Kelompok 3 Tugas Rekayasa Perangkat lunak

Untuk Memenuhi Tugas Rekayasa Perangkat lunak

Dosen Pengampu: Dr. Achmad Arwan, S.Kom., M.Kom.

Disusun Oleh:

Nada Nadhira Najwa Mazaya	256150100111005
Syahrul Budi Rahmadan	256150100111013
Muhamad Rizqi Duha Pramudya	256150100111016

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI
UNIVERSITAS BRAWIJAYA
FAKULTAS ILMU KOMPUTER
MAGISTER ILMU KOMPUTER
MALANG

2025

Transformasi Requirement → Class Diagram dengan 4C

Step 1: Identifikasi Objek

a. *Noun Analysis* (ekstraksi kata benda dari requirement card)
Dari requirement cards (Login, Upload Post, Story, Like/Comment, DM, Notifikasi, Moderasi):

- Nouns utama: **User, Account, Post, Photo, Video, Story, Comment, Message, Notification, Moderation, System**.

b. *Responsibility Analysis*

- **User** → login, membuat postingan/story, mengirim pesan, memberi komentar/like.
- **System** → autentikasi, simpan data, kirim notifikasi, moderasi konten.
- **Content (Post, Story, Comment, Message)** → mewakili entitas utama media sosial.

Hasil: objek: `User`, `Account`, `Post`, `PhotoPost`, `VideoPost`, `Story`, `Comment`, `DirectMessage`, `NotificationService`, `ModerationService`.

Step 2: Abstraksi Objek → Class

➔ Objek dengan karakteristik sama digabung jadi **class**.

- ◆ User individu → `User` class.
- ◆ Postingan → `Post` (superclass), dengan turunan `PhotoPost`, `VideoPost`.
- ◆ Story → `Story`.
- ◆ Komentar → `Comment`.
- ◆ Pesan → `DirectMessage`.
- ◆ Autentikasi → `Account`.

➔ Tambahkan atribut dasar (ID, timestamp, contentURL, dsb).

Hasil: Draft kelas inti.

Step 3: Analisis Asosiasi antar Class

- **User – Account** → one-to-one (satu user punya satu akun).
- **User – Post** → one-to-many (user bisa membuat banyak postingan).
- **User – Story** → one-to-many.
- **User – DirectMessage** → user bisa kirim/terima DM (one-to-many ke sesama user).
- **User – Comment – Post** → user menulis komentar pada postingan.
- **Post – Comment** → one-to-many (satu post punya banyak komentar).
- **Post – NotificationService** → trigger notifikasi saat ada like/comment.
- **Comment – ModerationService** → komentar discan sebelum ditampilkan.

Hasil: Relasi UML pada class diagram.

Step 4: Boundary–Control–Entity (BCE Pattern)

- **Entity Classes**: `User`, `Account`, `Post`, `Story`, `Comment`, `DirectMessage`.
- **Controller Classes**: implicit dalam method (misal: `User.createPost()`, `User.sendMessage()` bertindak sebagai kontrol logika bisnis).
- **Boundary Classes**: `NotificationService`, `ModerationService` (sebagai interface dengan sistem eksternal/aktor).

Hasil: Layer arsitektur lebih jelas, memisahkan domain, logika, dan layanan eksternal.

Step 5: Confirmation (Validasi dengan Stakeholder)

➔ Dicek apakah kebutuhan user sudah terwakili:

- **Login** → `Account` dengan method `login(), resetPassword()`.
- **Posting** → `Post` + subclass `PhotoPost`, `VideoPost`.
- **Story** → `Story` dengan `duration = 24h`.

- **Komentar & Like** → `Comment`, method `addComment()`, `addLike()`.
- **DM** → `DirectMessage` dengan `isRead`, `markAsRead()`.
- **Notifikasi** → `NotificationService`.
- **Moderasi** → `ModerationService`.

Hasil: Class diagram diverifikasi traceable ke semua requirement.

Step 6: Context (Konteks Bisnis–Teknis–Lingkungan)

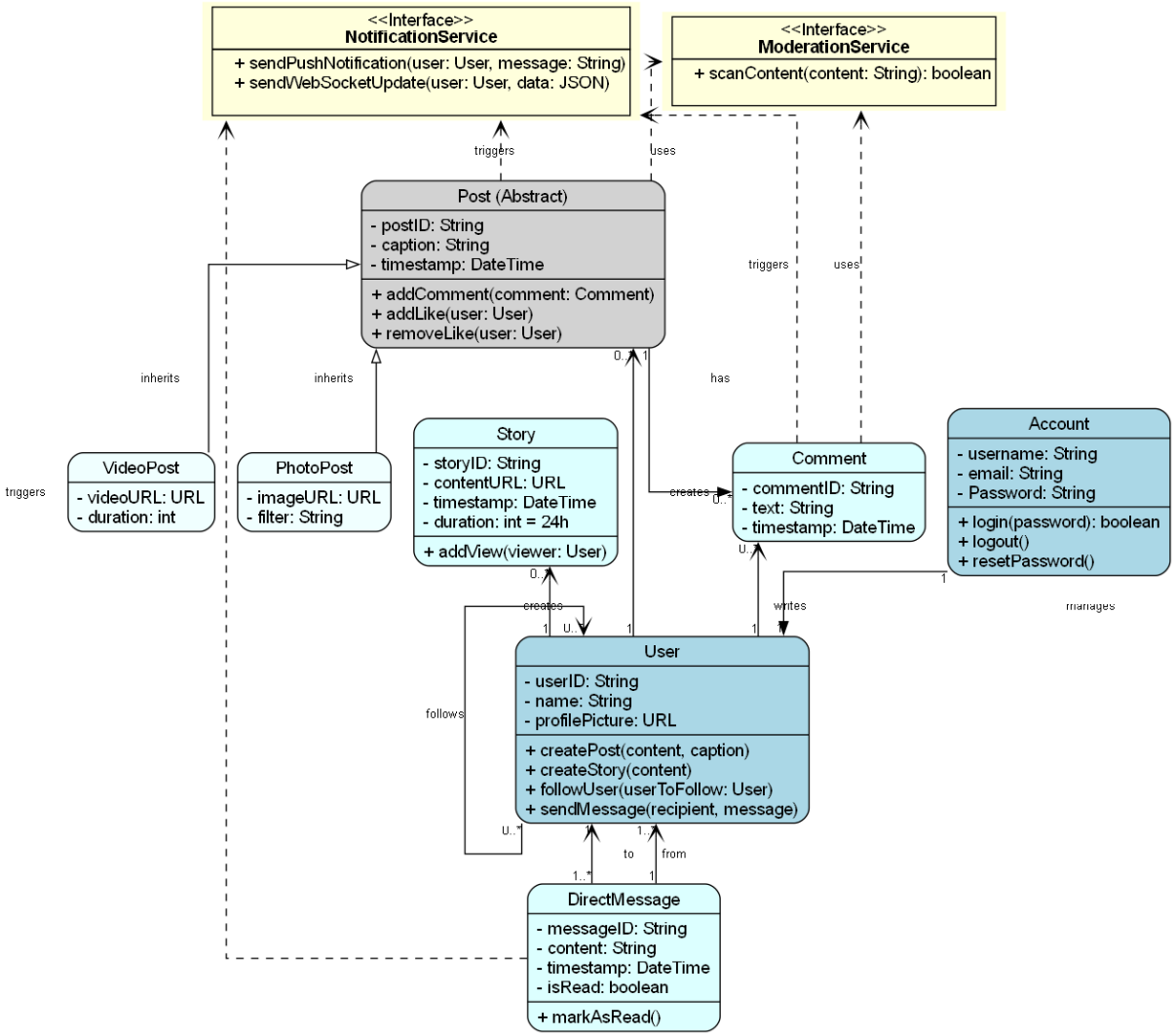
- **Business context**: menjaga engagement user (like, comment, story, DM).
- **Technical context**: butuh WebSocket untuk notifikasi real-time, moderasi AI untuk spam/konten negatif.
- **User context**: user menginginkan interaksi cepat (login < 3 detik, notifikasi < 2 detik).
- **Environmental context**: skalabilitas (jutaan user, like/detik).

Hasil: Kelas **NotificationService** & **ModerationService** ditambahkan untuk memenuhi kebutuhan konteks teknis.

Card (Requirement)	Mapping ke Class Diagram	Penjelasan Transformasi
REQ-IG-001: Login & Sign Up	User, Account	Dari kebutuhan login & registrasi, muncul kelas **User** (atribut profil) & **Account** (atribut autentikasi). Method: login(), logout(), resetPassword().
REQ-IG-002: Upload Foto/Video	Post, PhotoPost, VideoPost, User	Requirement "unggah konten" dimodelkan dengan kelas **Post** (superclass). Turunannya **PhotoPost** dan **VideoPost**. Relasi: **User → Post** (one-to-many).
REQ-IG-003: Stories (24 Jam)	Story, User	Kebutuhan konten sementara menghasilkan kelas **Story**. Atribut: storyID, duration. Relasi: **User → Story**.

REQ-IG-004: Pengaturan Akun	Account, User	Privasi akun dimodelkan dengan atribut isPrivate di **Account**. Relasi follow: **User ↔ User** (self-association).
REQ-IG-005: Direct Message (DM)	DirectMessage, User	DM diwakili kelas **DirectMessage** dengan atribut messageID, content, timestamp. Relasi: **User (sender → receiver)**.
REQ-IG-006: Like & Comment	Comment, User, Post	Interaksi sosial direpresentasikan dengan **Comment**. Relasi: **User → Comment → Post**. Method: addComment(), addLike().
Notifikasi Real-time	NotificationService (interface)	Dari konteks teknis muncul service ini, yang mengirim notifikasi ke **User** ketika ada **Like**, **Comment**, atau **DM**.
Moderasi Konten	ModerationService (interface)	Dari konteks keamanan/machine learning. Digunakan oleh **Comment** atau **Post** untuk validasi konten.

Class Diagram Media Sosial:



Penjelasan Class Diagram:

1. NotificationService (Interface)

- **Fungsi**: Layanan untuk mengirim notifikasi real-time.
- **Method**:
    - sendPushNotification(user: User, message: String) → kirim notifikasi push.
    - sendWebSocketUpdate(user: User, data: JSON) → update realtime via WebSocket.
- **Relasi**: Dipicu oleh aksi di kelas lain (misalnya saat ada komentar/like/DM).

2. ModerationService (Interface)

- **Fungsi**: Layanan moderasi konten (misal filtering teks yang tidak pantas).
- **Method**:
    - scanContent(content: String): boolean → mengecek apakah konten aman.
- **Relasi**: Digunakan oleh **Comment** dan **Post** sebelum dipublikasikan.

3. Post (Abstract Class)

- **Atribut**:
  - postID: String → identitas unik posting.
  - caption: String → teks keterangan posting.
  - timestamp: DateTime → waktu posting dibuat.
- **Method**:
  - addComment(comment: Comment) → tambahkan komentar.
  - addLike(user: User) → tambahkan like dari user.
  - removeLike(user: User) → hapus like dari user.
- **Relasi**:
  - *Inheritance* → diturunkan ke **PhotoPost** dan **VideoPost**.
  - *Trigger* → memicu **NotificationService** saat ada interaksi.

4. PhotoPost (Subclass dari Post)

- **Atribut**:
  - imageURL: URL → lokasi gambar.
  - filter: String → filter efek foto.
- **Fungsi**: Posting berbasis gambar.

5. VideoPost (Subclass dari Post)

- **Atribut**:
  - videoURL: URL → lokasi file video.
  - duration: int → durasi video (detik/menit).
- **Fungsi**: Posting berbasis video.

6. Story

- **Atribut**:
  - storyID: String → identitas unik story.
  - contentURL: URL → lokasi media story.
  - timestamp: DateTime → waktu story diposting.
  - duration: int = 24h → masa aktif story.
- **Method**:
  - addView(viewer: User) → mencatat siapa yang melihat story.
- **Relasi**:
  - **User → Story**: User membuat banyak story.
  - *Trigger* → notifikasi untuk interaksi story.

7. Comment

- **Atribut**:
  - commentID: String → identitas unik komentar.
  - text: String → isi komentar.
  - timestamp: DateTime → waktu komentar dibuat.
- **Relasi**:
  - **User → Comment**: User menulis komentar.

- **Comment → Post**: Komentar terkait posting tertentu.
- *Uses* → **ModerationService** untuk screening konten.

8. Account

- **Atribut**:
  - username: String
  - email: String
  - password: String
- **Method**:
  - login(password): boolean → verifikasi login.
  - logout() → keluar dari akun.
  - resetPassword() → ubah kata sandi.
- **Relasi**:
  - **User ↔ Account**: 1 user memiliki 1 akun.

9. User

- **Atribut**:
  - userID: String → identitas unik user.
  - name: String → nama user.
  - profilePicture: URL → foto profil.
- **Method**:
  - createPost(content, caption) → membuat posting.
  - createStory(content) → membuat story.
  - followUser(userToFollow: User) → mengikuti user lain.
  - sendMessage(recipient, message) → mengirim pesan.
- **Relasi**:
  - **User ↔ User** (self-association) → follow/following.
  - **User → Post/Story/Comment/DirectMessage**.

10. DirectMessage

- **Atribut**:
  - messageID: String → identitas unik pesan.
  - content: String → isi pesan.
  - timestamp: DateTime → waktu pesan dikirim.
  - isRead: boolean → status sudah dibaca atau belum.
- **Method**:
  - markAsRead() → ubah status pesan jadi “dibaca”.
- **Relasi**:
  - **User (sender → receiver)**.

Code :

https://github.com/SyahrulUB/Tugas_RPL_Kelompok_3.git