Design patterns (draft)

1. **Decorator :** Let's you attach new behaviours to the objects by placing them inside special wrapper objects that contain the behaviours. You can change the object at any moment and it works with inheritances. For example we can say that we have X number of guests and the menu for them is pizzas, depending what ingredients you want the pizza to be and the number of guests so the total price of adding can be calculated (ex: mushrooms in pizzas for all the guests, or one more salad in the table). Also you can add, remove objects/ingredients in this case.

2. **Observer :** When a customer books a business or planner it is necessary that business and planner should be notified, so instead of refreshing the list of bookings many times it will be more efficient if they are going to be notified at the moment when someone books them.

3. **Prototype :** In database we are going to store bookings that a customer has made, their account details, personal information etc., all of this in separate tables. Instead of writing the names all over again we are going to use prototype to copy the tables and use them again.

4. **Builder :** All the process of organizing an event should be a step-by-step process. First the customers select a date and then gives their preferences, then they can choose a planner if they want, then they find the venue, then they book a business etc. So this process is always done in this way and not by including objects in random order like booking a business first then selecting the date and then finding the venue because it is not efficient.