Question 1

**Using the airline reservation system as an example, under what conditions would a transparent**

**façade be more appropriate to use than an opaque façade?**

The artricle at <https://msdn.microsoft.com/en-us/library/orm-9780596527730-01-04> mentions the fact that an opaque façade is stringent, allowing access to only selected operations, and therefore a transparent façade is needed when users want to get at the individual operations of particular subsystems.

In the case of the Airline reservation system, an opaque façade would be appropriate to use when customers are using the system. This is because anything open to the public online or for someone other than employees of a company would have restrictions on what they could do. Customers and potential customers can view flight information and perform a fixed set of operations. Therefore an opaque façade for customers and potential customers is appropriate.

For the same system, employees have access to a wider range of operations, and therefore a transparent façade is more appropriate in their case. You would want to allow employees access to operations like adding flights and aircraft and other things unavailable to customers. So a transparent façade for employees would be appropriate.

It is also possible that different employees may require opaque facades that still allow the use of more operations than the opaque customer façade, one, i.e. a high level manager might use a transparent façade but a lower level employee might use one that is opaque but still allows the use of more methods than an opaque façade for customers.

In the case of additions or changes to the system, it would be necessary to determine which set of services is affected, since one set of services is for customers, any changes or additions to that part of the system would use the opaque façade for customers. In the case of the second set of services which is used by employees, changes would have to be made to the transparent façade or several opaque facades allowing different levels of access as described above.

From the article: https://msdn.microsoft.com/en-us/library/orm-9780596527730-01-04

The façade described in the preceding example is *opaque*, in that the subsystems cannot be accessed except via the Facade object. This requirement might be too stringent. Suppose some users want to get at the individual operations of particular subsystems. We can change all the internal modifiers to public, which will make the façade optional, or *transparent*. That is, as well as being able to go through the Facade, the client will be able to instantiate SubsystemA directly, for example, and then call A1.

So using a transparent façade might make the most sense when you need to complete a full operation and save data, such as actually creating a ticket and reservation object. This is different from performing an operation just for informatin gathering purposes, like displaying a list of current flights and rates. In cases like that, an opaque façade could be used because you only need to invoke certain methods to gather this information.

Also, security could be a concern, maybe opaque facades make sense to use for items available to the general public online whereas a transparent façade is used by employees logged in to a system, or possible different employees could have different facades of different levels of “opacity”, i.e. you make some methods public versus private as appropriate.