

|  |
| --- |
| BUSINESS DESCRIPTION  **Social media** |

Contents

[1 Business Description 3](#_Toc62212630)

[1.1 Business background 3](#_Toc62212631)

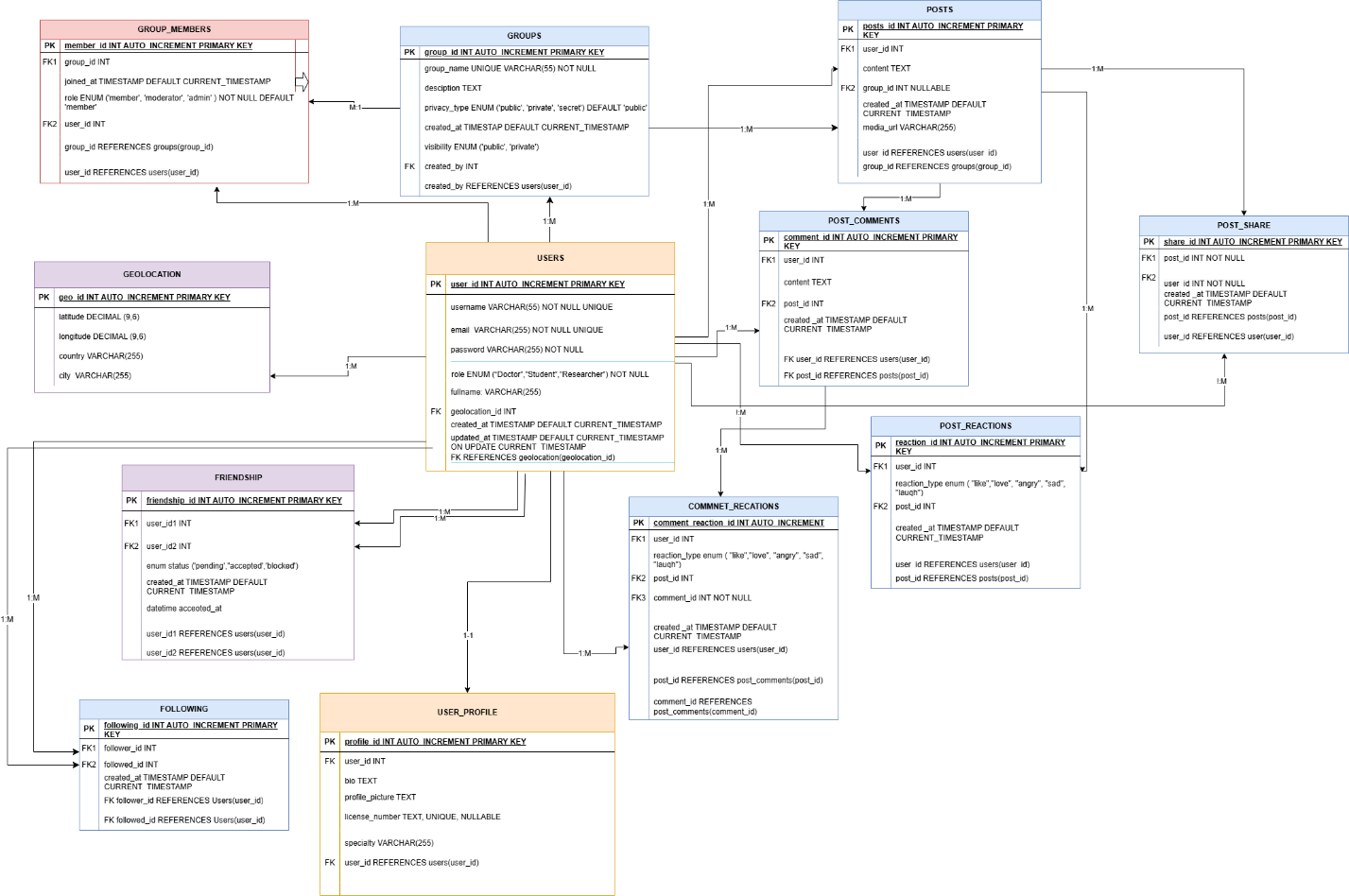
[1.2 Problems. Current Situation 3](#_Toc62212632)

[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 3](#_Toc62212637)

Business Description

Problems. Current Situation

Healthcare professionals often struggle with:

 **Limited networking opportunities** beyond their workplace or local conferences.

 **Difficulty sharing research, case studies, and medical advancements** in an organized and accessible way.

 **Fragmented medical discussions** spread across various platforms (LinkedIn, medical forums, WhatsApp groups).

 **Challenges in mentorship**—junior professionals lack direct access to experienced mentors.

 **Privacy concerns** when discussing sensitive medical topics on general social media platforms

the Benefits of implementing a database. Project Vision

A **secure, professional social media platform** for healthcare providers, researchers, and students to **connect, share medical knowledge, discuss cases, and collaborate**.

**Database Benefits:**

**Structured user profiles** categorized by specialty, experience, and certifications.

**Secure, role-based access** to ensure medical discussions remain professional and compliant.

**Easy knowledge sharing** via posts, articles, and case studies.

**Verified user system** to prevent misinformation and unqualified users.

**Real-time chat & discussion groups** for quick consultations and peer interactions.

Model description

Definitions & Acronyms

**PK**: Primary Key - A unique identifier for each record in a table

**FK**: Foreign Key - A field that refers to the primary key in another table

**1:1**: One-to-One relationship - Each record in Table A relates to exactly one record in Table B

**1:M**: One-to-Many relationship - One record in Table A can relate to multiple records in Table B

**M:N**: Many-to-Many relationship - Multiple records in Table A can relate to multiple records in Table B

**AUTO\_INCREMENT**: A property that automatically generates a unique number for new records

**NOT NULL**: Constraint that ensures a column cannot have a NULL value

**UNIQUE**: Constraint that ensures all values in a column are different

**DEFAULT**: Provides a default value for a column when none is specified

**ENUM**: A data type that allows a column to have one value from a predefined list

Logical Scheme

Objects

The Users table stores general information about each user. It serves as the central table for user management, containing user credentials and roles.

**Users**

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| Users | user\_id | Unique identifier for each user. PK, AUTO\_INCREMENT | INT |
| email | User’s email address. UNIQUE, NOT NULL | VARCHAR(255) |
| password | Hashed data password for authentication. NOT NULL | VARCHAR(255) |
| role | User role(Doctor, Student, Researcher, Admin). NOT NULL | ENUM (“Doctor”,”Student”,”Researcher”) |
| fullname | Full name of the user. NOT NULL | TEXT |
| created\_at | Account creation timestamp. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |
| updated\_at | Last profile update timestamp. ON UPDATE CURRENT\_TIMESTAMP | TIMESTAMP |
|  | geolocation\_id | FK REFERENCES geolocation(geolocation\_id) | INT |

The **User\_profile** has a field user\_id field that references that Users table. This creates one-to-one relationship between a user and their profile. This means each user can have only one profile and each profile is tied to a specific user.

Users has one-to-many relationship with groups table, meaning one user can create multiple groups, but each group is tied to a single creator.

Between group\_memebers and users there is one-to-many relationship because a **user** can be a **member of many groups**, but each entry in the **Group\_Members** table corresponds to **one user**. The user\_id in Group\_Members references the Users table.

A user is associated with a geolocation, but geolocation itself is not inherently tied to a specific user. One geolocation can be associated with many users. Each user is associated with only one geolocation at a time.

Users table has one-to-many relationship with Posts table. Because a user can create multiple posts, but each post is associated with a single user. The posts table includes a user\_id that ties the post to the user who created it.

Users table has one-to-many relationship with Post\_comment table. A user can comment on many posts. Each comment is tied to a specific user, and the post\_comments table has a user\_id to track the creator of each comment.

Comment\_reactions serves as a junction table with foreign keys to both Users and Comments. Users have a 1-to-M relationship with Comment\_reactions (one user can create many comment reactions).

Post\_Share serves as a junction table with foreign keys to both Users and Posts. Users have a 1-to-M relationship with Post\_Share (one user can share many posts)

The Friendship table represents a self-referential relationship within the Users table. Each row in the Friendship table connects two user records.

The relationship: Users have a 1-to-M relationship with Friendship through user\_id\_1 . Users have a 1-to-M relationship with Friendship through user\_id\_2.

Follows table is a self-referential relationship within the Users table. Users have a 1-to-M relationship with Follows through follower\_id, Users have a 1-to-M relationship with Follows through followed\_id. The Followers table functions as a junction table.

**Example with data**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| user\_id | email | password | role | fullname | created\_at | updated\_at |
| 1 | test@gmail.com | Bfahddncd1245dasda4324 | Doctor | Erin Brokovich | 2025-03-12 12:00:00 | 2025-03-12 12:00:00 |

User\_profile

The User\_profile table maintains additional details about each user.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| User\_profile | profile\_id | Unique profile identifier. PK, AUTO\_INCREMENT | INT |
|  | user\_id | References user. FK (user\_id) REFERENCES Users(user\_id) | INT |
| specialty | Medical specialty(cardiology, surgery, etc.). NULLABLE | VARCHAR(255) |
| license\_number | Medical license number. UNIQUE, NULLABLE | TEXT |
| bio | Short bio about user. **NULLABLE** | TEXT |
| profile\_picture | URL for profile picture. NULLABLE | TEXT |

User profile creating a one-to-one relationship with the Users table. Each user can have only one profile, and each profile belongs to a single user.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| profile\_id | user\_id | specialty | license\_number | bio | profile\_picture |
| 1 | 1 | cardiologist | A passionate cardiologist. | Erin Brokovich | erin\_brov\_pic.jpg |

**GROUPS**

The **Groups** table stores information about user-created groups.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| **GROUPS** | **group\_id** | **Unique identifier for groups. PK, AUTO\_INCREMENT** | **INT** |
| **group\_name** | **Name of the group. NOT NULL UNIQUE** | **VARCHAR(255)** |
| **description** | **Short desciption for the group.** | **TEXT** |
| **created\_at** | **When the group is created. DEFAULT CURRENT\_TIMTESTAMP** | **TIMESTAMP** |
| **created\_by** | **User who created the group. FOREIGN KEY (created\_by) REFERENCES Users(user\_id)** | **INT** |
| **visibility** | **Whether the group is public or private. NOT NULL DEFAUTLT ‘Public’** | **ENUM(‘public’, ‘private’)** |

Each group has a one-to-many relationship with the Users table through created\_by, meaning one user can create multiple groups, but each group is tied to a single creator.

The relationship between Groups and Group\_Members is a one-to-many relationship: One group can have many members. Also Users has one-to-many with Group\_Members. At the conceptual level: This creates a many-to-many relationship between Users and Groups (a user can be in many groups, a group can have many users)

The **Groups** table also has a one-to-many relationship with **Posts**, meaning posts can either be shared **within a group** or exist as standalone posts. One group can have many posts, each post belongs to at most one group.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| group\_id | group\_name | desctiprion | created\_at | created\_by | visibility |
| 1 | Cardiologist Network | A network for cardiologists to discuss research. | 2025-03-12 14:00:00 | 1 | Public |

**Group\_Members**

The GROUP\_MEMBERS table serves as a junction/bridge table that connects Users and Groups. It creates the many-to-many relationship between Users and Groups (users can belong to multiple groups, groups can have multiple users)

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| GROUP\_MEMBERS | member\_id | Unique identifier for group members. PK, AUTO\_INCREMENT | INT |
| group\_id | The group the user joined. FOREIGN KEY (group\_id) REFERENCES Groups(group\_id) | INT |
| user\_id | The user who joined.FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| joined\_at | When the user joined the group. DEFAULT CURRENT\_TIMTESTAMP | TIMESTAMP |
| role | Role of the meber. NOT NULL DEFAULT ‘member’ | ENUM(“Admin”, “Moderator”, “Member”) |

**One group** can have **many members** ,but each entry in **Group\_Members** corresponds to **one group**. The group\_id in Group\_Members references the Groups table. It establishes one-to-many realtionship.

Between group\_memebers and users there is one-to-many relationship because a **user** can be a **member of many groups**, but each entry in the **Group\_Members** table corresponds to **one user**. The user\_id in Group\_Members references the Users table.

At the conceptual level - It establishes a many-to-many relationship between Users and Groups, where a user can belong to multiple groups, and each group can have multiple users.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| member\_id | group\_id | user\_id | joined\_at | role |
| 1 | Cardiologist Netweok | A network for cardiologists to discuss research. | 2025-03-12 14:00:00 | Member |

**Geolocation**

The **Geolocation** table stores location information for users, including latitude, longitude, country, and city. This data represents the specific geographical position of a user at any given moment.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| Geolocation | geo\_id | Unique identifier for geolocation entry. PK, AUTO\_INCREMENT | INT |
| latitude | Latitude of the user’s location. | DECIMAL(9,6) |
| longitude | longitude ude of the user’s location. | DECIMAL(9,6) |
| country | Country where user lives. | VARCHAR(255) |
| city | City where user lives. | VARCHAR(255) |

The Geolocation table has a one-to-many relationship with the Users table. Users can have multiple location, but one entry in geol

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| geo\_id | latitude | Longitude | country | city |
| 2 | 40.712776 | -74.005974 | USA | LA |

Posts

The **Posts** table stores information about the posts created by users. It includes the content of the post, an optional media URL (for images or videos), and timestamps for when the post was created or last updated. Each post is linked to a specific user and group(optional), indicating who created the post and where it was shared.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| POSTS | post\_id | Unique identifier of post. PK, AUTO\_INCREMENT | INT |
| user\_id | The user who write a post. FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| content | The text or description of the post. | TEXT |
| media\_url | Optional image/video url attached | VARCHAR(255) |
| created\_at | Account creation timestamp. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |
| updated\_at | Last profile update timestamp. ON UPDATE CURRENT\_TIMESTAMP | TIMESTAMP |
| group\_id | The group where post shared.  FOREIGN KEY (group\_id) REFERENCES Groups(group\_id). NULLABLE | INT |

The Posts table has a one-to-many relationship with the Users table. This means that one user can write many posts, but each post is written by a single user.

The Posts table has a one-to-many relationship with the Groups table. This means that one group can have multiple posts, but each post belongs to a single group.

Post and Post\_comments have one-to-many relationship. Each post can have many comments, but each comment is associated with one specific post.

Post and Post\_Reactions have one-to-many relationship. Each post can have many reactions, but each reaction belongs to only one post.

Post\_Share serves as a junction table with foreign keys to both Users and Posts. Posts and Post\_share have **one-to-many** relationship. Each post can be shared by many users, but each share is tied to only one post.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| post\_id | user\_id | content | media\_url | created\_at | updated\_at | group\_id |
| 2 | 5 | Excited to share my research findings! | NULL | 2025-03-13 10:00:00 | 2025-03-13 10:00:00 | NULL |

Post\_comments

The **Post\_Comments** table stores comments made by users on posts. It includes the content of the comment, as well as timestamps for when the comment was created or last updated. Each comment is associated with both a specific post and a specific user, allowing for easy retrieval of all comments made by users on a particular post.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| POST\_COMMENTS | comment\_id | Unique identifier of comment. PK, AUTO\_INCREMENT | INT |
| post\_id | The post where user commented. FOREIGN KEY (post\_id) REFERENCES Posts(post\_id) | INT |
| user\_id | The user who wrote a comment. FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| content | The text or description of the comment. | TEXT |
| created\_at | Account creation timestamp. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |
| updated\_at | Last profile update timestamp. ON UPDATE CURRENT\_TIMESTAMP | TIMESTAMP |

The **Post\_Comments** table has a **one-to-many** relationship with the **Posts** table. This means that one **post** can have multiple **comments**, but each **comment** belongs to a single **post**.

The Post\_Comments table has a one-to-many relationship with the Users table. This means that one user can write multiple comments, but each comment belongs to a single user.

The Post\_Comments table has a one-to-many relationship with the Comment\_Reactions table. One comment can have multiple reactions, but each reaction belongs to a single comment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| comment\_id | post\_id | user\_id | content | created\_at | updated\_at |
| 2 | 5 | 2 | Awesome! | 2025-03-13 10:00:00 | 2025-03-13 10:00:00 |

**Post\_reactions**

Post\_reactions saves the information about post reactions.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| POST\_REACTIONS | reaction\_id | Unique identifier of reaction. PK, AUTO\_INCREMENT | INT |
| post\_id | The post that received a reaction. FOREIGN KEY (post\_id) REFERENCES Posts(post\_id) | INT |
| user\_id | The user who reacted. FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| reaction\_type | Type of reaction (Like ,love, etc. ) | ENUM(‘Like’, ‘Love’,’ |
| created\_at | Account creation timestamp. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |

Users can create many reactions, but each reaction is created by one user. This defines one-to-many relationship between users and post\_reaction tables.

Between posts and post\_reactions there is a one-to-many relationship. One post (from the Posts table) can receive many reactions (multiple entries in the Post\_Reactions table), but each reaction entry in Post\_Reactions corresponds to exactly one post (through the post\_id foreign key).

The Post\_Reactions table effectively serves as a junction table that implements a many-to-many relationship between Users and Posts at the conceptual level. This means:

* A user can react to multiple posts
* A post can receive reactions from multiple users

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| comment\_id | post\_id | user\_id | reaction\_type | created\_at | updated\_at |
| 2 | 5 | 2 | Like | 2025-03-13 10:00:00 | 2025-03-13 10:00:00 |

**Comment reactions**

Comment Reactions saves the information about post comment reactions.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| COMMENT\_REACTIONS | comment\_reaction\_id | Unique identifier of reaction. PK, AUTO\_INCREMENT | INT |
| comment\_id | The comment that received a reaction. FOREIGN KEY (comment\_id) REFERENCES Post\_Comments(comment\_id) | INT |
| user\_id | The user who reacted. FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| reaction\_type | Type of reaction (Like ,love, etc. ) | ENUM(‘Like’, ‘Love’,’ |
| created\_at | Account creation timestamp. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |
| updated\_at | Last profile update timestamp. ON UPDATE CURRENT\_TIMESTAMP | TIMESTAMP |

The Comment\_Reactions table has a one-to-many relationship with the Post\_Comments table. One comment can have multiple reactions, but each reaction belongs to a single comment.

The Comment\_Reactions table has a many-to-many relationship with the Users table. One user can react to multiple comments, and one comment can get multiple reactions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Comment\_reaction\_id | comment\_id | user\_id | reaction\_type | created\_at | updated\_at |
| 2 | 5 | 2 | Like | 2025-03-13 10:00:00 | 2025-03-13 10:00:00 |

**Post shares**

The **Post\_Share** table stores information about which users have shared which posts.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| POST\_SHARE | share\_id | Unique identifier of share. PK, AUTO\_INCREMENT | INT |
| post\_id | The post that that is tagged. FOREIGN KEY (post\_id) REFERENCES Posts(post\_id) | INT |
| user\_id | The user who sheared a post. FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| created\_at | Time when post shared. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |

Post\_Share, like Post\_Reactions, is functionally a junction table that implements a many-to-many relationship between Users and Posts at the conceptual level. However, at the database implementation level, this is represented by two one-to-many relationships:

Users to Post\_Share: One-to-many relationship (one user can share many posts)

Posts to Post\_Share: One-to-many relationship (one post can be shared by many users).

|  |  |  |  |
| --- | --- | --- | --- |
| share\_id | post\_id | user\_id | created\_at |
| 2 | 5 | 2 | 2025-03-13 10:00:00 |

**Friendship**

The **Friendship** table represents a relationship between two users, where one user can be friends with another.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| FRIENDSHIP | friendship\_id | Unique identifier of friendship. PK, AUTO\_INCREMENT | INT |
| user\_id\_1 | First user of frineship. FOREIGN KEY (user\_id\_1) REFERENCES Users(user\_id) | INT |
| user\_id\_2 | Second user of frindship. FOREIGN KEY (user\_id\_2) REFERENCES Users(user\_id) | INT |
| Status | Friendship status (e.g., Pending, Accepted, Blocked). NOT NULL | ENUM(‘pending’, ‘Accepted’,’Blocked’) |
| created\_at | Time when frienship created. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |

The table represents a friendship relationship between users. It includes two foreign keys, user\_id\_1 and user\_id\_2, both referencing the Users table. Each record tracks the relationship status between these two users. It defines one-to-many relationship, cause it represents user connection type.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| friendship\_id | user\_id\_1 | user\_id\_2 | status | created\_at |
| 2 | 5 | 2 | Accepted | 2025-03-13 10:00:00 |

Follows

The **Follows** table captures the **one-to-many** relationship between users in the context of following and being followed. One user (follower) can follow many users, and each follow record is linked to the user who is following and the user who is being followed. The **created\_at** field tracks when the following action occurred.

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Field name | Field Description | Data Type |
| Follows | following \_id | Unique identifier for the follow relationship.. PK, AUTO\_INCREMENT | INT |
| follower\_id | The user who follows another. FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| followed\_id | The user who is being followed. FOREIGN KEY (user\_id) REFERENCES Users(user\_id) | INT |
| created\_at | Time when follow relationship created. DEFAULT CURRENT\_TIMESTAMP | TIMESTAMP |

The Follows table functions as a junction table. Users have a 1-to-M relationship with Follows through follower\_id. Users have a 1-to-M relationship with Follows through followed\_id

|  |  |  |  |
| --- | --- | --- | --- |
| friendship\_id | user\_id\_1 | user\_id\_2 | created\_at |
| 2 | 5 | 2 | 2025-03-13 10:00:00 |