Build Privacy Setting System

Requirements clarification

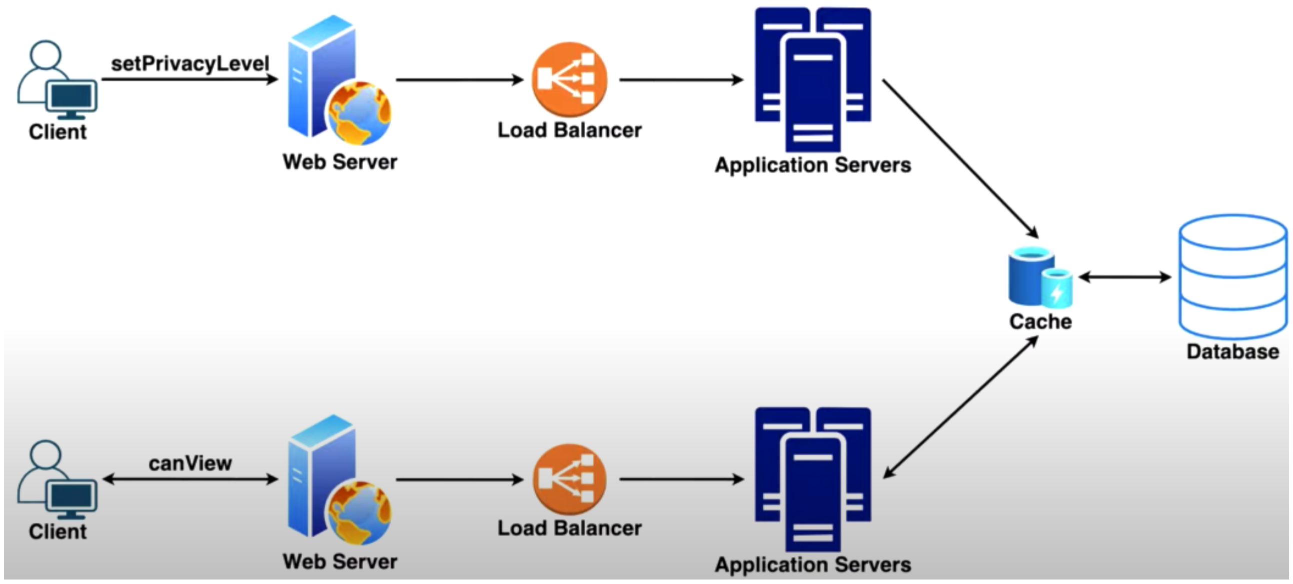
- **Functional requirements**
    - Set privacy level: Users can specify the different levels of privacy for a post so that it is only visible to a particular set of users.
    - Read privacy level: Feed generation service will make lots of requests to check privacy level of users for new posts.
- **Non-functional requirements**
    - Low latency.
    - High consistency (Same privacy setting should be applied to all the devices).
    - High availability is desirable (base on CAP theorem).

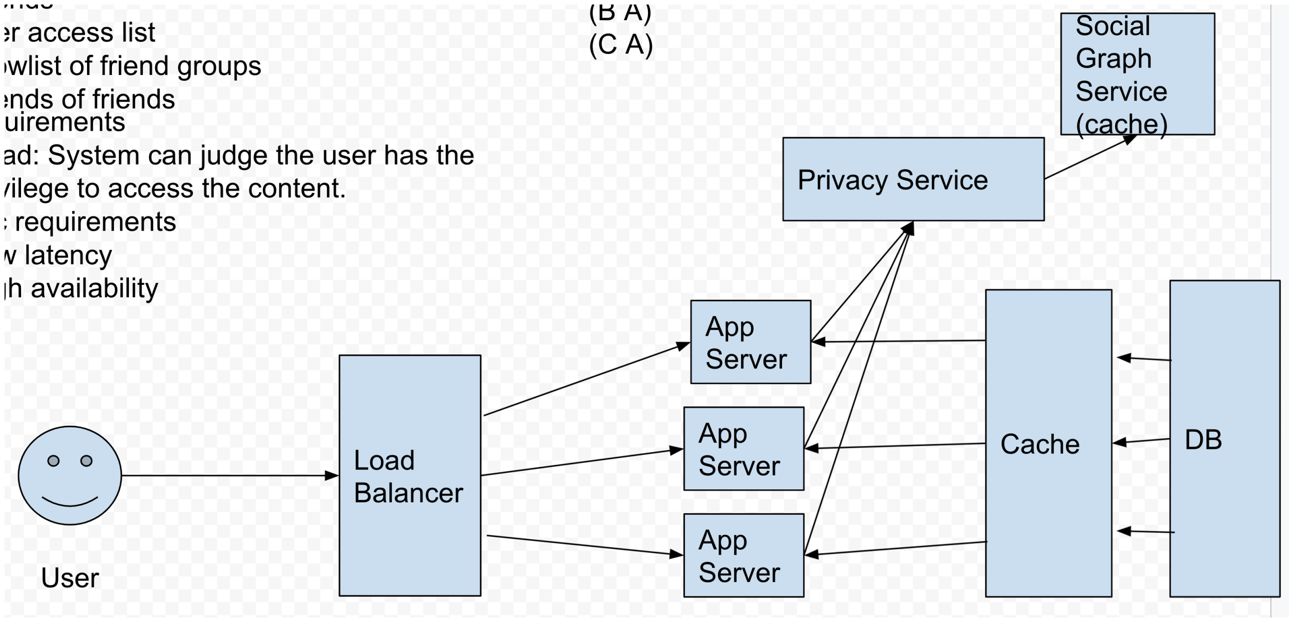
Estimation

- **Traffic estimation**
    - Our system will be read-heavy.

High-level design

- Use Enum data type to define the different privacy levels:
  - Public
  - Friends
  - Friends of friends
  - User access list
  - Allowlist of friend groups
  - Private
- Each post will have a privacy level field.
- Store the user relationship data in a key-value store (cache).
  - Key = UserID, Value = A set of all the friends.
  - Sharded across multiple instances.
- Friends of friends
  - Perform an intersection of the post's owner and the viewer's friend lists.
  - Add a record of friend of friend (A-B, B-C, so add A-C into the user relationship data).





Data model definition

- Schema
  - Table 1: Post
    - Description
      - Store post information.
    - Columns

Column Name	Column Type	PK	Description
PostID	int	PK	The post ID.
UserID	int		The user ID of the post creator.
PrivacyLevel	string		The privacy level of the post, the value can be: Public, Private, Friend, etc.

  - Table 2: FriendRelationship
    - Description
      - Store user's friend relationship.
    - Columns

Column Name	Column Type	PK	Description
RelationID	int	PK	The friend relation ID.

Column Name	Column Type	PK	Description
FirstUserID	int		The first user ID of one relationship.
SecondUserID	int		The second user ID of one relationship.

- Table 3: FriendGroup
  - Description
    - Define user's friend groups, which allow a group of friends to access a post directly.
  - Columns

Column Name	Column Type	PK	Description
UserID	int	PK	The user ID.
GroupID	int	PK	The user's friend group ID.
GroupName	string		The name of the friend group.
FriendUserID	int		The user's one friend's user ID.

Detailed design

- **How to store friend relationship**
  - Assumptions
    - Friendship is symmetrical (If A is friends with B that implies that B is also friends with A).
  - Options
    - Option 1: Use key-value store (cache).
      - Schema
        - Key is UserID
        - Value is set of all the friends.
      - Example

Key	Value
A	{B, C, D, E}

    - Option 2: Use SQL database.
      - Schema

        - RelationID: The primary key of relationship.
        - FirstUserID: The first user ID of one relationship.
        - SecondUserID: The second user ID of one relationship.
      - Example

RelationID	FirstUserID	SecondUserID
1	A	B
2	A	C
3	A	D
...	...	...

      - Optimization
        - Use 2 rows for one relationship (If A and B are friend, add A-B and B-A to the table).
    - Option 3: Use graph database.
- **How to figure out friend of friend from database**