Database schema project 2

Database:

```
DROP TABLE IF EXISTS users;

CREATE TABLE IF NOT EXISTS users (

    first_name TEXT,

    last_name TEXT,

    username TEXT,

    email_address TEXT,

    hashed_pass TEXT,

    salt TEXT,

    moderator INTEGER,

    PRIMARY KEY (username, email_address),

    UNIQUE (username)

);


DROP TABLE IF EXISTS passwords;

CREATE TABLE IF NOT EXISTS passwords(

    user_name TEXT PRIMARY KEY,

    previous_hashed_pass TEXT,

    FOREIGN KEY (user_name) REFERENCES users(username)

        ON DELETE CASCADE ON UPDATE CASCADE

);
```

```
DROP TABLE IF EXISTS posts;

CREATE TABLE IF NOT EXISTS posts(

   title TEXT,

   body TEXT,

   post_id INTEGER PRIMARY KEY,

   owner TEXT,

   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (owner) REFERENCES users(username)

      ON DELETE CASCADE ON UPDATE CASCADE

);


DROP TABLE IF EXISTS tags;

CREATE TABLE IF NOT EXISTS tags(

   post_id INTEGER,

   tags TEXT,

   FOREIGN KEY (post_id) REFERENCES posts(post_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE IF NOT EXISTS follows (

   follower_username TEXT NOT NULL,

   followed_username TEXT NOT NULL,
    PRIMARY KEY (follower_username, followed_username),

    FOREIGN KEY (follower_username) REFERENCES users(username)

        ON DELETE CASCADE ON UPDATE CASCADE,

    FOREIGN KEY (followed_username) REFERENCES users(username)

        ON DELETE CASCADE ON UPDATE CASCADE

);


CREATE TABLE IF NOT EXISTS likes (

    post_id INTEGER NOT NULL,

    username TEXT NOT NULL,

    PRIMARY KEY (post_id, username),

    FOREIGN KEY (post_id) REFERENCES posts(post_id)

        ON DELETE CASCADE ON UPDATE CASCADE,

    FOREIGN KEY (username) REFERENCES users(username)

        ON DELETE CASCADE ON UPDATE CASCADE

);
```

Explanation:

Users Table: The Users table serves as the central table in this database design. This table stores information about the users in terms of their first and last names, usernames, email addresses, hashed passwords, salts, and flags on whether they are moderators. The table uses a composite primary key, containing both username and email_address, making both fields together unique for the whole table. Also, the username field is uniquely indexed on its own to ensure no two users share a username.

Passwords Table: The password table stores the password history, if necessary, for the implementation of certain security features such as preventing the reuse of recent passwords. It engages with the users table through the field username, acting as a foreign key connected to the username of the users table. The ON DELETE CASCADE and the ON UPDATE CASCADE rules ensure that any change in the users table reflects here so that the referential integrity is maintained.

Posts Table: It stores user-generated content data. Each post is uniquely defined with a post_id, title of the post, body text, and metadata about who the owner of the post is and what time the post was created. The owner field in this table is a foreign key that points to the username field of the users table, connecting each post with its user. The cascading options on this foreign key keep the integrity of it, such that when a user is deleted, his posts will be deleted as well.

Tags Table: This table is utilized to tag posts, therefore assisting in the categorization of the content. The post_id field links back to the posts table so there can be more than one tag per post. Because of the cascade rules, upon deletion or modification of a tag, changes reflect further with the associated posts.

Follows Table: This table manages the relationship between the users in terms of "following" one another. Every record of this table would represent one-way follow relationships. Primary key will be composite comprising follower_username and followed_username, both foreign keys referencing users' table's username with update and delete cascades for integrity.

Likes Table: It keeps track of the users, and which posts they have liked, making the post_id and username composite, primary keys. This allows the table never to instance a user liking the same post more than once. Foreign keys create relationships with a given like attached to a specific post and user; this cascade on deletes or updates will remove extra likes should a post or user be deleted.