CICSOFT

WELCOME TO

systemSprint

STARTS AT 7:05PM

Mentimeter uses cookies

We do this to ensure that our website works well and provide you with relevant content and marketing campaigns. We also want to improve our features and research new ones.

Below you can accept all cookies or click 'Cookie Preferences' to choose which ones you want.

If you want to know precisely why we use cookies and other similar technologies or how you can withdraw consent, simply visit our Cookie Policy.

Accept All

Reject All

blue	white	yellow	blue	red
green	yellow	blue	green	black
yellow	red	blue	green	yellow
red	yellow	red	white	green
blue	green	red	yellow	black
blue	black	blue	black	white
black	yellow	white	blue	green
red	black	red	yellow	green
red	yellow	blue	white	blue
red	yellow	blue	yellow	green

LETS TALK SYSTEM

WTF IS SYSTEM DESIGN

CICSOFT

~~WTF~~

WHY IS SYSTEM DESIGN!!!

System design is the process of defining the architecture, interfaces, and data for a system that satisfies specific requirements.

CICSOFT

THANK YOU

JOIN US and bring your friends :)

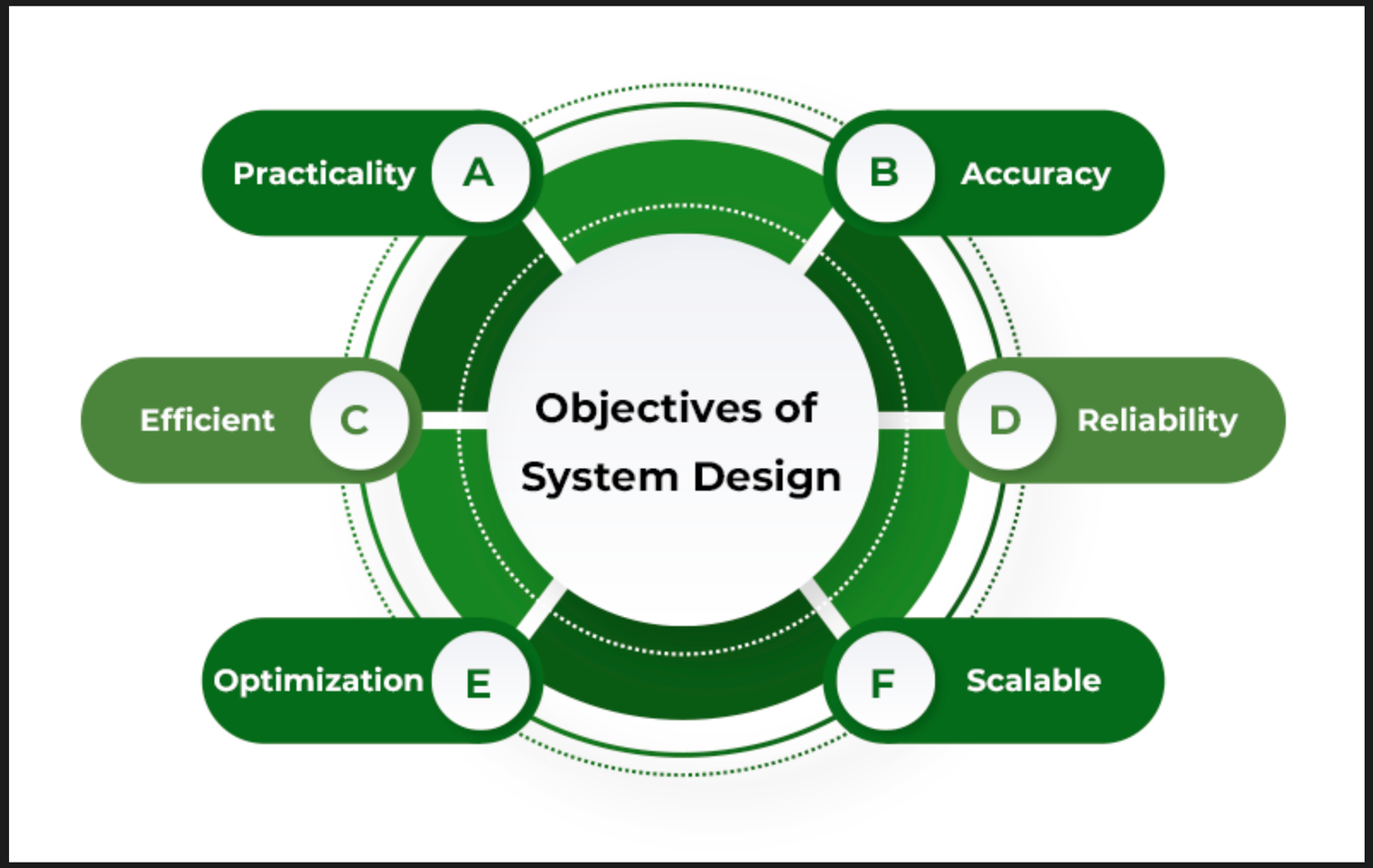
EVERY **MONDAY 7:00 PM** AT **ILC S231**

Feedback

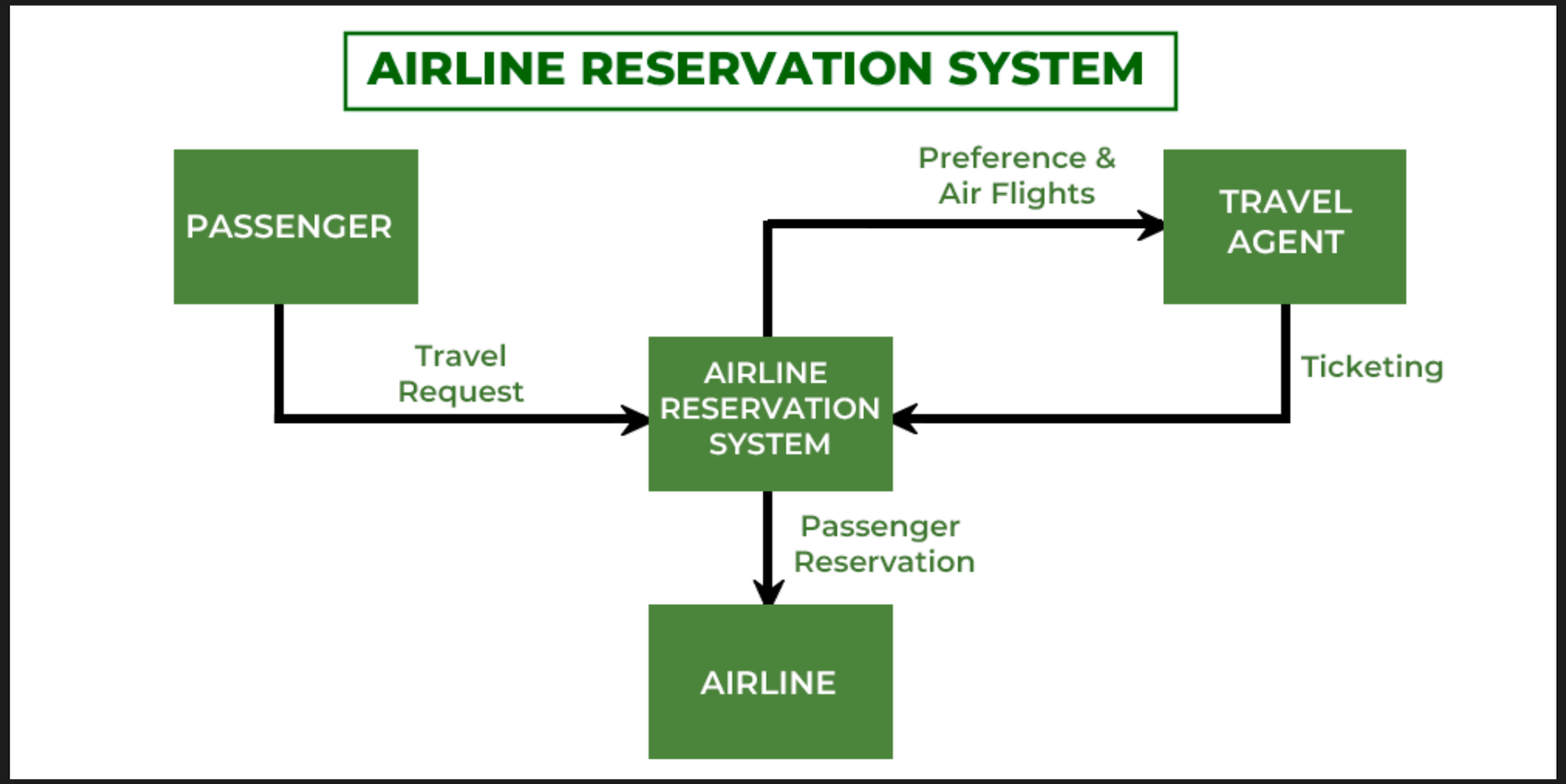
Resource GitHub

JK LOL!!!

WHY DO WE NEED SYSTEM DESIGN



EXAMPLE

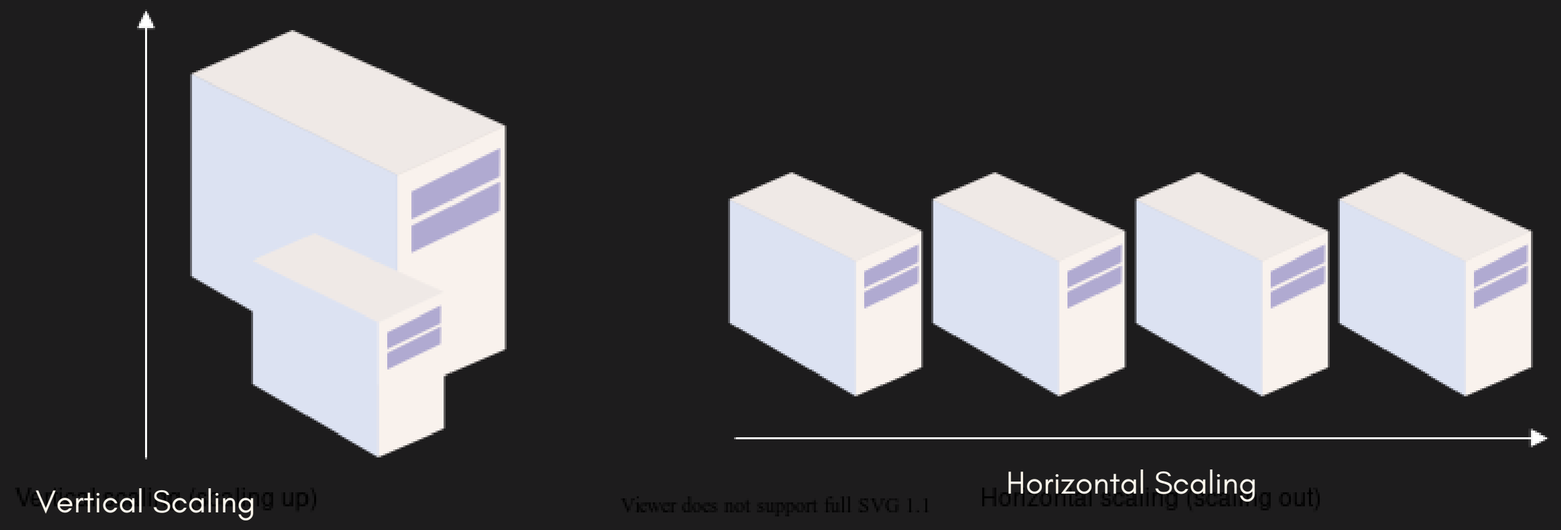

AIRLINE RESERVATION SYSTEM
PASSENGER
Preference &
Air Flights
TRAVEL
AGENT
Travel
Request
AIRLINE
RESERVATION
SYSTEM
Ticketing
Passenger
Reservation
AIRLINE

LETS DIVE DEEP

UNDERSTANDING SCALABILITY

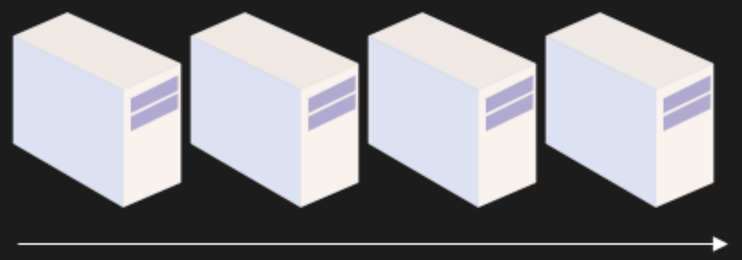
SCALABILITY

- Essential for applications to maintain performance under increased loads.
- Involves enhancing computing power to support more users, data, and transactions.



HORIZONTAL SCALING

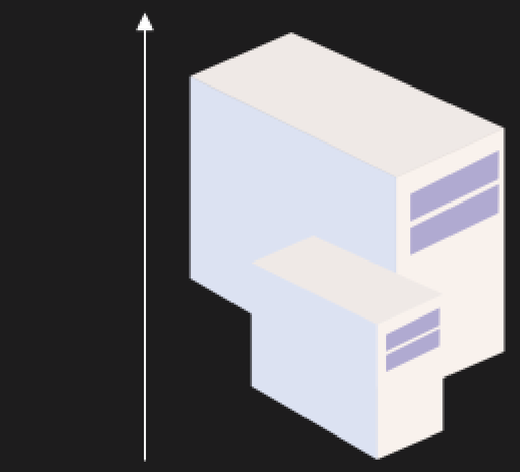
- Adds more machines or instances to the pool to distribute the load.
- Facilitates redundancy and failover, enhancing system reliability.
- More flexible; can scale according to demand fluctuations.
- Complexity in management and integration can increase.



er does not support full SVG 1.1 Horizontal scaling (scaling out)

VERTICAL SCALING

- Involves upgrading existing hardware (CPU, RAM, storage).
- Simpler to implement, with no need for modifying the application for distributed environments.
- Has physical and technological limits; not always feasible for long-term growth.
- Downtime may be required for hardware upgrades.



Vertical scaling (scaling up)

LOAD BALANCER

- Distributes large volume of incoming requests/workloads across multiple servers.
- Prevents any single server from being overloaded

Types of Load Balancers:

- Layer 4: Network layer, based on IP/port.
- Layer 7: Application layer, based on request content.
- Global: Across geographic locations.
- Application-Specific: For protocols like HTTP/HTTPS.

DATABASES

SQL

- Structured with predefined schemas like phone books.
- Data stored in rows and columns; each row is a single entity, columns are data points.
- Popular SQL databases: MySQL, Oracle, MS SQL Server, SQLite, PostgreSQL, MariaDB.
- SQL is used for data manipulation and supports complex queries with joins.

NoSQL

- Unstructured with dynamic schemas, accommodating diverse data types.
- Types include Key-value stores (Redis, DynamoDB), Document databases (MongoDB, CouchDB), Wide-column databases (Cassandra, HBase), Graph databases (Neo4J, InfiniteGraph).
- Flexible data models, suitable for unstructured data and rapid development.
- Emphasizes scalability, distributed computing, and ease of replication.

CHOOSE A DB

- **Data Structure:** SQL for structured, NoSQL for unstructured data.
- Scalability: SQL for vertical, NoSQL for horizontal scaling.
- **Query Complexity:** SQL excels in complex relationships; NoSQL for simplicity.
- **ACID vs BASE:** SQL follows ACID for integrity; NoSQL opts for BASE for performance.
- **Development:** NoSQL for speed and flexibility; SQL for stable schema environments.

Which Database should I use?

#GCPSketchnotes @PVERGADIA THECLOUDGIRL.DEV

07.10.2021

RELATIONAL

Cloud SQL

Managed MySQL, PostgreSQL, SQL Server

Cloud Spanner

Cloud-native with large scale, consistency, 99.999% availability

Bare Metal

Lift and shift Oracle workloads to Google Cloud

Good For:

- Cloud SQL: General purpose SQL DB
- Cloud Spanner: RDBMS+ scale, HA, HTAP
- Bare Metal: RDBMS+ scale, HA, HTAP

Use Case:

- Cloud SQL: Web frameworks, ERP, CRM, Ecommerce and web, SaaS application
- Cloud Spanner: Gaming, Global financial ledger, Supply chain/ inventory management
- Bare Metal: Legacy applications, Data center retirement

NON-RELATIONAL (NO SQL)

DOCUMENT | KEY VALUE

Firestore

Cloud Native, serverless, NoSQL document database, backend-as-a-service, global strong consistency, 99.999% SLA

Cloud Bigtable

Cloud-native NoSQL wide-column store for large scale, low-latency workloads

Good For:

- Firestore: Large scale, complex hierarchical data
- Cloud Bigtable: Heavy read + write, events

Use Case:

- Firestore: Mobile/web/ IoT applications, Real-time sync, Offline sync, Personalized apps
- Cloud Bigtable: Personalization, Adtech, Recommendation engines, Fraud detection

IN MEMORY

Memory Store

Fully managed Redis and Memcached for sub-millisecond data access

Good For:

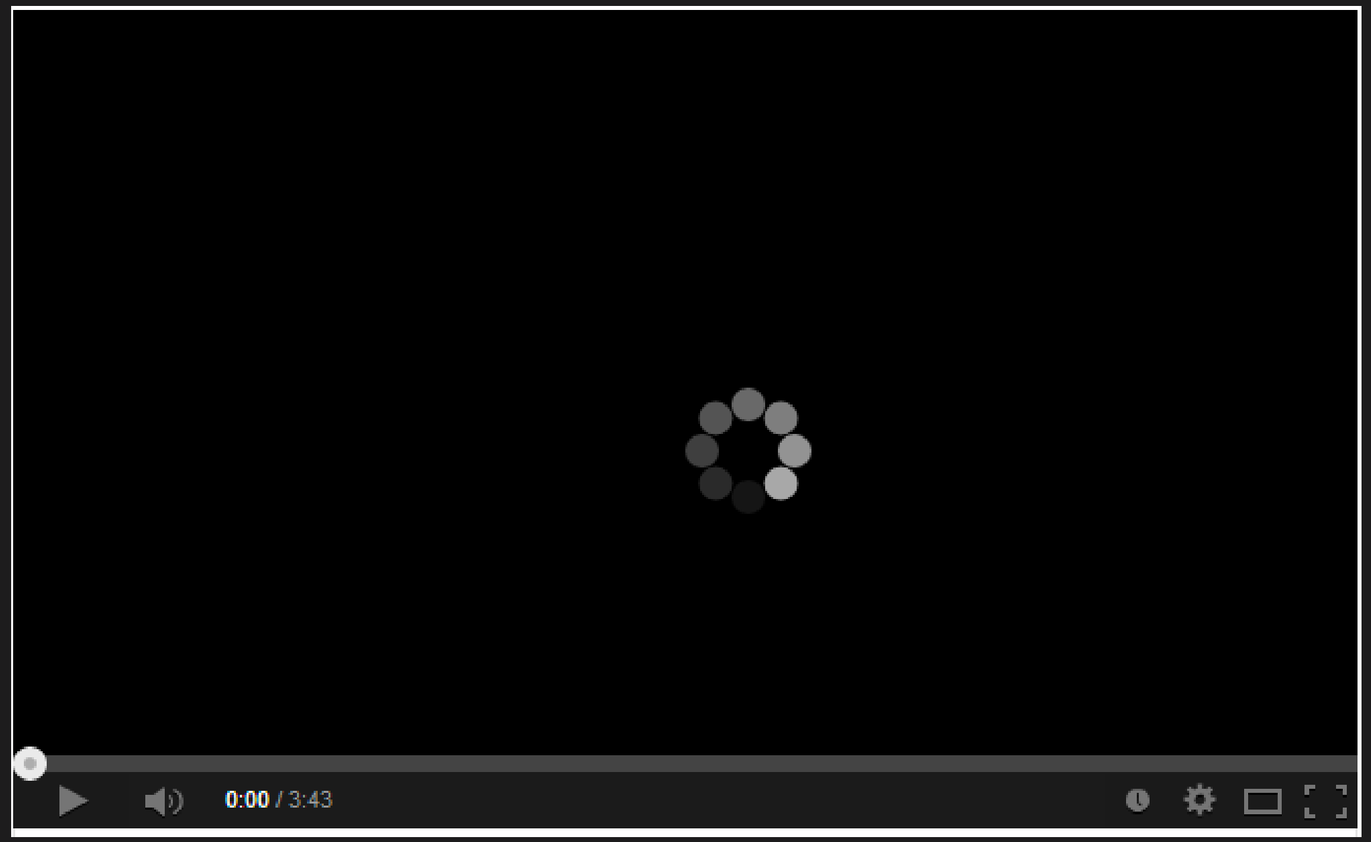
In-memory and Key-value store

Use Case:

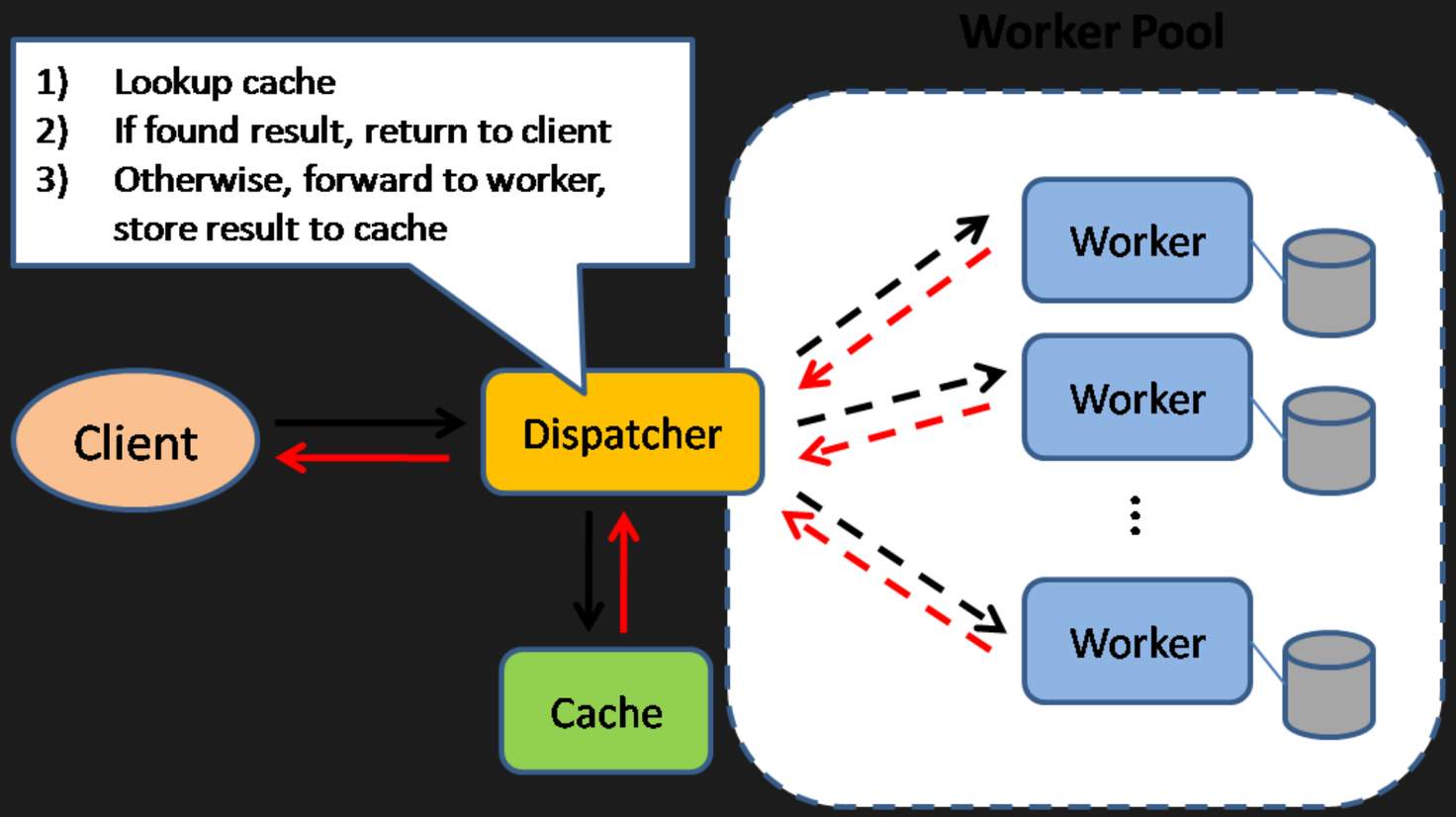
Caching, Session store, Gaming, Personalization, Leaderboard, Adtech, Social chat or news feed

WTF IS CACHE?

PERFORMANCE



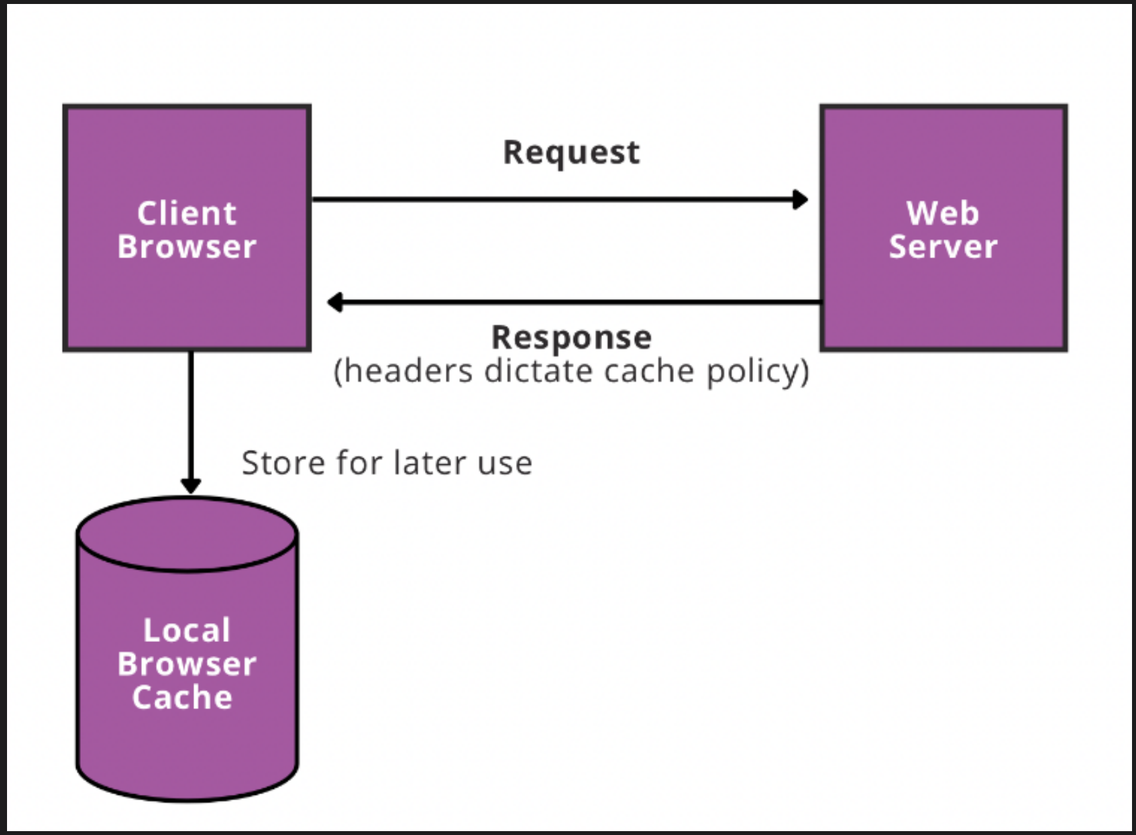
HOW DOES IT WORK???



CLIENT SIDE CACHING

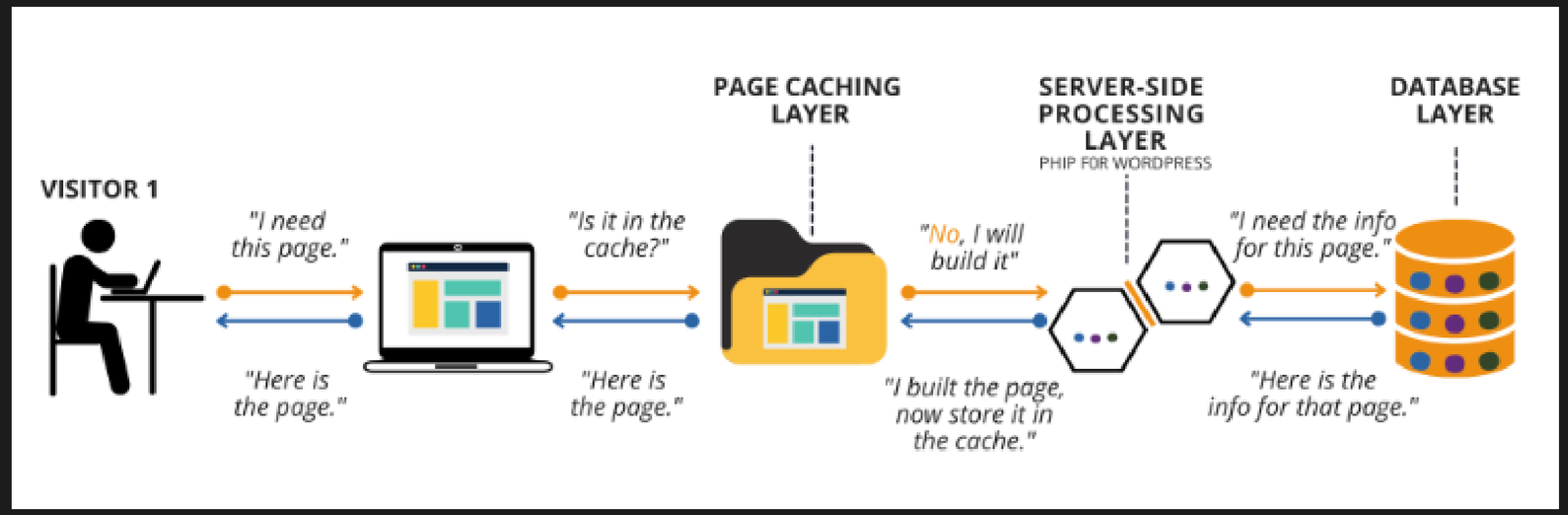
- Client-side caching occurs in the user's browser, where resources like images, scripts, and stylesheets are stored after the first visit to a website.
- On subsequent visits, the browser can load these resources from its cache rather than downloading them again from the server, which saves time and bandwidth.

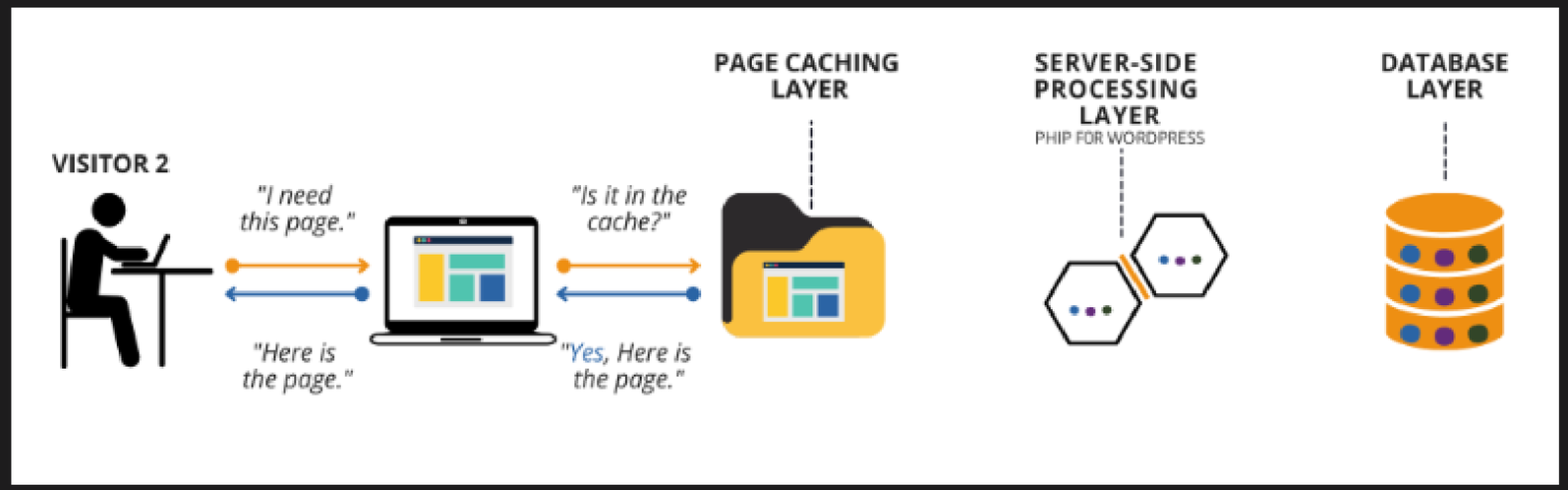
CLIENT SIDE CACHING



SERVER SIDE CACHING

- Server-side caching involves storing pre-generated responses to user requests on the server.
- When a user requests a webpage, the server can quickly deliver the stored response without having to generate it from scratch.


PAGE CACHING LAYER
SERVER-SIDE PROCESSING LAYER
PHIP FOR WORDPRESS
DATABASE LAYER
VISITOR 1
"I need this page."
"Is it in the cache?"
"No, I will build it"
"I need the info for this page."
"Here is the page."
"Here is the page."
"I built the page, now store it in the cache."
"Here is the info for that page."


PAGE CACHING LAYER
SERVER-SIDE PROCESSING LAYER
PHIP FOR WORDPRESS
DATABASE LAYER
VISITOR 2
"I need this page."
"Is it in the cache?"
"Here is the page."
"Yes, Here is the page."

API??

WHAT EVEN IS AN API?

- APPLICATION PROGRAMMING INTERFACE

- LET'S BREAK IT DOWN:
  - "INTERFACE": A MIDDLEMAN, A BROKER
  - UI - USER INTERFACE --> INTERFACE BETWEEN THE USER AND THE WEBSITE/APP
- AN API IS SIMPLY THE MIDDLEMAN BETWEEN TWO RUNNING APPLICATIONS/SERVICES.

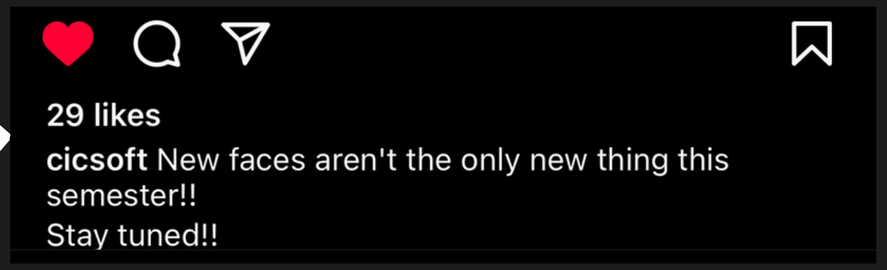
REAL-LIFE EXAMPLE OF “API”

LET'S LOOK AT THE ACTUAL USAGE OF API

- OPERATIONS WITH APPLICATIONS?
- WHAT HAPPENS WHEN A USER IS CREATED?
- WHAT ABOUT A LIKED INSTAGRAM POST? A COMMENT? YOU DELETED YOUR STORY?
- THESE OPERATIONS CAN BE DEFINED AS "CRUD"

CRUD

- CREATE
- READ
- UPDATE
- DELETE



RESTFUL API

- THINK OF IT AS A SET OF NOT-SO-STRICT RULES
- THE “NORM” FOR DESIGNING AN API.
- RESTFUL APIs ARE BASICALLY DOING “CRUD” THROUGH:
  - GET
  - POST
  - PUT
  - DELETE
- there are more, but these are the most important.
- guess what the 4 above mean?

RESTFUL API PRACTICES

- DATA: what format is this data? anyone knows the full form?

```
"relationships": {
  "author": {
    "links": {
      "self": "http://example.com/articles/1/relationships/author",
      "related": "http://example.com/articles/1/author"
    },
    "data": { "type": "people", "id": "9" }
  },
  "comments": {
    "links": {
      "self": "http://example.com/articles/1/relationships/comments",
      "related": "http://example.com/articles/1/comments"
    },
    "data": [
      { "type": "comments", "id": "5" },
      { "type": "comments", "id": "12" }
    ]
  }
```

RESTFUL API PRACTICES

- WHAT HAPPENS TO THIS DATA AFTER THE REQUEST IS COMPLETED?

```
"relationships": {
  "author": {
    "links": {
      "self": "http://example.com/articles/1/relationships/author",
      "related": "http://example.com/articles/1/author"
    },
    "data": { "type": "people", "id": "9" }
  },
  "comments": {
    "links": {
      "self": "http://example.com/articles/1/relationships/comments",
      "related": "http://example.com/articles/1/comments"
    },
    "data": [
      { "type": "comments", "id": "5" },
      { "type": "comments", "id": "12" }
    ]
  }
```

- STATELESSNESS: poof. gone.

RESTFUL API PRACTICES

- **ENDPOINT NAMING:**
  - **example.com/surveys**
  - **example.com/surveys/shivenpatel**
  - **example.com/surveys**

API RATE LIMITING

- IMAGINE YOUR LIGHT SWITCH.
- SWITCH IT ON AND OFF A THOUSAND TIMES, AND YOU MIGHT JUST BURN THE HOUSE DOWN.
- MAKE SURE YOUR API DOESN’T BURN YOUR APPLICATION.
- RATE LIMITING IS IMPORTANT TO PREVENT DOWNTIME AND KEEP YOUR APP SECURE.

API SECURITY

- AUTHENTICATION TOKEN
- API KEYS

T-Mobile API Breach (2022): T-Mobile has repo
of 37 million current postpaid and prepaid cust

LinkedIn API Breach (2021): In June 2021, a si
exposed. It resulted in the compromise of data
the platform for information on approximately

Facebook Data Breach (2019): More than 530
names, and Facebook IDs, was compromised i
Facebook applications that were publicly expos
out the attack. This allowed them to acquire ac

ACTIVITY TIME

GROUPS

- Make groups
- Introduce yourselves, talk about your favorite Large Scale Product!

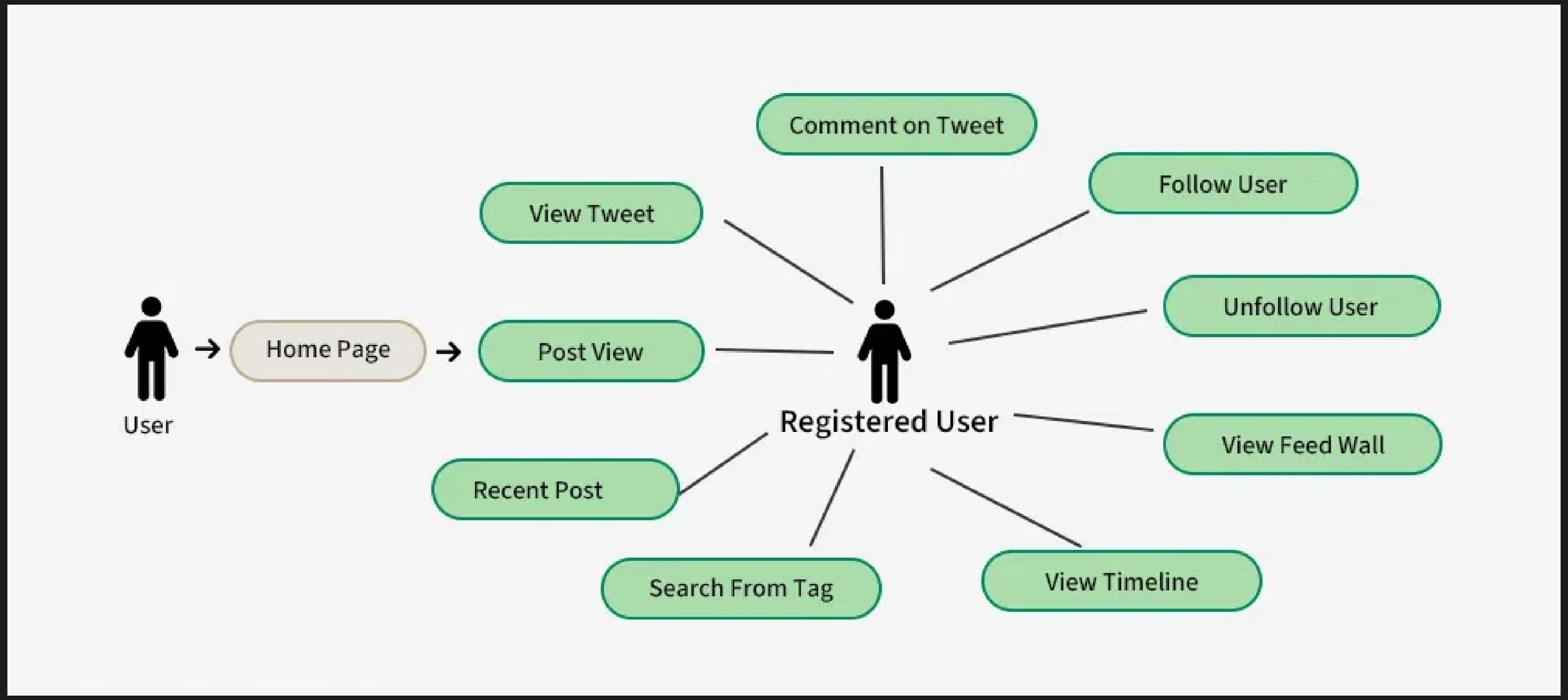
GOOD LUCK!

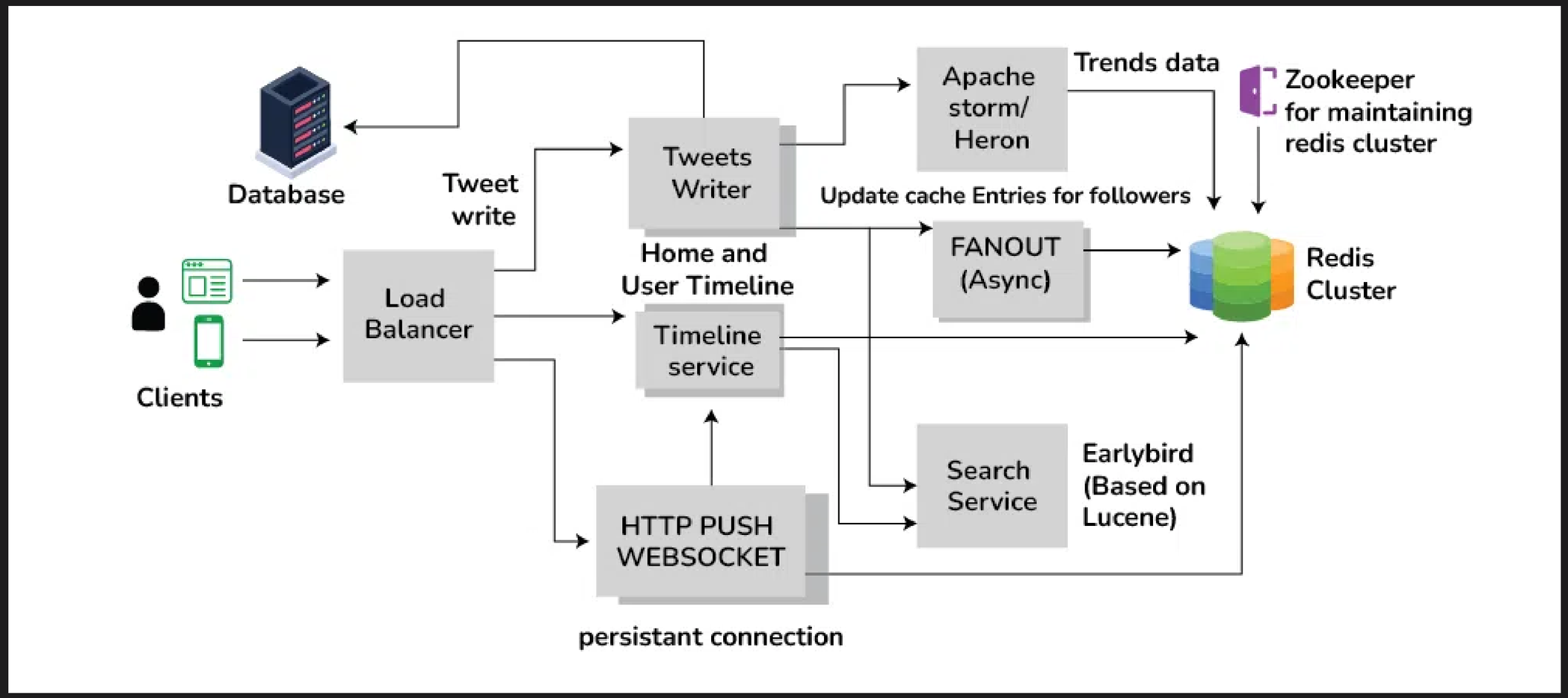
DESIGN X (FORMERLY TWITTER)

Problem: Build a system that shows users a feed of posts from other users they follow, with the ability for users to post new content.

Requirements:

- **Users should see a mixture of recent and popular content.**
- **Feed should update in real-time as new posts are made.**
- **Implement mechanisms to prevent spam and abuse.**
- **Provide personalized content based on user interests and interactions.**
- **Scale to support millions of users and posts.**


Comment on Tweet
Follow User
View Tweet
Unfollow User
User
Home Page
Post View
Registered User
View Feed Wall
Recent Post
Search From Tag
View Timeline


Database
Tweet write
Tweets Writer
Apache storm/ Heron
Trends data
Zookeeper for maintaining redis cluster
Update cache Entries for followers
FANOUT (Async)
Redis Cluster
Home and User Timeline
Load Balancer
Timeline service
Clients
Search Service
Earlybird (Based on Lucene)
HTTP PUSH WEBSOCKET
persistant connection

PREPARING FOR A SYSTEM DESIGN INTERVIEW

- Familiarize yourself with system design principles and patterns. Books such as "Designing Data-Intensive Applications" by Martin Kleppmann and "System Design Primer" by Donne Martin can be helpful for learning about these concepts.
- Practice system design questions and exercises. There are many websites and resources that offer practice system design questions, such as LeetCode and InterviewBit. You can also try working on personal projects or participating in hackathons that focus on system design.

- [ ] Review key concepts and technologies related to system design. This may include topics such as databases, networking, distributed systems, and scalability.
- [ ] Practice communicating your thought process and explaining your design choices. This is an important skill in system design interviews, as you may be asked to justify your design choices and discuss trade-offs.
- [ ] Research the company and the role you are applying for. Familiarize yourself with the company's products, services, and technology stack, as well as the specific responsibilities and requirements of the role you are applying for.
-

WHAT CAN I BE ASKED IN SUCH INTERVIEW

During a system design interview, you may be asked to:

- Define the problem and requirements for the system
- Break down the system into smaller components
- Determine the interfaces and dependencies between components
- Identify and evaluate trade-offs and constraints
- Design and justify the overall system architecture

CICSOFT

Write us Feedback through our website

CICSOFT

RESOURCE HUB

CICSOFT

THANK YOU

JOIN US and bring your friends :)

EVERY **MONDAY 7:00 PM** AT **ILC S231**

Feedback

Resource GitHub