**CADDi**

**Technical Architecture Work Sample**

# Brief project

* Customer: large manufacturing companies, there is about 500+ employees each company (total: n x 500)
* Main use case: Communication between mechanical engineers, procurement, and factory
* Vague requirements:
  + File storage and sharing, where the files are primarily PNG and JPEG photos
  + Be able to annotate the photos with circles and start a message thread
  + Allow multiple individuals to respond in the message thread
* Non-functions:
  + There is quite a bit of time pressure to show that this software is viable and delivers value to users
  + You are the first engineer assigned to this project, and no code has been written yet. Your design  
    document will drive the resourcing strategy, including engineer reassignments and new hiring.

# Solution draft and Development plan options: PhotoCom system

PhotoCom

System

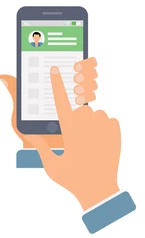
Postgres

GC Storage

CADDi Systems

ElasticSearch

System



Users

use mobile

## Functions: (minimum)

* Register/Login
* Upload photo
* Annotate photo (oval + comment)
* Thread circle (post, view, response)
* Display photo + annotation overlay

## Plan options:

### PhotoCom System:

* Frontend (Web app Mobile/Mobile): Upload photo, annotate (oval + comment), view + response thread, search. Responsive web app or mobile app is preferred because of using a mobile phone when factory worker notices a scratch on a part suddenly.
* Backend Service (Spring Boot hoặc Node.js):
  + Authentication + Authorization (OAuth2, JWT or CADDi user API?): a user only response another user’s threads that the both are belong to the same company.
  + Photo functions: upload, dowload
  + Annotation functions: save, load and overlay with photo
  + Thread functions: list thread, view thread, response thread
* Storage:
  + **Image files**: GC storage
  + **Metadata & Annotation**: PostgreSQL
  + Search (text): ElasticSearch
* CI/CD + DevOps:
  + Dockerized services, deploy bằng Kubernetes
  + Support operation (Logging)
  + Monitoring (optional): Prometheus + Grafana

### Plan for PhotoCom System

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Item** | **Plan 1 - Backend Monolith**  (Thread service only: contains all functions) | **Plan 2- Backend Microservices**  (Auth service, Photo service, Thread service, ..) |
| 1 | Developer | 1–2 person | 3–5 person |
|  | Tester | 1 person | 1-2 person |
| 2 | PoC | 01 month | 02 month |
| 3 | Scale | Module scale | Service scale |
| 4 | Total Project cost | Low | High |

\* **Suggestion plan:**

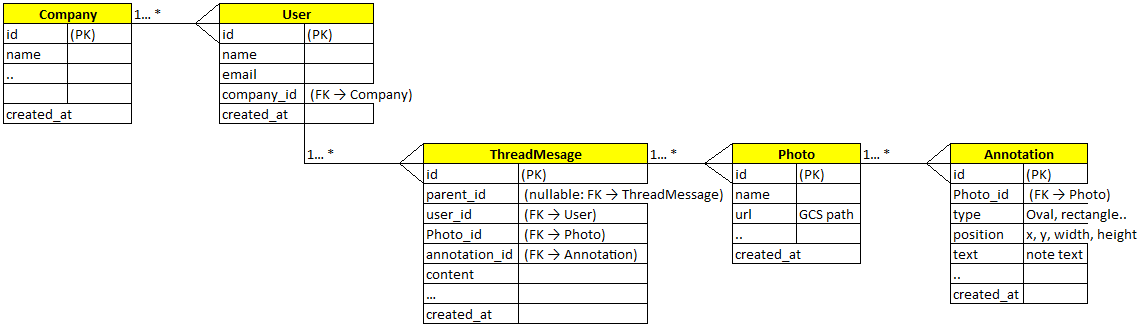
* If number of users <= 10 x 500 or low traffic: **Backend Monolith +** Frontend Web

=> then convert to **Backend Microservices** if number of users become large enough or high enough traffic (annual growth prediction)

* Otherwise:number of users > 10 x 500 or high traffic: **Backend Microservices +** Frontend Web

# Design system

## Database entity relationship



* User information: User, Company table contain this information or provided by CADDi user API (keycloak?)?
* Thread information:
  + Thread: start by root (parent\_id = null), then following messages that connected by parent\_id (if any limit only connects to the root message so maximum thread depth size is only 2)
  + Thread Message: its description is built up its own content + photo + annotations

## Services Integration

* API flows: call API login first to get user information => check authorization that based on the user information when calling to any other APIs
* Security: Frontend + Backend should be configured to prevent CORS security issue.
* Backend is integrated with APM + ELK: to support Devops to operate and to supply API performance measurements such as response time to monitor operation latter.
* If Search text function is preferred: should be isolated to APIs, so implemented independently such as feed newly created thread message by a background job (that use time slice window technique to feed to ElasticSearch,)
* Deploy the system on CADDi infrastructure so use existing feature for security such DDOS, Load Balancing, ..