1. **What should a requirement include?**
   1. In general – a list of all of the things the code should allow to do
      1. Tech only – be generic so it will be easier for us to add things in the future – **Non Functional**
      2. User related – the user should be able to do this and that (each and every action and scenario) – **Functional**
   2. They should be separated by individual values to the **stakeholder**, which are listed by value priority to the user.
2. **How can I use requirements to measure my team's performance?**
   1. We measure **only** value to the user (we can call them UserStories).  
      If a codebase was written but it help nothing by itself, we get a 0.
   2. At the end of the sprint:
      1. Compare the product we have to the list of requirements at the beginning.
      2. Understand if there are things we didn’t know that we didn’t implement (if the product misses requirements from the list and we didn’t decide on that)
      3. After production, understand if we got what the users wanted (maybe we missed requirements? Maybe the spec wasn’t good enough?)
   3. It works when 1B is implemented
3. **How can I teach my team to specify requirements when they take on a task?**
   1. At first you need to ask to see the specified requirements to catch the mistakes early
   2. Then by measuring them by Value to a user they will understand that good requirements are key for succeeding at bring value to the users.
4. **Problems:**
   1. Never on deadlines
   2. Barely delivering to production
   3. The soldiers don’t get a sense of progression (b related)
   4. Teams are not working… as a team. Each one has a different project (Daily is boring and useless, when someone is missing the project is stuck… etc)