1. create an account
2. upload my images
3. follow with my friends,
4. share my pics with my friends
5. be able to like
6. able to repost some pic

non func:
1. scalable, 10 million users
2. be reliable

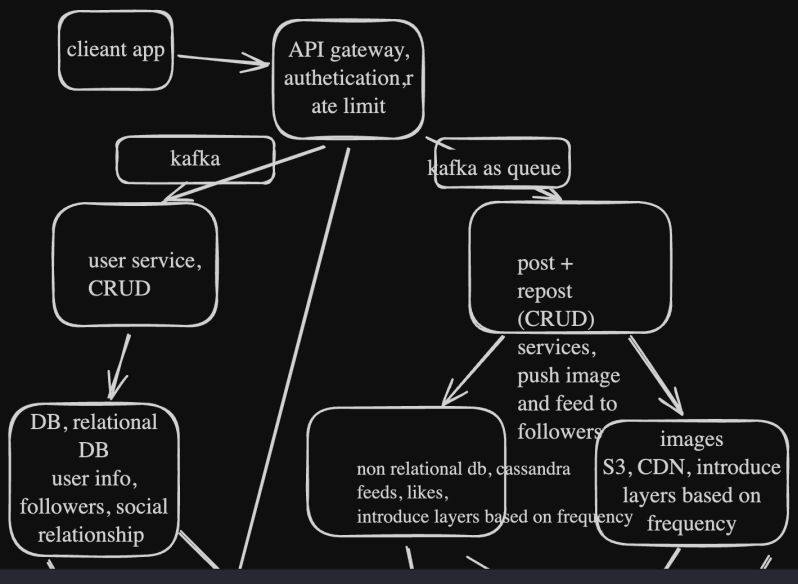
we divide to 4 steps
1. business req
function req:
1. create an account
2. upload my images
3. follow with my friends,
4. share my pics with my friends
5. be able to like
6. able to repost some pic

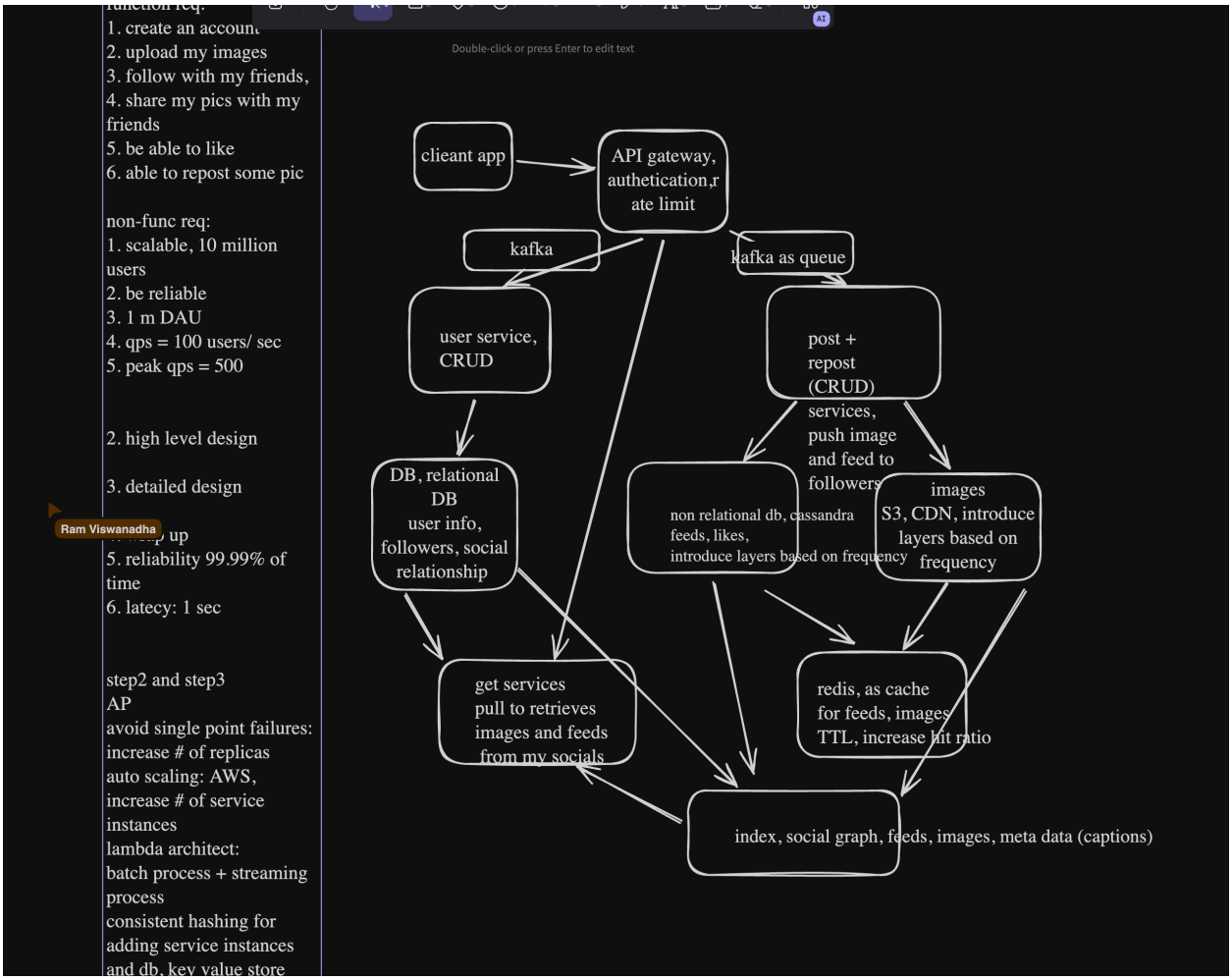
non-func req:
1. scalable, 10 million users
2. be reliable
3. 1 m DAU
4. qps = 100 users/ sec
5. peak qps = 500

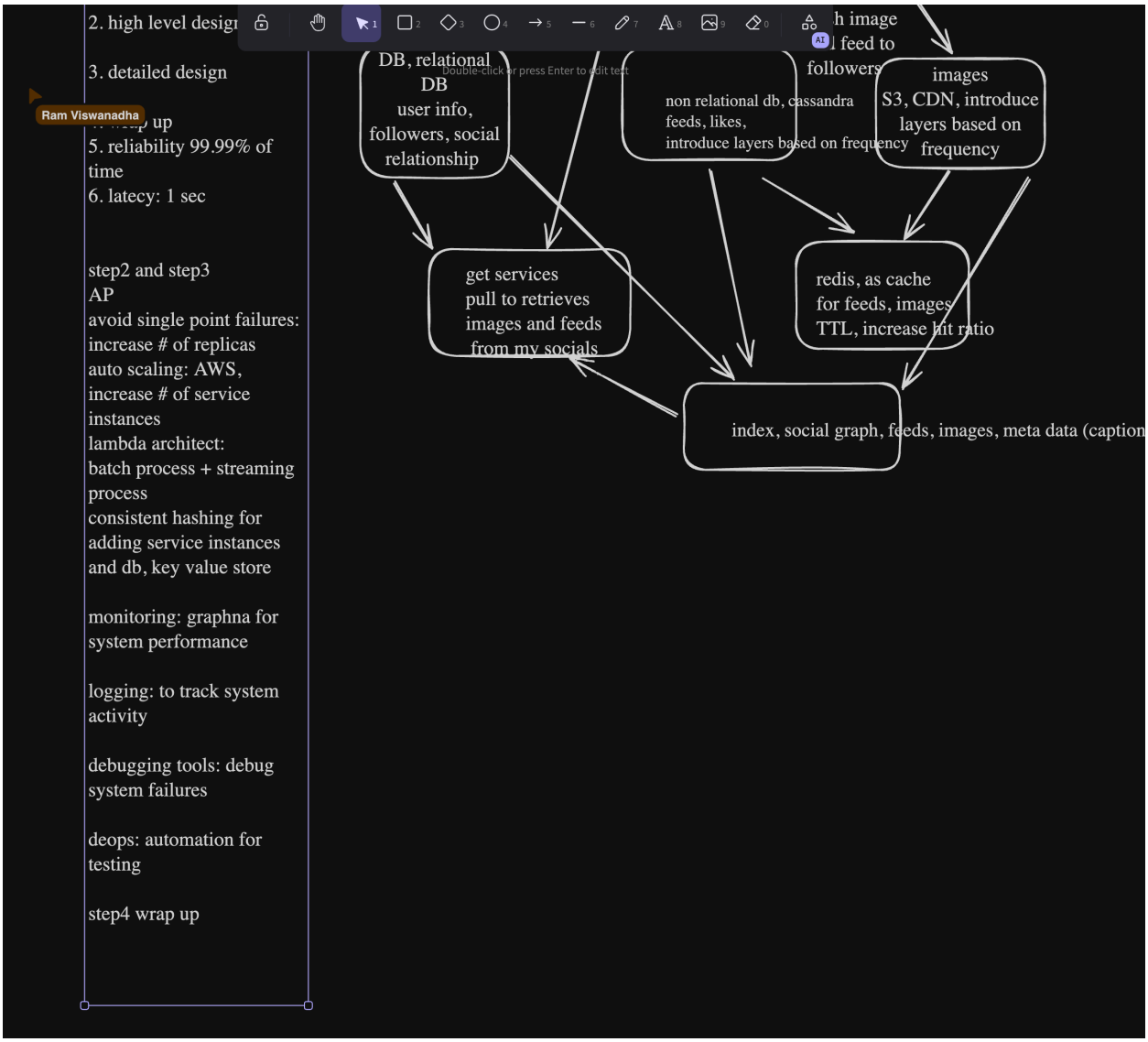
2. high level design

3. detailed design

up
5. reliability 99.99% of time




function req:
1. create an account
2. upload my images
3. follow with my friends,
4. share my pics with my friends
5. be able to like
6. able to repost some pic
non-func req:
1. scalable, 10 million users
2. be reliable
3. 1 m DAU
4. qps = 100 users/ sec
5. peak qps = 500
2. high level design
3. detailed design
Ram Viswanadha
5. reliability 99.99% of time
6. latecy: 1 sec
step2 and step3
AP
avoid single point failures:
increase # of replicas
auto scaling: AWS,
increase # of service instances
lambda architect:
batch process + streaming process
consistent hashing for adding service instances
and db, key value store
Double-click or press Enter to edit text
clieant app
API gateway, authetication,rate limit
kafka
kafka as queue
user service, CRUD
post + repost (CRUD) services, push image and feed to followers
DB, relational DB user info, followers, social relationship
non relational db, cassandra feeds, likes, introduce layers based on frequency
images S3, CDN, introduce layers based on frequency
get services pull to retrieves images and feeds from my socials
redis, as cache for feeds, images TTL, increase hit ratio
index, social graph, feeds, images, meta data (captions)


2. high level desig
3. detailed design
Ram Viswanadha
up
5. reliability 99.99% of time
6. latecy: 1 sec
step2 and step3
AP
avoid single point failures:
increase # of replicas
auto scaling: AWS,
increase # of service instances
lambda architect:
batch process + streaming process
consistent hashing for adding service instances and db, key value store
monitoring: graphna for system performance
logging: to track system activity
debugging tools: debug system failures
deops: automation for testing
step4 wrap up
Double-click or press Enter to edit text
h image
l feed to
followers
DB, relational DB
user info, followers, social relationship
non relational db, cassandra
feeds, likes,
introduce layers based on frequency
images
S3, CDN, introduce layers based on frequency
get services
pull to retrieves
images and feeds
from my socials
redis, as cache
for feeds, images
TTL, increase hit ratio
index, social graph, feeds, images, meta data (caption

non-func req:
1. scalable, 10 million users
2. be reliable
3. 1 m DAU
4. qps = 100 users/ sec
5. peak qps = 500

2. high level design

3. detailed design

4. wrap up
5. reliability 99.99% of time
6. latecy: 1 sec

Ram Viswanadha

step2 and step3
AP
avoid single point failures:
increase # of replicas
auto scaling: AWS, increase # of service instances
lambda architect:
batch process + streaming process
consistent hashing for adding service instances and db, key value store

monitoring: graphna for system performance

logging: to track system activity

debugging tools: debug system failures

deops: automation for testing