TEST SUITE

for

Egret

A Transport Company Computerization Software

Version 1.0 approved

Prepared by

Parth Jindal
Mayank Kumar
Shristi Singh

Department of Computer Science and Engineering,
IIT Kharagpur

March 27, 2021

1. Test Cases for Address Class

-------------------------- Constructor Testing ---

GIVEN

city = “Delhi”
addrLine = “C-28,Model Town-3”
zipCode = “110009”

THEN

assert if all values are correctly assigned or not.
This can be done by checking each attribute of Address class

GIVEN

city = “Gorakhpur”
addrLine = “A-76, Buddha Vihar”

THEN

assert if all values are correctly assigned to each attribute of Address class.

GIVEN

city = “Kharagpur”

THEN

assert if all values are correctly assigned to each attribute of Address class.

---------------------------------Method Testing ---

1. **getID()**

GIVEN

Three objects of Address class A1, A2, A3

THEN

assert that the IDs of all the objects are unique and the ID of the first constructed object is 1.

2. **getCity()**

GIVEN

Two objects of Address class A1, A2

THEN

assert that the string returned is the same as the city name given at the time of constructing each object .

3. **getAddressLine()**

GIVEN

Three objects of Address class A1, A2, A3

THEN

assert that the string returned is the same as the address given at the time of constructing each object .

4. **getZIP()**

GIVEN

Three objects of Address class A1, A2, A3

THEN

assert that the string returned is the same as the zip given at the time of constructing each object .

2. Test Cases for Bill Class

-------------------------- Constructor Testing ---

GIVEN

date = datetime.date(2021,03,27)
amount = 2500
paymentID = “573708TS34”

THEN

assert if all values are correctly assigned to each attribute of Bill class.

----------------------------------Method Testing --

1. **getDate()**

GIVEN

An object of the Bill class

THEN

assert that the date returned is the same as the date given at the time of constructing the object .
assert that the date is not a future date.

2. **getAmount()**

GIVEN
An object of the Bill class

THEN
assert that the integer returned is the same as the amount given at the time of constructing the object .

3. **getPaymentID()**

GIVEN
Two objects of the Bill class

THEN
assert that the string returned is the same as the paymentID given at the time of constructing each object.
assert that the paymentID of both the objects are unique.

3. Test cases for Consignment Class

--------------------------- Constructor Testing --

GIVEN
addr1 = Address(city = “Delhi”)
addr2 = Address(city = “Kolkata”)
volume = 400
destinationID = 1
consign = Consignment(volume, addr1, addr2, destinationID)

THEN
assert if all values are correctly assigned or not.
This can be done by checking each attribute of Address class (white-box-testing)

-----------------------------------Method Testing ---

1. **getID()**

GIVEN

Two Consignment class objects C1,C2

WHEN

Both objects are committed to database Db

THEN

assert that C1.getID() should not be equal to C2.getID()
assert that C1.getID() should be equal to 1.

2. **getCharge()**

GIVEN

charge = 1 rs per kilometer,volume
consignment from SourceBranch S to Destination Branch D with distance 300 km,
and volume 300

WHEN

System sets the charge of the consignment using setCharge(value)

THEN

*assert if the consignment.getCharge() is equal to 300*300*1 Rs.*

3. **viewAssignedTrucks()**

GIVEN

Consignment C(srcBranch = 1,dstBranch = 2)
Trucks(T[1.*]) at Branch with non-specific volume

WHEN

Consignment is allotted to Trucks

THEN

Trucks should be allotted in a manner such that trucks only take upto 500 units of volume .Thus Trucks can have multiple consignments and each consignment can be in multiple trucks.

Case 1:

Consignment can fit into one truck and truck has not been allotted yet.Truck T out of all trucks is the best candidate if it has not been allotted to any destination yet.Its volume will also be zero.

C has volume = 400
T = Truck(currentBranch = 1,volume = 0)
Since T.volume + 400 < 500
assert that C.viewAssignedTrucks contains T

Case2:

Consignment cannot fit into one truck and truck has not been allotted yet to any Branch. The consignment then must be divided into multiple trucks with each Truck being

C has volume = 600
T1 = Truck(currentBranch = 1,volume = 0),T2 = Truck(currentBranch = 1,volume = 0)
assert that C.viewAssignedTrucks() contains T1, T2

Case3:

Consignment fits into a truck already containing some volume and having same destinationBranchID as before

C has volume = 200
T1 = Truck(dstBranchId = 2,currentVolume = 250)

assert that **C.viewAssignedTrucks()** contains only T1

Case 4:

Consignment cannot fit into one truck and only one truck has space left that goes to branch
C has volume = 400
T = Truck(dstBranch = 1,volume = 300)

In such a case some part of Consignment is left at branch and other goes with T

C. viewAssignedTrucks() must contain T then and another truck that comes in future

Case 5: Consignment has no matching fit for any truck due to size or destination

C has Volume = 600
T1 = Truck(dst = 2,volume = 500) It is full
T2 = Truck(dst = 3,volume = 200) destination is different

Assert **C.viewAssignedTrucks()** is empty list

4. **getStatus()**

GIVEN

A consignment C(destinationBranchID = 2) of SourceBranchID = 1

WHEN

consignment is allotted to trucks or attempt is made to allot
consignment is received by the branch

Case 1:

Consignments status will change from PENDING TO ASSIGNED when all of its volume has been allotted to trucks and

```
C{ volume = 200}
Truck T1{
    volume = 200
}
T1 is allotted to C
asserted C.getStatus() == ASSIGNED
```

Case 2:

Consignment status will change from ASSIGNED TO ENROUTE when trucks carrying it are finally dispatched

```
C{ volume = 200}
Truck T1{
    volume = 300
}
T1 is allotted to C
T1.getStatus() == ENROUTE
assert C.getStatus() == ENROUTE
```

Case 3: Consignment status will change from ENROUTE TO RECEIVED when trucks carrying it arrive at destination branch

```
C{volume = 200}

Truck trucks[1.*]  = C.viewAssignedTrucks()
for t in trucks:
    if t.getStatus() != RECEIVED:
        return
assert C.getStatus() == RECEIVED
```

4. Test Cases for Truck Class

-------------------------- Constructor Testing ---

GIVEN

plateNo = ”AB12CD1314”
branchID = 1
t1 = Truck(plateNo, branchID)

THEN

assert if all values are correctly assigned or not.
This can be done by checking each attribute of Address class (white-box-testing)

----------------------------------Method Testing --

1. **getID()**

GIVEN

Two Truck class objects T1, T2

WHEN

Both objects are committed to database Db

THEN

assert that T1.getID() should not be equal to T2.getID()

assert that T1.getID() should be equal to 1.

2. **getStatus()**

GIVEN

A Truck T with T.consignments = None

Case 1: When T is added to branch B, it's status is AVAILABLE

Case 2: When T is assigned any consignment C, it's status changes to ASSIGNED

Case 3: When T.isFull() is True, the truck is dispatched and it's status changes to ENROUTE

3. **addConsignment()**

GIVEN

T = Truck(currentBranch = 1, volume = 0)
volume = 200
addr1 = Address(city = "Delhi")
addr2 = Address(city = "Kolkata")
C = Consignment(volume, addr1, addr2, destinationID)

WHEN

Truck T is assigned to a Consignment C

THEN

Truck is assigned such that it cannot take more than 500 units volume. More than one Truck will be assigned if its available volume is less than volume of a consignment. The consignments allotted to one truck will have the same destination.

4. **viewConsignments()**

GIVEN

T = Truck(currentBranch = 1, volume = 0)
volume = 200
addr1 = Address(city = "Delhi")
addr2 = Address(city = "Kolkata")
C1 = Consignment(volume, addr1, addr2, destinationID)
C2 = Consignment(volume, addr1, addr2, destinationID)

WHEN

Truck T is assigned to a Consignment C

THEN

assert that the list T.consignments displays both consignments C1 and C2

5. **isFull()**

GIVEN

Truck object T

THEN

assert that T.isFull() is True if its volumeConsumed is 500 units, and False if its volumeConsumed is less than 500 units.

6. **emptyTruck()**

GIVEN

clist = T.emptyTruck()

WHEN

Truck object is emptied and returns a list of consignments

THEN

assert that the list clist has all consignments assigned to the truck T
assert that volumeConsumed is 0, isFull() is False and status is AVAILABLE

5. Test Cases for Employee Class

-------------------------- Constructor Testing ---

GIVEN

name = "Mayank Kumar"
email = "mayankkumar1205@gmail.com"
branchID = 2

THEN

assert if all values are correctly assigned to each attribute of Employee class.

-----------------------------------Method Testing ---

1. **getName()**

GIVEN

An object of the Employee class E1

THEN

assert that the string returned is the same as the name given at the time of constructing the object .

2. **getEmail()**

GIVEN

An object of the Employee class E1

THEN

assert that the string returned is the same as the email given at the time of constructing the object .

3. **getBranchID()**

GIVEN

An object of the Employee class E1

THEN

assert that the integer returned is the same as the branchID given at the time of constructing the object .

4. **set_password()**

GIVEN

An object of the Employee class E1
A string which is to be set as password

THEN

assert that the password_hash is the same as the calculated hash value of the password string.

5. **check_password()**

GIVEN

An object of the Employee class E1
A string which is to be set as password

THEN

assert that the function returns true if the hash value of the password string is equal to password_hash.

6. **RequestForTruck()**

GIVEN

An object of the Employee class E1

THEN

assert that a mail has been sent to the Manager requesting a truck for its branch after analyzing the waiting period and current number of trucks present in the branch.

7. **DispatchTruck()**

GIVEN:

An object of the Employee class E1
An integer variable storing the id of the Truck to be dispatched

THEN:

assert that the truck requested for dispatch is full.

6. Test Cases for Manager Class

-------------------------- Constructor Testing ---

GIVEN

name = “Parth Jindal”
email = “ pmjindal@gmail.com ”
branchID = 3

THEN

assert if all values are correctly assigned to each attribute of Manager class.

-----------------------------------Method Testing ---

Since Manager class is derived from Employee class, all the method tests in Employee class will comply with the Manager class.In addition to that, the following methods are tested

1. **viewWaitingPeriod()**

GIVEN

An object of the Manager class M

THEN

assert that the average waiting period returned for a consignment is the same as the golden output.

2. **viewWaitingTime()**

GIVEN

An object of the Manager class M

THEN

assert that the waiting time of a truck as returned by the function is correct.

3. **viewIdleTime()**

GIVEN

An object of the Manager class M

THEN

assert that the idle time of a truck as returned by the function is correct.

4. **changeRate()**

GIVEN

An object of the Manager class M and an integer variable storing the new rate

THEN

assert that the rate is changed to the given value .

5. **buyNewTruck()**

GIVEN

An object of the Manager class M
B1 = BranchOffice(addr1, phone)

THEN

assert that the truck is added to the given branch.

6. **viewTruckStatus()**

Case1:

GIVEN

An object of the Manager class M
T1 =Truck(currentBranch=1, volume=150)

THEN

assert that the status of the Truck is ASSIGNED.

Case2:

GIVEN

An object of the Manager class M
T2 =Truck(currentBranch=2, volume=500)

THEN

assert that the status of the Truck is ENROUTE.

Case3:

GIVEN

An object of the Manager class M
T3 =Truck(currentBranch=1, volume=0)

THEN

assert that the status of the Truck is AVAILABLE.

7. **viewTruckUsage()**

GIVEN

An object of the Manager class M
An object of the Truck class T

THEN

assert that the usage time returned for the truck is correct.

7. Test Cases for BranchOffice Class

------------------------- Constructor Testing ---

GIVEN

addr = Address(addressLine=”XYZ Road”, city=”ABC”, zipCode=”124578”)
phone = “9876543210”
b1 = Branch(addr, phone)

THEN

assert if all values are correctly assigned to each attribute of BranchOffice class.

----------------------------------Method Testing --

1. **getID()**

GIVEN

Two BranchOffice class objects B1, B2

WHEN

Both objects are committed to database Db

THEN

assert that B1.getID() should not be equal to B2.getID()
assert that B1.getID() should be equal to 1.

2. **addEmployee()**

GIVEN

An Employee object E

WHEN

The employee is added to branch B1

THEN

assert that the employee is added with his branchID 1 (branchID of B1)

3. **addTruck()**

GIVEN

A Truck object T

WHEN

The truck is added to branch B1

THEN

assert that the truck is added with his branchID 1 (branchID of B1)

4. **addTransaction()**

GIVEN

A Bill object bl1

WHEN

The bill is added to branch B1

THEN

assert that the Bill object bl1 is valid

5. **viewTransactions()**

GIVEN

A Branch object B1

THEN

assert that B1.viewTransactions() shows all the transactions of branch B1 only

6. **receiveTruck()**

GIVEN

A Truck object T

WHEN

The truck is received at branch B1

THEN

assert that the Truck object T1 is properly added to list of other trucks in B1, the truck is empty and it's status is AVAILABLE. Also, all the consignments allotted to this truck should have its status DELIVERED.

7. **removeTruck()**

GIVEN

A Truck object T

WHEN

The truck is removed from branch B1

THEN

assert that the Truck object T1 initially had its currentBranchID as 1 (same as branchID of B1), it had volumeConsumed 500 units and it's status was ENROUTE.

8. Test Cases for HeadOffice Class

------------------------- Constructor Testing --

GIVEN

addr = Address(addressLine="XYZ Road", city="ABC", zipCode="124578")
phone = "9876543210"
H = HeadOffice(addr, phone)

THEN

assert if all values are correctly assigned to each attribute of HeadOffice class.

----------------------------------Method Testing ---

All the tests for the BranchOffice are to be complied for HeadOffice

1. **setRate()**

GIVEN

An object of the HeadOfficeClass and an integer variable rate

THEN

assert that the static constant of rate is correctly changed to the given value.

9. Test Cases for Authorization Blueprint

--Login ---

GIVEN:

login credentials
URL of login

WHEN:

Employee/Manager tries to login

THEN:

Response should be appropriate with correct status code and RESPONSE HTML

Case 0:

Email doesn't pass email validation test
email: pmjindal.com
password: aaaaaa

Input:
POST METHOD
Response
Assert response.status == 401
Assert "invalid email" in response.data

Case 1:

Login credentials are correct
email:- pmjindal@gmail.com
password:-aaaaaa

Input:
POST METHOD

On submitting

Response:
Assert response.status == 200
Assert "Successfully logged in" in response.HTML

Case 2:

Login credentials are wrong
email:- pmjindal@gmail.com
password:- aaaaab
Input:
POST METHOD with credentials
Response:
Assert response.status == 401
Assert "Incorrect password" in response.HTML

Case 3:

email doesn't exist
email:- pmjindsl@gmail.com
password: aaaaaa

Input:
POST METHOD

Response:
Assert response.status == 404
Assert "email not registered" in response.HTML

--- Register --

GIVEN:

Manager
New Employee details,

WHEN:

Manager tries to create a new account for the employee

THEN:

Response should have its status code 200 if user is created
Response should have its status code as 403 if user is already created
Response should have its status code as 404 if email validation fails or any other form validation fails

----

Test Cases for User Blueprint

Given:
Employee
When:
Employee tries to create consignment/view consignment /truck
Then:
Case 1:
Employee is not logged in
Response:
Employee must be redirected to login page
Assert "next" in response.args

Database Testing

Database testing will be done by creating objects of all model classes and committing them to the database using the ORM interface.This is followed by querying from the database back to check the validity of the commits.

1. **Creating Consignment**
   db: Database ORM object (where everything will be committed)

GIVEN:
a = Address(city = “Delhi”)
c = Consignment(volume = 100, senderAddress = a, receiverAddress = Address(city = “Kharagpur”)

WHEN:
 c is committed to Database db
db.session.add(c)
db.session.commit(c)

THEN:
Database when queried should return the object with the same fields. In Addition since Consignment **has a** sender Address, receiverAddress , a should be automatically committed to the database
c_ = Consignment.query.filter_by(id = 1)

Assert all fields of c match to c_.

a_ = c_.getSenderAddress()

Assert all fields of a match to a_

2.Assigning Trucks to consignment
GIVEN:
Truck T1(srcBranch = 1,dstBranch = 2,plateNo = “ABCD1103)
Consignment C(dstBranch = 2)
WHEN:
T1.addConsignment(C)
db.session.add(T1)
db.session.commit(T1)

THEN:
Querying:
T1_ = T1.query.filter_by(plateNo = “ABCD1103”).first()
Assert T1 has all attributes same as in T1_
C_ = T1_.getConsigments()[0]
Assert C_ has all attributes same as C

Test Cases for Application

1. curr = datetime.now()

2. Distance between branches (constant)
   B1 to B2: b12 = 100 km
   B1 to B3: b13 = 150 km
   B2 to B3: b23 = 200 km

3. Average speed of trucks (constant)
   speed = 40 km/hr

4. Rate per km in Rupees (constant)
   rate1 = 0.25

5. Rate per unit volume (constant)
   rate2 = 0.15

6. Rate of a consignment
   rate = int (rate1*distance + rate2*volume)

7. When the Manager registers, it is checked that the email entered is valid and is not already present in the database.
   Manager M1:
   Name: M1
   Email: tccs_manager_m1@gmail.com
   Password: 12345678

8. The Manager logs in.
   Email: tccs_manager_m1@gmail.com
   Password: 12345678

9. The Manager should provide unique email to all employees while adding them.
   Employee E1:
   Name: E1
   Email: tccs_employee_e1@gmail.com
   Password: abcdefgh
   Branch ID: 1

   Employee E2:

Name: E2
Email: tccs_employee_e2@gmail.com
Password: ijklmnop
Branch ID: 2

Employee E3:
Name: E3
Email: tccs_employee_e3@gmail.com
Password: qrstuvwx
Branch ID: 3

10. The employees logs in.
    Login for Employee E1:
    Email: tccs_employee_e1@gmail.com
    Password: abcdefgh
    Login Successful

    Login for Employee E2:
    Email: tccs_employee_e@gmail.com
    Password: ijklmnop
    Incorrect email, login Failed

    Login for Employee E2:
    Email: tccs_employee_e2@gmail.com
    Password: ijklmnoq
    Incorrect password, login Failed

    Login for Employee E2:
    Email: tccs_employee_e2@gmail.com
    Password: ijklmnop
    Login Successful

    Employee E3 Forgot Password:
    Email: tccs_employee_e@gmail.com
    Incorrect Email, Password Reset Failed

    Employee E3 Forgot Password:
    Email: tccs_employee_e3@gmail.com
    E3 used link sent through email to reset password:
    New Password: qrstuvwxyz
    Password Reset Successful

    Login for Employee E3
    Email: tccs_employee_e3@gmail.com

Password: qrstuvwxyz
Login Successful

Employee E1 logs out
Log Out Successful

Login for Employee E1:
Email: tccs_employee_e1@gmail.com
Password: abcdefgh
Login Successful

11. The Manager adds trucks.
    Truck T1:
    Plate No.: AB12CD3456
    Truck ID: 1
    Current Branch ID: 1
    Volume Consumed: 0
    Status: AVAILABLE

    Truck T2:
    Plate No.: AB13CE3457
    Truck ID: 2
    Current Branch ID: 2
    Volume Consumed: 0
    Status: AVAILABLE

    Truck T3:
    Plate No.: AB14CF3458
    Truck ID: 3
    Current Branch ID: 3
    Volume Consumed: 0
    Status: AVAILABLE

12. Employee E1 places consignments
    Consignment C1:
    Consignment ID: 1
    C1.status = PENDING
    Volume: 200 units
    Sender Address: Address(addressLine=”ABC Palace”, city=”DEF”, zipCode=”123456”)
    Receiver Address: Address(addressLine=”XYZ Palace”, city=”GHI”, zipCode=”654321”)
    Source Branch: B1
    Destination Branch: B2
    C1.charge = rate1*b12 + rate2*volume = 55
    pid = XYZ1234

Bill bill_C1 = Bill(date=date.today(), amount=C1.charge, paymentID=pid)

Trucks: [T1]
C1.status = ALLOTTED

Truck T1 updated as:
Volume Consumed: 200 units
Consignments: [C1]
Status: ASSIGNED
Idle Time: datetime.now() - curr
curr_t1 = datetime.now()
Usage Time: 0

Consignment C2:
Consignment ID: 2
C2.status = PENDING
Volume: 300 units
Sender Address: Address(addressLine=”ABCD Palace”, city=”DEF”, zipCode=”123456”)
Receiver Address: Address(addressLine=”WYZ Palace”, city=”GHI”, zipCode=”654321”)
Source Branch: B1
Destination Branch: B2
C2.charge = rate1*b12 + rate2*volume = 70
pid = XYZ1235
Bill bill_C2 = Bill(date=date.today(), amount=C2.charge, paymentID=pid)

Trucks: [T1]
C2.status = ALLOTTED

Truck T1 updated as:
Volume Consumed: 500 units
Consignments: [C1]
Status: ENROUTE
B1.trucks = []
Idle Time: Idle Time + datetime.now() - curr_t1
curr_t1 = datetime.now()
Usage Time: 0
B1.transactions = [bill_C1, bill_C2]
B1.revenue = B1.revenue + C1.charge + C2.charge = 125

13. Employee E2 receives Truck T1
    B2.trucks = [T2, T1]
    C1.status = DELIVERED
    C2.status = DELIVERED

Truck T1 is updated as:
Volume Consumed: 0 units
Status: AVAILABLE
Idle Time: unchanged
curr_t1 = datetime.now()
Usage Time: b12 / speed
Consignments = []

14. Employee E2 places consignments
Consignment C3:
Consignment ID: 3
C3.status = PENDING
Volume: 600 units
Sender Address: Address(addressLine=”XYZ Palace”, city=”GHI”, zipCode=”654321”)
Receiver Address: Address(addressLine=”PQR Palace”, city=”JKL”, zipCode=”134679”)
Source Branch: B2
Destination Branch: B3
C3.charge = rate1*b23 + rate2*volume = 140
pid = ABC1236
Bill bill_C3 = Bill(date=date.today(), amount=C3.charge, paymentID=pid)

Trucks: [T2, T1]
C3.status = ALLOTTED

Truck T1 updated as:
Volume Consumed: 100 units
Consignments: [C3]
Status: ASSIGNED
Idle Time: Idle Time + datetime.now() - curr_t1
curr_t1 = datetime.now()
Usage Time: unchanged

Truck T2 updated as:
Volume Consumed: 500 units
Consignments: [C3]
Status: ENROUTE
B2.trucks = [T1]
Idle Time: datetime.now() - curr
curr_t2 = datetime.now()
Usage Time: unchanged
B1.transactions = []

Consignment C4:
Consignment ID: 4

C4.status = PENDING
Volume: 400 units
Sender Address: Address(addressLine=”XYR Palace”, city=”GHI”, zipCode=”654321”)
Receiver Address: Address(addressLine=”PQS Palace”, city=”JKL”, zipCode=”134679”)
Source Branch: B2
Destination Branch: B3
C4.charge = rate1*b23 + rate2*volume = 110
pid = ABC1237
Bill bill_C4 = Bill(date=date.today(), amount=C4.charge, paymentID=pid)

Trucks: [T1]
C4.status = ALLOTTED

Truck T1 updated as:
Volume Consumed: 500 units
Consignments: [C3, C4]
Status: ENROUTE
B2.trucks = []
Idle Time: Idle Time + datetime.now() - curr_t2
curr_t1 = datetime.now()
Usage Time: unchanged
B2.transactions = [bill_C3, bill_C4]
B2.revenue = B2.revenue + C3.charge + C4.charge = 250

15. Employee E3 receives trucks T1 and T2
B3.trucks = [T3, T1, T2]
C3.status = DELIVERED
C4.status = DELIVERED

Truck T1 is updated as:
Volume Consumed: 0 units
Status: AVAILABLE
Idle Time: unchanged
curr_t1 = datetime.now()
Usage Time: Usage Time + b23 / speed
Consignments = []

Truck T2 is updated as:
Volume Consumed: 0 units
Status: AVAILABLE
Idle Time: unchanged
curr_t1 = datetime.now()
Usage Time: b23 / speed
Consignments = []

16. Employee E1 places consignments
    Consignment C5:
    Consignment ID: 5
    C5.status = PENDING
    Volume: 500 units
    Sender Address: Address(addressLine="ABC Palace", city="DEF", zipCode="123456")
    Receiver Address: Address(addressLine="XYZ Palace", city="GHI", zipCode="654321")
    Source Branch: B1
    Destination Branch: B2
    C5.charge = rate1*b12 + rate2*volume = 100
    pid = XYZ1236
    Bill bill_C5 = Bill(date=date.today(), amount=C5.charge, paymentID=pid)

    Employee E1 requests Manager to send a truck.

17. Manager checks E1's request.
    Manager checks branch B1's revenue
    B1.revenue = 125
    Manager decides to buy truck T4 for branch B1
    Truck T4:
    Plate No.: AB12CZ3465
    Truck ID: 4
    Current Branch ID: 1
    Volume Consumed: 0
    Status: AVAILABLE
    B1.trucks = [T5]
    curr_t4 = datetime.now()

18. Consignment C5 is dispatched

    Trucks: [T4]
    C5.status = ALLOTTED

    Truck T4 updated as:
    Volume Consumed: 500 units
    Consignments: [C5]
    Status: ENROUTE
    B1.trucks = []
    Idle Time: Idle Time + datetime.now() - curr_t4
    curr_t4 = datetime.now()
    Usage Time: unchanged
    B1.transactions = [bill_C1, bill_C2, bill_C5]
    B1.revenue = B1.revenue + C5.charge = 225

19. Employee E2 receives truck T4
    B2.trucks = [T2, T1]
    C5.status = DELIVERED

    Truck T4 is updated as:
    Volume Consumed: 0 units
    Status: AVAILABLE
    Idle Time: unchanged
    curr_t4 = datetime.now()
    Usage Time: b12 / speed
    Consignments = []

20. Employee E1 logs out
    Employee E2 logs out
    Employee E3 logs out
    Manager logs out