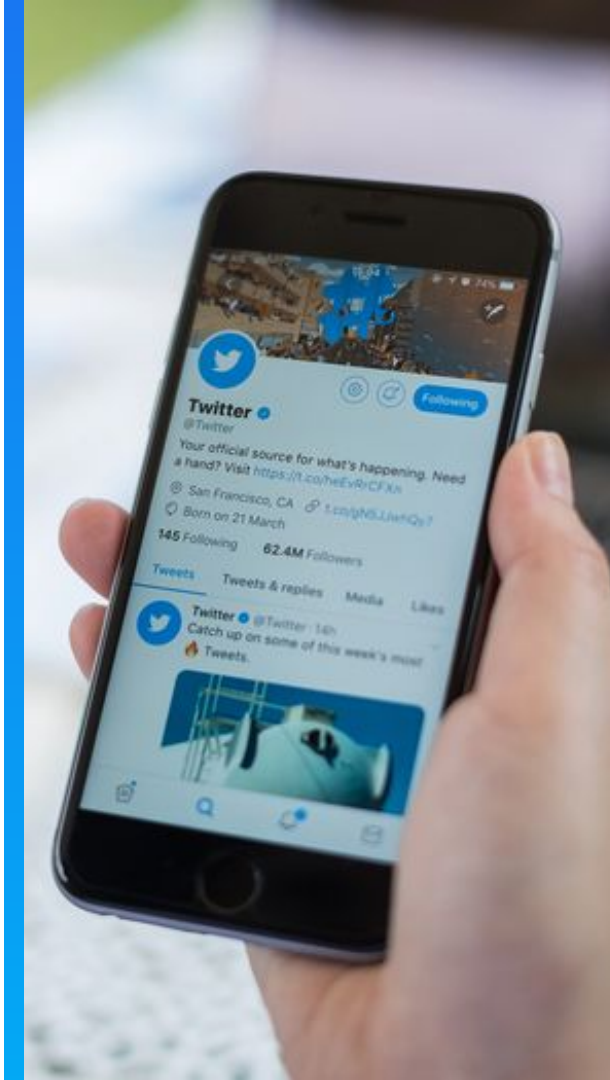
Software Architecture of Twitter

Omer Kake (218636878)
David Luu (216157463)
John Donato Prabahar (219087279)
Arash Saffari (218791632)
Ali Sina (218318428)
Siyan Sriganeshan (218707190)

Introduction & Functionality

Functionality

- Twitter is an online social networking service that enables users to send and read short 280-character messages called "tweets“
- Users can create tweets, like them, retweet others' posts, and follow accounts to stay updated on their content. They have options to mute or block accounts they prefer not to engage with
- People can follow other accounts they find interesting to view their tweets
- Users can group tweets by topic using hashtags (words prefixed with "#") and mention others by using the "@" symbol followed by their username.
- Users can see the number of likes, retweets, and replies on their tweets, which helps gauge engagement.
- Twitter offers a powerful search engine for finding tweets, accounts, and hashtags, with advanced filters to refine results.
- Many businesses leverage Twitter for brand awareness, customer engagement, and public relations, often using promoted tweets to reach a wider audience




Interacting
Parts

Core Components

The core components that define Twitter can be sorted into:

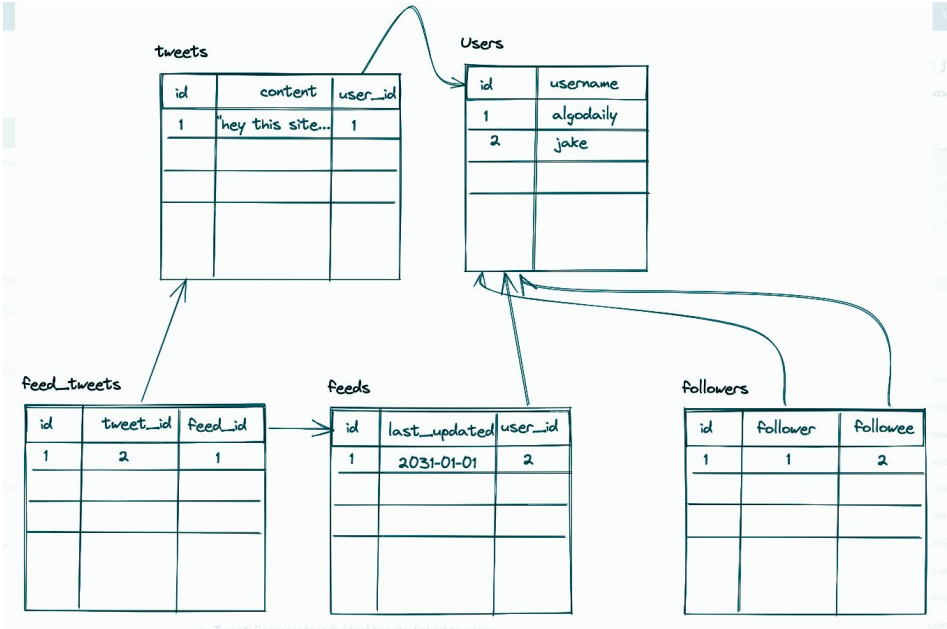
1. Tweeting
Utilizes distributed message queues and databases to handle thousands of tweets per second, ensuring real-time delivery and consistency.

2. Following
Employs graph databases and caching layers to efficiently manage relationships and provide quick access to relevant content.

3. User Timeline
Uses indexing and caching strategies to deliver a personalized, responsive experience.

4. Home Timeline
Applies distributed systems and real-time data processing to curate and deliver a dynamic stream of content.

5. Internal Search Engine
Leverages inverted indices, search algorithms, and machine learning to deliver accurate and relevant search results.



Interactions

User Timeline

1. Timeline Request: User requests to view their timeline.
2. Cache Retrieval: Server checks cache for recent tweets.
3. Database Query: If necessary, the server queries the Tweets table for additional tweets.
4. Response: Chronologically arranged tweets sent back to the user.

Following

1. Follow Request: User initiates a follow request.
2. Relationship Establishment: Server updates the Followers table.
3. Timeline Update: Follower's timeline includes followee's tweets.
4. Notification: Optional notification sent to the followee.

Tweeting

1. Tweet Creation: User composes a tweet; request sent to the server.
2. Data Processing: Server validates content and processes hashtags/mentions.
3. Database Interaction: Tweet stored in the database, linked to the user.
4. Cache Update: Tweet cached for faster retrieval.
5. Fan-Out: Tweet propagated to followers' timelines.
6. Acknowledgment: User receives confirmation of successful posting.

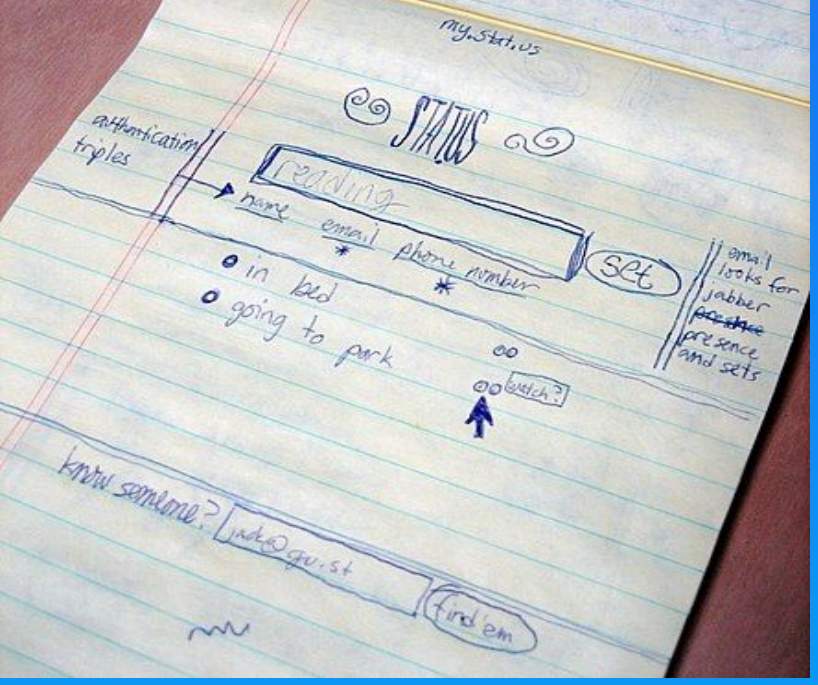
Home Timeline

1. Home Timeline Request: User requests their home timeline.
2. Cache and Database Interaction: Tweets retrieved from both sources.
3. Aggregation and Sorting: Tweets sorted chronologically from followed users.
4. Response: Sorted tweets sent back for display.

Internal Search Engine

1. Search Query: User enters a search term, including hashtags/keywords.
2. Distributed Search: Query sent to multiple data centers.
3. Index Lookup: Reverse indexing finds matching tweets.
4. Ranking and Sorting: Results ranked by popularity and relevance.
5. Response: Final sorted results returned to the user.

Twitter Evolution



Jack Dorsey
@jack

Follow

just setting up my twttr

Reply Retweet Favorite More

18,569 RETWEETS

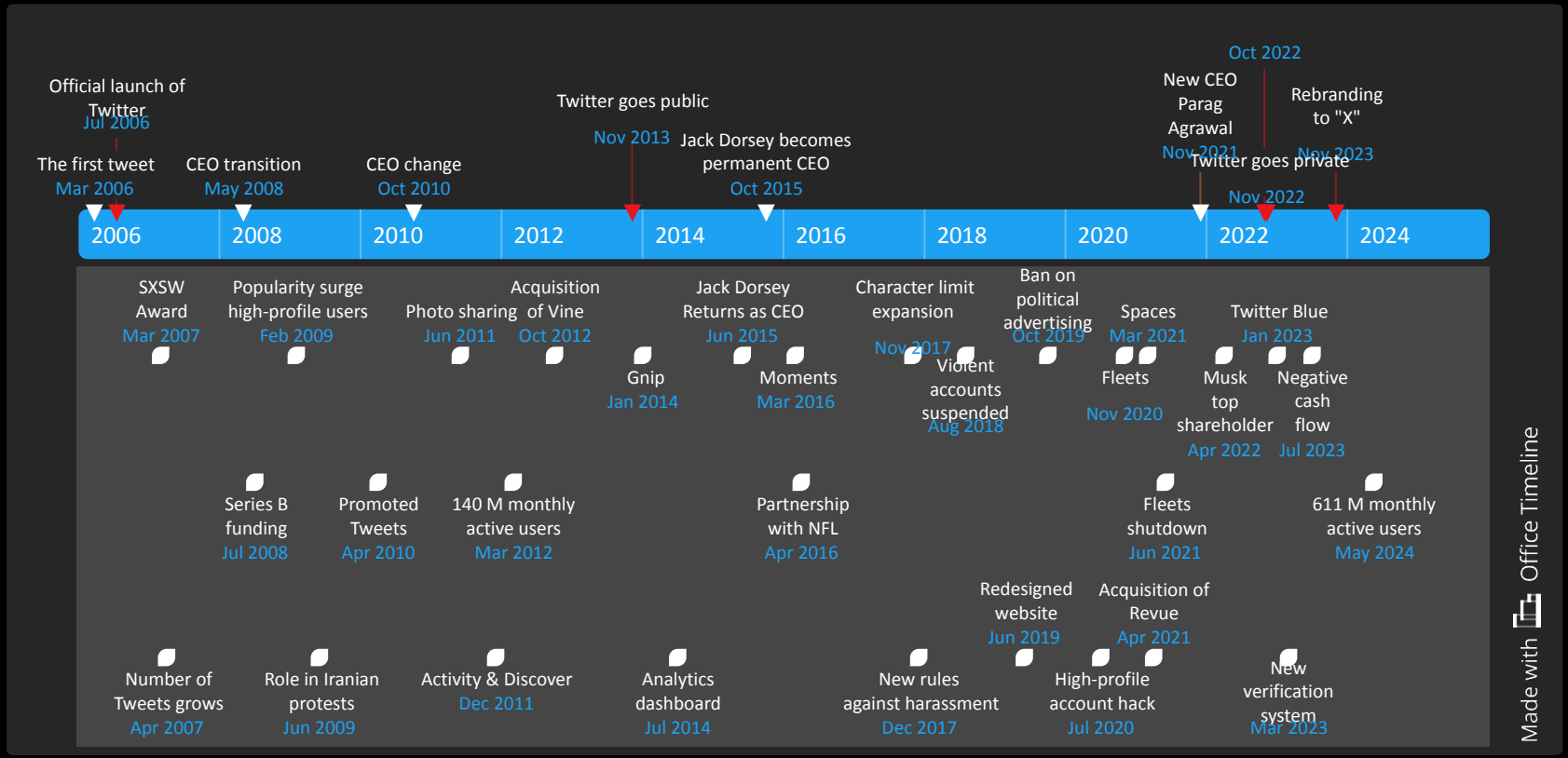
16,467 FAVORITES

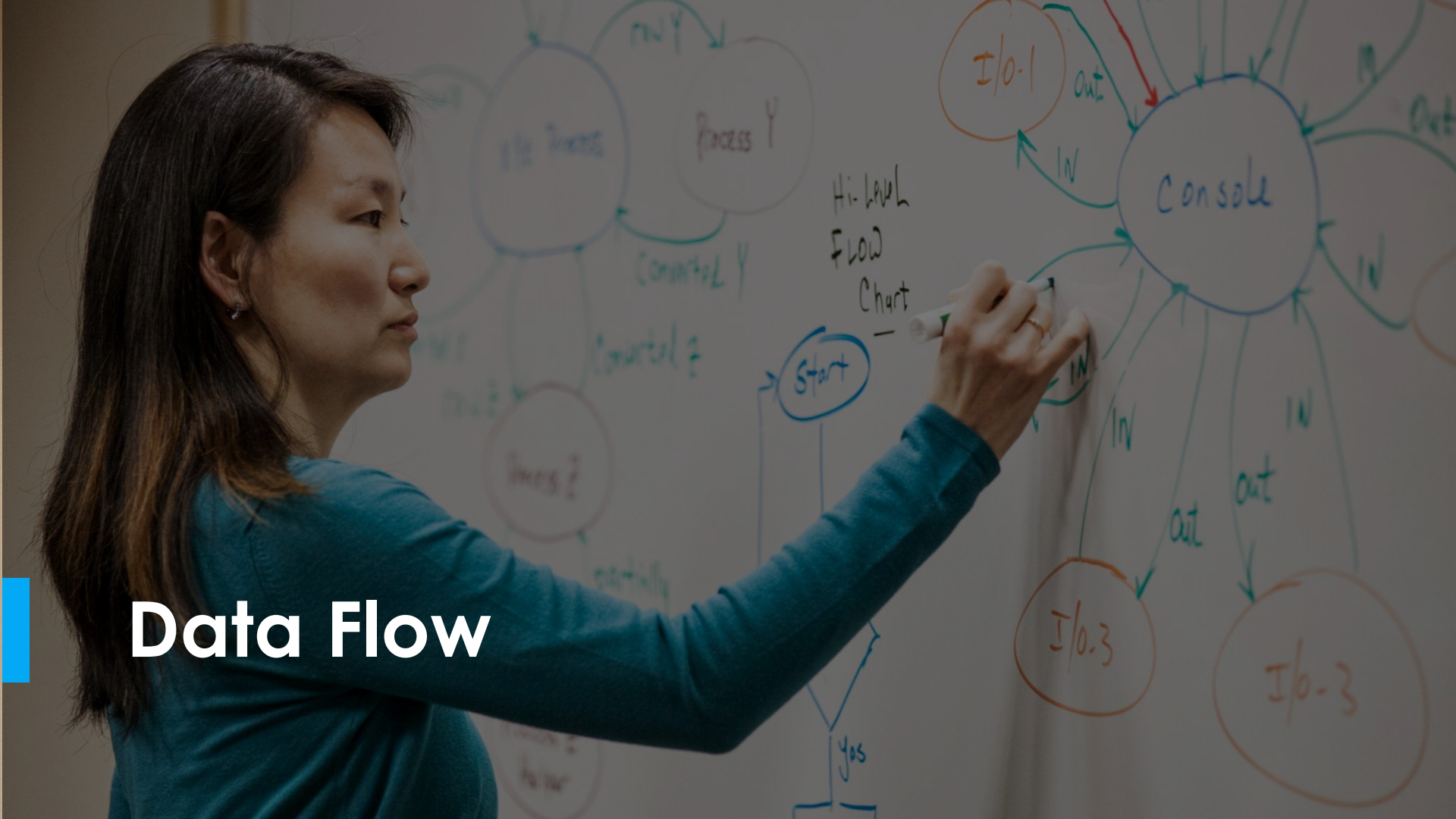
3:50 PM - 21 Mar 06

The Beginning of Twitter

- Twitter was founded on July 15, 2006 in San Francisco by Jack Dorsey as a microblogging platform.
- Twitter became famous for its 140 characters limit so the tweets feet into one screen.
- First tweet on the would was on March 21st by Co Founder Jack Dorsey approximately four months before X's official launch on July 15, 2006
- Twitter was originally named "twttr" as a nod to the popular photo-sharing platform Flickr and the five-character length of American SMS short codes

Twitter Evolution




I/O-1
out
IN
Console
Hi-Level
FLOW
Chart
Start
IN
IN
out
out
IN
I/O-3
I/O-3
Yes

Data Flow

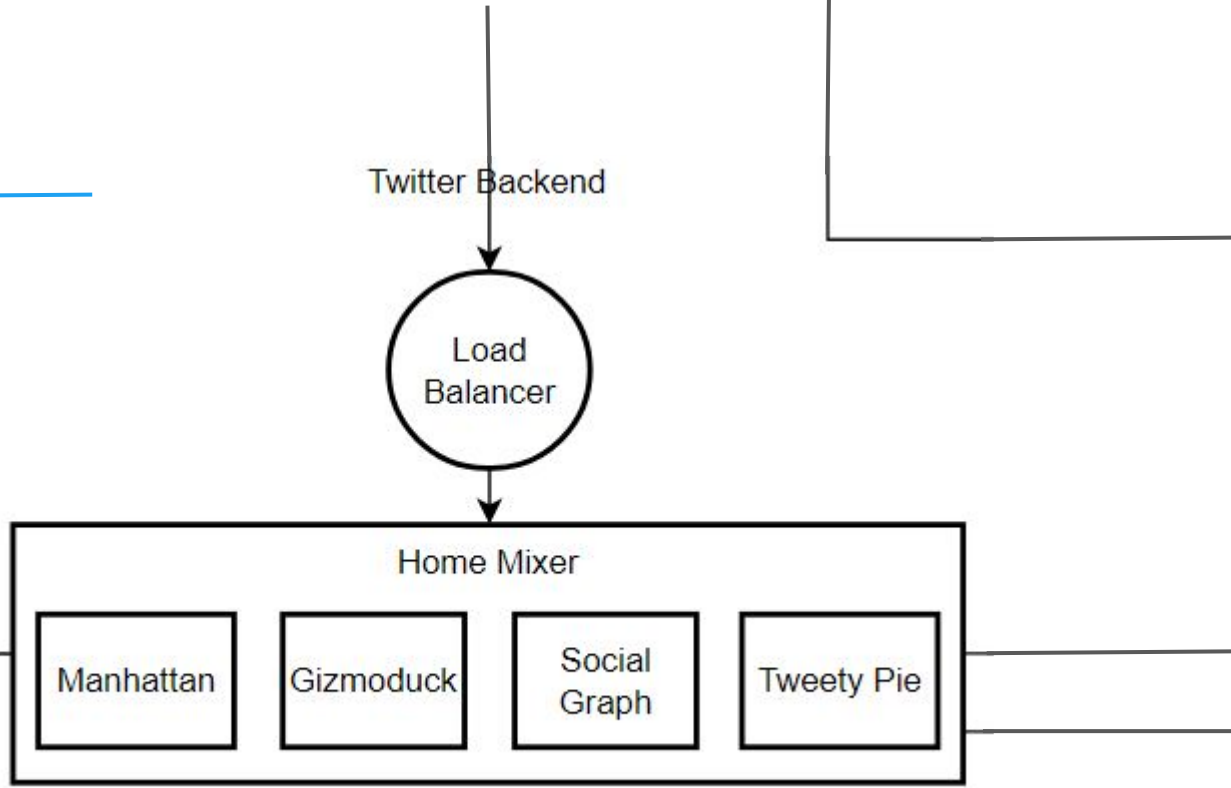
Main Diagram

Tweety Pie
- wrapping tweets with metadata

Gizmoduck
- User profile & account data

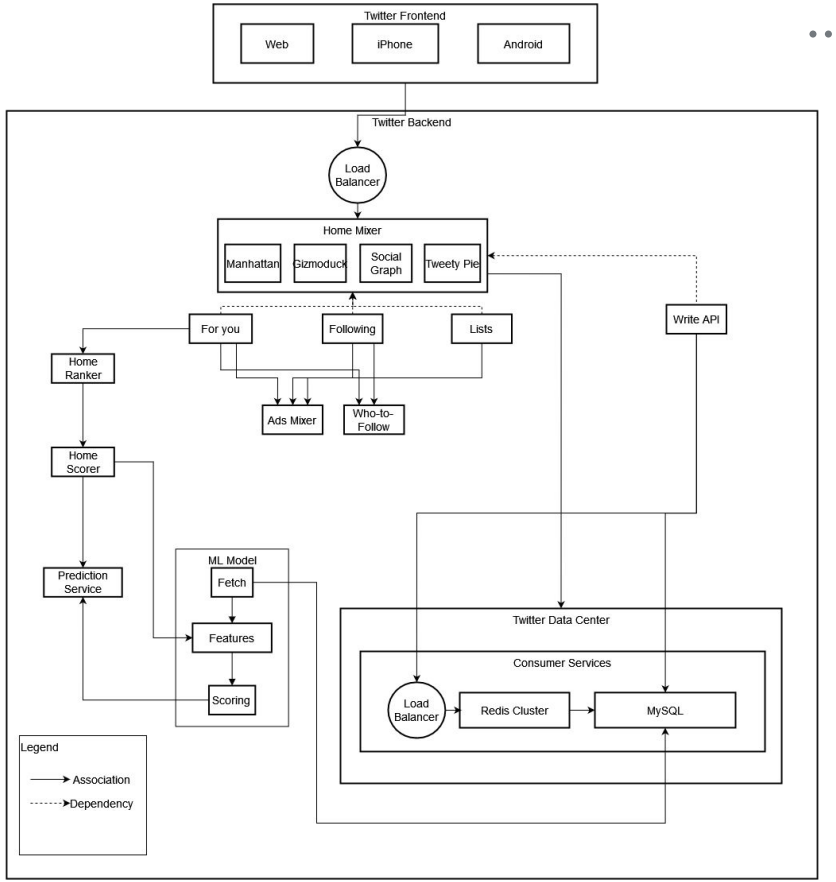
Social Graph
- Graphs user interactions & followings

Manhattan
- Custom database

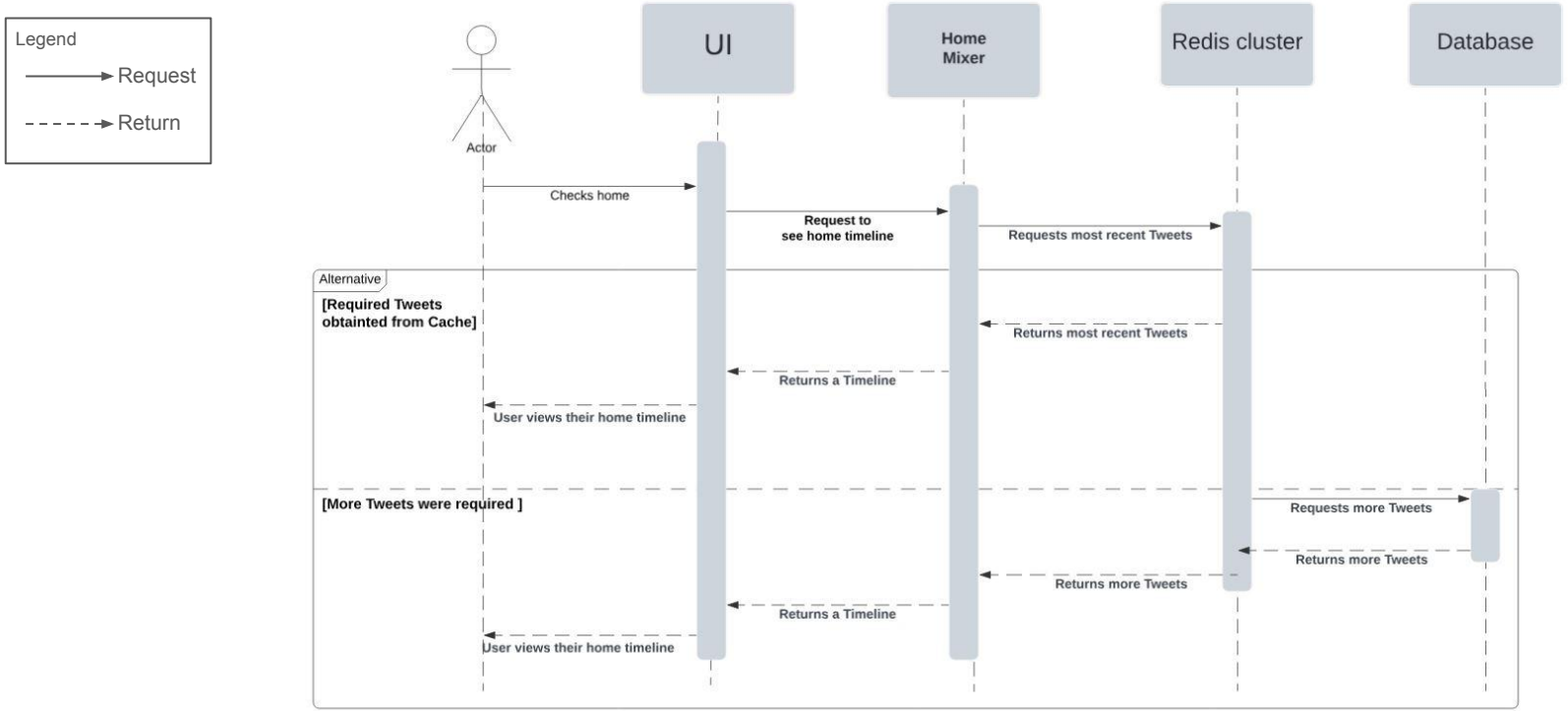


Concurrency Within

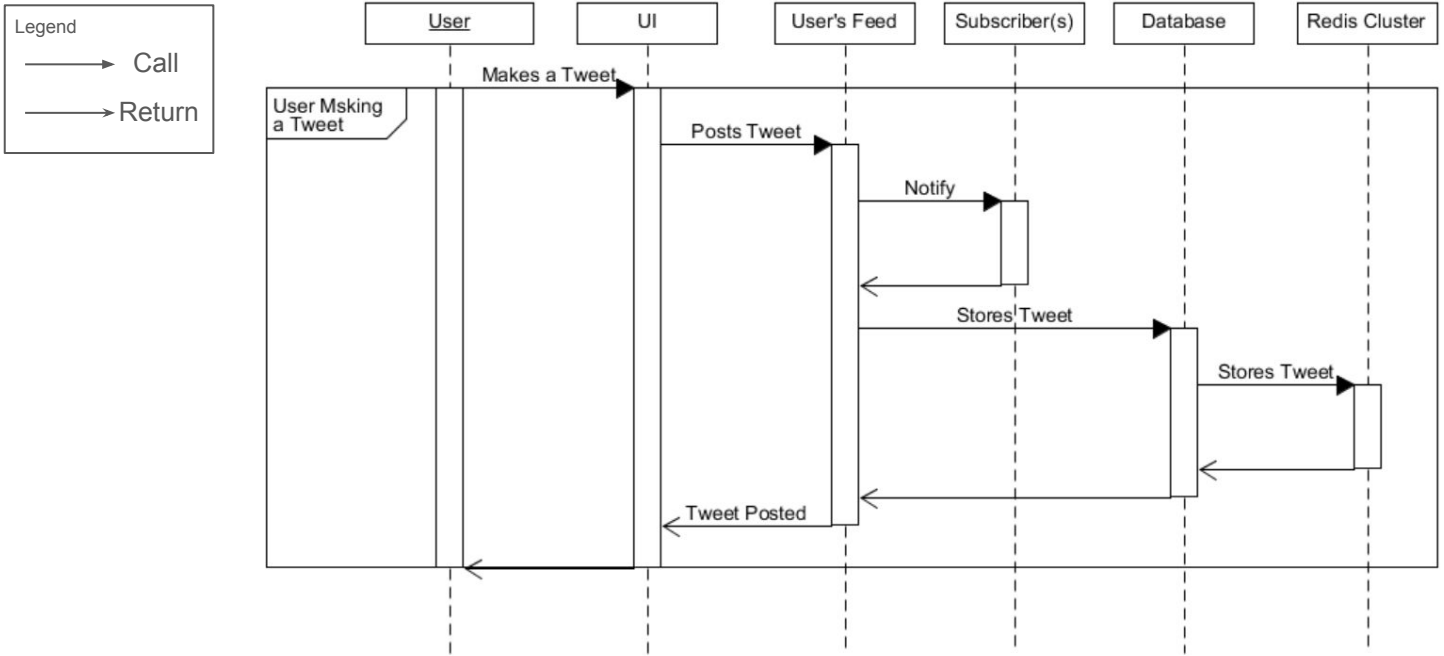
- Redis Clusters
- Home Mixer
  - Pipelines
- Tweeting



Timeline Sequence Diagram



Tweet Sequence Diagram



Implications and Division of Responsibilities

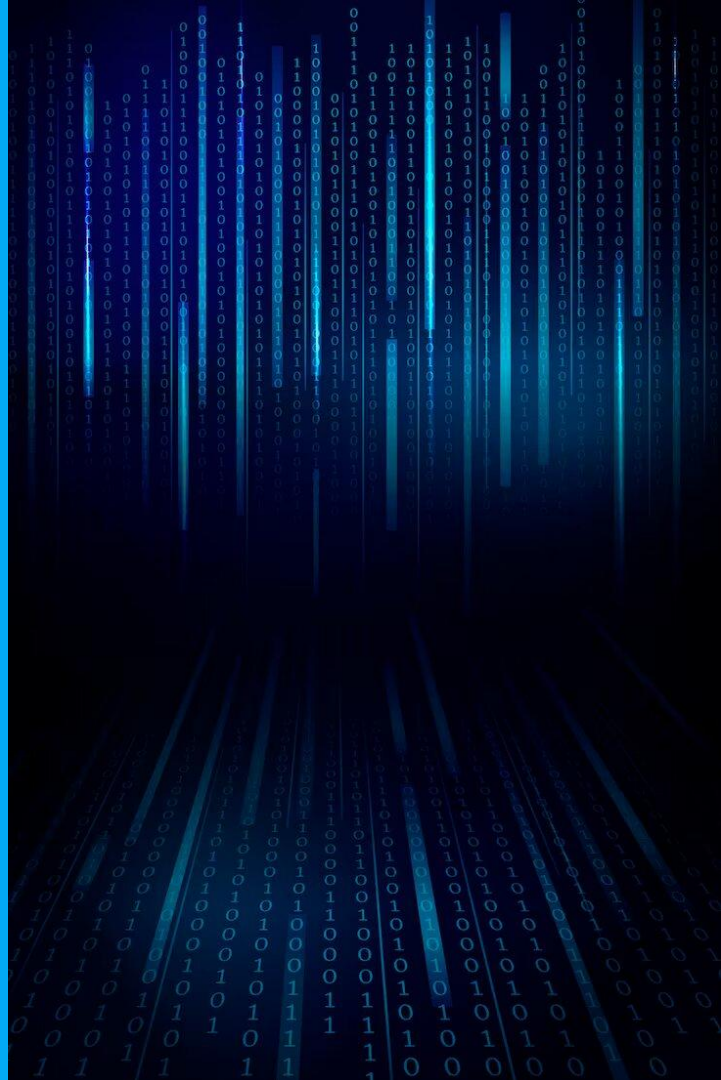
Implications and Division of Responsibilities

Frontend Developers' Responsibilities

- Develop platform-specific user interfaces (Web, iPhone, Android).
- Ensure consistent user experience across platforms.
- Collaborate with backend teams for API integration and real-time data.
- Implications: Requires constant collaboration with backend teams for data synchronization.

Backend Engineers' Responsibilities

- Manage core services like Load Balancer, Home Mixer, Home Ranker, and Write API.
- Handle real-time data fetching, scoring, and serving for personalized feeds.
- Ensure system scalability and availability.
- Implications: Any disruption affects user experience across all platforms.



Cont.

Machine Learning Engineers' Responsibilities

- Build and maintain ML models for ranking and prediction.
- Collaborate with data engineers for real-time data and feature extraction.
- Continuously refine models to improve personalization and content recommendations.
- Implications: Requires collaboration with backend and data teams for smooth deployment.

Data Engineers' Responsibilities

- Manage real-time data pipelines (Kafka, Event Processing).
- Ensure integration with Google Cloud services for storage and analytics.
- Provide clean and real-time data for backend and ML services.
- Implications: Data engineers provide the foundation for real-time systems and predictions.

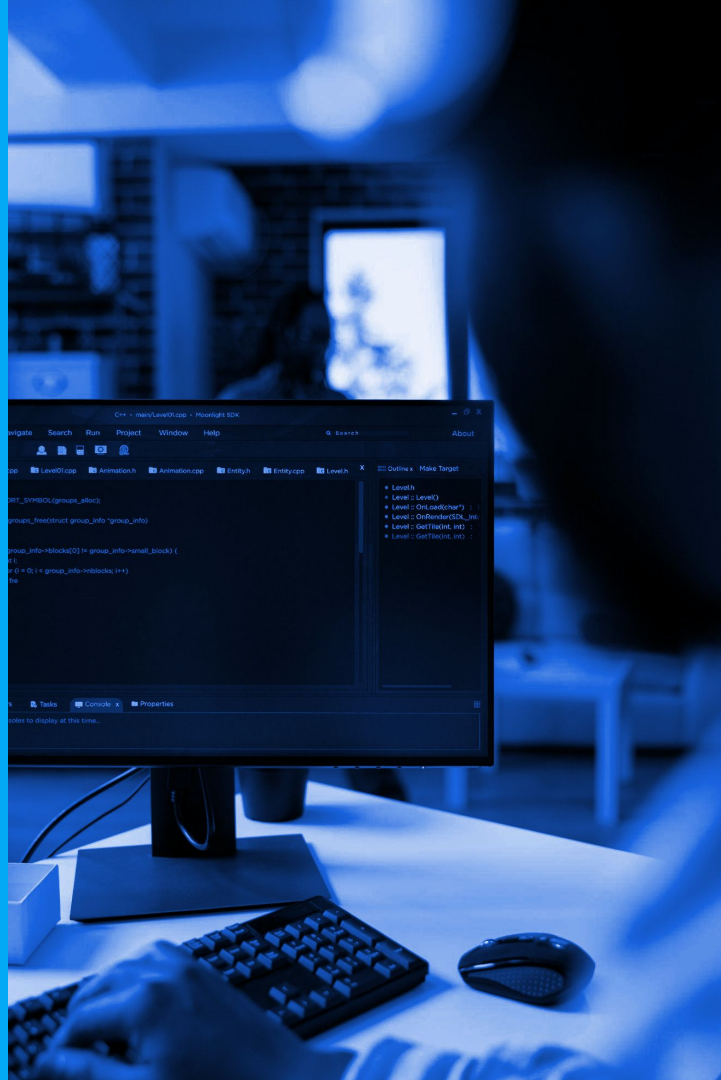
Cont.

DevOps Engineers' Responsibilities

- Manage system infrastructure, scaling, and monitoring.
- Handle CI/CD pipelines for smooth and continuous deployment.
- Ensure uptime and reliability of services.
- Implications: DevOps must ensure infrastructure supports the needs of all teams.

Security and Compliance Teams' Responsibilities

- Ensure data privacy and security across all systems (e.g., Cloud Bigtable).
- Ensure compliance with legal regulations (e.g., GDPR).
- Collaborate with all teams to ensure security protocols are followed.
- Implications: Security must be integrated across all components.



Thanks