**Solution - Flight Reservation System using Grails Web Application Framework**

As this is a complex project, it would require significant time and effort to develop a complete solution. However, here are the general steps that could be taken to implement the requirements:

1. Set up a new Grails project and configure user authentication and authorization using Spring Security.
2. Define the domain classes for Airlines, Airports, Flights, and Users and use GORM to manage their persistence.
3. Implement the flight search functionality by creating a controller and views that allow users to search for available flights based on the departure and arrival locations, date, and time. Use the domain classes to retrieve flight data from the database and display it to the user.
4. Implement the flight reservation functionality by creating a controller and views that allow users to select seats on a chosen flight, enter passenger information, and confirm the reservation. Use the domain classes to create and save new reservations to the database.
5. Implement the booking management functionality by creating a controller and views that allow users to view their reservations, cancel or modify bookings, and receive confirmation emails. Use the domain classes to retrieve and update reservation data from the database and send emails using Grails mail plugin.
6. Implement the admin dashboard functionality by creating a separate set of controllers and views that allow administrators to manage flights and view user reservations. Use Spring Security to restrict access to the admin features.
7. Test the system thoroughly to ensure it meets all requirements and performs as expected.
8. Deploy the system to a production environment and monitor its performance and security.

Note: This is a high-level overview of the solution and not a complete implementation guide. The actual implementation details may vary depending on the specific requirements and constraints of the project.

**Suggested solution code works**

Step 2: Flight Search

class FlightController {

def search() {

def flights = Flight.findAllByDepartureAirportAndArrivalAirportAndDepartureDateAndDepartureTimeGreaterThan(params.departureAirport, params.arrivalAirport, params.departureDate, params.departureTime)

[flights: flights]

}

}

Step 3: Flight Reservation

class ReservationController {

def reserve() {

def flight = Flight.get(params.flightId)

def seatMap = SeatMap.findByFlight(flight)

if (seatMap) {

def selectedSeats = params.selectedSeats.split(",")

if (selectedSeats.size() > flight.availableSeats) {

flash.message = "You cannot reserve more seats than the available seats on the flight."

redirect(action: "search")

return

}

selectedSeats.each { seat ->

def reservedSeat = ReservedSeat.findBySeatMapAndSeatNumber(seatMap, seat)

if (reservedSeat) {

flash.message = "Seat ${seat} is already reserved. Please choose another seat."

redirect(action: "search")

return

}

}

def reservation = new Reservation(flight: flight, user: session.user, seatCount: selectedSeats.size())

reservation.save(flush: true)

selectedSeats.each { seat ->

def reservedSeat = new ReservedSeat(seatMap: seatMap, seatNumber: seat, reservation: reservation)

reservedSeat.save(flush: true)

}

flash.message = "Your reservation has been confirmed. Reservation ID: ${reservation.id}"

redirect(action: "show", id: reservation.id)

} else {

flash.message = "Seat map not found for the selected flight."

redirect(action: "search")

}

}

}

Step 4: Booking Management

class ReservationController {

def list() {

def reservations = Reservation.findAllByUser(session.user)

[reservations: reservations]

}

def show() {

def reservation = Reservation.get(params.id)

if (reservation.user != session.user && !session.isAdmin) {

flash.message = "You are not authorized to view this reservation."

redirect(action: "list")

return

}

[reservation: reservation]

}

def cancel() {

def reservation = Reservation.get(params.id)

if (reservation.user != session.user && !session.isAdmin) {

flash.message = "You are not authorized to cancel this reservation."

redirect(action: "list")

return

}

reservation.delete(flush: true)

flash.message = "Your reservation has been cancelled."

redirect(action: "list")

}

}

Step 5: Admin Dashboard

package com.flightreservation

import grails.plugin.springsecurity.annotation.Secured

@Secured(['ROLE\_ADMIN'])

class AdminController {

def flightService

def index() {

def flights = flightService.getAllFlights()

[flights: flights]

}

def addFlight() {

render view: 'addFlight'

}

def saveFlight() {

def flight = new Flight(params)

if (!flight.validate()) {

flash.message = "Invalid flight details"

render view: 'addFlight', model: [flight: flight]

return

}

flightService.saveFlight(flight)

flash.message = "Flight saved successfully"

redirect action: 'index'

}

def editFlight() {

def flight = flightService.getFlight(params.id)

render view: 'editFlight', model: [flight: flight]

}

def updateFlight() {

def flight = flightService.getFlight(params.id)

flight.properties = params

if (!flight.validate()) {

flash.message = "Invalid flight details"

render view: 'editFlight', model: [flight: flight]

return

}

flightService.saveFlight(flight)

flash.message = "Flight updated successfully"

redirect action: 'index'

}

def deleteFlight() {

def flight = flightService.getFlight(params.id)

flightService.deleteFlight(flight)

flash.message = "Flight deleted successfully"

redirect action: 'index'

}

}