Airport: Each airport has a unique airport code hence the underline. This is how it is identified. It also has a name, city and state it resides in.

Airplane Type: Each airplane type has a unique TypeName. It also has max seats- and a company it belongs to.

Airplane: Has a unique Id of AirplaneID. It also has a fixed amount of total number of seats on the plane.

Can Land: This is a relationship that relates the airport to the airplane type where they can land at a time

Type: Relates Airplane to airplane type. This is a hard relationship meaning Type relationship needs airplane to exist to exist itself.

Flight: Identified by unique number

Has day of flight and airline that the flight is on

Flight Leg: Has a unique ID of leg number

Leg Instance: Has unique ID date. Along with No-OfAvail-Seats, Leg Distance is an instance of Flight leg for when a flight travels. Information for airplane used and available seats is kept in leg instance.

Reservation: In leg instance, all the reservations for every customer include the customer name, cell phone, seat and seat #

Problem Four (one solution per team):

a. What is the name of your team?

R^2

b. Provide a description (less than one page) of the system you'll be building for your final project.

We are still deciding but we want to do and other ryan will have a definite answer when he submits his work.

This description should focus on the REQUIREMENTS and GOALS of the system.... in other words WHAT the system will do. For example, the library system has two principle users: librarian and patron. Each user has a different kind of user interface. The librarian has an administrative panel allows the books to be added/removed from the inventory, ... etc.

The description should not discuss HOW the system will meet these requirements. For example, don't say "we will use the Java mapset collection to maintain a local cache of a patron's books".

I'll use this description to help ensure your team isn't taking on too much or too little for the final project.