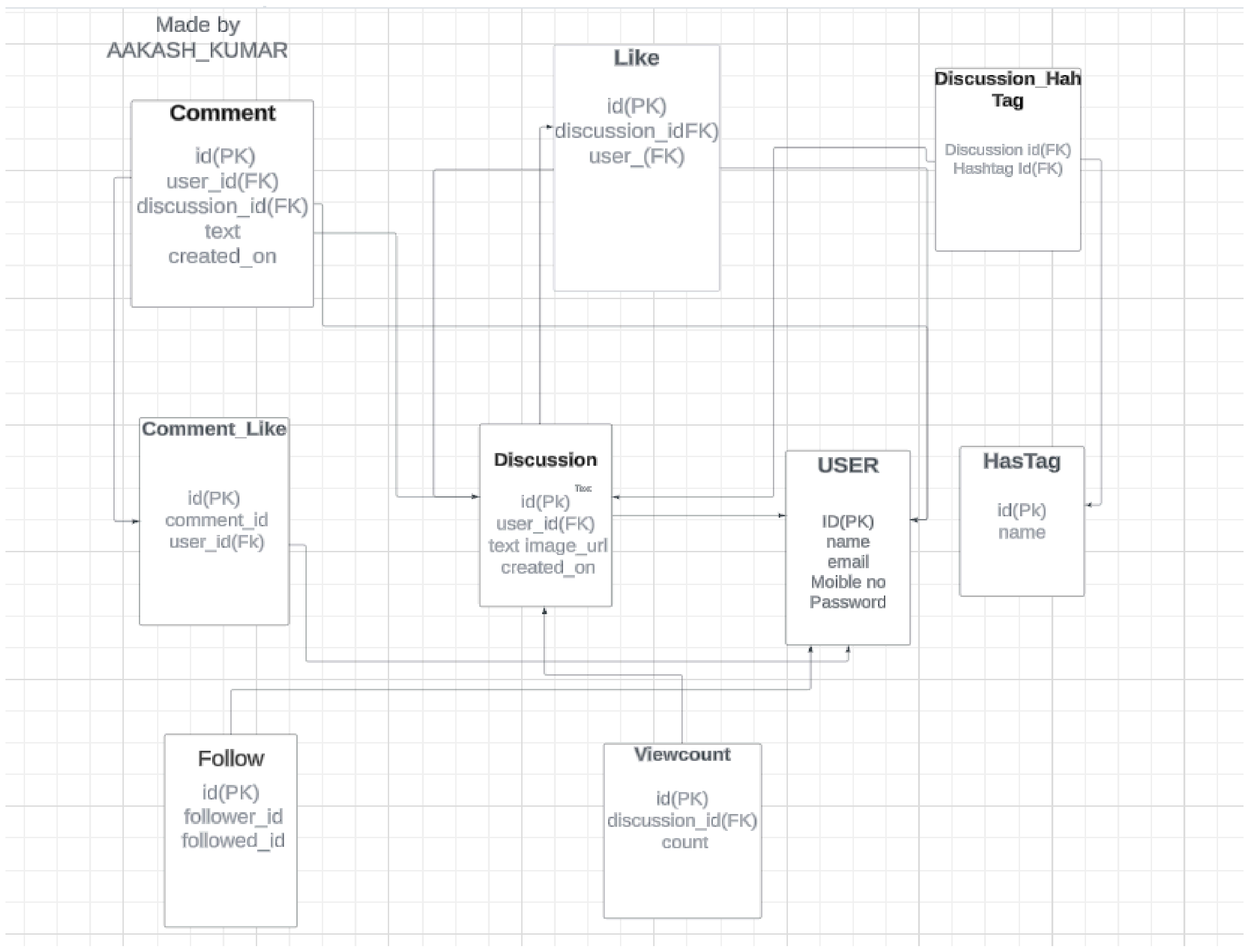
Made by AAKASH KUMAR

Database Schema Documentation

The database schema for the discussion application consists of several tables with relationships defined among them. Below are the table/collection definitions and relationships.



1. User Table

Table Name: `users`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the user.
- `name`: `VARCHAR(255) NOT NULL` - Name of the user.
- `mobile`: `VARCHAR(15) UNIQUE NOT NULL` - Mobile number of the user, unique for each user.
- `email`: `VARCHAR(255) UNIQUE NOT NULL` - Email address of the user, unique for each user.
- `password`: `VARCHAR(255) NOT NULL` - Hashed password of the user.

Relationships:

- One-to-Many relationship with `discussions` (a user can have multiple discussions).
- One-to-Many relationship with `comments` (a user can have multiple comments).
- One-to-Many relationship with `likes` (a user can like multiple discussions).
- One-to-Many relationship with `comment_likes` (a user can like multiple comments).
- Many-to-Many relationship with `users` through the `follows` table (users can follow each other).

2. Discussion Table

Table Name: `discussions`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the discussion.
- `user_id`: `BIGINT NOT NULL` - Foreign key referencing the `users` table.
- `text`: `TEXT NOT NULL` - Text content of the discussion.
- `image_url`: `VARCHAR(255)` - URL of the image associated with the discussion.
- `created_on`: `TIMESTAMP DEFAULT CURRENT_TIMESTAMP` - Timestamp of when the discussion was created.

Relationships:

- Many-to-One relationship with `users` (each discussion is posted by one user).
- Many-to-Many relationship with `hashtags` through the `discussion_hashtags` table (each discussion can have multiple hashtags).

- One-to-Many relationship with `comments` (a discussion can have multiple comments).
- One-to-Many relationship with `likes` (a discussion can have multiple likes).
- One-to-Many relationship with `view_counts` (a discussion can have multiple view counts).

3. HashTag Table

Table Name: `hashtags`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the hashtag.
- `name`: `VARCHAR(255) NOT NULL` - Name of the hashtag.

Relationships:

- Many-to-Many relationship with `discussions` through the `discussion_hashtags` table (each hashtag can be associated with multiple discussions).

4. DiscussionHashTag Table

Table Name: `discussion_hashtags`

Columns:

- `discussion_id`: `BIGINT NOT NULL` - Foreign key referencing the `discussions` table.
- `hashtag_id`: `BIGINT NOT NULL` - Foreign key referencing the `hashtags` table.

Relationships:

- Many-to-Many relationship between `discussions` and `hashtags`.

5. Comment Table

Table Name: `comments`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the comment.
- `discussion_id`: `BIGINT NOT NULL` - Foreign key referencing the `discussions` table.
- `user_id`: `BIGINT NOT NULL` - Foreign key referencing the `users` table.
- `text`: `TEXT NOT NULL` - Text content of the comment.

- `created_on`: `TIMESTAMP DEFAULT CURRENT_TIMESTAMP` - Timestamp of when the comment was created.

Relationships:

- Many-to-One relationship with `discussions` (each comment is associated with one discussion).
- Many-to-One relationship with `users` (each comment is posted by one user).
- One-to-Many relationship with `comment_likes` (a comment can have multiple likes).

6. Like Table

Table Name: `likes`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the like.
- `discussion_id`: `BIGINT NOT NULL` - Foreign key referencing the `discussions` table.
- `user_id`: `BIGINT NOT NULL` - Foreign key referencing the `users` table.

Relationships:

- Many-to-One relationship with `discussions` (each like is associated with one discussion).
- Many-to-One relationship with `users` (each like is given by one user).

7. CommentLike Table

Table Name: `comment_likes`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the comment like.
- `comment_id`: `BIGINT NOT NULL` - Foreign key referencing the `comments` table.
- `user_id`: `BIGINT NOT NULL` - Foreign key referencing the `users` table.

Relationships:

- Many-to-One relationship with `comments` (each like is associated with one comment).
- Many-to-One relationship with `users` (each like is given by one user).

8. Follow Table

Table Name: `follows`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the follow relationship.
- `follower_id`: `BIGINT NOT NULL` - Foreign key referencing the `users` table (the user who follows).
- `followed_id`: `BIGINT NOT NULL` - Foreign key referencing the `users` table (the user being followed).

Relationships:

- Many-to-One relationship with `users` (each follow relationship involves two users).

9. ViewCount Table

Table Name: `view_counts`

Columns:

- `id`: `BIGINT AUTO_INCREMENT PRIMARY KEY` - Unique identifier for the view count.
- `discussion_id`: `BIGINT NOT NULL` - Foreign key referencing the `discussions` table.
- `count`: `INT NOT NULL` - Number of views for the discussion.

Relationships:

- Many-to-One relationship with `discussions` (each view count is associated with one discussion).

ER Diagram

Below is the Entity-Relationship (ER) diagram illustrating the schema:

In the above diagram:

- A `User` can have multiple `Discussion`, `Comment`, `Like`, `Follow`, and `CommentLike`.
- A `Discussion` can have multiple `Comment`, `Like`, `ViewCount`, and can be associated with multiple `HashTag` through `DiscussionHashTag`.
- A `HashTag` can be associated with multiple `Discussion` through `DiscussionHashTag`.

- A `Comment` can have multiple `CommentLike`.

This schema ensures that the application can handle user management, discussions, comments, likes, follows, and view counts efficiently.