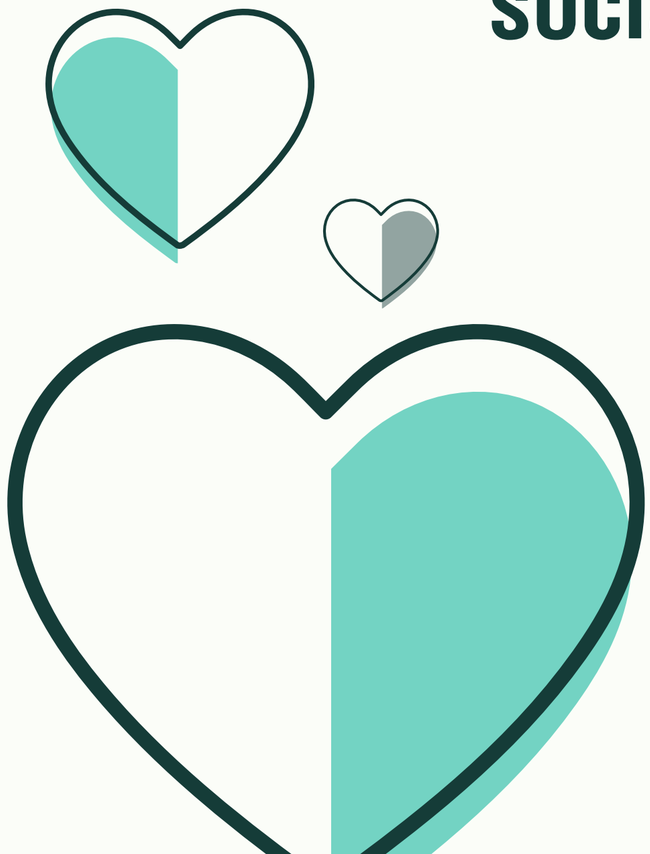
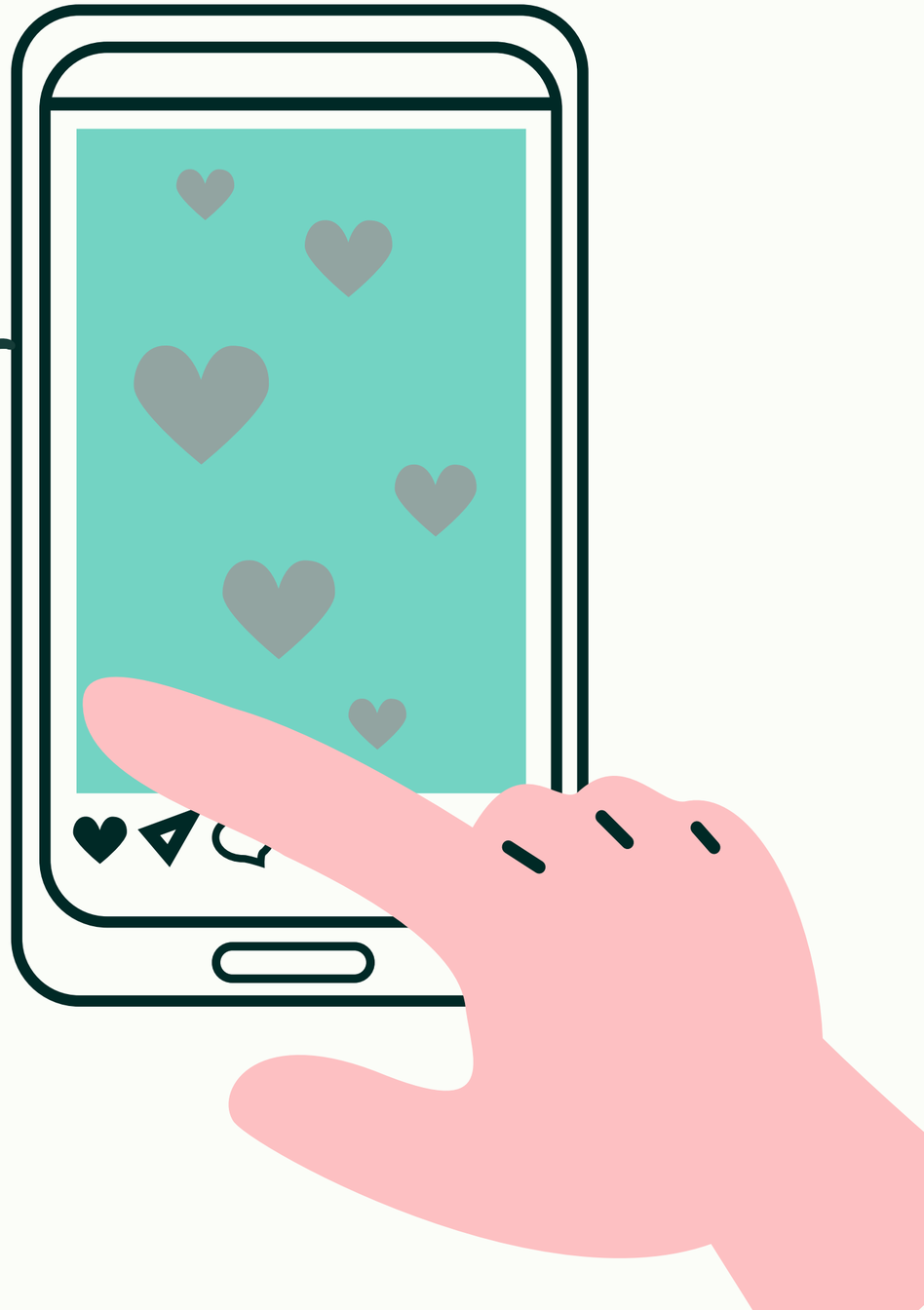
Yumelink

social media *project*





Borcelle Agency
2023

Project outline

- Real world - business
- UML
- Translate to Relation
- 1 -3 NF
- Django queries
- DEMO

REAL WORLD - BUSINESS

Business selection: Social Media Platform

- **Objective:** A social media web application designed to showcase the process of implementing database knowledge into building a functional software product. The focus is on learning and practical experience.

List Business Activities/Processes.

User Profiles

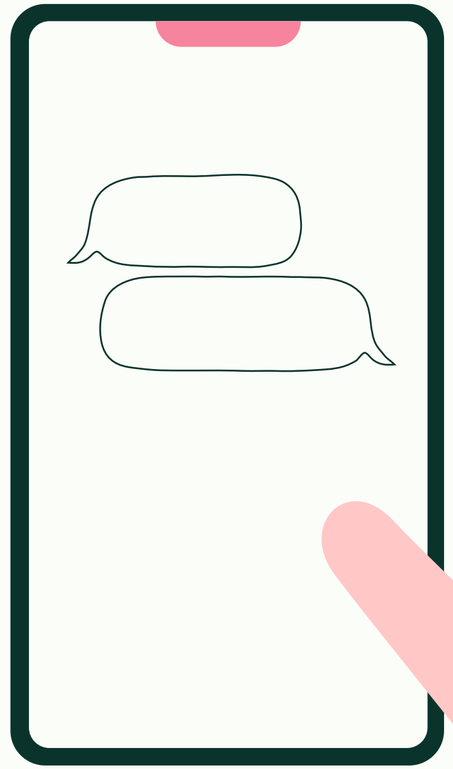
Activities/Processes:

- Users fill out personal information, create bios, and upload profile pictures.
- Users can edit their bios or change their profile pictures.

User Relationship

Activities/Processes:

- Users can follow other users to see their posts or block users to prevent interactions.



List Business Activities/Processes.

Content

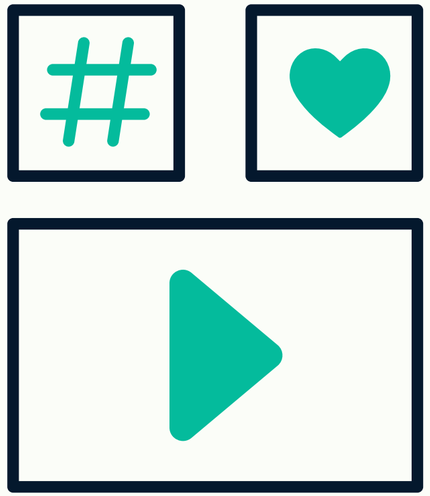
Activities/Processes:

- Users post text, images, or videos (e.g., status updates or media posts).

Message

Activities/Processes:

- Users can send private messages to each other, either individually or in groups



List Business Activities/Processes.

User Activity

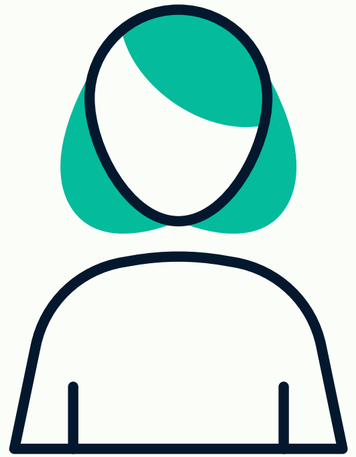
Activities/Processes:

- Users can like, share, or comment on post

Tags

Activities/Processes:

- Users can add tags (keywords) to their posts for categorization.



Processes necessitate

User Profiles

Collect:

- Collect user details (e.g., name, email, bio, profile picture) during registration and updates.

Manage (Insert, Update, Delete):

- Add new user profile data when they sign up.
- Allow users to update their bio, profile picture, and other personal details.

Data Use:

- Display user profiles on the use interface.

Processes necessitate

User Relationship

Collect:

- Collect and store data about who is following whom, and who has blocked whom.

Manage (Insert, Update, Delete):

- A new relationship (e.g., following or blocking)
- Users can change their relationships (e.g., unfollow someone or unblock them)

Data Use:

- Displaying the correct feed of posts for each user based on who they follow.

Processes necessitate

Content

Collect:

- Collect data related to user posts (timestamps, post ID, etc.).

Manage (Insert, Update, Delete):

- New posts are inserted into the database when users upload content.
- Users can edit, delete or update their posts

Data Use:

- Displaying posts to users (on their feed or specific user pages).

Processes Necessitate

Message

Collect:

- Collect message data (sender, receiver, timestamp, message content).

Manage:

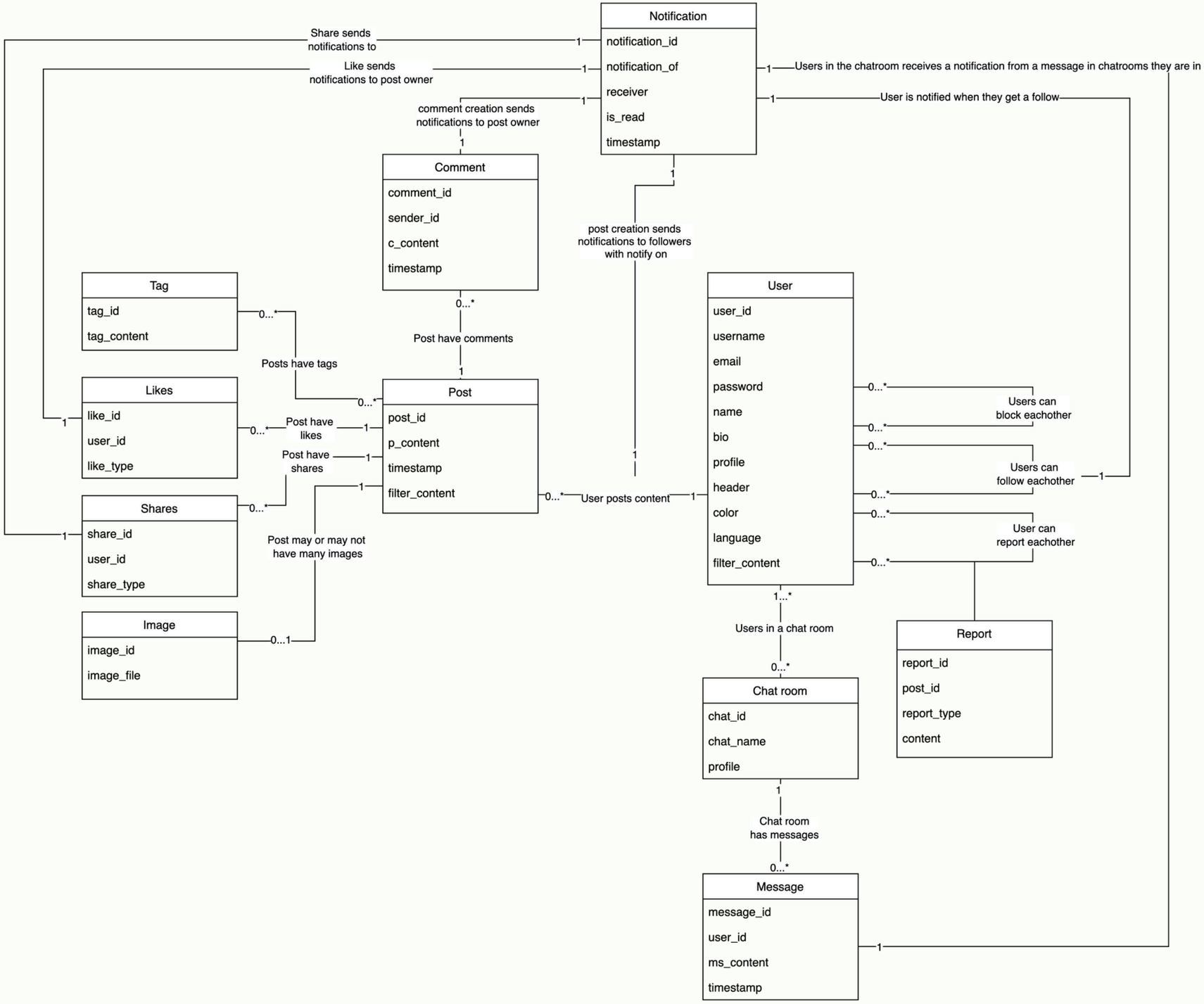
- Each new message sent between users is added to the database.

Data Use:

- Displaying messages and conversations on the user interface.

DATABASE

VML


Notification
notification_id
notification_of
receiver
is_read
timestamp
Share sends notifications to
Like sends notifications to post owner
comment creation sends notifications to post owner
Users in the chatroom receives a notification from a message in chatrooms they are in
User is notified when they get a follow
Comment
comment_id
sender_id
c_content
timestamp
post creation sends notifications to followers with notify on
Tag
tag_id
tag_content
Post have comments
Posts have tags
Likes
like_id
user_id
like_type
Post have likes
Post have shares
Post
post_id
p_content
timestamp
filter_content
User posts content
User
user_id
username
email
password
name
bio
profile
header
color
language
filter_content
Users can block eachother
Users can follow eachother
User can report eachother
Shares
share_id
user_id
share_type
Post may or may not have many images
Image
image_id
image_file
Users in a chat room
Report
report_id
post_id
report_type
content
Chat room
chat_id
chat_name
profile
Chat room has messages
Message
message_id
user_id
ms_content
timestamp

Translate to Relation

red- Foreign key
blue- Primary key that is not Foreign key
underline - Primary key
black- Mon-key attribute

Relation	Attribute	Foreign keys
Follow	• user_id • follower_id • notify	• user_id: User(user_id) • follower_id: User(user_id)
Blocker	• blocker_id • blocked_id	• blocker_id: User(user_id) • blocked_id: User(user_id)
Report	• report_id • post_id • user_id • reporter_id • report_type • content	• post_id : Post(post_id) • user_id : User(user_id) • reporter_id : User(user_id)

Relation	Attribute	Foreign keys
User	• user_id • username • email • password • name • bio • profile • header • color • language • filter_content	
UserChat	• user_id • chat_id	• user_id: User(user_id) • chat_id: ChatRoom(chat_id)

red- Foreign key
blue- Primary key that is not Foreign key
underline - Primary key
black- Mon-key attribute

Relation	Attribute	Foreign keys
Post	• post_id • user_id • p_content • timestamp • filter_content	• user_id: User(user_id)
PostImage	• postimage_id • post_id • image	• post_id : Post(post_id)
Tag	• tag_id • tag_content	
PostTag	• tag_id • post_id	• tag_id : Tag(tag_id) • post_id : Post(post_id)

red- Foreign key
blue- Primary key that is not Foreign key
underline - Primary key
black- Mon-key attribute

Relation	Attribute	Foreign keys
Comment	• comment_id • user_id • post_id • c_content • timestamp	• user_id : User(user_id) • post_id : Post(post_id)
ChatRoom	• chat_id • chat_name • profile	
Message	• message_id • user_id • chat_id • ms_content • timestamp	• user_id : User(user_id) • chat_id : ChatRoom(chat_id)

red- Foreign key
blue- Primary key that is not Foreign key
underline - Primary key
black- Mon-key attribute

red- Foreign key
blue- Primary key that is not Foreign key
underline - Primary key
black- Mon-key attribute

Relation	Attribute	Foreign keys
Likes	• like_id • user_id • post_id • like_type	• user_id : User(user_id) • post_id : Post(post_id)
Shares	• share_id • user_id • post_id • share_type	• post_id : Post(post_id) • user_id : User(user_id)
Notification	• notification_id • notification_of • receiver_id • is_read • timestamp	• notification_of : Any Class object • receiver_id : User(user_id)

1-3 nf

Notification(notification_id, notification_of, receiver_id, is_read, timestamp)

1NF: All attributes contain atomic values.

- notification of

notification_of is a model object with id, type

Thus, we seperate it into 2 attributes:

id(Integer) and content_type(ContentType)

*ContentType is a Django model for model object class Types

2NF: No partial dependency on any subset of a primary key.
Since notification_id is the primary key, all non-key attributes (notification_id, notification_of, receiver_id, is_read, timestamp) are fully dependent on notification_id.
3NF: No transitive dependency on non-key attributes.
There are no dependencies between non-key attributes; therefore, no transitive dependency exists.

DJANGO QUERIES

We will discuss Django Queries in DEMO, for example;

- current_user = User.objects.get(id=self.request.user.id)
- blocking_users = Block.objects.filter(blocked=current_user).values_list('blocker', flat=True)
- posts = Post.objects.all().exclude(user__id__in=excluded_users).order_by('-timestamp')
- posts = Post.objects.all().order_by('-timestamp')
- 'post_images': PostImage.objects.filter(post=post)
- is_blocked = Block.objects.filter(blocker=post_owner, blocked=current_user).exists()
- for comment in Comment.objects.filter(post=post).order_by('-timestamp')
- 'owns': User.objects.get(id=self.request.user.id) == comment.user
- context['tags'] = Tag.objects.filter(posttag__post=post)
- context['post_tags'] = PostTag.objects.filter(post=post)
- context['likes'] = Like.objects.filter(post=post, type=LikeType.like.name).count()
- context['is_block'] = Block.objects.filter(blocker=user, blocked=viewed_user).exists()
- ...


DEMO

TEAm MEmBEr

6510545276 Kantapon Hemmadhun

6510545535 Nantawan Paramapooti

6510545616 Phatthadon Suwanpattanawech

THANK YOU

for watching!

More Details of the Project on

YumeLink Github