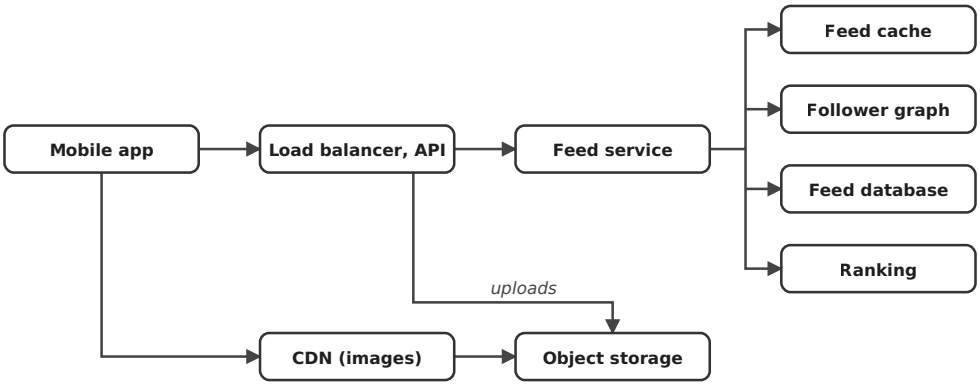
Photo Feed · Defend or Recover?

Scenario: (A) (B) (C) (D)

Andrew IDs: _____ _____ _____ _____ _____

Your group runs the photo-sharing app from last week. Your group analyzes one of the four scenarios on page 3; read the others for the debrief. Decide what can keep working during the failure and what the team must recover. Use brief bullets; complete sentences are optional.



Facts

- The feed cache answers about **95%** of feed loads; misses go to the follower graph and feed database.
- Ranking orders the posts on **every** feed load, in about 100 ms.
- The feed database holds each post's caption and image link, and has one replica. The photos themselves are in object storage.
- Everything runs in **one cloud region** of **one cloud account**.
- **Feed SLO:** 99% of feed loads finish in under 1 second.

Key terms

- **Resilience pattern:** a design element that stops a failure from spreading: a timeout, retry with backoff, circuit breaker, bulkhead, load shedding, or graceful degradation.
- **Disaster:** a failure the system cannot ride through, so the team must recover.
- **Failure domain:** a set of components that a single fault can take down together, such as one disk, one account, or one credential.
- **RPO:** the most data, measured in time, the app can afford to lose.
- **RTO:** the longest it can afford to be down.
- **Blast radius:** the set of users and systems a rehearsal can affect.
- **Abort condition:** an observable threshold, agreed in advance, that ends a rehearsal at once.

1 · What would users see?

2 · Could a resilience pattern contain it? Which one, and what would users see instead?

3 · What still needs recovery? Propose RPO and RTO targets with assumptions. What must be prepared in advance? If no data recovery is needed, explain why.

RPO, RTO, and assumptions

Prepared in advance

4 · Which rehearsal would show that the defense or plan works? What are its blast radius, abort condition, and way back?

Rehearsal

Blast radius

Abort condition

Way back

Scenarios

Scenario A · Ranking slows down

After a model update, the ranking service takes about **3 seconds** instead of 100 ms for half of its requests.

Every other service is healthy. The feed service waits for ranking using its client library's default settings.

Scenario B · The CDN stops serving images

The CDN returns errors for **every image request**. Captions and uploads still work, and the photos are safe in object storage.

The CDN normally answers 90% of image requests without contacting object storage, which can handle only **twice** its normal image reads. An engineer proposes sending every image request straight to object storage.

Scenario C · A migration corrupts the feed database

A Friday afternoon migration rewrote the image links in the posts table. A bug pointed **30% of posts at the wrong photos**, and the replica copied every change. Users report it the next morning.

A database snapshot is taken nightly and stored in the same account. Nobody has ever restored one.

Scenario D · The storage credential expires

At 02:00 on Sunday, the credential the API uses to write new photos to object storage expires. **Every upload fails**; existing photos still load.

Only one engineer can create a new credential, and they are on a flight.