

|  |
| --- |
| Business Template  **Subject areas** |
| **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

Social media platforms are an essential part of daily life, that’s why it’s important to make sure users have better experience and can freely express themselves.

This project focuses on designing a structured and efficient database that will support our goals.

## Problems. Current Situation

Without a structured database, social media platforms will face several challenges, including:

* Data redundancy and inconsistency
* Slow query performance
* Lack of structured relationships
* Limited personalization

## the Benefits of implementing a database. Project Vision

The goal is to create a database for social media that enables

* efficient data retrieval
* elimination of redundancy
* support for future scalability

This ensures that social media platforms will remain optimized, responsive and adaptable.

# Model description

## Definitions & Acronyms

## Logical Scheme

A screenshot of a computer

AI-generated content may be incorrect.

## Objects

Table: User

Stores user account details.

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| UserID | PK, unique identifier for user | Bigint |
| Username | Unique user’s chosen name | Varchar(255) |
| Email | Unique user’s chosen email | Varchar(255) |
| Password | User’s password | Varchar(255) |

Table: Post

Stores posts made by user

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| PostID | PK, unique identifier for a post | Bigint |
| UserID | FK, user who made the post | Bigint |
| Content | The text content of the post | Text |

Table: Like

Tracks user likes on posts.

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| LikeID | PK, unique identifier for a like | Bigint |
| UserID | FK, user who liked the post | Bigint |
| PostID | FK, the post that is being liked | Bigint |

Table: Comment

Stores user comments on posts.

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| CommentID | PK, unique identifier for a comment | Bigint |
| UserID | FK, user who liked the post | Bigint |
| PostID | FK, the post that is being liked | Bigint |

Table: Share

Tracks users sharing posts.

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| ShareID | PK, unique identifier for a share | Bigint |
| UserID | FK, user who liked the post | Bigint |
| PostID | FK, the post that is being liked | Bigint |

Table: Hashtag

Stores hashtags used in posts.

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| HashtagID | PK, unique identifier for a hashtag | Bigint |
| Hashtag | Hashtag text | Varchar(255) |

Table: PostHashtag

Links hashtags with posts

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| PostHashtagID | PK, unique identifier for the relationship | Bigint |
| PostID | FK, related post | Bigint |
| HashtagID | FK, related hashtag | Bigint |

Table: FriendshipBridge

Stores user friendships

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| FriendshipBridgeID | PK, unique identifier for the friendship | Bigint |
| User1ID | FK, first user in the friendship | Bigint |
| User2ID | FK, second user in the friendship | Bigint |
| StatusID | Friendship status | Bigint |

Table: Friendship

Defines different types of user relationships

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| StatusID | PK, unique identifier for status | Bigint |
| StatusName | Status type | Varchar(255) |

Table: Geolocation

Tracks user location.

|  |  |  |
| --- | --- | --- |
| Field name | Field Description | Data Type |
| LocationID | PK, unique identifier for a location | Bigint |
| UserID | FK, related user | Bigint |
| Latitude | User’s latitude coordinate | Decimal(9,6) |
| Longitude | User’s longitude coordinate | Decimal(9,6) |

Most relationships in the database follow a one-to-many (1:N) structure, ensuring efficient data organization. However, there are two key many-to-many (N:M) relationships:

* **User and Friendships**: A user can have multiple friendships, and each friendship involves two users. To manage this, a bridge table (FriendshipBridge) is used to track relationships and their statuses.
* **Post and Hashtag**: A single post can include multiple hashtags, while each hashtag can be associated with multiple posts. This many-to-many relationship is implemented using a junction table (PostHashtag) to efficiently store and retrieve hashtag associations.

Example with data

User Table

For user safety these passwords should be hashed but for simplicity they are represented as text.

|  |  |  |  |
| --- | --- | --- | --- |
| UserID | Username | Email | Password |
| 1 | John | John@email.com | 1234 |
| 2 | Joe | [Joe@email.com](mailto:Joe@email.com) | Password |
| 3 | Jane | [Jane@email.com](mailto:Jane@email.com) | Password123 |

Post Table

|  |  |  |
| --- | --- | --- |
| PostID | UserID | Content |
| 101 | 2 | I’m going out tonight! |
| 102 | 1 | Learning is fun. |
| 103 | 2 | Just got back. |

FriendshipStatus Table

|  |  |
| --- | --- |
| StatusID | StatusName |
| 1 | Friends |
| 2 | Pending invite |
| 3 | Blocked |
| 4 | Not friends |