# Welcome!

WWCode San Francisco - Backend Study Group

March 16, 2023

- We'll start in a moment :)
- We are **RECORDING** tonight's event
- We may plan to take screenshots for social media
- If you are comfortable, turn the video ON. If you want to be anonymous, then turn the video off
- We'll make some time for Q&A at the end of the presentation
- Feel free to take notes
- Online event best practices:
    - Don't multitask. Distractions reduce your ability to remember concepts
    - Mute yourself when you aren't talking
    - We want the session to be interactive
    - Use the 'Raise Hand' feature to ask questions
- **By attending our events, you agree to comply with our Code of Conduct**

WOMEN WHO CODE /san-francisco

# Introduction & Agenda

- Welcome from WWCode!
- Our mission: Empower diverse women to excel in technology careers
- Our vision: A tech industry where diverse women and historically excluded people thrive at any level
- Backend Study Group: Learn and discuss backend engineering concepts

System Design - Series Part 3 of 3:
- How to interview?
- Solve a system design question
- Q & A

Prachi Shah
Instructor
Senior Software Engineer, Unity
Director, WWCode San Francisco

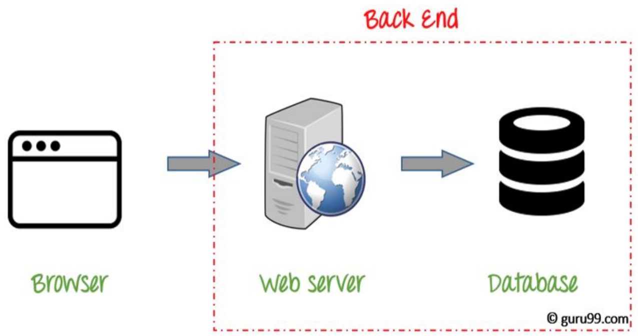
Anjali Bajaj
Co-Host
Lead, WWCode San Francisco

WOMEN WHO CODE® /san-francisco

# Backend Engineering

- Design, build and maintain server-side web applications

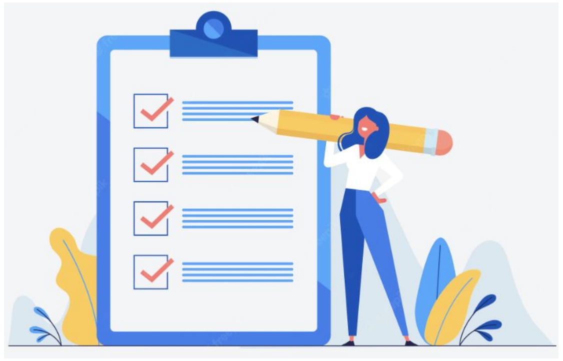
- Concepts: Client-server architecture, networking, APIs, web fundamentals, microservices, databases, security, operating systems, etc.



- Tech Stack: Java, PHP, .NET, C#, Ruby, Python, REST, AWS, Node, SQL, NoSQL, etc.

# System Design

- Solve a problem or build a product
- Defining the architecture, modules, interfaces and data flow
- Architecture: Defines behavior and view of a system
- Modules: Each module corresponds to a task
- Interfaces: Defines the communication between modules
- Data flow: Flow of data and information between systems
- Define the input, output, business rules, data schema



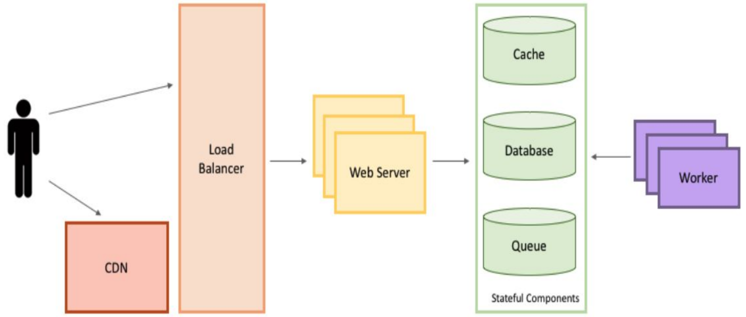
WOMEN WHO CODE /san-francisco

# Interview

- Product-design, components, data flow, integrations
- Examples: Frontend, Backend, APIs, Data Models, Database, Server, etc.
- Framework:
    - Open-ended questions
    - Understanding of systems, components and interaction
    - Communication and clarity - it's a discussion
    - Pros/cons and unknowns
    - Short-term vs. long-term thinking

# Design Considerations

- Scaling: Change in performance as per changing application demands
- Availability: System uptime and downtime
- Reliability: System performs the tasks as expected
- Robustness: Functional when errors or disturbances
- Load Balancing: Network traffic distribution across servers
- Caching: Data storage layer
- Data Partitioning: Distribute data across systems to improve querying performance
- SQL vs. NoSQL: Relational vs. Non-relational data model
- Performance: Glitch-free* and fast
- Extensibility: Future growth
- Error Handling and Security: UX and secure data



** requirements may vary*

## Do’s

- Ask questions to understand the requirements. Any domain knowledge?
- Note all the topics you want to discuss
- Visualization helps! Draw a component diagram
- Make some assumptions and frame a recommendation
- Do what the interviewer asks you to focus on
- There is no right or wrong answer
- It’s OK to mention that you do not have familiarity with a technology/ tool/ concept
- Discuss trade-offs

# Don'ts

- Don’t start coding
- No need to code the API *
- No need to code the data models *
- Don’t be quiet. This is a collaboration
- Don’t delve into a technology *
- Don’t delve into a module/domain *

** suggested by the interviewer*

# Question

- Design Instagram for ten users. Then scale up to millions of users

- Components:
    - Requirements:
        - Functional: Consider product requirements
        - Non-functional: Consider system design principles
        - Data Models: SQL/NoSQL, caching, data partitioning, load balancing, etc.
    - Backend Services: Microservices, internal vs. external integration
    - API and Frontend: REST APIs and user experience
    - Security and beyond: AuthN, AuthZ, extensibility, etc.

- Draw the Building Blocks

- Discuss design and iterate

instagram

6,368 Posts | 350 M Followers | 101 Following

**Instagram**
Bringing you closer to the people and things you love. ❤️
www.facebook.com/covidsupport

WOMEN WHO CODE /san-francisco

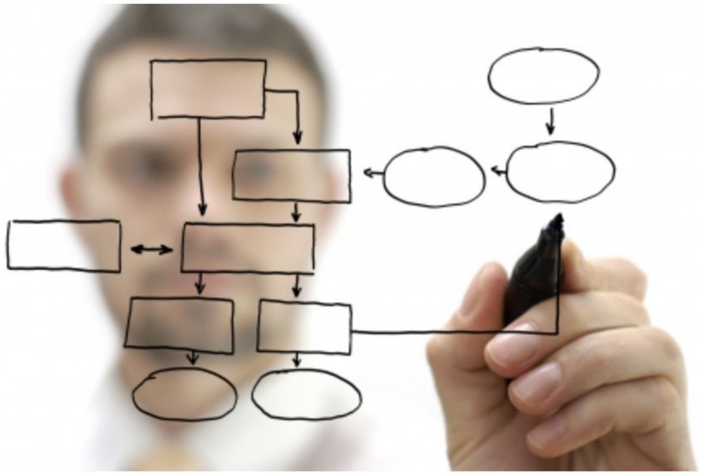
# Requirements

Functional:

- Create a post with images and videos
- Update a post
- Delete a post
- Follow and unfollow an account
- View account profile
- Feed of posts
- User management
- Notifications

Non-functional:

- Ten users
- Mobile app and website
- Process images and videos
- Limit text to 255 characters
- Limit posts to 6 media (images and videos)



# Services

Services:

- Post service
- Media service
- Feed service
- User management service
- Account management service
- Authentication service
- Analytics service

Datastore:

- User database - SQL
- User feeds database - SQL
- Posts database - SQL
- Media datastore - NoSQL
- Account management datastore - SQL
- Analytics datastore - SQL

WOMEN WHO CODE® /san-francisco

# Data

Tables:

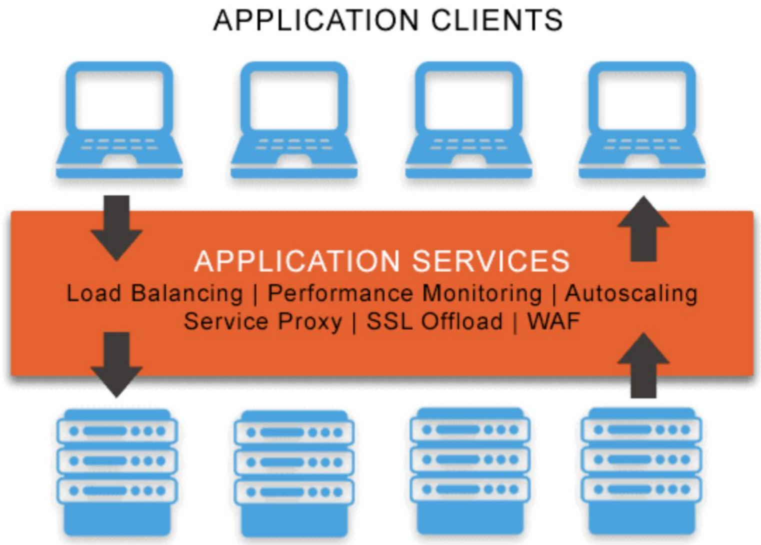
- User database - profile (description), photo, friends, media, posts, primary key (ID)
- User feeds database - list of friends, posts, timestamp, primary key (ID)
- Posts database - timestamp, post, media, user, primary key (ID)
- Media datastore - media link, user, timestamp, storage (AWS S3), key (ID)
- Account management datastore - user, account status, timestamp, notifications
- Analytics management datastore - logs, timestamp, user activity, post activity

Relationships:

- User database - links to post, media and account management
- User feeds database - links to users, posts, media
- Posts database - links to user, media
- Media datastore - links to user, post
- Account management datastore - links to user, posts
- Analytics management datastore - links to user, feeds, posts, media, account

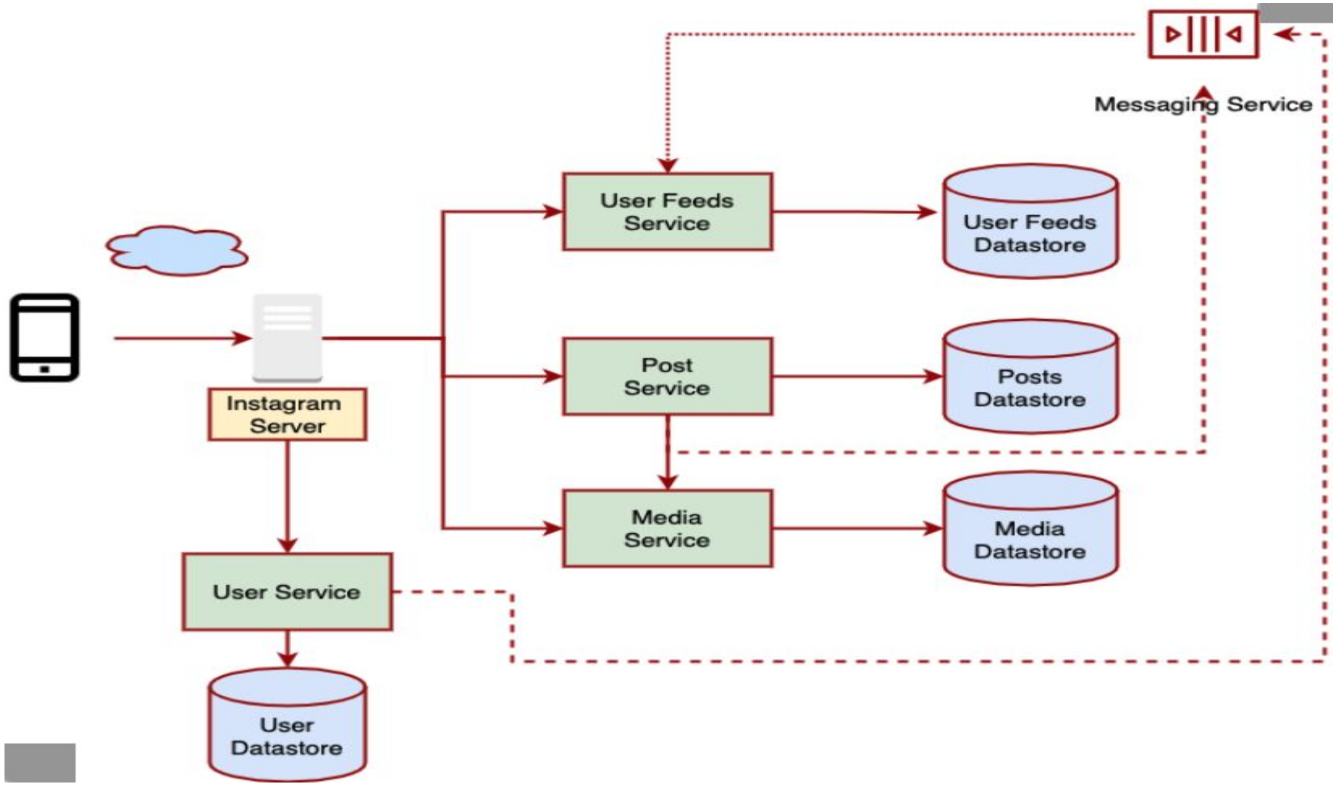
# Infrastructure

- REST APIs for features/functionality, Rate limited to 50 requests per user profile
- Gateway and load balancer (web servers)
- Authentication Service
- Caching (images, videos, post)
- Data partitioning (geography)
- Data in cloud
- Application servers for hosting application
- Docker and kubernetes for container orchestration
- External service integrations (Splunk and Datadog for monitoring)

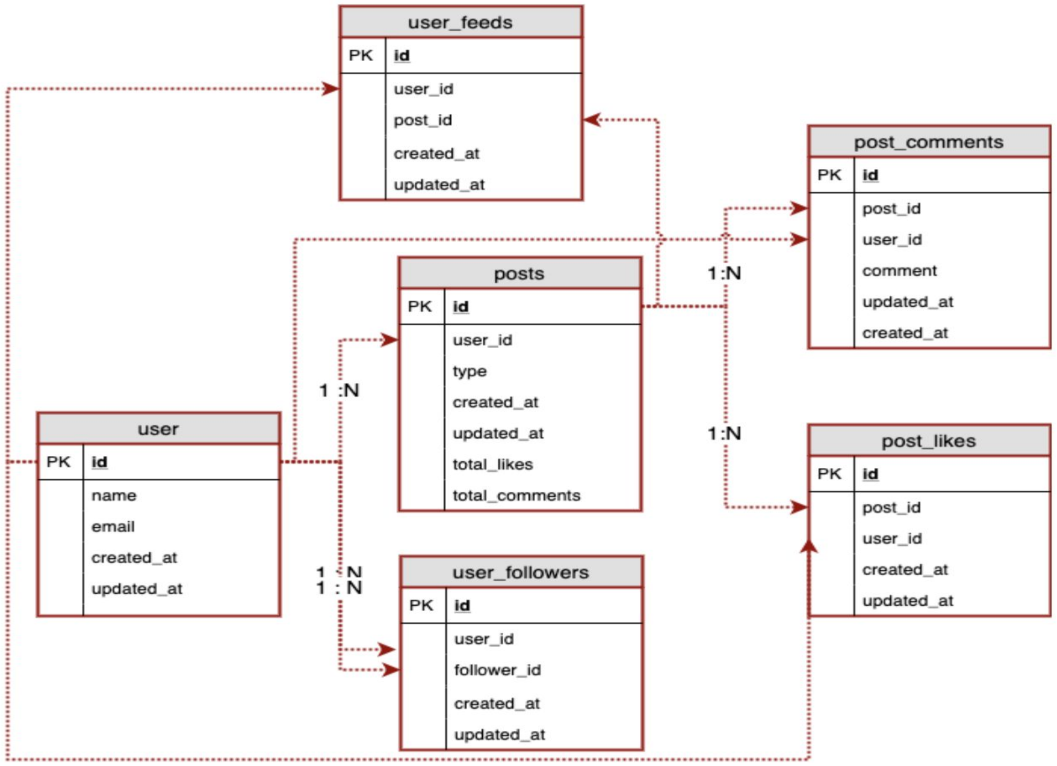

APPLICATION CLIENTS
APPLICATION SERVICES
Load Balancing | Performance Monitoring | Autoscaling
Service Proxy | SSL Offload | WAF

WOMEN WHO CODE® /san-francisco

## Architecture

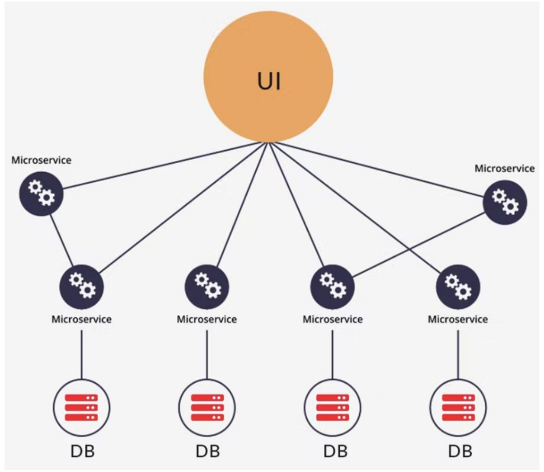


# Data Models


user_feeds
PK id
user_id
post_id
created_at
updated_at
post_comments
PK id
post_id
user_id
comment
updated_at
created_at
1:N
posts
PK id
user_id
type
created_at
updated_at
total_likes
total_comments
1 :N
1:N
post_likes
PK id
post_id
user_id
created_at
updated_at
user
PK id
name
email
created_at
updated_at
1 : N
1 : N
user_followers
PK id
user_id
follower_id
created_at
updated_at

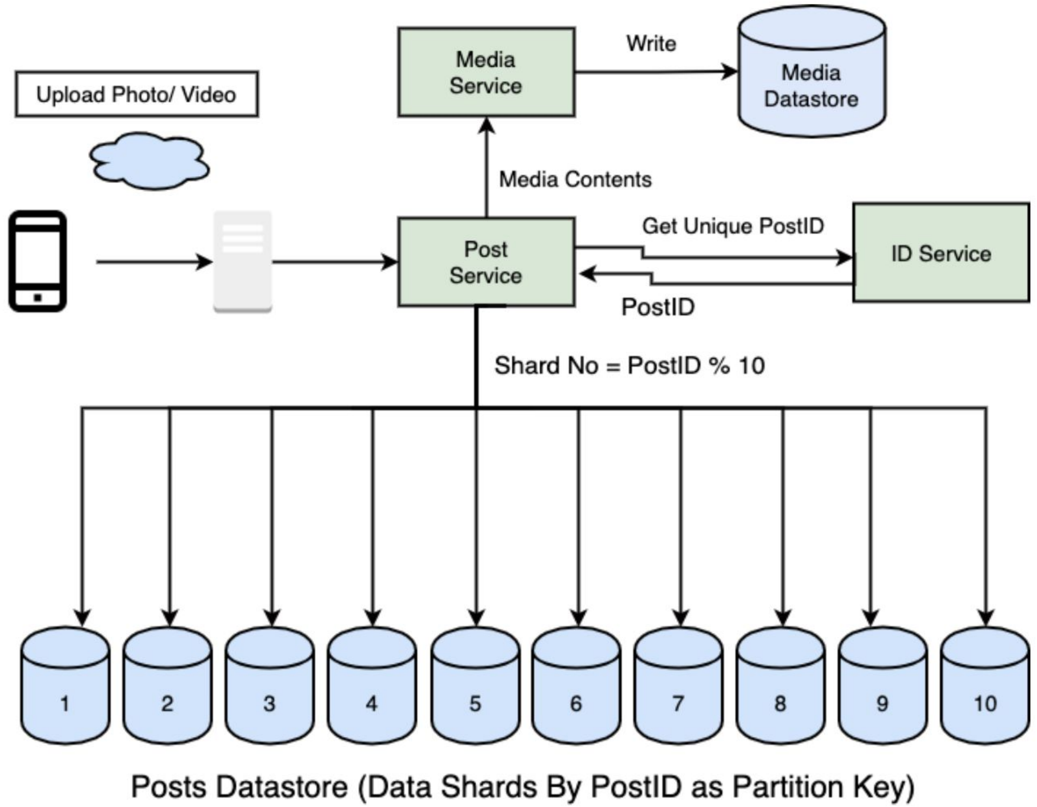
# Scaling Up

- Load Balancing:
  - Assign incoming client requests to distributed resources
  - Prevents overloading resources (servers, database)
- Caching:
  - Optimize system for read operations (ideally)
  - Reduces load on servers and databases
  - Optimizes distributed traffic management
- Data Partitioning:
  - Splitting data across multiple tables and datastores
  - Improve maintainability, performance, availability, load balancing and cost effectiveness
  - Horizontal partitioning (Sharding) vs. Vertical Partitioning
- Distributed storage: SQL and NoSQL databases and CAP theorem
- On-premises vs. Cloud infrastructure
- Monolith vs. Microservice

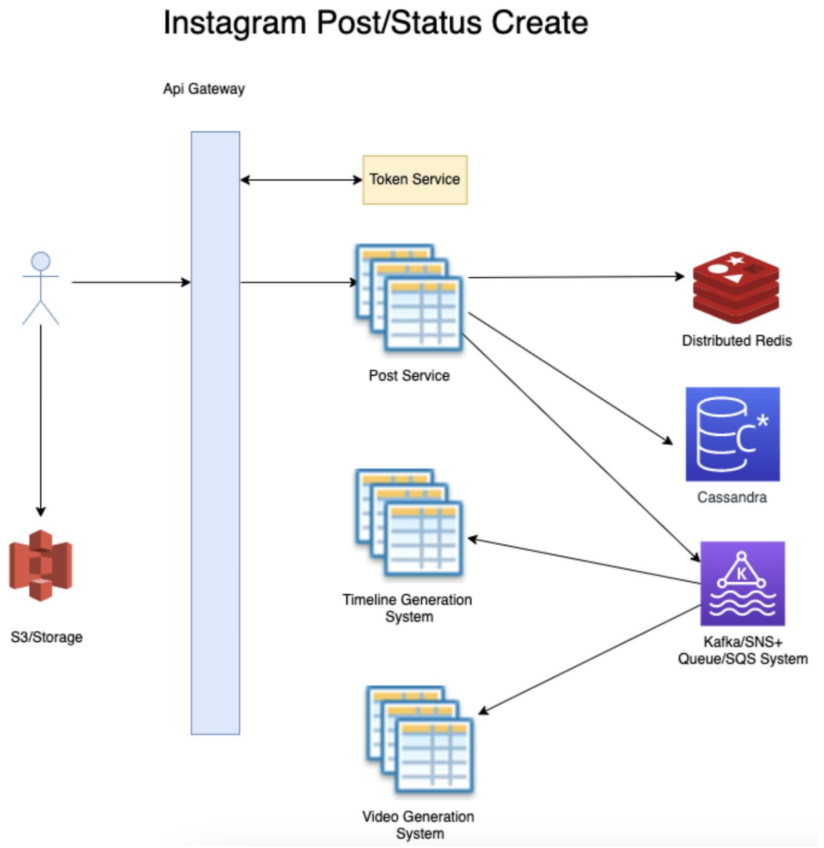


# Scaling

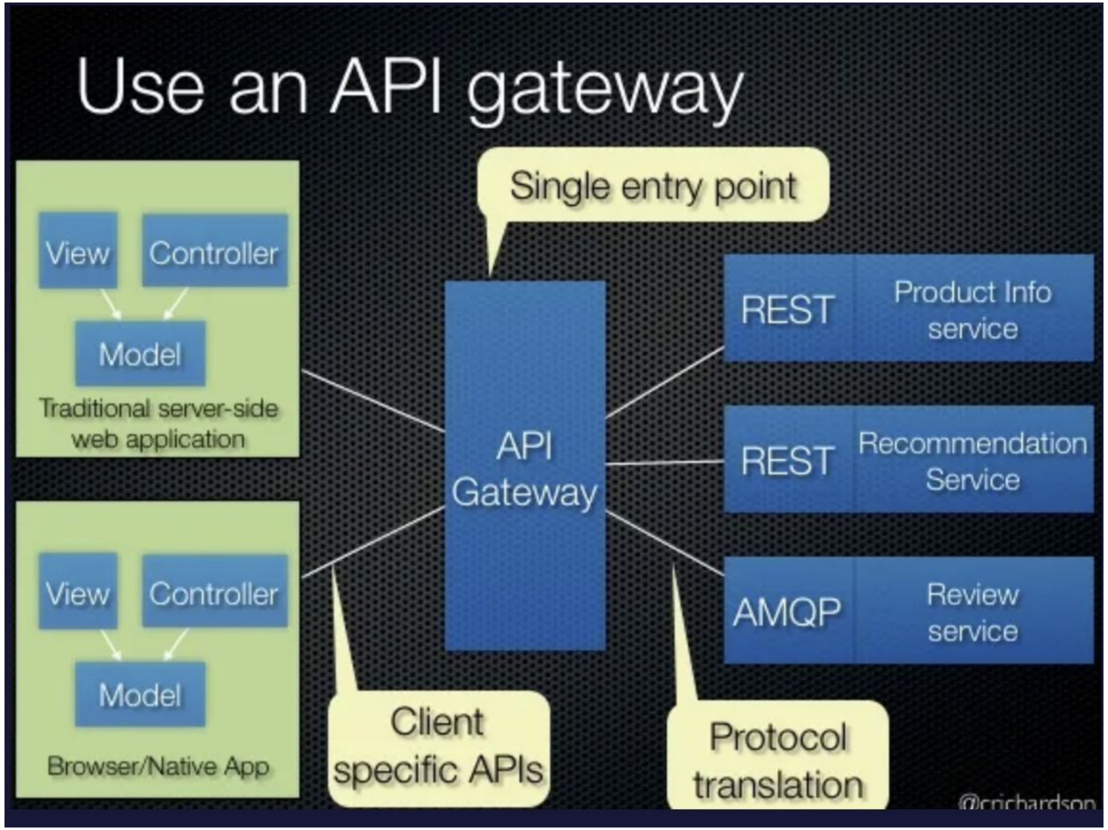
- Shard by Users or Posts



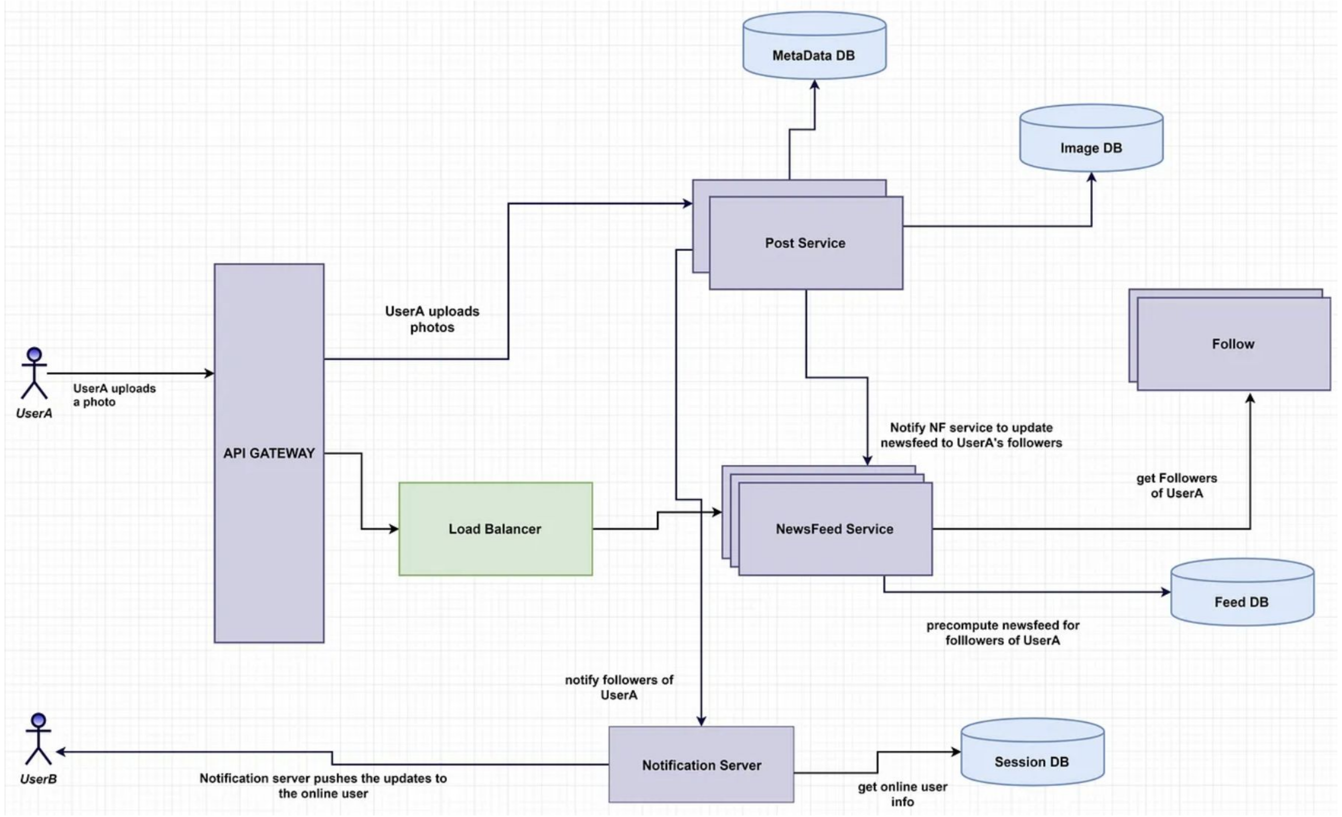
# Design


Instagram Post/Status Create
Api Gateway
Token Service
Distributed Redis
Post Service
Cassandra
Timeline Generation System
Kafka/SNS+ Queue/SQS System
S3/Storage
Video Generation System

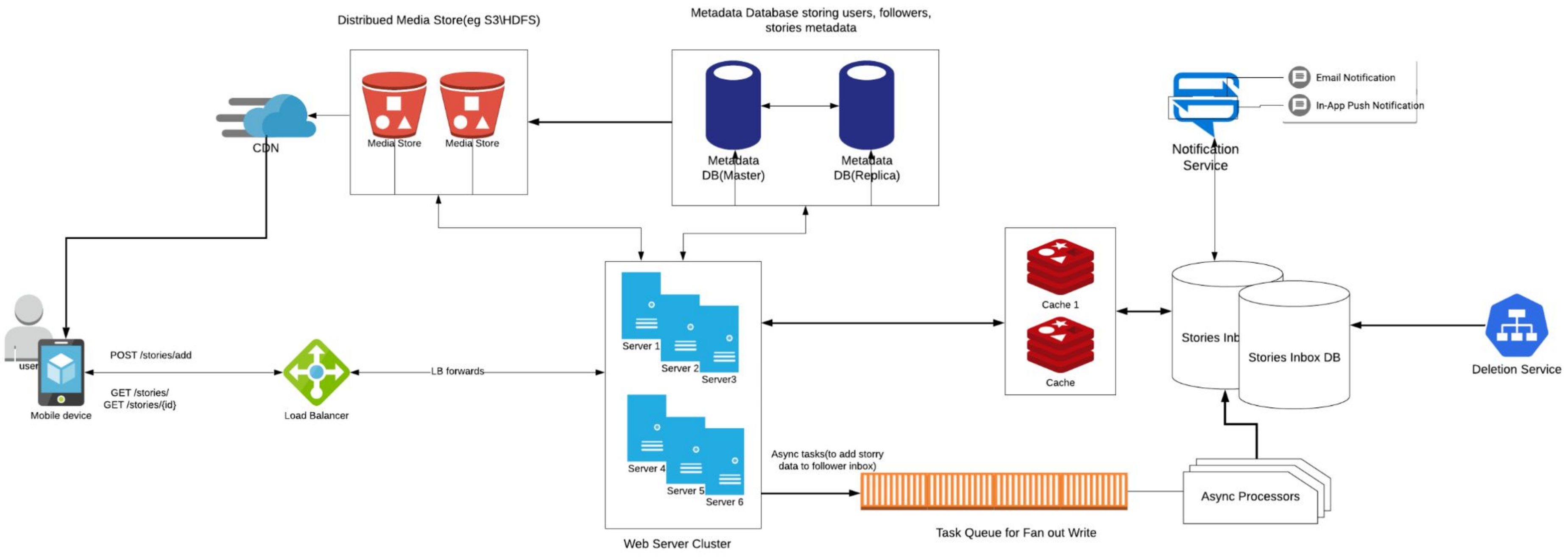
# Design


Use an API gateway
Single entry point
View
Controller
Model
Traditional server-side web application
View
Controller
Model
Browser/Native App
API Gateway
REST
Product Info service
REST
Recommendation Service
AMQP
Review service
Client specific APIs
Protocol translation
@crichardson

# Design


MetaData DB
Image DB
Post Service
UserA uploads photos
API GATEWAY
UserA uploads a photo
UserA
Follow
Notify NF service to update newsfeed to UserA's followers
get Followers of UserA
Load Balancer
NewsFeed Service
Feed DB
precompute newsfeed for folllowers of UserA
notify followers of UserA
UserB
Notification Server
Session DB
Notification server pushes the updates to the online user
get online user info

# Design


Distribued Media Store(eg S3\HDFS)
Metadata Database storing users, followers, stories metadata
Media Store
Media Store
CDN
Metadata DB(Master)
Metadata DB(Replica)
Email Notification
In-App Push Notification
Notification Service
Cache 1
Cache
Stories Inbox
Stories Inbox DB
Deletion Service
user
Mobile device
POST /stories/add
GET /stories/
GET /stories/{id}
Load Balancer
LB forwards
Server 1
Server 2
Server3
Server 4
Server 5
Server 6
Web Server Cluster
Async tasks(to add storry data to follower inbox)
Task Queue for Fan out Write
Async Processors

# Backend Study Group

**References:**

- System Design Primer
- Design Instagram

**Backend Study Group**:

- Presentations on GitHub and session recordings available on WWCode YouTube channel
- March 23rd, 2023: Introduction to GPT3
- April 6th, 2023: SQL Queries 101

**Women Who Code:**

- Technical Tracks and Digital Events for more events
- Join the Digital mailing list for updates about WWCode
- Contacts us at: contact@womenwhocode.com
- Join our Slack workspace and join *#backend-study-group*!

*You can unmute and talk or use the chat*

WOMEN WHO CODE /san-francisco