**RegisteredUser**

**Phone number of Registered User should be of 10 digits.**

Context RegisteredUser

inv: self.phone\_number.size()=10

**Registered User password should be alphanumeric, contain Special character and size should not be less than 8 or greater than 15.**

Context RegisteredUser

inv: password = password.matches((?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%]).{8,15})

**No two flights can have same flight number.**

Context RegisteredUser

inv: self.search-for->ForAll (F1,F2=Flight|F1<>F2

ImpliesF1.flight\_number<>F2.flight\_number)

**The RegisteredUser can only cancel the flight if there exist booking id respective to there name.**

Context RegisteredUser

inv: self.cancel->cancelFlight() implies self.book-for.booking\_id->nonEmpty()

**Administrator**

**Only the Administrator can perform update operation.**

Context Administrator::updateInformation()

pre:self.type\_of\_user=’Administrator’

**Only the Administrator can perform Manage operation.**

Context Administrator::manageProfile()

pre:self.type\_of\_user=’Administrator’

**Flight**

**The departure date of flight should be before its arrival date.**

Context Flight

inv: self.departure\_date<=self.arrival\_date

**The departure city of flight cannot be same as its arrival city**

Context Flight

inv: self.departure\_city<>self.arrival\_city

**FlightInformation**

Context Flight

inv: self.departure\_date<=self.arrival\_date

Context Flight

inv: self.departure\_city<>self.arrival\_city

**BookFlight**

**The booking id issued to one RegisteredUser should not match to the other RegisteredUser.**

Context BookFlight

inv: self.AllInstances->ForAll (B1,B2=BookFlight|B1<>B2

Implies B1.booking\_id<>B2.booking\_id)

Context BookFlight

inv: self.departure\_city<>self.arrival\_city

**The booking id should generates once the payment is made.**

Context BookFlight

inv: self.booking\_id->nonempty implies self.book-through->paymentConfirm()

**Payment**

**For payment to be valid there should be sufficient balance in account.**

Context Payment::paymentConfirm():

Pre: self.amount->sufficientAmount()

**The notification message will be issued on payment and cancel status.**

**Alert**

Context Alerts::generateAlert()

Pre:self.notification.cancel\_status->nonEmpty()

Or self.notification.payment\_status->nonEmpty()