**Layout assumptions:**

* Assumed AWS stack
* Expect recalls to come in X per hour.
* Org is ok w/webservers.
* All files should be stored in the laid-out model. Assuming pdfs, pngs, jpeg. Misshaped pngs, or non-standardized files/images are not accounted for.
  + How many pics, files, are the users are uploading from the vendor, or third party (i.e. FDA, internal teams, etc.)?
* Server and platform choices.
* Grafana is a visualization tool, can be replaced by PowerBI or other tool.
* Working assumption is that the business has an application performance monitoring system that is standard across the org and a reference architecture to support this baseline.
* Org is setup for standard practice for DevOps and test automation to setup maintenance windows.
* Create section for a decision matrix.
  + Decision - Refer to critical choice the org supports AWS as primary supplier for cloud vendor the recommendation was to stick to AWS because of cost benefits, and potential data leakages (i.e. b/c of Microsoft’s relationship with OpenAI there’s a potential chance or a data leak and at the same time this is based on a standard referential architecture).
  + Decision - The patterns and anti-patterns, what does this look like from a data governance perspective (i.e. privacy, consumer rights, legalities, etc).
  + Decision to use queues vs streaming message infra is due the expected load of notifications not being clearly defined from the biz requirements perspective so this assumption was made.
    - If a streaming infra is required the architecture can be adapted, but it’s not out of scope.
  + Decision - Deployment model was made to use containers vs VMs, this was due to cost and tech choices.
  + Decision - Choice to use small tech language the is event based and modern enough for business to adapt long term.
  + Decision - Closed source vs open source: the system will preclude the use of open source tech unless authorized by governance/legal/compliance team.
  + Decision – services will be using a standard REST API due to the adaptability of the framework and auto generation of client code (e.g. we can build to OpenAPI standard 3)