Designing Software

What's Design?

- The mapping of a certain concept into implementation.
- We're trying to implement software systems to solve a certain problem in an application or a particular domain.

When should we create design?

- In the olden days → **Big Upfront Design (BUF)**.
  - Design everything upfront and then code everything.
  - Doesn't really work.
  - Bad idea.
- Design is so important that we do it continuously in Agile Development.
  - Evolve/Refactor design, each day if needed.
- Think through some details and have some initial design, that can be updated/changed as time goes on.

Features aka User Stories

- **Value** → give a value score to the features. Eg. a library application *needs* the feature to check books out, but sorting the books a user has ever checked out is not needed (it'd be good to have, so can be implemented later on).
- **Impact** → what impact will the feature have on the architecture.
- To decide what features to implement first, choose the ones with the most value & impact.
  - Same value? Choose the one with the most impact.
  - Same impact? Choose the one with the most value.

How to know if the design is good?

- If the design is easy to evolve and make change, and maintain it.
- **Entropy** → tendency of a system to rot and go bad. Software has quite a bit of entropy. So, we have to maintain it.