**Constraints:**

1)**Class person**

* Constraint1: person “age” must be a non-negative number
* Constraint 2 : person objects must have a non-empty **name**.
* Constraint 3 : person **id** must be unique across all **Person** objects.
* Constraint 4: password must be non-empty and secure.

2)**Class Manager**

* Constraint 5 : manager must have a salary
* Constraint 6 : each manager manages at least one staff member
* Constraint 7: staff salary must not exceed manager salary

3)**Class Staff**

* Constraint 8 : staff must have a salary
* Constraint 9: staff manages at least one order
* Constraint 10 : staff must access all the reservations

4)**Class user**

* Constraint 1 1: user only access product that are available
* Constraint 12 : feedback added by user must not exceed 1000 characters
* Constraint 13 : when a user adds an order, it must include the user's ID

5)**Class order**

* Constraint14 :each order’s cost is not null
* Constraint 15: each order’s id must be unique
* Constraint 16 : each order includes at least one product

6**)Class Products**

* Constraint 17: Products must have a unique id
* Constraint 18: Products price must be not null
* Constraint 19: Products name not empty

7)Class reservation

* Constraint 20 : reservations must have a unique id .
* Constraint 21 : each reservation associated with a user .
* Constraint 22 : reservation day should be within a valid range [1-31].

8) Class feedback

* Constraint 23 : feedback must include a user id.

1. **person “age” must be a non-negative number**

Context : Person

Invariant : Self.age >=0

**2)Person object must be a non empty name**

Context : person

Invariant : not self.name.isEmpty()

**3)person id must be unique across all Person objects.**

Context : Person

Invariant : Person.allInstances()->isUnique(id)

**4) Password must be non-empty and secure**

Context : Person

Invariant : not self.password.isEmpty() and self.password.matches('[a-zA-Z0-9]{8,}')

**5)Manager must have a salary**

Context : Manager

Invariant :self.salary <> null

**6) each manager manages at least one staff member**

Context : Manager

Invariant : self.staff->notEmpty()

**7) staff salary must not exceed manager salary**

Context :Manager

Invariant : self.staff->forAll(s | s.salary <= self.salary)

**8) staff must have a salary**

Context : Staff

Invariant : self.salary <> null

**9) staff manages at least one order**

Context : Staff

Invariant : self.orders->notEmpty()

**10) Staff must access all the reservation**

Context : Staff

Invariant: Reservation.allInstances()->forAll(r | self.showsReservation(r))

**11) User only access products that are available**

Context : User

Invariant : self.showsProduct->forAll(p | p.available)

**12) Feedback added by user must not exceed 1000 characters**

Context : User

Invariant :self.feedback->forAll(f | f.text.size() <= 1000)

**13) When a user adds an order, it must include the user's ID**

Context : User

Invariant: self.orders->forAll(o | o.userid = self.id)

**14) Each order's cost is not null**

Context : Order

Inveriant:self.cost <> null

**15) each order’s id must be unique**

Context : Order

Inveriant: Order.allInstances()->isUnique(id)

**16) each order includes at least one product**

Context : Order

Inveriant:self.products->notEmpty()

**17) Products must have a unique id**

Context : Products

Inveriant:Products.allInstances()->isUnique(id)

**18) Products price must be not null**

Context : Products

Inveriant: self.price <> null

**19) Products name not empty**

Context : Products

Inveriant: not self.name.isEmpty()

**20) reservations must have a unique id**

Context : Reservation

Inveriant: Reservation.allInstances()->isUnique(id)

**21) each reservation associated with a user .**

Context : Reservation

Inveriant: self.userid <> null

**22) reservation day should be within a valid range [1-31]**

Context : Reservation

Inveriant:

self.day >= 1 and self.day <= 31

**23) feedback must include a user id.**

Context : Feedback

Inveriant:

self.userid <> null