

|  |
| --- |
| Social Media |
|  |

Contents

[1 social Media Description 3](#_Toc62212630)

[1.1 Social Media background 3](#_Toc62212631)

[1.2 Problems. Current Situation 3](#_Toc62212632)

[1.3 The benefits of implementing a database. Project Vision 3](#_Toc62212633)

[2 Model description 3](#_Toc62212634)

[2.1 Definitions & Acronyms 3](#_Toc62212635)

[2.2 Logical Scheme 3](#_Toc62212636)

[2.3 Objects](#_Toc62212637) 4

# 

# 1. social Media Description

## social Media background

A social network where users create profiles, publish posts (optionally with media and location), interact through comments and reactions, share others’ posts, follow people, and discover content via hashtags. Privacy and preferences are stored per user.

## Problems. Current Situation

We have designed an ER diagram for a Social Media platform that already goes beyond the basics: it’s normalized (3NF), uses clear PK/FK constraints, relationships M:N, and includes rules to prevent duplicates and keep references clean.

## the Benefits of implementing a database. Project Vision

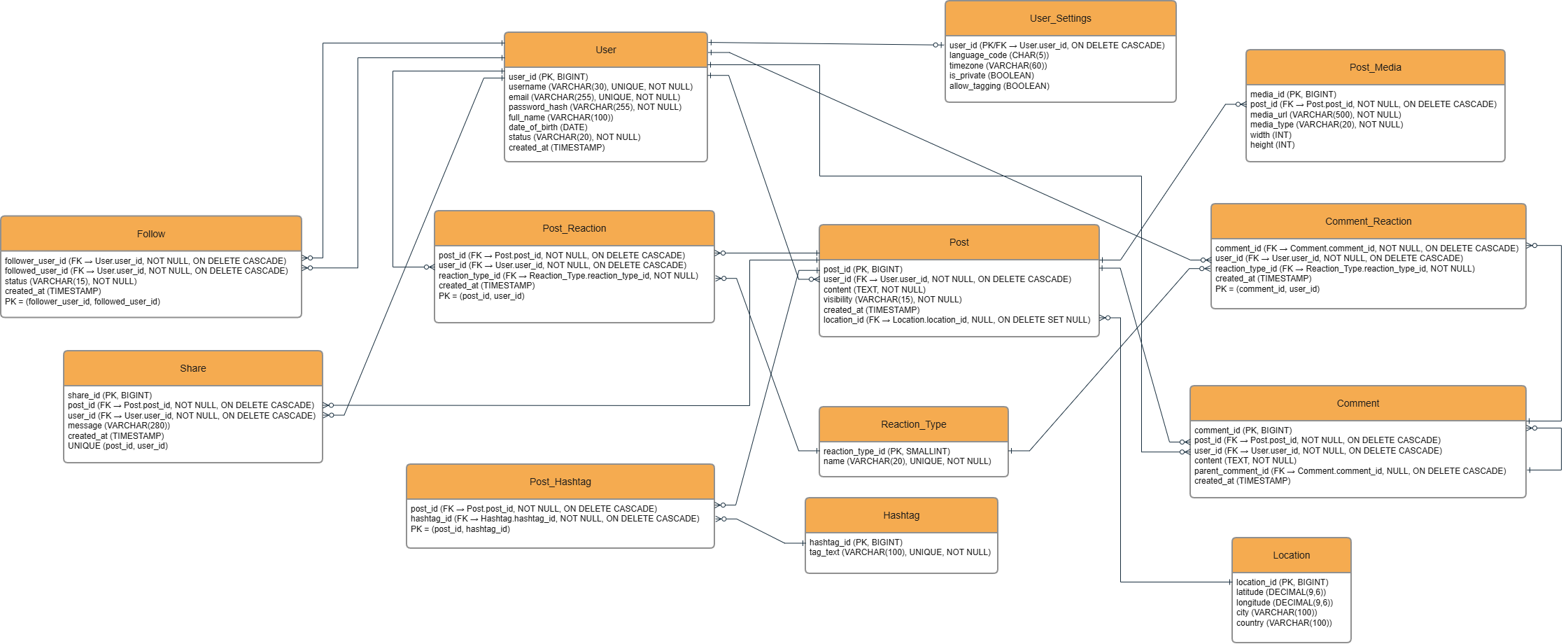
The current schema is production-ready and can be implemented in a relational database as is. It’s also easy to extend—new features (e.g., stories, direct messages, groups, extra media/analytics) can be added by attaching new tables without breaking existing ones, thanks to the modular design and explicit relationships.

# Model description

## Definitions & Acronyms

PK – Primary Key; FK – Foreign Key; UQ – Unique; NN – Not Null;

## Logical Scheme



## Objects

Table Description

The User table stores core information about each registered user of the social media platform. It contains authentication data (username, email, password hash), personal details (full name, date of birth), account status, and the creation timestamp.

user\_id is the PK. username and email are UQ and NN to ensure uniqueness and data integrity.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| User | user\_id | Primary key (unique identifier for each user), **PK** | BIGINT |
| username | Unique username for login, **UQ, NN** | VARCHAR(30) |
| email | User’s email address (must be unique), **UQ, NN** | VARCHAR(255) |
| password\_hash | Encrypted password of the user, **NN** | VARCHAR(255) |
| full\_name | User’s full name (optional field) | VARCHAR(100) |
| date\_of\_birth | User’s date of birth | DATE |
| status | User’s account status (e.g., active, suspended), **NN** | VARCHAR(20) |
| created\_at | Date and time the user account was created | TIMESTAMP |

Comments on table relationships

1 : 0..1 — User ↔ User\_Settings

1 : 0..N — User ↔ Post

1 : 0..N — User ↔ Comment

1 : 0..N — User ↔ Share

1 : 0..N — User ↔ Post\_Reaction

1 : 0..N — User ↔ Comment\_Reaction

M:N — User ↔ User (Follow) via join table Follow

Follow.follower\_user\_id — FK → User.user\_id (NOT NULL, ON DELETE CASCADE)

Follow.followed\_user\_id — FK → User.user\_id (NOT NULL, ON DELETE CASCADE)

Composite PK (follower\_user\_id, followed\_user\_id) enforces one follow per pair (prevents duplicates).

User ↔ User (Follow) is M:N because any user can follow many users and be followed by many. Implemented with join table Follow (PK (follower\_user\_id, followed\_user\_id), FKs to User, ON DELETE CASCADE) which also stores status and created\_at.

3NF note for User:

All columns are atomic (1NF). The PK is a single column user\_id, so there are no partial dependencies (2NF).

Every non-key attribute (username, email, password\_hash, full\_name, date\_of\_birth, status, created\_at) depends only on user\_id and not on other non-keys (no transitive dependencies → 3NF).

Example with data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| user\_id | username | email | password\_hash | full\_name | date\_of\_birth | status | created\_at |
| 1002 | milena.dodic | milena.dodic@example.com | $2b$12$Wk7x1... | Milena Dodić | 1998-02-04 | active | 2025-10-04 15:05:00 |