

|  |
| --- |
| Business Template  **Social media** |
| **Logo / Image** |

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

## Business background

This project involves creating a social media platform where users can interact by posting content, liking, sharing, commenting, and messaging. The platform focuses on connecting users with various social networking functionalities such as user profiles, posts, friendships, and hashtags.

## Problems. Current Situation

Currently, social media platforms have to handle large amounts of data related to users, posts, interactions, and privacy settings. The lack of a structured database can lead to issues in managing relationships between these elements efficiently.

## the Benefits of implementing a database. Project Vision

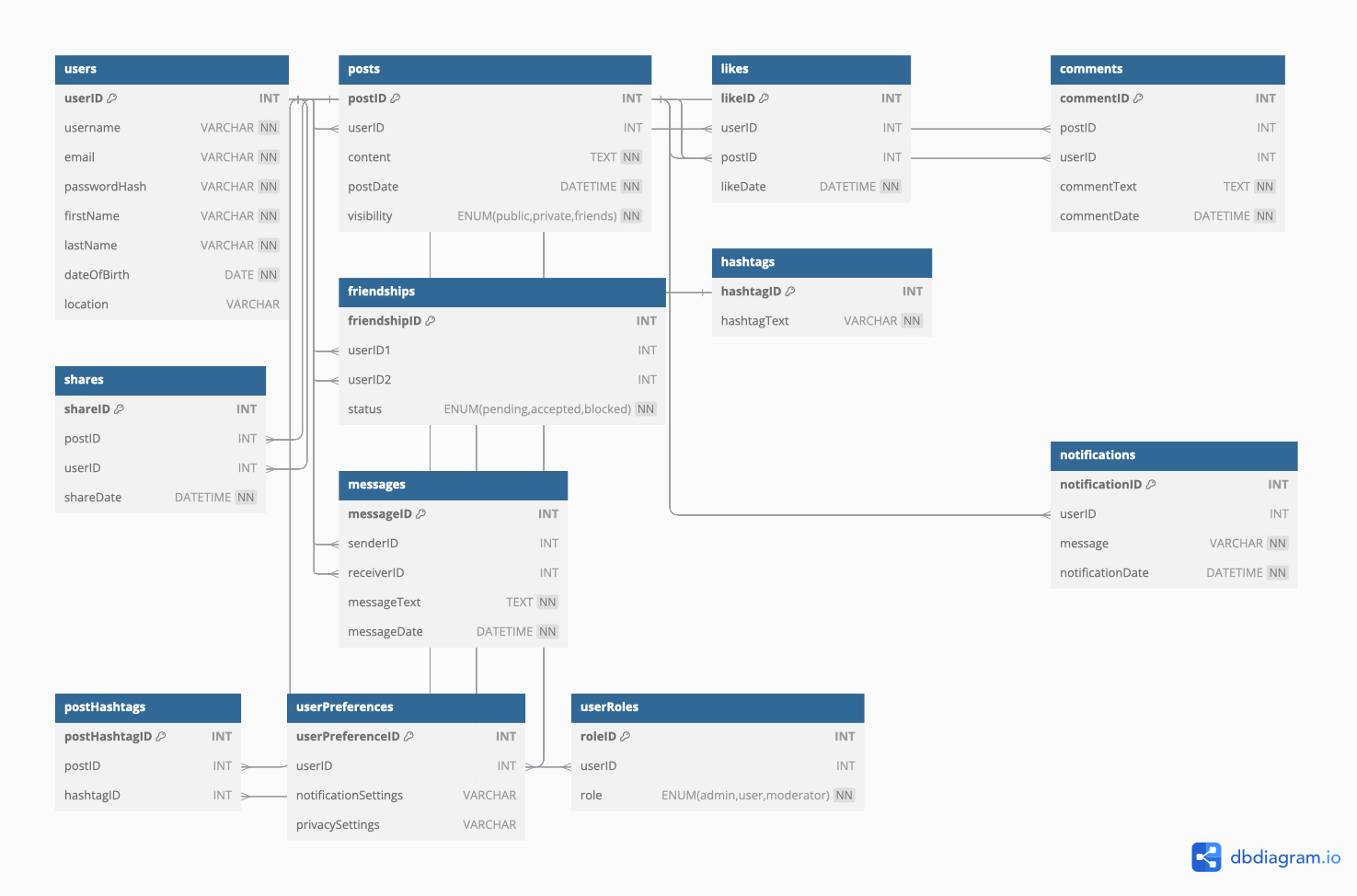
Implementing this database will allow the platform to handle user data securely and efficiently. It will enable proper relationships between users, posts, likes, shares, comments, and notifications. The project aims to create a scalable, normalized, and optimized database that supports the growth of the platform.

# Model description

## Definitions & Acronyms

The database is designed in 3rd normal form and includes relationships between users, posts, interactions (likes, shares, comments), hashtags, friendships, preferences, and notifications.

## Logical Scheme



## Objects

Table Description

Stores information about the users of the social media platform

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Users Table | userID | Unique identifier for each user, PK | Int, Auto\_Increment |
| username | Unique username for each user | Varchar 255, Unique, Not Null |
| email | User's email address | Varchar 255, Unique, Not Null |
| passwordHash | Hashed password for security | Varchar 255, Not Null |
| firstName | User's first name | Varchar 100, Not Null |
| lastName | User's last name | Varchar 100, Not Null |
| dateOfBirth | User's date of birth | Date, Not Null |
| location | User's location | Varchar 255, Null |

Comments on table relationships

One-to-Many: A single user can have multiple posts, likes, comments, shares, friendships, messages, preferences, roles, and notifications.

Primary Key (userID) is referenced by foreign keys in the Posts, Likes, Comments, Shares, Friendships, UserPreferences, Messages, UserRoles, and Notifications tables.

Example with data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| userID | username | email | passwordHash | firstName | lastName | dateOfBirth | location |
| 1 | gigi.ge | gigi@gmail.com | $2y$10$examplehashedpassword | Giorgi | Gagoshidze | 1990-05-12 | Tbilisi, Georgia |

Table Description

Contains all posts made by users.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Post Table | postID | Unique identifier for each post. PK | Int, Auto\_Increment |
| userID | ID of the user who created the post FK | Int, Not Null, references Users |
| content | The content of the post | Text, Not Null |
| postDate | The date and time the post was made | Enum, Not Null |
| visibility | Post visibility (Public, Private, Friends) | Enum, Not Null |

Comments on table relationships

One-to-Many: Each post can have multiple likes, comments, and shares.

Foreign Key (userID) references the Users table to link a post to its creator.

Primary Key postID) is referenced by foreign keys in the Likes, Comments, Shares, and PostHashtags tables.

Example with data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| posID | userID | content | postDate | visibility |
| 1 | 1 | "Hello, world!" | 2024-10-21 10:00:00 | Public |

Table Description

Records the likes that users give to posts.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Likes Table | likeID | Unique identifier for each like. PK | Int, Auto\_Increment |
| userID | ID of the user who liked the post. FK | Int, Not Null, references Users |
| postID | ID of the post being liked. FK | Int, Not Null, reference Posts |
| likeDates | The date and time the post was liked | Datetime, Not Null |

Comments on table relationships

Many-to-One: Each like is associated with a single post and a single user, but a user can like multiple posts, and a post can be liked by multiple users.

Composite Unique Key (userID, postID) ensures that a user can like a post only once.

Example with data

|  |  |  |  |
| --- | --- | --- | --- |
| likeID | userID | postID | likeDates |
| 1 | 1 | 1 | 2024-10-21 10:05:00 |

Table Description

Stores comments made by users on posts.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Comments Table | commentID | Unique identifier for each comment. PK | Int, Auto\_Increment |
| postID | ID of the post being commented on. FK | Int, Not Null, reference Posts |
| userID | ID of the user who commented. FK | Int, Not Null, references Users |
| commentText | The text of the comment | Text, Not Null |
| commentDate | The date and time the comment was made | DateTime |

Comments on table relationships

Many-to-One: Each comment is linked to a single post and a single user, but a user can comment on multiple posts, and a post can have multiple comments.

The Foreign Key (postID) links to the Posts table, and the Foreign Key (UserID) links to the Users table.

Example with data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| commentID | postID | userID | commentText | commentDate |
| 1 | 1 | 1 | This is my first post! | 2024-10-21 10:10:00 |

Table Description

Records when users share posts.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Share Table | shareID | Unique identifier for each share. PK | Int, Auto\_Increment |
| postID | ID of the post being shared. FK | Int, Not Null, references Posts |
| userID | ID of the user who shared the post. FK | Int, Not Null, references Users |
| shareDate | The date and time the post was shared | Datetime, Not Null |

Comments on table relationships

Many-to-One: Each share is associated with a single post and a single user, similar to likes and comments.

The Foreign Key (postID) links to the Posts table, and the Foreign Key (UserID) links to the Users table.

Example with data

|  |  |  |  |
| --- | --- | --- | --- |
| shareID | postID | userID | shareData |
| 1 | 1 | 1 | 2024-10-21 10:15:00 |

Table Description

Manages friendship relationships between users.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Friendships Table | friendshipID | Unique identifier for each friendship. PK | Int, Auto\_Increment |
| userID1 | First user in the friendship. FK | Int, Not Null, references Users |
| userID2 | Second user in the friendship. FK | Int, Not Null, references Users |
| status | Status of the friendship | Enum, Not Null |

Comments on table relationships

Many-to-Many: This table establishes a relationship between two users. Each friendship consists of two users (userID1 and userID2), and both can have multiple friendships with different users.

Composite Unique Key (userID1, userID2) ensures a unique relationship between two users.

Example with data

|  |  |  |  |
| --- | --- | --- | --- |
| friendshipID | userID1 | userID2 | status |
| 1 | 1 | 2 | Accepted |

Table Description

Stores hashtags used in posts.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Hashtag Table | hashtagID | Unique identifier for each hashtag. PK | Int, Auto\_Increment |
| hashtagText | The hashtag text | Varchar 255, Unique, Not Null |

Comments on table relationships

One-to-Many: Each hashtag can be associated with multiple posts.

The Primary Key (hashtagID) is referenced by the PostHashtags table.

Example with data

|  |  |
| --- | --- |
| hashtagID | hashtagText |
| 1 | #HelloWorld |

Table Description

Links posts to hashtags.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| PostHashtags Table | postHashtagID | Unique identifier for each post-hashtag relationship. PK | Int, Auto\_Increment |
| postID | ID of the post. FK | Int, Not Null, references Posts |
| hashtagID | ID of the hashtag. FK | Int, Not Null, references Hshtags |

Comments on table relationships

Many-to-One: This is a junction table that links posts and hashtags. Each record in this table associates a single post with a single hashtag.

Composite Unique Key (postID, hashtagID) ensures a unique relationship for hashtags associated with a post.

Example with data

|  |  |  |
| --- | --- | --- |
| postHashtagID | postID | hashtagID |
| 1 | 1 | 1 |

Table Description

Stores user-specific preferences.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| UserPreferences Table | userPerferenceID | Unique identifier for each user's preferences | Int, Auto\_Increment |
| userID | ID of the user | Int, Not Null, references Users |
| notificationSettings | User's notification settings | Varchar 255, Null |
| privacySettings | User's privacy settings | Varchar 255, Null |

Comments on table relationships

One-to-One: Each user can have one set of preferences.

The Foreign Key (userID) links to the Users table, ensuring that preferences are specific to each user.

Example with data

|  |  |  |  |
| --- | --- | --- | --- |
| userPreferenceID | userID | notificationSettings | privacySettings |
| 1 | 1 | "Email" | "Friends Only" |

Table Description

Records messages sent between users.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Messages Table | messageID | Unique identifier for each message. PK | Int, Auto\_Increment |
| senderID | ID of the user who sent the message. FK | Int, Not Null, references Users |
| receiverID | ID of the user who received the message. FK | Int, Not Null, references Users |
| messageText | The text of the message | Text, Not Null |
| messageDate | The date and time the message was sent | Datetime, Not Null |

Comments on table relationships

Many-to-One: Each message is sent from one user to another user. A user can send and receive multiple messages.

The Foreign Key (senderID) and Foreign Key (receiverID) both link to the Users table.

Example with data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| messageID | senderID | receverID | messagetext | messageDate |
| 1 | 1 | 2 | "Hi, how are you?" | 2024-10-21 10:20:00 |

Table Description

Manages user roles within the platform.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| UserRoles Table | roleID | Unique identifier for each user role | Int, Auto\_Increment |
| userID | ID of the user | Int, Not Null, references Users |
| role | User role | Enum, Not Null |

Comments on table relationships

One-to-One: Each user can have only one role at a time (e.g., Admin, User, Moderator).

The Foreign Key (userID) links to the Users table, ensuring each user is assigned a specific role.

Example with data

|  |  |  |
| --- | --- | --- |
| roleID | userID | role |
| 1 | 1 | User |

Table Description

Records notifications sent to users.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Notifications Table | notificationID | Unique identifier for each notification. PK | Int, Auto\_Increment |
| userID | ID of the user receiving the notification. FK | Int, Not Null, references Users |
| message | The content of the notification | Varchar 255, Nut Null |
| notificationDate | The date and time the notification was created | Datetime, Not Null |

Comments on table relationships

One-to-Many: A user can receive multiple notifications.

The Foreign Key (UserID) links to the Users table to associate notifications with the respective user.

Example with data

|  |  |  |  |
| --- | --- | --- | --- |
| notificationID | userID | message | notificationDate |
| 1 | 1 | "You have a new friend request." | 2024-10-21 10:25:00 |