A assignment was given to me to create the back end structure for an application used to generate user names and passwords for guests on a network. The main challenge of this was to keep the framework as expandable and flexibly as possible since future additions were likely. A few JavaScript libraries that did this already existed, but they were very bulky and overpowered for what was needed, and often had to be altered to fit the data model. I decided to write my own and its worked very well. I worked by building a very specifc framework, then abstracting out every component that I could which resulted in a system that could expand in almost every way possible. A few instances of people wanting to add things and being able to do so easily have occurred so I'd say it was a success.

A less successful assignment given to me was to change the way our tables sorted themselves. The framework that we were using did not support what we were trying to do with it so altering they way it worked involved changing some very important core functions that resulted in a fragile system that caused many problems later on. In hidsight there were better ways to do it.

As a convergent problem solver I approach problems very systematically, starting with a working solution, then improving on it through many iterations. As seen in my first example this results in a system that works from the start and only improves as it goes until it meet the requirements of the assignment. A downside of this is a unwillingness to branch away from a working system and just dump everything and start new. This would have probably been the best solution in my second example. I bring to the team a problem solver that will always have at least something that works, and never be completely without results.