Web Technologies and Security

[CSC-40046-2023-SEM2]

Report

Name of Student – Rafiqkhan Ayubkhan Lohani

Student Id – 23018024

Student Mail – y0y93@students.keele.ac.uk

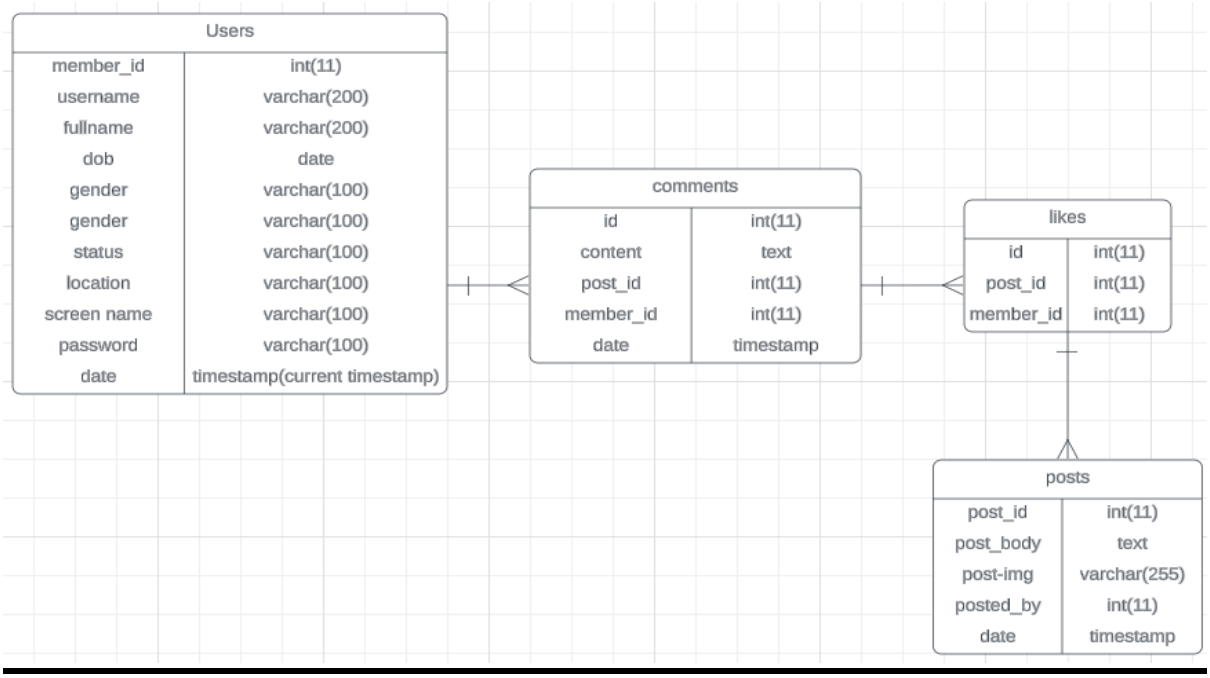
Contact Number – 07748977557

- **Entity Relationship Diagram.**

  An entity relationship diagram (ERD) is a visual representation for the between entities and the database. It helps to illustrate the structure of the database by showing how data entities relate to each other and the attributed associated with each entity.

  In the context of the social network it will help to typically include entities such as users, posts, comments, messages and the relationship between them, here is the simplified of social network website.

- **Users:** This entity represents the users of the social networking website. It would include as User ID, username, email, password, profile picture, etc.
- **Likes:** It would allow to like the post or the content which is been updated or been uploaded by the user or any other user.
- **Posts:** This entity represents the posts created by users on the website. It would include attributes such as post Id, content, timestamp, user ID.
- **Comments:** This entity represents the comments made by users on the post. It would include attributes such as comment id, content, timestamp, user Id, Post id etc.
- **Messages:** This entity represent the private messages exchanged between the users on the website. It would include attribute such as message id, content, timestamp, sender id, receiver id.



- **Site Map**

  A site map that shows the structure of our sites and details and the filenames of all the pages we plan to create and how relate to each other. This does not show all the included php files. But it should show all the pages a user can visit and they are interlinked.

Social Network/

- Social Network
- Javascript
- ----------script.js
- Img
- ----------head_sun_flower.png
- Include
- ----------foot.php
- ----------head.php
- db-server.php
- errors.php
- home.php
- index.php
- my-posts.php
- profile.php
- register.php

- **<u>SQL Queries</u>**

- **Users Table**

```
CREATE TABLE `users` (
  `member_id` int(11) NOT NULL,
  `username` varchar(100) NOT NULL,
  `full_name` varchar(100) NOT NULL,
  `screen_name` varchar(100) NOT NULL,
  `dob` date NOT NULL,
  `gender` varchar(30) NOT NULL,
  `status` varchar(100) NOT NULL,
  `location` varchar(100) NOT NULL,
  `password` varchar(255) NOT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

- **Posts Table**

```
CREATE TABLE `posts` (
  `post_id` int(11) NOT NULL,
```

```
  `post_body` text NOT NULL,
  `post_img` varchar(255) NOT NULL,
  `posted_by` int(11) NOT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

- **Comments Table**

```
CREATE TABLE `comments` (
  `id` int(11) NOT NULL,
  `content` text NOT NULL,
  `post_id` int(11) NOT NULL,
  `member_id` int(11) NOT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

- **Likes Table**

```
CREATE TABLE `likes` (
  `id` int(11) NOT NULL,
  `post_id` int(11) NOT NULL,
  `member_id` int(11) NOT NULL
);
```

Security Concern

MySQL remains a widely used database for web development since its introduction in mid 90s compared to the other SQL and non-SQL databases. MYSQL offers advantage like user friendliness, adaptability, scalability, and cost-free licensing.

Common Security Risks in MySQL:

• Mismanagement of Account Access: Assigning inappropriate account privileges is a significant risk. Users should not be granted excessive permissions, such as root access, which can lead to severe data damage if exploited by malicious actors. Best practices include granting minimal necessary privileges to users and restricting access to critical features like the user table to root accounts only.

• Weak Passwords: Using strong passwords is imperative to prevent unauthorized access. Enforcing password strength requirements for user accounts helps mitigate the risk of breaches and data loss.

• SQL Injection Attacks: SQL injection attacks pose a prevalent threat where attackers insert malicious queries/scripts into the database, potentially compromising data integrity. Mitigation involves implementing proper data validation mechanisms for input received via forms.

• Use of Insecure URLs: Insecure URLs can be exploited by attackers to inject malicious scripts into the database. Utilizing secure submission methods, such as POST instead of GET, is essential to mitigate vulnerabilities and prevent potential exploits.

• Weak Audit Trails: Inadequate auditing increases the risk of security breaches and poses significant challenges in maintaining control over database security.

Additional security threats include exposure of database backups, malware, ineffective permissions management, misconfigurations, privilege threats, accessible backups, and credential threats.

Protective Measures for Databases:
To mitigate database security risks, implement the following strategies:

• Manage user access rights by assigning appropriate permissions, reducing privileges, and deactivating inactive accounts.
• Employ measures to block malicious web requests.
• Mask sensitive data fields to prevent unauthorized access.
• Stay informed about database vulnerabilities and implement necessary patches and updates.
• Archive external data securely.
• Manage user access rights by assigning appropriate permissions, reducing privileges, and deactivating inactive accounts
• Monitor database access and usage patterns in real-time.
• Provide employees with training on the latest risk mitigation techniques.

Credentials

Website Link – teach.scam.keele.ac.uk/msc/y0y93

Database username – y0y93

Database password – y0y93y0y93

Test User Details – rafiq@786

Test User Password – Momdad@786