	MH	Low	V	V
15	Workers	User Management	The system will save the personal information, and work details of	Must Have	Low	Yes	V

			each and every worker.				
16	Workers	User Management	Human resource manager and his replacement (and only them) may update a worker Info/details.	Must Have	Low	Yes	V
17	Workers	Logistics	A Workday will be Divided Into dayshift and nightshift.	Must Have	Low	No	V
18	Workers	Administration	The system will store all employee's personal constraints regarding what dates and when they can work.	Must Have	High	Yes	V
19	Workers	Logistics	Each shift must have a shift manager.	Must Have	Low	No	V
20	Workers	Logistics	A shift manager must be an employee that is authorized to cancel purchases.	Must Have	Low	No	V
21	Workers	Logistics	Each employee has at least one job and he can only work in the specified jobs he is certified to do.	Must Have	Low	No	V

22	Workers	User Management	An employee must be able to login.	Must Have	High	Yes	V
23	Workers	Administration	The system will notify workers on a weekly basis on their upcoming shift schedule.	Must Have	High	Yes	V
24	Workers	Shift Management	The system will create shifts automatically.	Must Have	High	Yes	V
25	Workers	Shift Management	HR Manager and his replacement (and only them) can alter the shifts.	Must Have	Low	Yes	V
26	Workers	Shift Management	Shifts may only be altered up until the end of the previous week.	Must Have	Low	No	V
27	Workers	Shift Management	A storekeeper must be present in the Store branch that is currently accepting a delivery.	Must Have	High	Yes	V
28	Workers	User Management	The System will supply a summary of all the workers on shift and their corresponding active in position the shift	Must Have	Low	Yes	V

Question ID	Topic	Description
1	HR	Are the drivers employees of the company or employees of a contractor?
2	Transportation	What is the minimum amount for which the system will approve a new delivery?
3	HR	Is it possible to edit a transport form after it has been received in the system? And if so, until when?
4	Transportation	Is it necessary to prioritize the transportation of expensive products over cheap products?
5	Transportation	Should the system support returned items?
6	HR	Can the driver have multiple driver's licenses?
7	Shift management	Q: Are there any other constraints the shift controller should consider when creating a default shift other than personal constraints of the user? For example: an employee may not work more than 6 days in a week.
8	Shift management	Q: Assuming there is enough roles to fill for recommended positions for a certain shift, should the manager be able to overrule the shift and not meet the recommended standard “lineup” of the shift?
9	Shift management	Should it be possible to edit a shift even after the supposed deadline? for instance, an employee was injured during the work week but still has shifts he is assigned to during that week.

Question ID	Topic	Description
1	Transportation	Q:According to what are the delivery areas divided? A: The areas will be divided into 3 sections : North , Middle , South.
2	Transportation	Q: What is the minimum amount for which the system will approve a new delivery? A: At least half a truck full of goods.
3	Transportation	Q: Is it important to determine the driver for the transport? That is, for long trips would we prefer drivers with more experience? Or enough that the .driver has a truck license. A: Any driver with a suitable license may make any .trip.
4	Transportation	Q: What happens when you are overweight? A: It will not be possible to close the transport form.
5	Transportation	Q: Does the lead come from a single place? Can transport pass through several different sections? A: The transport leaves from the same place, and it is also possible that the transport will pass through several different sections
6	Transportation	Q: Who has the permit to take out trucking? Any driver or have a trucking manager?

		A: Will be in charge of transports who is responsible for forwarding the requests for transport to the system
7	Transportation	Q: What happens when a driver's license is changed? A: The license can gain weight but not decrease
8	Worker constraints	Q: are worker constraints set, or may they be changed over time? A: I think the constraints should be able to be updated on a weekly basis. Constraints and personal issues change but I do need those before Thursday night, else I won't be able to take them into consideration.
9	Shift requirements	Q: apart from being certified on cancelling purchases, are there any other requirements in order to be a Shift manager. A: for the time being, each certified purchase canceler, is also certified to be shift manager.
10	Employee	Q: when stated "…wage, and terms of employment…" What are the terms of employment of a worker that the system should save? A: wage, advanced study funds, no. sick days, no. days off.
11	Employee	Q: should wage be set per job or per employee? Example: Sam and Sandra are both cashiers, Sam wage is 15$ per hour while Sandra makes 20$ per hour. A: We currently have different work contracts with different employees. We have 1 contract per employee and the Employee's wage is a set amount listed there which refers to all types of work the employee might provide.

Presentation layer

Main

Driver_Menu

- controller: Facade

+ mainMenu(controller:Facade): void
- loadData(controller:Facade): void
- printDeliveryMenu(controller:Facade): void
- removeDeliveryDocument(controller:Facade): void
- addDeliveryDoc(controller:Facade): void
- removeDelivery(controller:Facade): void
- addNewDelivery(controller:Facade): void
- printUpdateDeliveryDocumentMenu(controller:Facade): void
- printUpdateDeliveryMenu(controller:Facade): void
- printTruckMenu(controller:Facade): void
- removeTruck(controller:Facade): void
- addNewTruck(controller:Facade): void
- printTruckUpdateMenu(controller:Facade): void
- updateDeliveryDriver(controller:Facade): void
- updateDeliveryTruck(controller:Facade): void
- updateDeliveryDayDeprature TimeArrivalTime(controller:Facade): void
- updateDeliveryMaxWeight(controller:Facade): void
- updateTruckMaxWeight(controller:Facade): void
- printLocationMenu(controller:Facade): void
- addNewSupplyToDocument(controller:Facade): void
- removeSupplyFromMenu(controller:Facade): void
- removeLocation(controller:Facade): void
- addNewLocation(controller:Facade): void
- printLocationUpdateMenu(controller:Facade): void
- updateLocationContactName(controller:Facade): void
- updateLocationPhone(controller:Facade): void
- updateLocationAddress(controller:Facade): void
- printDriverMenu(controller:Facade): void
- removeDriver(controller:Facade): void
- addNewDriver(controller:Facade): void
- printUpdateDriverMenu(controller:Facade): void
- printDriverLicenseType(controller:Facade): void

Employee_Menu

facade: ApplicationFacade

scanner: Scanner

+deploy(): void

FacadeLayer

DriverFacade

- tpController: TPController
- tpController: TPController

+ addDelivery(departureDay:LocalDate, departureTime:LocalTime,arrivalTime:LocalTime,truckNum:int,source:LocationFacade,destination::List<LocationFacade>,truckWeight:double,driverID:int): ResponseT<DeliveryFacade>
+ addDocument(id:int, dest:String,supplies:List<SuppliesFacade>,deliveryID:int): ResponseT<DeliveryDocFacade>
+ addTruck(licenseNum:int, model:String,netoWeight:double,maxWeight:double): ResponseT<TruckFacade>
+ addNewSupplyToDoc(deliveryID:int,docID:int,supplies:int): ResponseT<TruckFacade>
+ getSupply(id:int): ResponseT<SuppliesFacade>
+ addLocation(id:int, name:String,phone:String,contactName:String,idNameOfSection:int): ResponseT<LocationFacade>
+ addNewDriver(id:int, name:String,licenseType:String): ResponseT<DriverFacade>
+ removeDelivery(id:int): Response
+ removeTruck(licenseNum:int): Response
+ updateDeliveryTime(deliveryID:int, departureDay:LocalDate,departureTime:LocalTime,arrivalTime:LocalTime): Response
+ updateDeliveryTruckWeight(deliveryID:int, newWeight:double): Response
+ updateDeliveryTruck(deliveryID:int, truckID:int): Response
+ updateLocationAddress(id:int, name:String): Response
+ updateLocationPhone(id:int, phone:String): Response
+ updateLocationContactName(id:int, contactName:String): Response
+ updateLocationLicense(driverID:int, newType:String): Response
+ updateDeliveryDriver(deliveryID:int, driverID:int): Response
+ removeLocation(id:int): Response
+ removeSupplyFromDocument(deliveryID:int,docID:int,supplies:int): Response
+ removeDeliveryDoc(id:int): Response
+ removeDriver(id:int): Response
+ updateLocationAddress(id:int,name:String): Response
+ updateLocationPhone(id:int,phone:String): Response
+ updateLocationContactName(id:int,contactName:String): Response
+ updateTruckMaxWeight(licenseNum:int,maxWeight:double): Response

Response<T>

+value:T

Response

+ errorMessage: string
+ errorOccured: boolean

ApplicationFacade

employeeController: EmployeeController
shiftController: ShiftController
+ApplicationFacade()
+createShift()
+getShift(Date date, ShiftType shiftType): Response
+getEmployeesForJob(String input): Response
+upDateShift(Date date, ShiftType shiftType, int upDateShiftOp): Response
+registerRegWorker(String f_Name, String l_Name, int id, int wage, double monthlyDayOff, double monthlySickDays, int bankId, String branch, int accountNumber, int advancedStudyFunds, Branch workplace, boolean shiftManager, List<Job> certifiedWorks): Response
+registerHRManager(String f_Name, String l_Name, int id, int wage, double monthlyDayOff, double monthlySickDays, int bankId, String branch, int accountNumber, int advancedStudyFunds,Branch workplace): Response
+change_in_the_number_of_employees_on_shift(int dayOfWeek, ShiftType shiftType, Job job, int number): void
+get_number_of_employee_for_job_on_shift(int i,ShiftType shiftType,Job job): int
+get_number_of_employees_on_shift(int i,ShiftType shiftType): String
+sLogin(int id): Response
+isActive(): Response
+thisWeekShifts()
+nextWeekShifts()
+getSpecificShift(Date date): Response
+logout(): void
+changeConstraints(boolean[][] processedConstraints): Response
+showConstraints(): String
+changeConstraints(int dayOfWeek, int shiftType, boolean constraint): Response
+remove(Date date, ShiftType shiftType, int upDateShiftOp): Response
+add(Date date, ShiftType shiftType, int add,String input): Response
+getShiftManager(Date date, ShiftType shiftType): Response
+replice(Date date, ShiftType shiftType, int id): Response
+FireEmployee(int id): Response

EmployeeFacadeObjects

<<abstract class>>
FacadeEmployee

f_Name:String
l_Name:String
id: Int
wage: Int
monthlyDaysOff: double
monthlySickDays: double
advancedStudyFunds: int
+getSpecificShift: Shift
+thisWeekShifts: String
+nextWeekShifts: String

FacadeShift

-type
-day:int
-shiftManager: FacadeRegularWorker
-regularWorkerList: Map<FacadeRegularWorker,Job>

FacadeShift

-type
-day:int
-shiftManager: FacadeRegularWorker
-regularWorkerList: Map<FacadeRegularWorker,Job>

FacadeHuman Resources Manager

Employee's Bank info

FacadeRegular worker

shift manager: bool
personalConstraints: bool[][]
certifiedWorks: List<Job>
personalShiftRec: Map <Date, Shift>
+checkAvailability: boolean
+checkIfOccupiedInDate: boolean
+insertShiftRecord: void
+removeShift: void

FacadeBankInfo

bankId: int
Branch: string
Account Number: int

DriverFacadeObjects

SectionFacade
+ areas: List<LocationFacade>

DeliveryDocFacade
+ id: int
+ dest: String
+ supplies: List<SuppliesFacade>

TruckFacade
+ licenseNum: int
+ model: String
+ netoWeight: double
+ maxWeight: double

LocationFacade
+ id: int
+ address: String
+ phone: String
+ contactName: String

DriverFacade
+ id: int
+ name: String
+ licenseType: String

SuppliesFacade
+ id: int
+ name: String

DeliveryFacade
+ id: int
+ departureDay: LocalDate
+ departureTime: LocalTime
+ arrivalTime: LocalTime
+ truckNum: int
+ source: LocationFacade
+ destintaion: List<LocationFacade>
+ docs: List<DeliveryDocFacade>
+ truckWeight: double
+ driverName: String
+ driverID: int

BusinessLayer

controlledBy

Drivers

DriverController

+ addDriver(id:int,name:String,licenseType:String): Driver
+ removeDriver(id:int): void

Drivers

Deliveries

TpController

+ addDelivery(departureDay:LocalDate, departureTime:LocalTime,arrivalTime:LocalTime,truckNum:int,source:LocationFacade,destination::List<LocationFacade>,deliversDocs:List<DeliveryDocFacade>,truckWeight:double,driverID:int): ResponseT<DeliveryFacade>
+ addDocumentation(id:int,dest:String,supplies:List<supplies,deliveryID:int): DeliveryDoc
+ removeDelivery(id:int): void
+ updateDeliveryTime(delivery:Delivery): void
+ updateDriver(deliveryID:int,driverID:int,driverName:String): void
+ addLocation(id:int,name:String,phone:String,contactName:String,secName:String): Location
+ addSupplyToDoc(deliveryID:int,docID:int,supplies:int): void
+ removeLocation(id:int): void
+ checkWeightLicense(license:String,maxWeight:double): void
+ removeSupplyFromDoc(deliveryID:int,docID:int,supplies:int): void
+ removeDeliveryDoc(id:int): void
+ updateNameOfLocation(id:int,newName:String): void
+ checkWeightLicense(license:String,maxWeight:double): void
+ checkDeliveryDest(nameLoc:String): void
+ updatePhoneOfLocation(id:int,phone:String): void
+ updateDeliveryTruckWeight(deliveryID:int,newWeight:double): void
+ updateDeliveryTime(deliveryID:int,departureDay:LocalDay,departureTime:LocalTime,arrivalTime:LocalTime,truckNum:int): void
+ updateTruckOfTheDelivery(deliveryID:int,truckNum:int): void
+ timeCheck(deliveryID:int): void
+ updateContactName(id:int,newName:String): void
+ addTruck(licenseNum:int,model:String,netoWeight:double,maxWeight:double): Truck
+ removeTruck(licenseNum:int): void
+ updateTruckMaxWeight(licenseNum:int,maxWeight:double): void

Trucks

Truck

- licenseNum : int
- model: string
- netoWeight: double
- maxWeight: double

1

Truck

Deliveries

1...n

Delivery

- id: int
- deprtureDay: LocalDate
- deptureTime: LocalTime
- arrivalTime: LocalTime
- truckNum: int
- truckWeight: double
- driverName: String
- driverID: int

0...n

Locations

Section

+ removeLocation(location:Location): void
+ addLocation(areaName:String,location:Location): void

1

Locations

1...n

Deliverydocs

1

1...n

DeliveryDoc

- id: int
- dest: String

- addSupply(supply:Supplies): void
- removeSupplies(id:int): void

Location

- id: int
- address: string
- phone: string
- contactName: string

1...n

Branch Location

Supplies

- id: int
- name: string

Supplies

1...n

Business layer

<<singleton>>
EmployeesController

+Login: boolean
+Logout: boolean
+empExists: boolean
+registerRegularWorker: void
+testRegisterRegularWorker: boolean
+registerHRManager: boolean
+getEmployeedById: Employee
+isHRManager: Employee
+getShiftManagerJob: Employee
+jobCertifiedAvailableWorkers: List<Employee>
+isActive: boolean
+changeWorkersInfo: boolean
+getSpecificShift: Shift

0...*

Employee list

current Worker

1

<<abstract class>>
Employee

f_Name:String
l_Name:String
id: Int
wage: Int
monthlyDaysOff: double
monthlySickDays: double
advancedStudyFunds: int
+getSpecificShift: Shift
+thisWeekShifts: String
+nextWeekShifts: String

1

Human Resources Manager

jobList: List<job>

Employee's Bank info

Employee's Positions

Regular worker

shift manager: bool
personalConstraints: bool[][]
certifiedWorks: List<Job>
personalShiftRec: Map <Date, Shift>
+checkAvailability: boolean
+checkIfOccupiedInDate: boolean
+insertShiftRecord: void
+removeShift: void

BankInfo

bankId: int
Branch: string
Account Number: int

Driver

- licenseType: string

loads data

<<enumeration>>
job

Courier
Cashier
Stockkeeper

<<singleton>>
ShiftController

+createMultiShift(): boolean
+createShift(LocalDate localDate, ShiftType shiftType, int day, Map<RegularWorker, AtomicInteger> workerIntegerMAP): boolean
+change_in_the_number_of_employees_on_shift(int i, ShiftType shiftType, Job job, int number):void
+addShift(Tuple<Date, ShiftType> typeTuple, Shift shift)
+FireEmployee(int id): boolean

<<enumeration>>
ShiftType

Day
Night

recommendedLineUp

1

0...*

current Shift

Shift History

loads data

Shift

type: ShiftType
day: int
+Shift(FacadeShift facadeShift)
+Registration_for_shift(RegularWorker regularWorker,Job job)
+addManager(RegularWorker regularWorker, Job job)
+getJob(RegularWorker regularWorker): Job
+removeWorker(RegularWorker replace): boolean

Data Layer

<<singleton>>
Repository

DTO

Shift - Assigned WorkerOTO

-shift date: real
- type: string
-job: string
-Employee id: int

Shift - RecommendedLineUpDTO

- day: int
- type: string
-job: string
-number of employee: int

RegularWorkerDTO

- id: int
-shiftManager
-SUNDAY day: boolean
-SUNDAY night: boolean
-MONDAY day: boolean
-MONDAY night: boolean
-TUESDAY day: boolean
-TUESDAY night: boolean
-WEDNESDAY day: boolean
-WEDNESDAY night: boolean
-THURSDAY day: boolean
-THURSDAY night: boolean
-FRIDAY day: boolean
-FRIDAY night: boolean
-SATURDAY day: boolean
-SATURDAY night: boolean

EmployeeDTO

- id: int
- f_name: string
- l_ name: string
-wage: int
-monthlyDayOffs: int
-monthlySickDays: int
-advancedStudyFunds: int
-bankId: int
-branch: string
-accountNumber: int
-locationId: int

ShiftDTO

- day: int
- type: string
shiftmanagerId: int
-date: real

SectionDTO

- id: int
- location: string

DeliveryDocDTO

- id: int
- location: int

TruckDTO

- licenseNum : int
- model: string
- netoWeight : double
- maxWeight: double

DriverDTO

- id: int
- name: string
- licenseType: string

LocationDTO

- id: int
- address: string
- phone: string
- contactName: string

DeliveryDTO

- id: int
- date: Date
- deptureTime: Date
- driverName: string
- truckNum: int
- source: Location

Employee -

an external actor to the system. A general term describing all actors which Divides into sub-categories listed below.

HR Manager (HRM) -

an Employee with the exclusive authority to modify employee's\shift's details. Updating any personal information of an employee, or changing the employee's working in a specific shift etc.

Registration of a new Employee of any kind are executed by the HR manager. Creation of all shifts are executed by the HR manager.

HR manager may also alter the recommended line up of a shift (explained below).

Regular Worker (RW) -

The employee's which are assigned to the basic roles in "Super LI". Each RW must be qualified for at least one type of the jobs listed: Storekeeper, Driver, Guard, Cashier, Usher. the RW can change his personal constraints, look at the shifts he is assigned to for this week and the upcoming weeks, and can look at the history a specific shift he worked in by entering the shift details.

Recommended lineup -

for each type of shift (e.g. "Monday Night shift"), a default list of each worker's usually needed for that shift is referred to the recommended lineup. The recommended lineup is used for the creation of a shift and the RW occupying the shift, automatically.

Personal constraints -

The constraints of all RW concerning on what days and shift type each RW is available to work what. RW and Only RW have personal constraints.

Shift –

Stores the information regarding which employees are on duty, when, where and what job they are currently assigned to.
date, type of shift (day or night), shift manager (id of employee) and list of employees registered for this specific shift and the type of job they perform in that specific shift (an employee may be authorized to perform several jobs in "Super LI").
Each shift has a recommended lineup. Each shift must have a shift manager. Without a shift manager it is not possible to create one a shift.

it is possible to create a shift with deficiencies in the amount employees required according to the recommended lineup. The system will notify the HRM in such case.

Truck : Super Lee has several branches to which goods need to be delivered, in order to do this there is a registration of trucks that perform transports, each truck has a characterization of weight that can be carried and thus we will know how to fit each driver to a relevant truck and also know that its weight does not exceed.

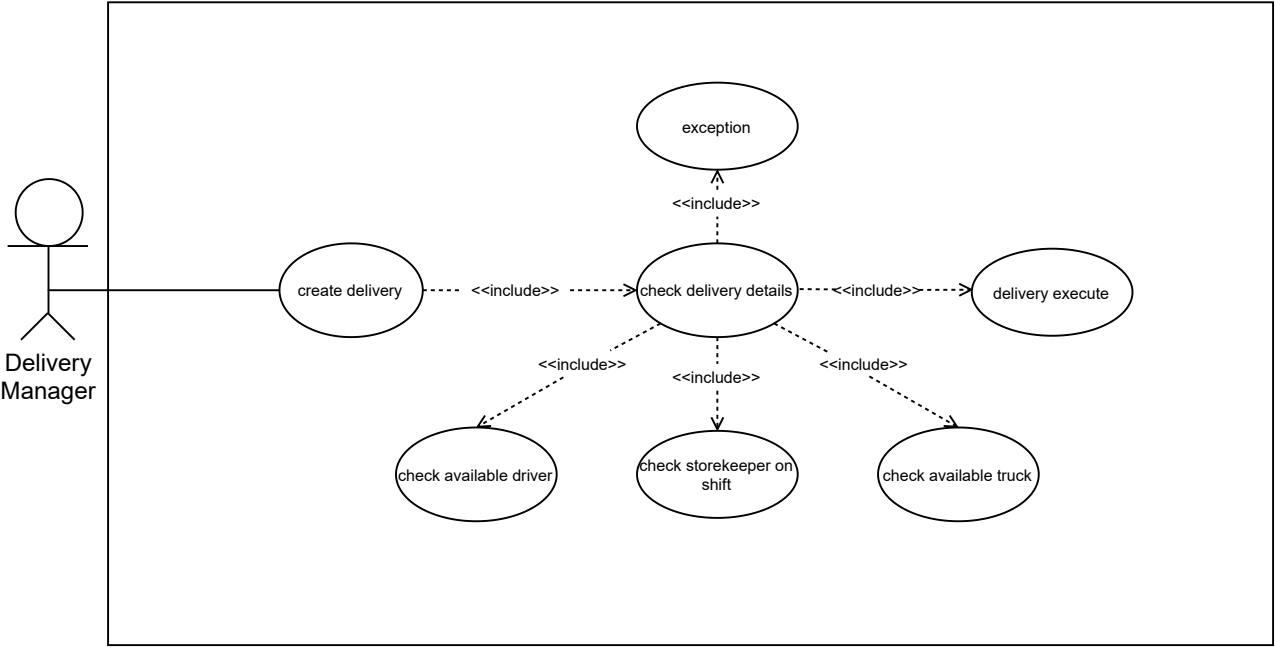
Location : Location is designed to store branch information.

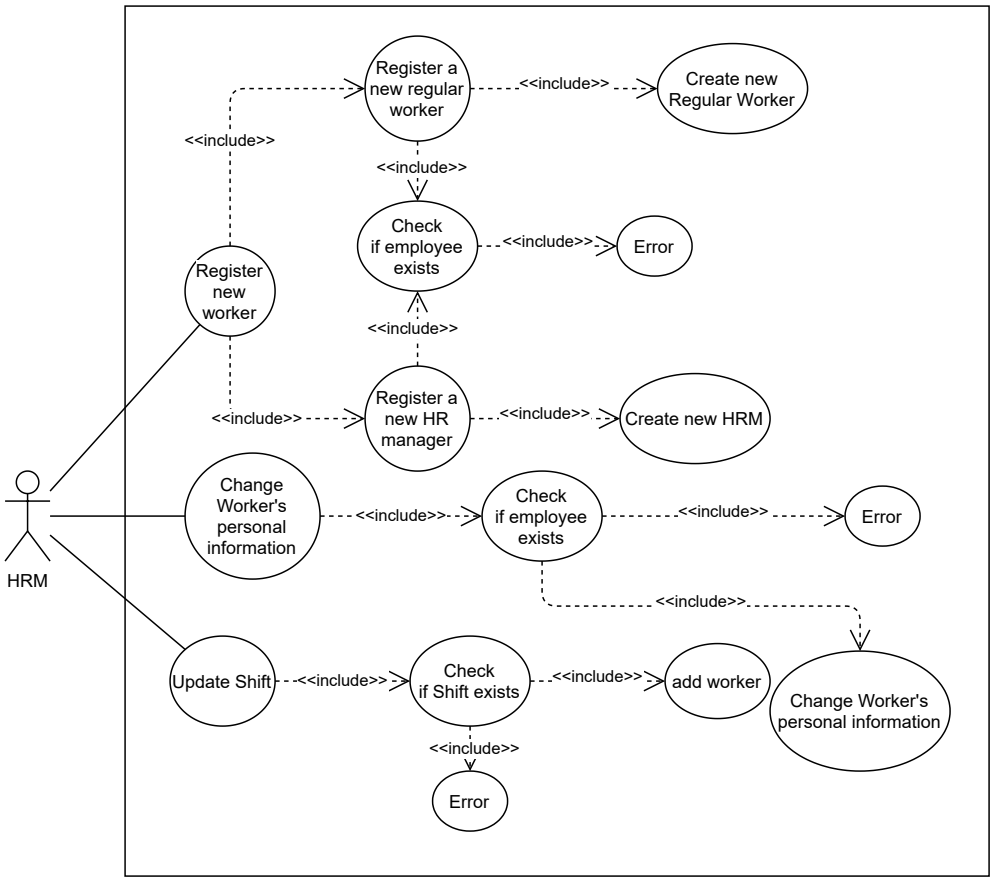
Section : The locations of the branches are divided into 3 shipping areas north, center and south.

Driver : Designed to carry out transports using the trucks in the system, each driver has his own details and there is his own type of license, according to the license we will know which truck to fit the specific driver.

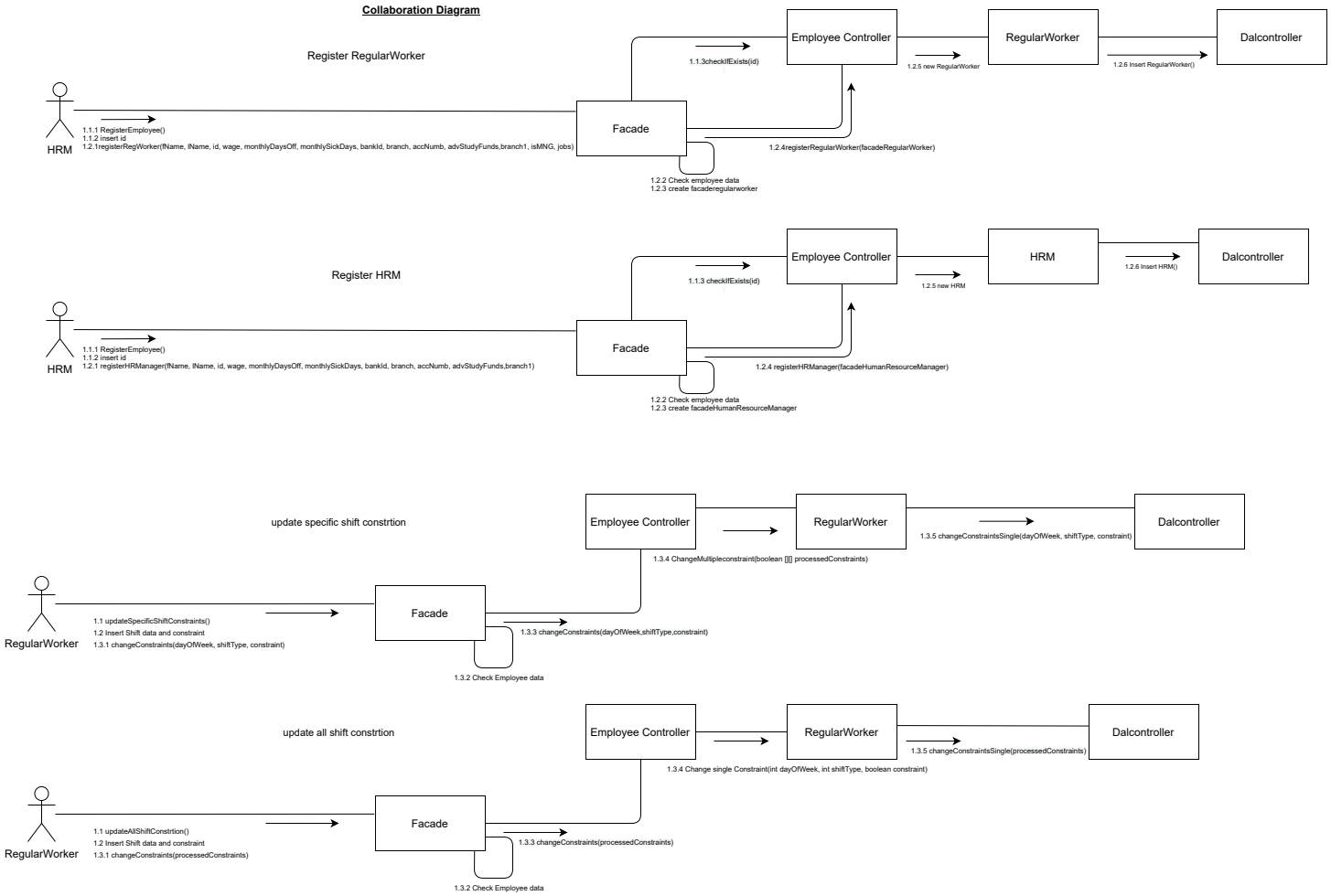
Delivery : The person in charge of transport is the one who enters the details of the transport that leaves. For each shipment he will have to give the date of shipment and the exact weight of the truck leaving, as well as each shipment he will have shipping documents for each destination.

DeliveryDoc : For each delivery destination the person in charge of trucking will need to enter the details of the products we are transporting to the destination, so that we can get a list for each destination with the products in an orderly fashion.

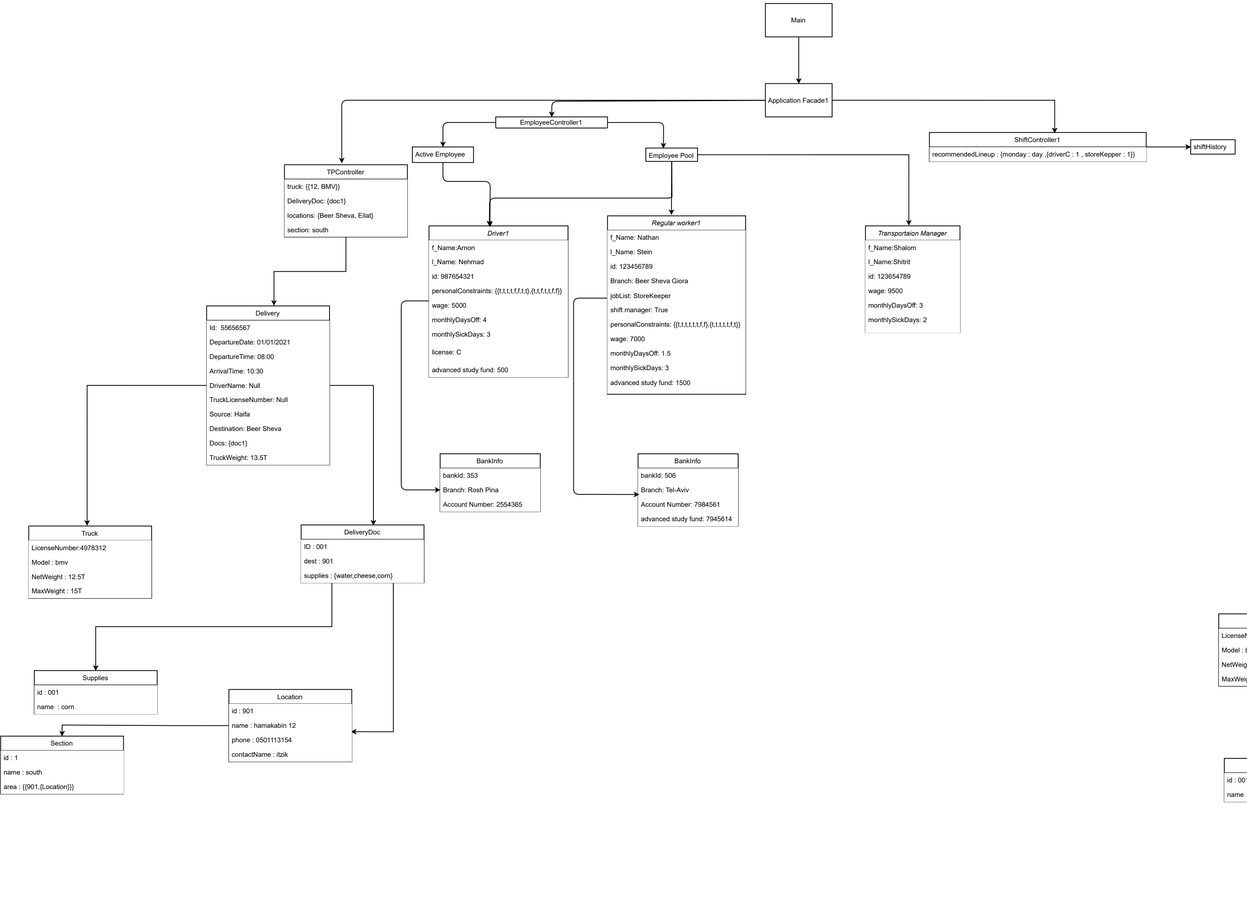

exception
<<include>>
Delivery Manager
create delivery
<<include>>
check delivery details
<<include>>
delivery execute
<<include>>
<<include>>
<<include>>
check available driver
check storekeeper on shift
check available truck


Register a new regular worker
<<include>>
Create new Regular Worker
<<include>>
<<include>>
Check if employee exists
<<include>>
Error
Register new worker
<<include>>
<<include>>
Register a new HR manager
<<include>>
Create new HRM
HRM
Change Worker's personal information
<<include>>
Check if employee exists
<<include>>
Error
<<include>>
Update Shift
<<include>>
Check if Shift exists
<<include>>
add worker
Change Worker's personal information
<<include>>
Error

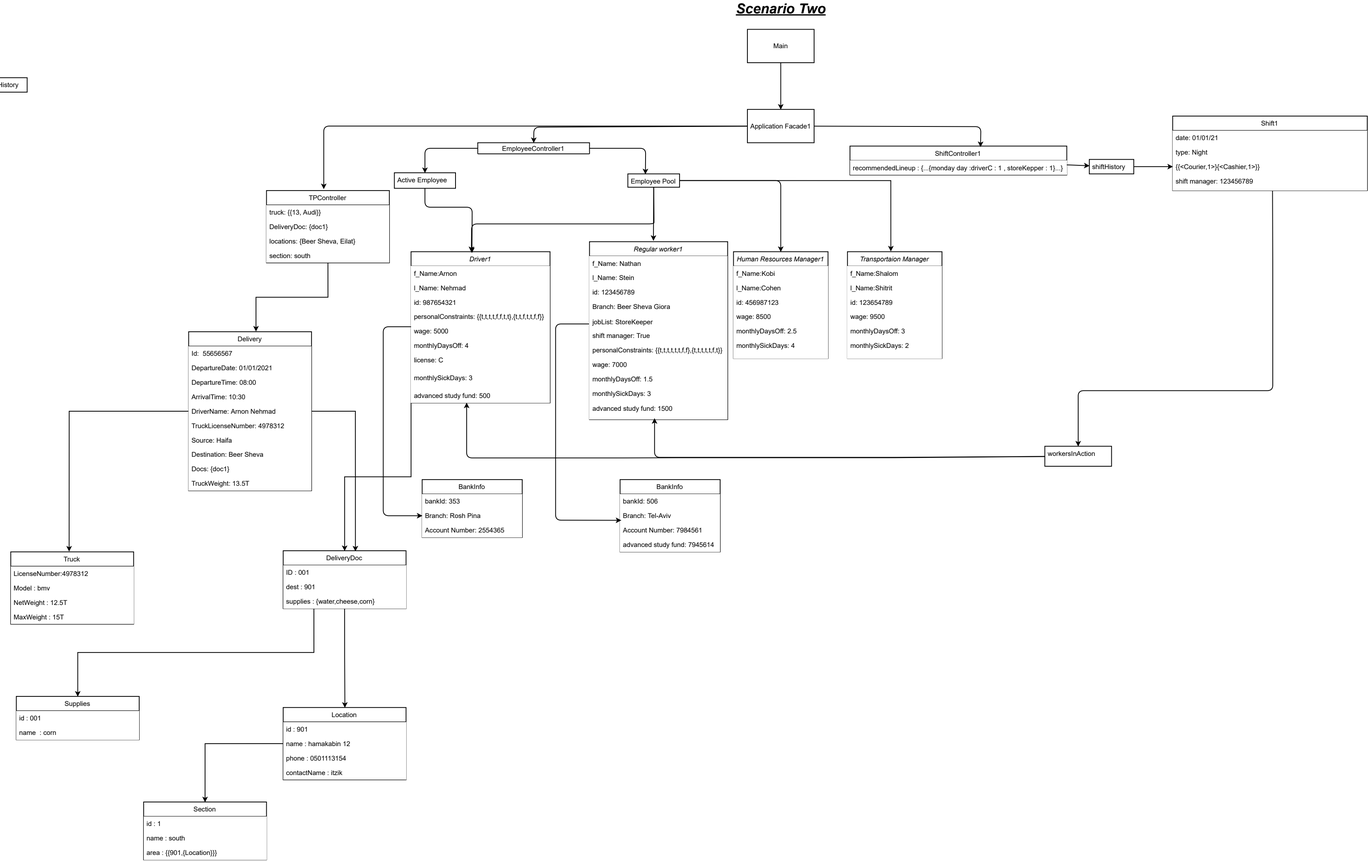
Use case name	Add worker to shift
Textual Description	Adding Regular worker to an existing shift
List of Actors	HRM
Pre-conditions	There is at list one worker(that can do the selected job) that are available. The update shift (the shift we adding the worker to) are exists.
Post-conditions	The worker are registered to the update shift. The shift enroll to the shift History of the added worker.
Main success scenario	1. The HRM login to the system. 2. The HRM select Update shift. 3. The HRM enters the date and type of the shift. 4. The HRM select to add worker. 5. The HRM select type of Job for the shift. 6. The system print list of worker who can work at that shift at this job. 7. The HRM enters id of one of the employee that in the available list. 8. The system register the worker to the shift and the shift enroll to the worker shift history list.
Alternatives/Extensions	If shift is not exists the system returns to the main menu of HRM. If worker is not exists or can not work at this shift the system returns to the main menu of HRM


Collaboration Diagram
Register RegularWorker
Employee Controller
RegularWorker
Dalcontroller
1.1.3checkIfExists(id)
1.2.5 new RegularWorker
1.2.6 Insert RegularWorker()
HRM
1.1.1 RegisterEmployee()
1.1.2 insert id
1.2.1registerRegWorker(fName, lName, id, wage, monthlyDaysOff, monthlySickDays, bankId, branch, accNumb, advStudyFunds,branch1, isMNG, jobs)
Facade
1.2.4registerRegularWorker(facadeRegularWorker)
1.2.2 Check employee data
1.2.3 create facaderegularworker
Register HRM
Employee Controller
HRM
Dalcontroller
1.1.3 checkIfExists(id)
1.2.5 new HRM
1.2.6 Insert HRM()
HRM
1.1.1 RegisterEmployee()
1.1.2 insert id
1.2.1 registerHRManager(fName, lName, id, wage, monthlyDaysOff, monthlySickDays, bankId, branch, accNumb, advStudyFunds,branch1)
Facade
1.2.4 registerHRManager(facadeHumanResourceManager)
1.2.2 Check employee data
1.2.3 create facadeHumanResourceManager
update specific shift constrtion
Employee Controller
RegularWorker
Dalcontroller
1.3.4 ChangeMullipleconstraint(boolean [][] processedConstraints)
1.3.5 changeConstraintsSingle(dayOfWeek, shiftType, constraint)
RegularWorker
1.1 updateSpecificShiftConstraints()
1.2 Insert Shift data and constraint
1.3.1 changeConstraints(dayOfWeek, shiftType, constraint)
Facade
1.3.3 changeConstraints(dayOfWeek,shiftType,constraint)
1.3.2 Check Employee data
update all shift constrtion
Employee Controller
RegularWorker
Dalcontroller
1.3.4 Change single Constraint(int dayOfWeek, int shiftType, boolean constraint)
1.3.5 changeConstraintsSingle(processedConstraints)
RegularWorker
1.1 updateAllShiftConstrtion()
1.2 Insert Shift data and constraint
1.3.1 changeConstraints(processedConstraints)
Facade
1.3.3 changeConstraints(processedConstraints)
1.3.2 Check Employee data

Scenario One



Scenario Two



TPM – Transport Manager

HRM – Human resource Manager

Sequence 1:

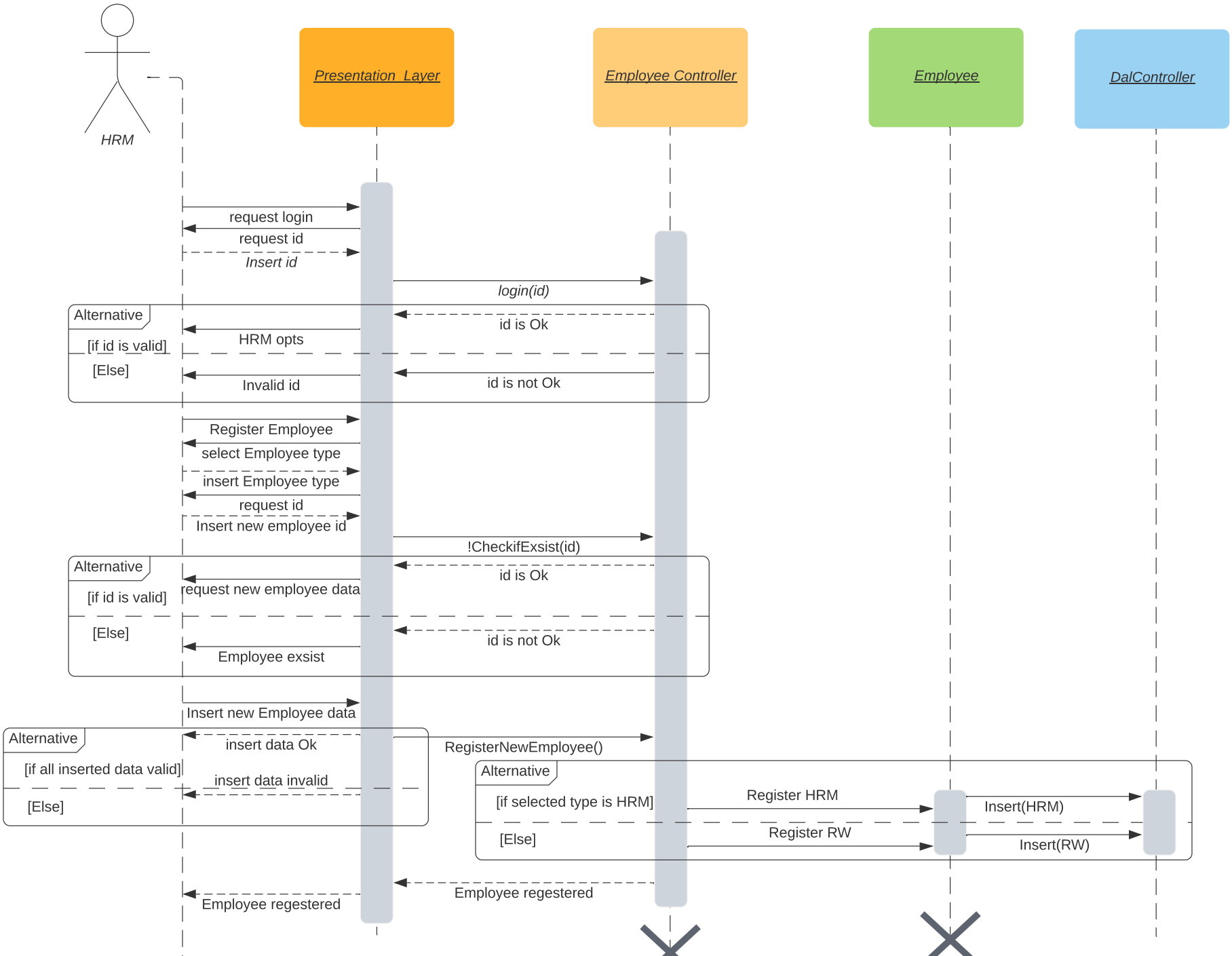
1. The TPM logins into the system using his ID
2. The TPM requests to schedule a new Delivery to Monday next week.
3. The System requests details of the Transport. The Date: 01, departure time, arrival time, driver's name, source of the delivery, destination, docs info, and truck weight in ton.
4. The TPM fills in the requested details
5. The Shift Controller Checks if a driver that is available on shift during the delivery hours. The Shift Controller did not find any driver on shift during those hours that meets the criteria.
6. The Shift Controller Changes the recommended lineup for the upcoming shift on Monday Morning, incrementing the number of drivers with the fitting license needed, by one.
7. If no storekeeper in the recommended lineup of the shift In the Destination's location, the Shift Controller inserts that 1 storekeeper to the recommended line up.
8. The TPM Logs out.

Sequence 2:

1. HRM logs in.
2. On Thursday, HRM creates shifts.
3. The Employee controller returns lists of all available workers (considering personal constraints) for the jobs needed.
4. The Shift controller Assigns workers automatically in according to the recommended line up of the shift and succeeds.
5. HRM logs out
6. TPM logs in
7. after the shifts have been made, the TPM requests to assign drivers to the upcoming deliveries.
8. The TP controller automatically Assigns driver to the delivery from the drivers on duty and that are available during the delivery hours.
9. TPM logs out.

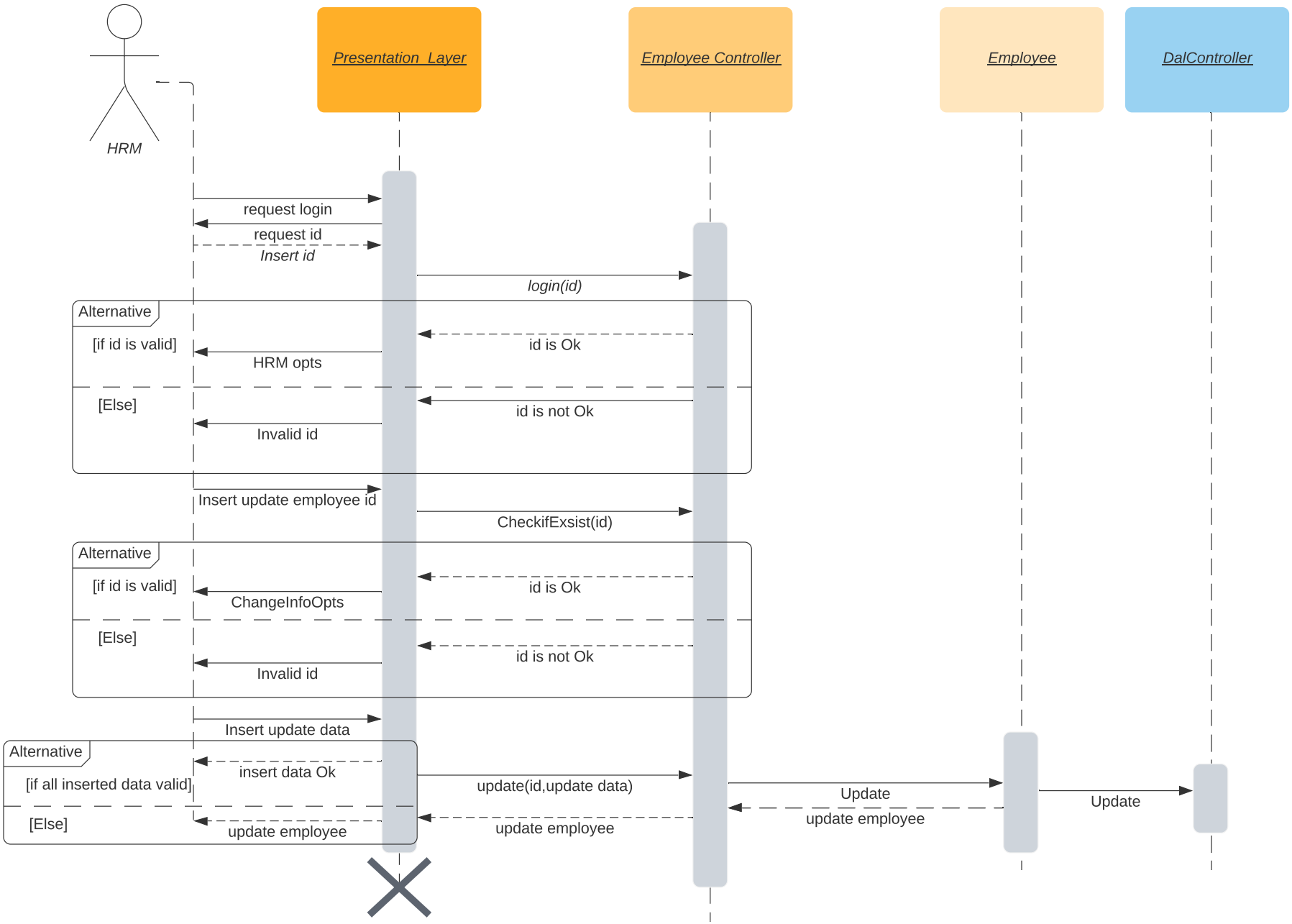
Sequence diagram - Register Employee

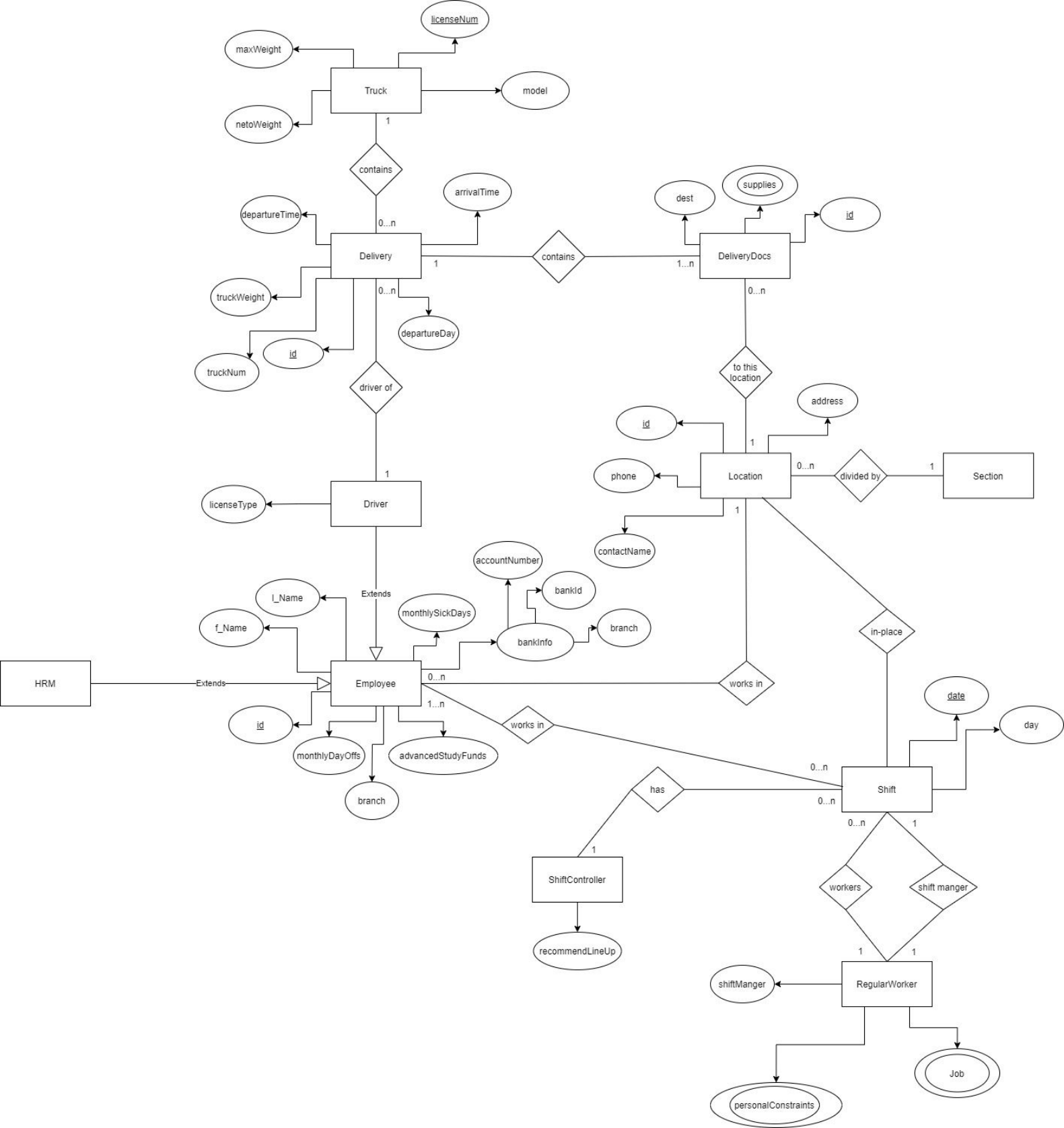
Nitay Vitkin | April 28, 2021



Sequence diagram - update Employee - final

Nitay Vitkin | April 28, 2021




licenseNum
maxWeight
Truck
model
netoWeight
1
contains
arrivalTime
departureTime
0...n
Delivery
1
contains
1...n
DeliveryDocs
dest
supplies
id
truckWeight
0...n
departureDay
id
truckNum
driver of
0...n
to this location
address
id
1
phone
Location
0...n
divided by
1
Section
1
licenseType
Driver
1
contactName
accountNumber
bankId
Extends
l_Name
monthlySickDays
f_Name
bankInfo
branch
in-place
HRM
Extends
Employee
0...n
works in
1...n
works in
date
day
id
monthlyDayOffs
advancedStudyFunds
branch
0...n
has
Shift
0...n
0...n
1
1
ShiftController
workers
shift manger
recommendLineUp
1
1
shiftManger
RegularWorker
Job
personalConstraints

Appendix changes in uml:

1.After integrating the systems, we decided that we would create an operation that would select a driver for transportation, unlike the first job where the person in charge of transportation had to select the appropriate driver.

2. Driver is now integrated to the employee, redundant fields covered by the parent class have been removed.

3. We have added the data layer, in which we will save the details of the system.

4. Added an employee class which is extended by all sub types of Employees that have been added and that will be added upon further integration of the project.