Transport Company Computerization Software UML Diagram

Sayan Mandal

14CS30032

Sourav Pal

14CS10062

CONTENTS

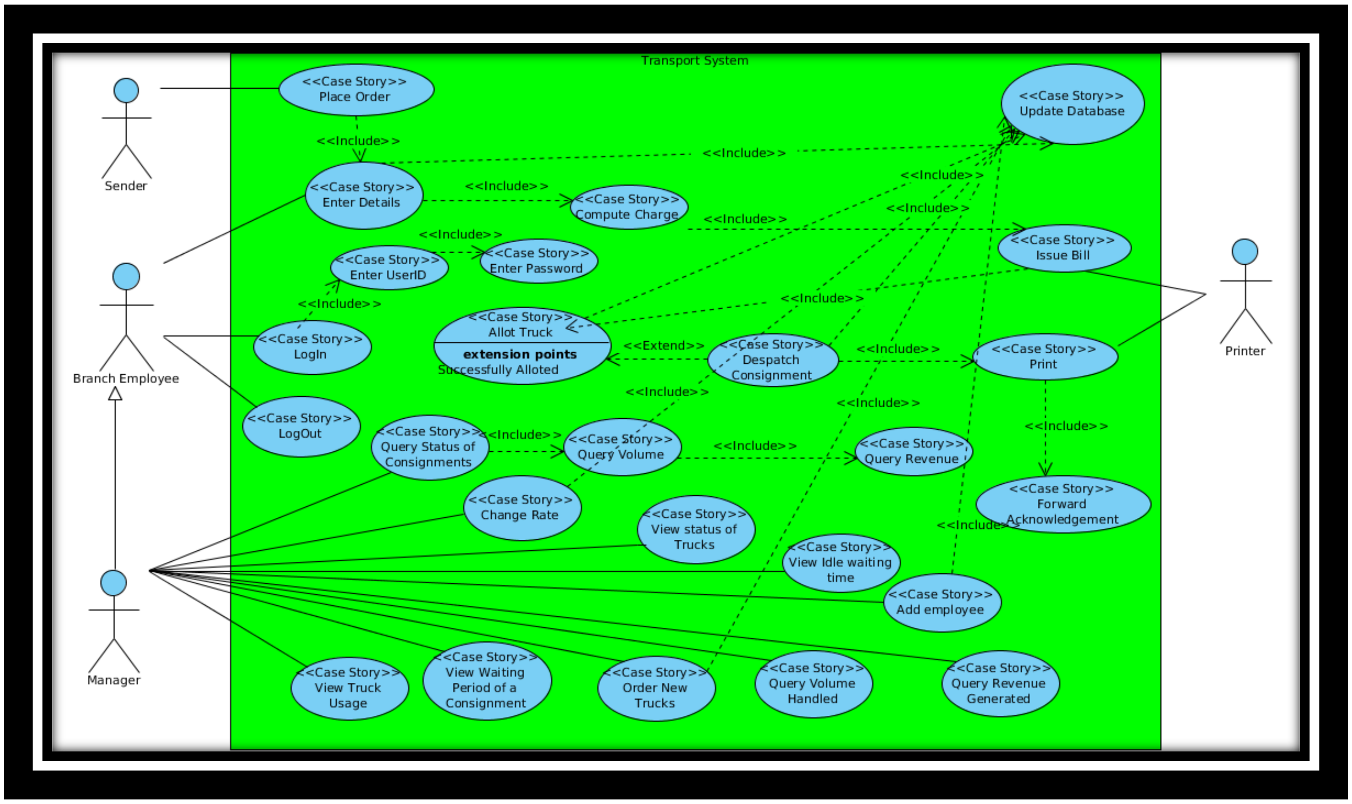
1. Use Case Diagram

The purpose of this diagram is to demonstrate how objects will interact with TCCS and map out the basic functionality of the system. Below is a list of the elements that you will see in the diagram on the next page as well what is included in the use case templates that follow.

Elements	Description
Actors	Shown in the diagram as stick figures with a name underneath. They represent elements that will be directly interacting with the system.
Use Cases	Oval shapes that have their names in the center. These represent direct functionality within the system that must be implemented.
Interactions	Lines that connect the actors with the different Use Cases. These show that there is some form of direct interaction between the actor and that specific functionality.
Includes	Dotted lines labelled "<<include>>" that connect two use cases and have an arrow pointing towards one. This means that the use case without the arrow calls on the functionality of the use case with the arrow.
System Boundary	The large rectangle that contains the Use Cases. Everything within the rectangle is what the system is responsible for implementing

Type	A field in the use case template that states whether or not the use case is directly interacted with by an actor (Primary) or not (Secondary) as well as whether or not it is essential to having a functioning system.

Use-Cases	A field in the use case templates that state which other use cases must be executed prior to that particular use case.


Transport System
Sender
Branch Employee
Manager
Printer
<<Case Story>> Place Order
<<Case Story>> Update Database
<<Include>>
<<Case Story>> Enter Details
<<Case Story>> Compute Charge
<<Case Story>> Enter UserID
<<Case Story>> Enter Password
<<Case Story>> Issue Bill
<<Case Story>> LogIn
<<Case Story>> Allot Truck
extension points
Successfully Alloted
<<Extend>>
<<Case Story>> Despatch Consignment
<<Case Story>> Print
<<Case Story>> LogOut
<<Case Story>> Query Status of Consignments
<<Case Story>> Query Volume
<<Case Story>> Query Revenue
<<Case Story>> Forward Acknowledgement
<<Case Story>> Change Rate
<<Case Story>> View status of Trucks
<<Case Story>> View Idle waiting time
<<Case Story>> Add employee
<<Case Story>> View Truck Usage
<<Case Story>> View Waiting Period of a Consignment
<<Case Story>> Order New Trucks
<<Case Story>> Query Volume Handled
<<Case Story>> Query Revenue Generated

2.2 Use Cases

In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human or an external system. In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals.

2.2.1 Use Case: Log in

Actors: Managers and Employees

Type: Primary and Essential

Description: Using the login panel, Manager and Employees log into their account. In this panel they are first asked to give their username and password.

Includes: Enter password

Use-Cases: None

2.2.2 Use case: Place Order

Actors: Senders

Type: Primary and essential

Description: In this use case, the sender places the consignment to be sent to the office.

Includes: Enter details of the consignment

Use-Cases: None

2.2.3 Use case: Enter details

Actors: Employees and Managers

Type: Primary and essential

Description: In this use case, Employees enter the details of the consignment received at the office, such as volume, destination address, and sender address into the computer.

Includes: Compute Charge, Update database

Use-Cases: Place order

2.2.4 Use case: Compute Charge

Actors: None

Type: Primary and essential

Description: In this use case, the computer would compute the transport charge depending upon the volume of the consignment and its destination and would issue a bill for the consignment.

Includes: Issue Bill

Use-Cases: Enter details

2.2.5 Use Case: Issue Bill

Actors: Printer

Type: primary and essential

Description: In this use case, the computer would compute the transport charge depending upon the volume of the consignment and its destination and would issue a bill for the consignment.

Includes: Allot truck

Use-Cases: Compute Charge

2.2.6 Use Case: Allot truck

Actors: None

Type: Primary and Essential

Description: In this use case, once the volume of any particular destination becomes 500 cubic meters, the computerization system should automatically allot the next available truck.

Includes: Update Database

Extends: Dispatch Consignment

Use-Cases: Issue Bill

2.2.7 Use Case: Update Database

Actors: None

Type: Primary and Essential

Description: In this use case, the computer updates the database every time it allots truck or gets any details.

Includes: None

Use-Cases: Allot truck, Dispatch Consignment, Change rate, Order new trucks, Add Employee

2.2.8 Use Case: Dispatch Consignment

Actors: None

Type: Secondary and Non-Essential

Description: In this use case, when the consignment is ready for dispatch and the truck is fully loaded, the computer prints the details and the consignment is dispatched.

Extension Point: When the Consignment is ready for dispatch and the truck is fully loaded.

Includes: Print

Use-Cases: Allot truck

2.2.9 Use Case: Print

Actors: Printer

Type: Secondary

Description: The computer system should print the details of the consignment number, volume, sender's name and address, and the receiver's name and address to be forwarded along with the truck.

Includes: Forward Acknowledgement

Use-Cases: Dispatch Consignment

2.3.0 Use Case: Forward Acknowledgement

Actors: None

Type: Primary and essential

Description: When a truck is available and the required consignment is available for dispatch, the computer system should print the details of the consignment number, volume, sender's name and address, and the receiver's name and address to be forwarded along with the truck.

Includes: None

Use-Cases: Print

2.3.1 Use Case: Enter Password

Actors: Employees and manager

Type: Primary and Essential

Description: In this case, employees and manager enter password. If the password is valid, then only they can log into their account.

Includes: None

Use-Cases: Log In

2.3.2 Use Case: Log Out

Actors: Employees and Manager

Type: Primary and Essential

Description: In this case, Employees or managers log out from their account.

Includes: None

Use-Cases: Log In

2.3.3 Use Case: Query Status of Consignments

Actors: Managers

Type: Primary

Description: Manager queries the status of consignments

Includes: Query Volume of Consignments

Use-Cases: None

2.3.4 Use Case: Query Volume of Consignments

Actors: Managers

Type: Primary

Description: Manager queries the volume of consignments being sent

Includes: Query Revenue

Use-Cases: Query status of consignments

2.3.5 Use Case: Query Revenue

Actors: Manager

Type: Primary

Description: Manager Queries the revenue generated for the consignment

Includes: None

2.3.6 Use Case: Change Rate

Actor: Manager

Type: Primary

Description: Manager can change the rate of sending consignments between any two branches.

Includes: None

2.3.7 Use Case: View Status Of trucks

Actor: Manager

Type: Primary

Description: Manager can view the status of trucks at any time

Includes: None

2.3.8 Use Case: View Idle waiting time

Actor: Manager

Type: Primary

Description: Manager can see the average idle time of the truck in the branch for a given period for future planning.

Includes: None

2.3.9 Use Case: Add Employee

Actor: Manager

Type: Primary

Description: Manager can add employee to the directory

Includes: Update Database

2.4.0: Use Case: Order New Trucks

Actor: Manager

Type: Primary

Description: Manager can order new trucks for better transportation system.

Includes: Update Database

2.4.1: Use Case: View Waiting period of a truck

Actor: Manager

Type: Primary

Description: Manager can view the average waiting period for different consignments. This statistics is important for him since he normally orders new trucks when the average waiting period for consignments becomes high due to non-availability of trucks.

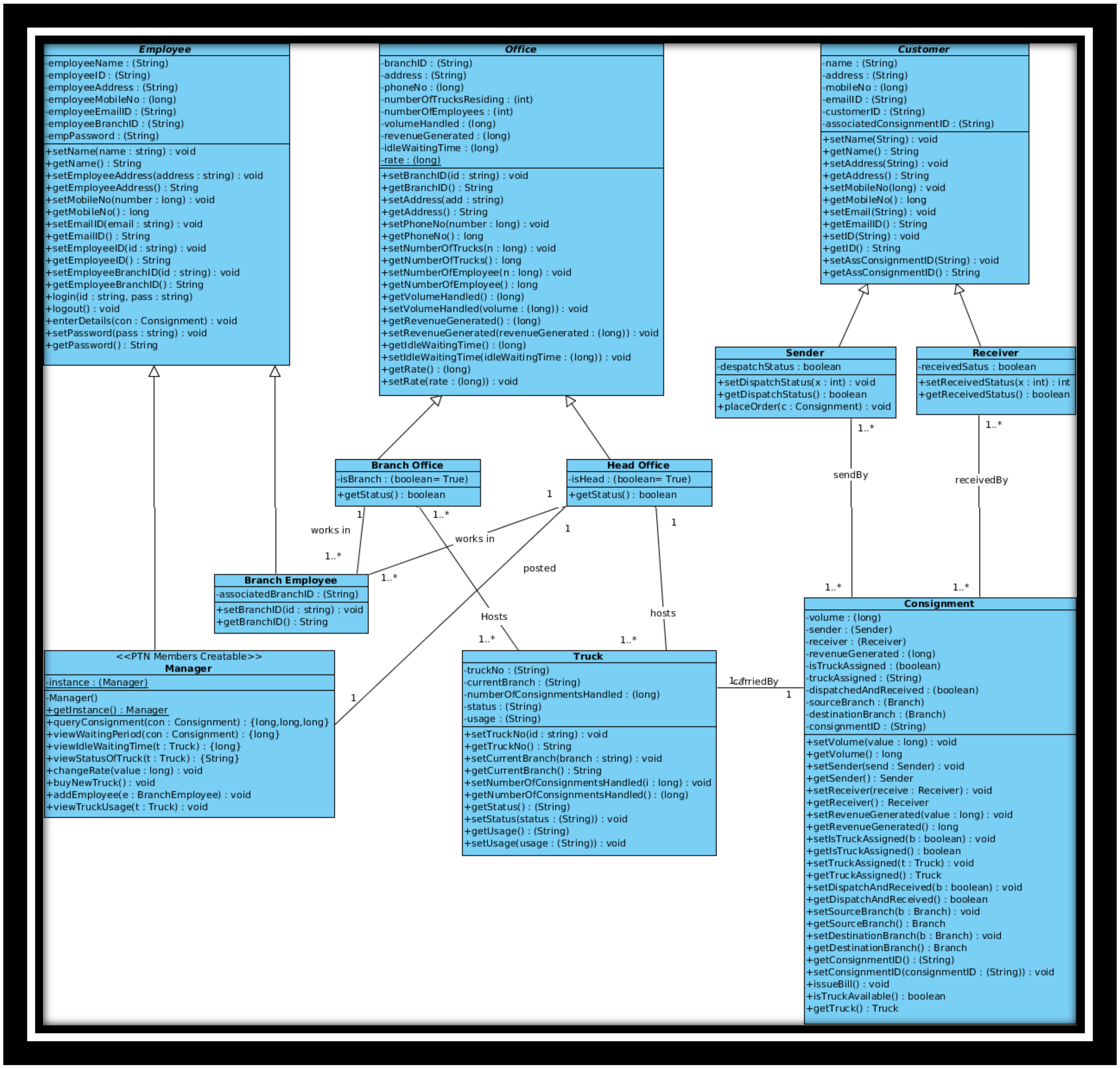
Includes: Update Database

2. Class Diagrams

The purpose of this diagram is to show how objects within the TCCS system will interact with each other in order to achieve the functionality required by the Use Case diagram. Below is a list of what you will see in the diagram itself as well as the class descriptions that follow.

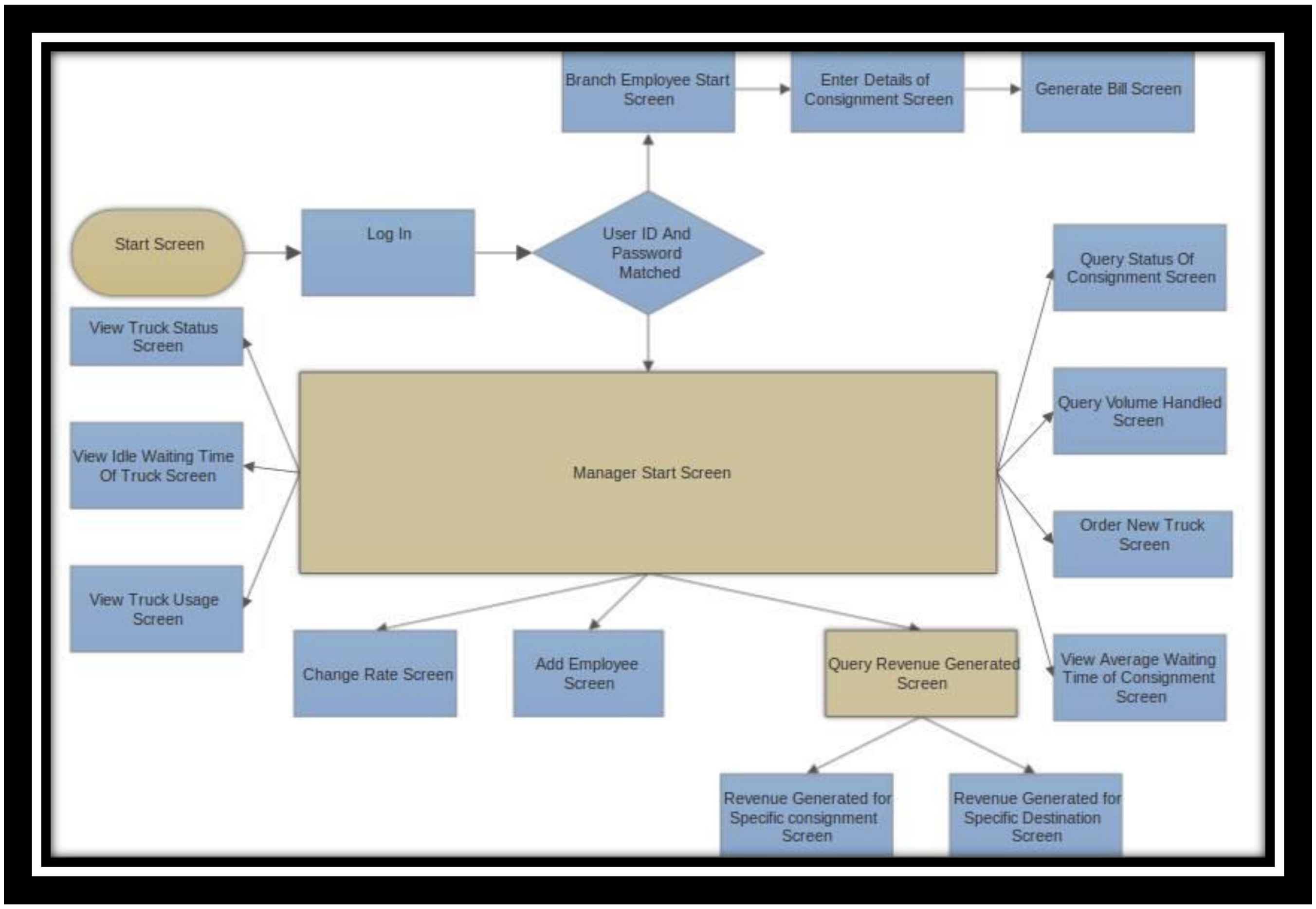
Elements	Description
Classes	Rectangles in the diagram that are split into three parts. The top section is the name of the class, the middle section is the list of variables that are stored in the class and the bottom section is the list of functions in the class. These rectangles represent objects within the system.
Variables	These have a name followed by a semicolon and then a type. The type denotes what kind of data can be stored in the variable.
Functions	These have a name followed by a list of any variable that the function receives in-between the parenthesis "()". After that there is a semicolon and any variables that the function may return, if none it will be void.
Generalization	Shown using a line from one object to the other with an unfilled triangle on one end. The object without the triangle inherits the functionality and variables from the object that has the triangle pointing towards it.
Aggregations	Lines that have an unfilled diamond on one end. This means the object with the diamond contains the object(s) without the diamond. This may have numbers on the ends (multiplicities).
Association	Lines connecting two classes that can have a name beside it, may point in one direction, and may have numbers at the ends (multiplicities). These designate some relationship between the objects. Arrows are simply

	there to assist you in recognizing which direction the name of the association is read.


Employee
-employeeName : (String)
-employeeID : (String)
-employeeAddress : (String)
-employeeMobileNo : (long)
-employeeEmailID : (String)
-employeeBranchID : (String)
-empPassword : (String)
+setName(name : string) : void
+getName() : String
+setEmployeeAddress(address : string) : void
+getEmployeeAddress() : String
+setMobileNo(number : long) : void
+getMobileNo() : long
+setEmailID(email : string) : void
+getEmailID() : String
+setEmployeeID(id : string) : void
+getEmployeeID() : String
+setEmployeeBranchID(id : string) : void
+getEmployeeBranchID() : String
+login(id : string, pass : string)
+logout() : void
+enterDetails(con : Consignment) : void
+setPassword(pass : string) : void
+getPassword() : String
Office
-branchID : (String)
-address : (String)
-phoneNo : (long)
-numberOfTrucksResiding : (int)
-numberOfEmployees : (int)
-volumeHandled : (long)
-revenueGenerated : (long)
-idleWaitingTime : (long)
-rate : (long)
+setBranchID(id : string) : void
+getBranchID() : String
+setAddress(add : string)
+getAddress() : String
+setPhoneNo(number : long) : void
+getPhoneNo() : long
+setNumberOfTrucks(n : long) : void
+getNumberOfTrucks() : long
+setNumberOfEmployee(n : long) : void
+getNumberOfEmployee() : long
+getVolumeHandled() : (long)
+setVolumeHandled(volume : (long)) : void
+getRevenueGenerated() : (long)
+setRevenueGenerated(revenueGenerated : (long)) : void
+getIdleWaitingTime() : (long)
+setIdleWaitingTime(idleWaitingTime : (long)) : void
+getRate() : (long)
+setRate(rate : (long)) : void
Customer
-name : (String)
-address : (String)
-mobileNo : (long)
-emailID : (String)
-customerID : (String)
-associatedConsignmentID : (String)
+setName(String) : void
+getName() : String
+setAddress(String) : void
+getAddress() : String
+setMobileNo(long) : void
+getMobileNo() : long
+setEmail(String) : void
+getEmailID() : String
+setID(String) : void
+getID() : String
+setAssConsignmentID(String) : void
+getAssConsignmentID() : String
Sender
-despatchStatus : boolean
+setDispatchStatus(x : int) : void
+getDispatchStatus() : boolean
+placeOrder(c : Consignment) : void
Receiver
-receivedSatus : boolean
+setReceivedStatus(x : int) : int
+getReceivedStatus() : boolean
Branch Office
-isBranch : (boolean= True)
+getStatus() : boolean
Head Office
-isHead : (boolean= True)
+getStatus() : boolean
Branch Employee
-associatedBranchID : (String)
+setBranchID(id : string) : void
+getBranchID() : String
<<PTN Members Creatable>>
Manager
-instance : (Manager)
-Manager()
+getInstance() : Manager
+queryConsignment(con : Consignment) : {long,long,long}
+viewWaitingPeriod(con : Consignment) : {long}
+viewIdleWaitingTime(t : Truck) : {long}
+viewStatusOfTruck(t : Truck) : {String}
+changeRate(value : long) : void
+buyNewTruck() : void
+addEmployee(e : BranchEmployee) : void
+viewTruckUsage(t : Truck) : void
Truck
-truckNo : (String)
-currentBranch : (String)
-numberOfConsignmentsHandled : (long)
-status : (String)
-usage : (String)
+setTruckNo(id : string) : void
+getTruckNo() : String
+setCurrentBranch(branch : string) : void
+getCurrentBranch() : String
+setNumberOfConsignmentsHandled(i : long) : void
+getNumberOfConsignmentsHandled() : (long)
+getStatus() : (String)
+setStatus(status : (String)) : void
+getUsage() : (String)
+setUsage(usage : (String)) : void
Consignment
-volume : (long)
-sender : (Sender)
-receiver : (Receiver)
-revenueGenerated : (long)
-isTruckAssigned : (boolean)
-truckAssigned : (String)
-dispatchedAndReceived : (boolean)
-sourceBranch : (Branch)
-destinationBranch : (Branch)
-consignmentID : (String)
+setVolume(value : long) : void
+getVolume() : long
+setSender(send : Sender) : void
+getSender() : Sender
+setReceiver(receive : Receiver) : void
+getReceiver() : Receiver
+setRevenueGenerated(value : long) : void
+getRevenueGenerated() : long
+setIsTruckAssigned(b : boolean) : void
+getIsTruckAssigned() : boolean
+setTruckAssigned(t : Truck) : void
+getTruckAssigned() : Truck
+setDispatchAndReceived(b : boolean) : void
+getDispatchAndReceived() : boolean
+setSourceBranch(b : Branch) : void
+getSourceBranch() : Branch
+setDestinationBranch(b : Branch) : void
+getDestinationBranch() : Branch
+getConsignmentID() : (String)
+setConsignmentID(consignmentID : (String)) : void
+issueBill() : void
+isTruckAvailable() : boolean
+getTruck() : Truck
works in
works in
posted
Hosts
hosts
sendBy
receivedBy
carriedBy
1
1..*
1..*
1
1
1
1..*
1..*
1..*
1..*
1..*
1..*
1..*
1..*
1
1
1

3. Interface Diagram

In UML modeling, *interfaces* are model elements that define sets of operations that other model elements, such as classes, or components must implement. An implementing model element realizes an interface by overriding each of the operations that the interface declares. Interfaces support the hiding of information and protect client code by publicly declaring certain behavior or services.

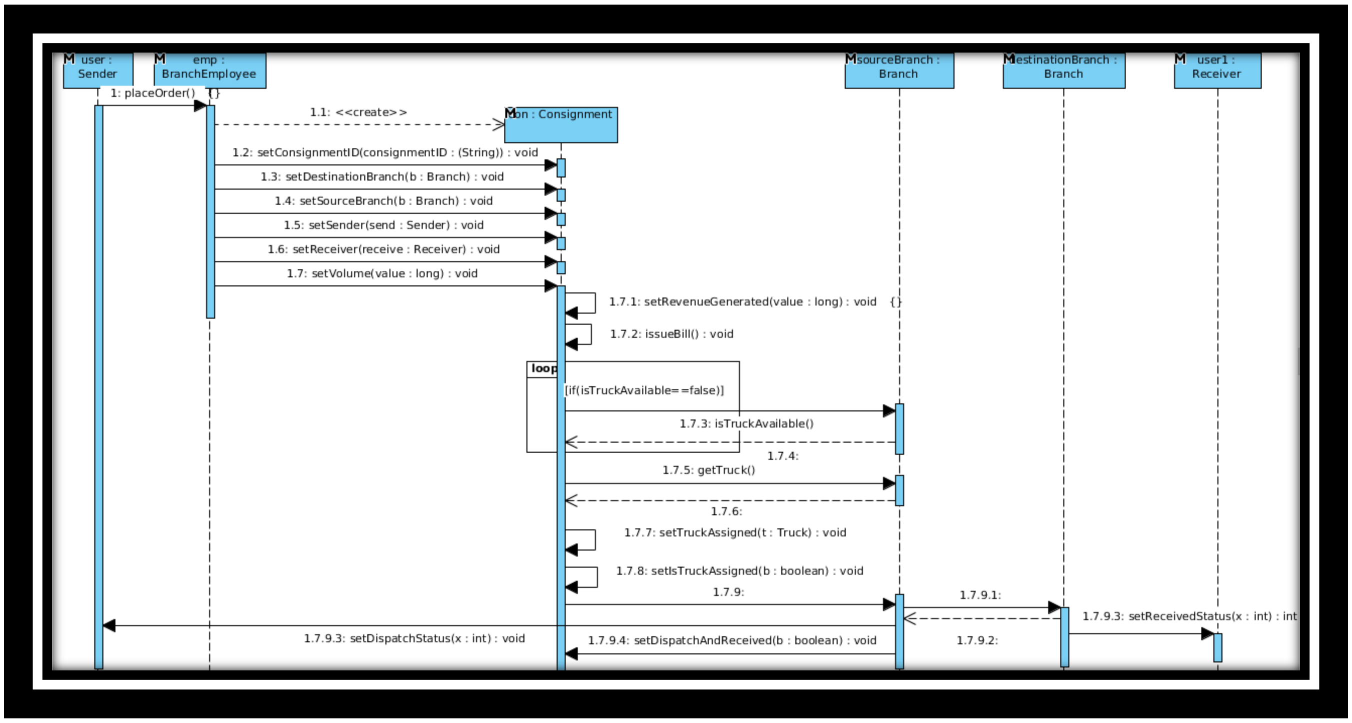

Branch Employee Start Screen
Enter Details of Consignment Screen
Generate Bill Screen
Start Screen
Log In
User ID And Password Matched
Query Status Of Consignment Screen
View Truck Status Screen
Query Volume Handled Screen
View Idle Waiting Time Of Truck Screen
Manager Start Screen
Order New Truck Screen
View Truck Usage Screen
Change Rate Screen
Add Employee Screen
Query Revenue Generated Screen
View Average Waiting Time of Consignment Screen
Revenue Generated for Specific consignment Screen
Revenue Generated for Specific Destination Screen

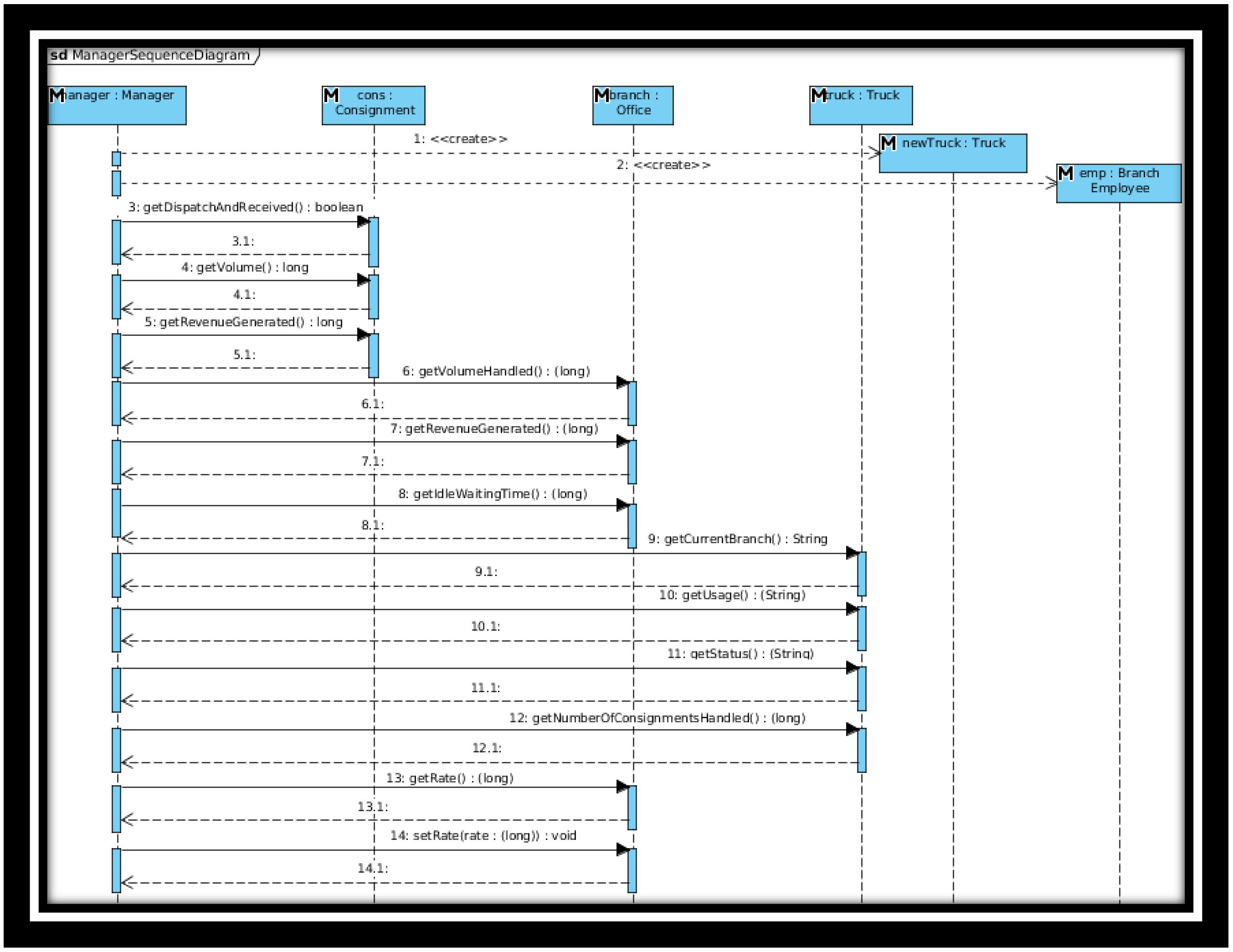
4. Sequence Diagrams

The sequence diagrams use the class diagram and demonstrate specific sequences of actions in the system. The purpose is to ensure that the TCCS system runs in an expected way and that the class structure is sufficient to accomplish the tasks needed. Below is a list of the items that you will see in the diagram and their definitions.

Elements	Description
Axis's	The X-axis identifies movement between objects and the Y-axis identifies time.
Instances	Solid boxes along the top that have dotted lines that stretch vertically below them. These are specific instances of an object. The first part of the title is the name of that specific instance and the object it is an instance of follows it. (Special Note: If there are multiple instances that have the same title then they are actually the same instance and are only there to diagram calls onto themselves.)
Calls	Lines that have filled triangles at the ends. These are transitions from one instance to another and have a label above them that is a function call, a variable being set, or both. It can also have a guard statement that precedes it.

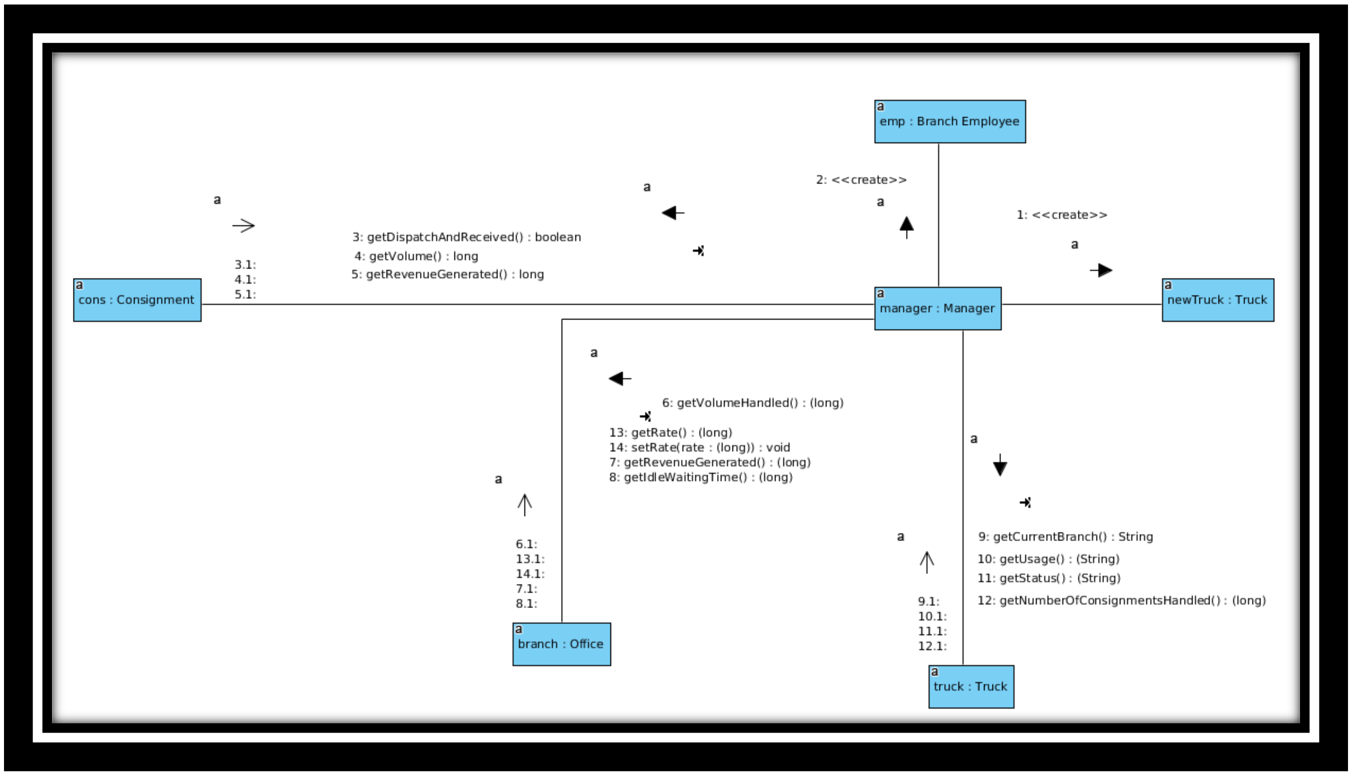
Object Execution Time	This is shown with the solid white boxes that run vertically along the dotted lines. These simply represent the execution time for the objects. (Special Note: the first object has a solid box all the way down this is a special case and should not be there and is due to the application used to create the diagram).

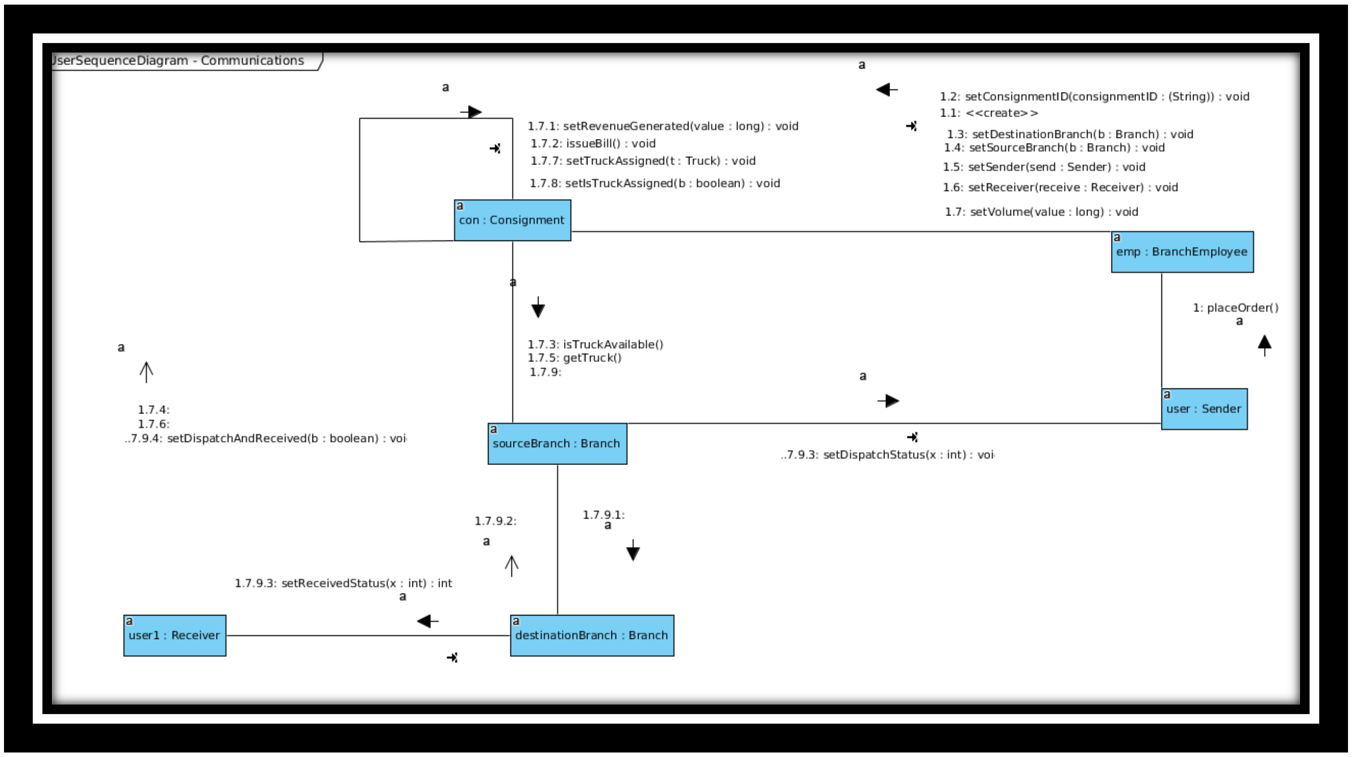

user : Sender
emp : BranchEmployee
sourceBranch : Branch
destinationBranch : Branch
user1 : Receiver
con : Consignment
1: placeOrder()
1.1: <<create>>
1.2: setConsignmentID(consignmentID : (String)) : void
1.3: setDestinationBranch(b : Branch) : void
1.4: setSourceBranch(b : Branch) : void
1.5: setSender(send : Sender) : void
1.6: setReceiver(receive : Receiver) : void
1.7: setVolume(value : long) : void
1.7.1: setRevenueGenerated(value : long) : void
1.7.2: issueBill() : void
loop
[if(isTruckAvailable==false)]
1.7.3: isTruckAvailable()
1.7.4:
1.7.5: getTruck()
1.7.6:
1.7.7: setTruckAssigned(t : Truck) : void
1.7.8: setIsTruckAssigned(b : boolean) : void
1.7.9:
1.7.9.1:
1.7.9.3: setReceivedStatus(x : int) : int
1.7.9.2:
1.7.9.3: setDispatchStatus(x : int) : void
1.7.9.4: setDispatchAndReceived(b : boolean) : void


sd ManagerSequenceDiagram
manager : Manager
cons : Consignment
branch : Office
truck : Truck
newTruck : Truck
emp : Branch Employee
1: <<create>>
2: <<create>>
3: getDispatchAndReceived() : boolean
3.1:
4: getVolume() : long
4.1:
5: getRevenueGenerated() : long
5.1:
6: getVolumeHandled() : (long)
6.1:
7: getRevenueGenerated() : (long)
7.1:
8: getIdleWaitingTime() : (long)
8.1:
9: getCurrentBranch() : String
9.1:
10: getUsage() : (String)
10.1:
11: getStatus() : (String)
11.1:
12: getNumberOfConsignmentsHandled() : (long)
12.1:
13: getRate() : (long)
13.1:
14: setRate(rate : (long)) : void
14.1:

5. Collaboration Diagrams

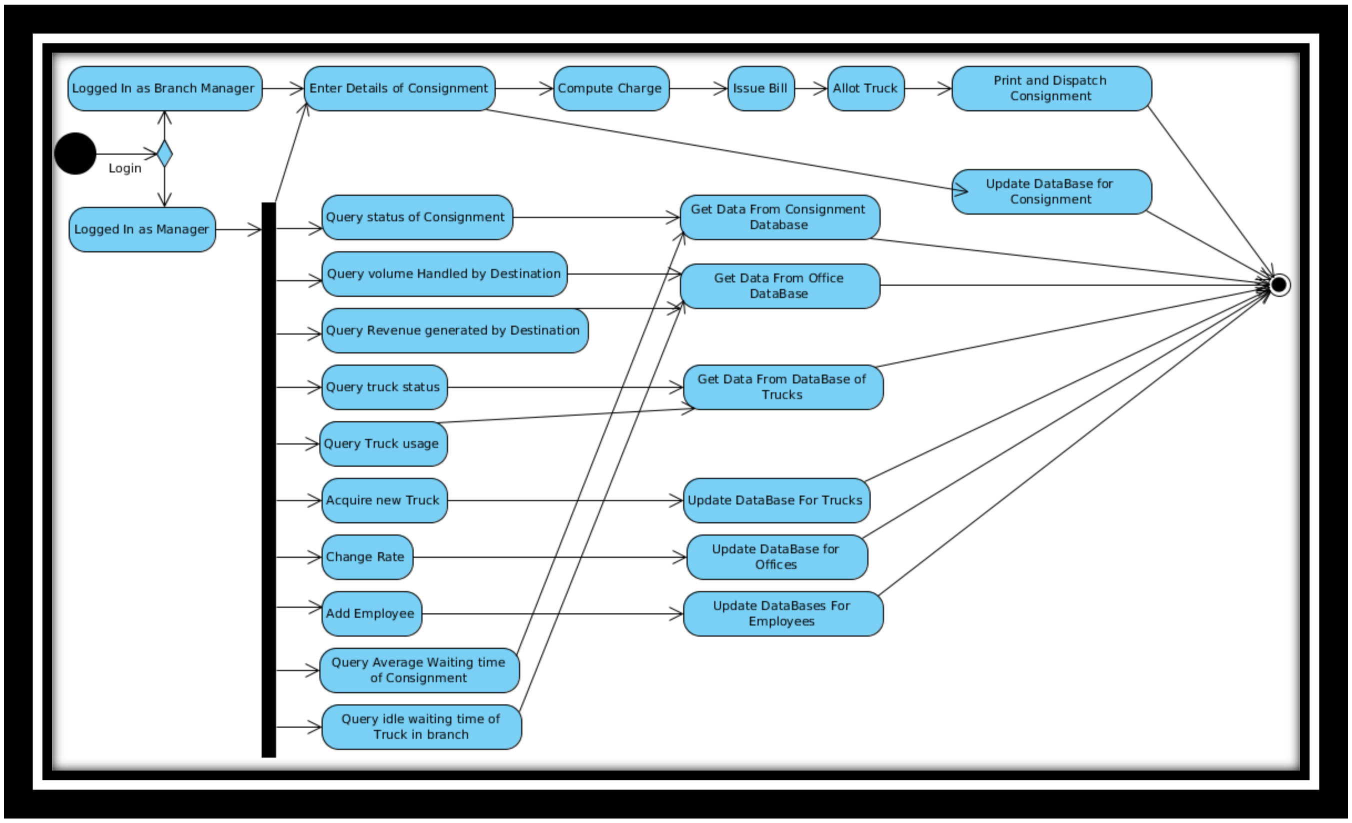
A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.


emp : Branch Employee
2: <<create>>
1: <<create>>
3: getDispatchAndReceived() : boolean
4: getVolume() : long
5: getRevenueGenerated() : long
3.1:
4.1:
5.1:
cons : Consignment
manager : Manager
newTruck : Truck
6: getVolumeHandled() : (long)
13: getRate() : (long)
14: setRate(rate : (long)) : void
7: getRevenueGenerated() : (long)
8: getIdleWaitingTime() : (long)
6.1:
13.1:
14.1:
7.1:
8.1:
branch : Office
9: getCurrentBranch() : String
10: getUsage() : (String)
11: getStatus() : (String)
12: getNumberOfConsignmentsHandled() : (long)
9.1:
10.1:
11.1:
12.1:
truck : Truck


UserSequenceDiagram - Communications
1.7.1: setRevenueGenerated(value : long) : void
1.7.2: issueBill() : void
1.7.7: setTruckAssigned(t : Truck) : void
1.7.8: setIsTruckAssigned(b : boolean) : void
1.2: setConsignmentID(consignmentID : (String)) : void
1.1: <<create>>
1.3: setDestinationBranch(b : Branch) : void
1.4: setSourceBranch(b : Branch) : void
1.5: setSender(send : Sender) : void
1.6: setReceiver(receive : Receiver) : void
1.7: setVolume(value : long) : void
con : Consignment
emp : BranchEmployee
1: placeOrder()
1.7.3: isTruckAvailable()
1.7.5: getTruck()
1.7.9:
1.7.4:
1.7.6:
..7.9.4: setDispatchAndReceived(b : boolean) : voi
user : Sender
sourceBranch : Branch
..7.9.3: setDispatchStatus(x : int) : voi
1.7.9.2:
1.7.9.1:
1.7.9.3: setReceivedStatus(x : int) : int
user1 : Receiver
destinationBranch : Branch

6. Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

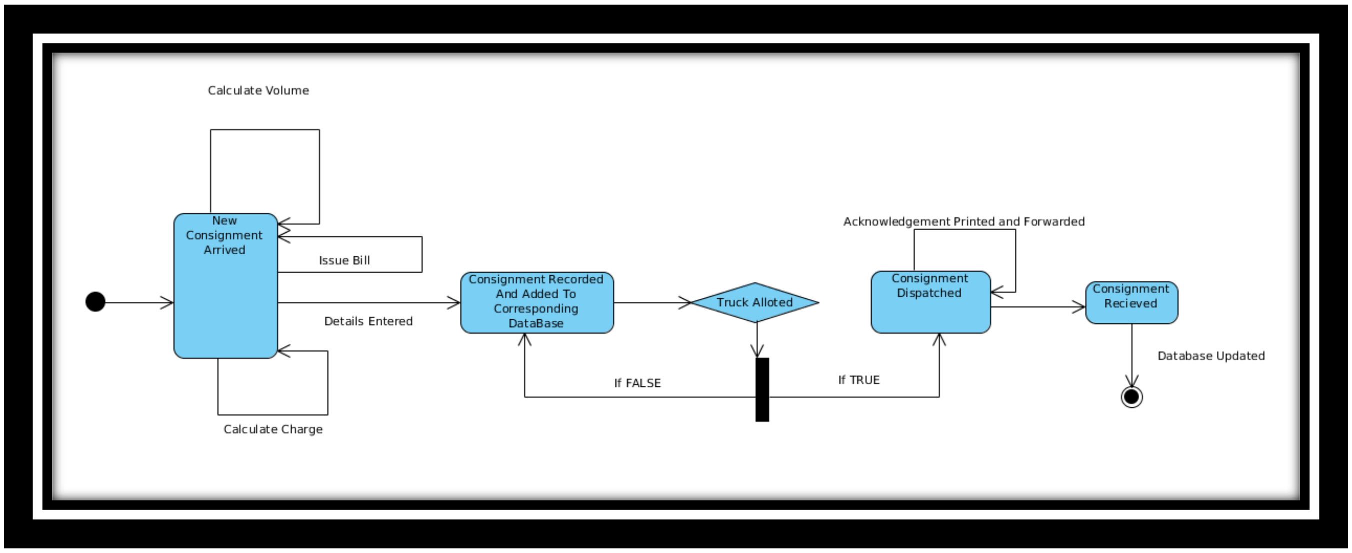

Logged In as Branch Manager
Enter Details of Consignment
Compute Charge
Issue Bill
Allot Truck
Print and Dispatch Consignment
Login
Update DataBase for Consignment
Logged In as Manager
Query status of Consignment
Get Data From Consignment Database
Query volume Handled by Destination
Get Data From Office DataBase
Query Revenue generated by Destination
Query truck status
Get Data From DataBase of Trucks
Query Truck usage
Acquire new Truck
Update DataBase For Trucks
Change Rate
Update DataBase for Offices
Add Employee
Update DataBases For Employees
Query Average Waiting time of Consignment
Query idle waiting time of Truck in branch

7. State Chart Diagrams

The state diagrams take all of the functionality found in the previous diagrams and combines them together to demonstrate the possible changes in the state within TCCS. These state diagrams contain all the possible scenarios shown in the sequence diagrams.

Elements	Description
Starting point	This is where the system starts at and it is represented with a filled circle that has an arrow that points towards the start state.

States	These are represented by a box with round edges. Inside these boxes there can be states encapsulated, but these state diagrams do not contain any encapsulated state diagrams within.
Transitions	The arrows that connect a state to another state and have a label of that occurs which triggers the transition. These triggers are normally functions but they can also be a new call that is the event of creating a new object;
Objects	The larger boxes that can contain several states and transitions. They are classes within the state diagram. The text at the top of the class is the name of the object.



Thank You...