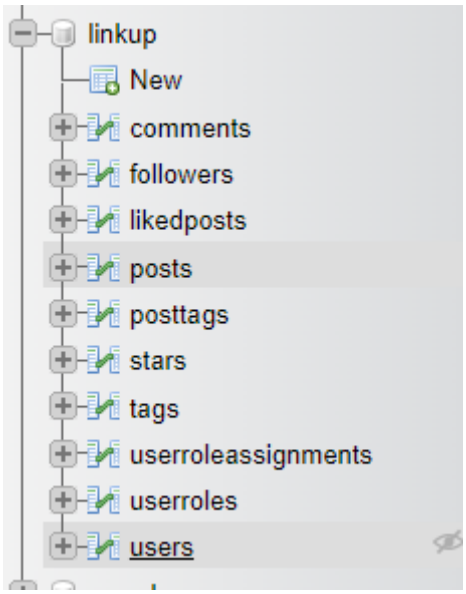
LinkUp

Deacon Mizzi

Task 3

Database Setup

The database for the LinkUp website was set up using phpMyAdmin via the XAMPP application. The database LinkUp consists of several tables that are essential for the platform's functionality. These tables include "users", "posts", "comments"," tags", "posttags", "stars", "followers", "likedposts", "userroleassignments", "userroles".



Each table has its own specific fields and constraints to ensure integrity and efficient data management. As an example, the table "users" includes fields such as "user_id" (Primary Key), "username" (VARCHAR), "email" (VARCHAR), "password" (VARCHAR), "profile_picture" (VARCHAR), and "bio" (TEXT).

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	user_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	username	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
3	email	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
4	password	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	bio	text	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	profile_picture	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Other key tables include:

- posts: Stores the posts created by users.
    - post_id: Primary key, auto-incremented integer.
    - user_id: Foreign key referencing users(user_id).
    - title: A string up to 255 characters.
    - content: Text field for the post content.
    - image_path: A string to store the path to the post's image.
- comments: Stores comments on posts.
    - comment_id: Primary key, auto-incremented integer.
    - post_id: Foreign key referencing posts(post_id).
    - user_id: Foreign key referencing users(user_id).
    - content: Text field for the comment content.
- followers: Manages follower-followee relationships.
    - id: Primary key, auto-incremented integer.
    - follower_id: Foreign key referencing users(user_id).
    - followee_id: Foreign key referencing users(user_id).
- likedposts: Tracks which posts are liked by which users.
    - like_id: Primary key, auto-incremented integer.
    - user_id: Foreign key referencing users(user_id).
    - post_id: Foreign key referencing posts(post_id).
- stars: Tracks star ratings for posts.
    - star_id: Primary key, auto-incremented integer.
    - post_id: Foreign key referencing posts(post_id).
    - user_id: Foreign key referencing users(user_id).
- tags: Stores tags for categorizing posts.
    - tag_id: Primary key, auto-incremented integer.
    - tag_name: A unique string.
- posttags: Associates tags with posts.
    - post_id: Foreign key referencing posts(post_id).
    - tag_id: Foreign key referencing tags(tag_id).
- userroles: Defines different user roles.
    - role_id: Primary key, auto-incremented integer.
    - role_name: A unique string.
- userroleassignments: Associates users with roles.
    - assignment_id: Primary key, auto-incremented integer.

- user_id: Foreign key referencing users(user_id).
- role_id: Foreign key referencing userroles(role_id).

While foreign keys ensure referential integrity by enforcing relationships between tables, primary keys guarantee that each entry may be uniquely identified. To improve query efficiency, indexes are applied to columns that are often requested. As an example of foreign keys, posttags has foreign keys referencing posts and tags.

The database was set up using MySQL, and tables were created with appropriate data types and constraints to ensure data integrity. Relationships between tables were established using foreign keys.

Techniques for Data Manipulation

Prepared Statements: Used to prevent SQL injection and ensure secure database operations. For example, when inserting or updating user data, prepared statements are used to sanitize inputs.

Session Management: Sessions are used to manage user authentication, ensuring that users stay logged in across different pages and actions. Session data is used to verify user identity and permissions.

File Handling: For profile pictures and post images, file upload handling is implemented. The files are validated, renamed to avoid conflicts, and stored in the designated directories.

HTML Forms and PHP: HTML forms are used for user input across various functionalities, such as registration, login, post creation, and comments. PHP scripts handle form submissions, process data, and interact with the database to perform CRUD operations.

Local Server Setup

The application is developed and tested locally using XAMPP, which provides an Apache server, MySQL database, and PHP environment. This setup mimics a live server environment, allowing for thorough testing and debugging before deployment. XAMPP is configured to run Apache and MySQL services, enabling the local execution of PHP scripts and database interactions.

Dynamic Web Application Techniques

HTML Forms: Used for user input across various functionalities, such as registration, login, post creation, and comments.

PHP: Server-side scripting language used for handling form submissions, processing data, and interacting with the database. PHP scripts manage CRUD operations and session handling.

JavaScript: Enhances user experience by adding client-side validation and dynamic interactions without needing a page reload.

CSS: Ensures a responsive and visually appealing design across all pages.

CRUD Operations

The website performs various CRUD (Create, Read, Update, Delete) operations to manage data within the database effectively.

1. Users

Create:

- **Register User**: In the register.php file, when a new user registers, their details (username, email, password) are saved into the users table. Passwords are hashed for security.

Read:

- **Fetch User Details**: In various files like profile.php, user details are fetched based on the user_id to display profile information, including username, email, bio, and profile picture.

Update:

- **Update Profile**: In the update_profile.php file, users can update their profile information such as username, bio, and profile picture.
- **Change Password**: In the change_password.php file, users can update their password.

Delete:

- **Delete User**: In the delete_user.php file, users can delete their account. Admins and moderators can also delete user accounts.

2. Posts

Create:

- **Create Post**: In the create_post.php file, users can create new posts. This includes adding the title, content, and associated tags. Images can also be uploaded and associated with posts.

Read:

- **Fetch Posts:** On the index.php page, all posts are fetched and displayed. Posts are also fetched on the profile.php page to show user's liked posts.

Delete:

- Delete Post: In the delete_post.php file, Admins and moderators can delete posts.

3. Comments

Create:

- **Add Comment:** In the add_comment.php file, users can add comments to posts. These comments are saved in the comments table.

Read:

- **Fetch Comments**: Comments are fetched and displayed below the posts on the post.php page.

Delete:

- **Delete Comment:** In the delete_comment.php file, Admins and moderators can delete comments.

4. Tags

Create:

- **Add Tag:** In the add_tag.php file, new tags can be created and added to the tags table.

Read:

- **Fetch Tags**: Tags are fetched in the index.php page to be displayed and selected when creating or editing posts.

Update:

- **Update Tag**: Tags can be updated in the update_tag.php file.

Delete:

- **Delete Tag**: In the delete_tag.php file, tags can be deleted.

5. User Roles

Create:

- **Assign Role**: In the assign_role.php file, roles can be assigned to users and stored in the userroleassignments table.

Read:

- **Fetch Roles**: Roles are fetched to check user permissions and displayed on user profiles.

Update:

- **Update Role**: User roles can be updated in the update_role.php file.

Delete:

- **Remove Role:** In the remove_role.php file, roles can be removed from users.

6. Followers

Create:

- **Follow User**: In the follow.php file, users can follow other users. This relationship is stored in the followers table.

Read:

- **Fetch Followers and Following**: Followers and following lists are fetched and displayed on the profile.php page.

Delete:

- **Unfollow User**: In the unfollow.php file, users can unfollow other users.

7. Likes (Stars)

Create:

- **Like Post**: In the add_star.php file, users can like (star) posts. This relationship is stored in the stars table.

Read:

- **Fetch Likes**: The number of likes is fetched and displayed next to each post on the index.php and post.php pages.

Delete:

- **Unlike Post:** In the remove_star.php file, users can remove their like from a post.

Test Cases

Test Case ID	Title	Description	Preconditions	Test Steps	Expected Result	Actual Result	Status
TC01	User Registration Form Validation	Verify that the user registration form validates input fields correctly	The registration page is accessible.	1.Navigate to the registration page 2. Enter a valid email address. 3. Enter a valid username. 4. Enter a valid password. 5. Confirm the password. 6. Click "Register".	The user account is created successfully, and user is taken to log in.	The user account is created successfully, and user is taken to log in.	Pass
TC02	User Registration Form Error Handling	Verify that the user registration form displays error messages for invalid inputs.	The registration page is accessible.	1. Leave the email field empty and click "Register". 2. Enter an invalid email format and click "Register". 3. Leave the username field empty and click "Register". 4. Enter a password that does not meet complexity requirements and click "Register".	Appropriate error messages are displayed for each invalid input.	Appropriate error messages are displayed for each invalid input.	Pass

				5. Enter a different password in the confirm password field and click "Register".			
TC03	User Login Validation	Verify that the login form validates user credentials correctly.	The login page is accessible.	1. Navigate to the login page. 2. Enter a valid email and password. 3. Click the "Login" button.	The user is logged in successfully and redirected to the index.	The user is logged in successfully and redirected to the index.	Pass
TC04	User Login Error Handling	Verify that the login form displays error messages for incorrect credentials	The login page is accessible.	1. Navigate to the login page. 2. Enter an invalid email and password. 3. Click the "Login" button.	Resets the fields.	Resets the fields.	Pass
TC05	Post Creation	Verify that a user can create a post successfully.	The user is logged in and on the home page.	1. Enter a title for the post. 2. Enter content for the post. 3. Click the "Post" button.	The post is created and displayed on the home page.	The post is created and displayed on the home page.	Pass
TC06	Commenting on a Post	Verify that a user can comment on a post successfully.	The user is logged in and viewing a post.	1.Enter a comment in the comment field. 2.Click the "Post Comment" button.	The comment is added and displayed below the post.	(The comment is added and displayed below the post.	Pass
TC07	Commenting Error Handling	Verify that the commenting	The user is logged in and	1.Leave the comment field empty.	An error message is displayed	An error message is displayed	Pass

		form displays an error message for an empty comment field.	viewing a post.	2.Click the "Post Comment" button.	indicating that the comment field cannot be empty.	indicating that the comment field cannot be empty.	
TC08	Liking a Post	Verify that a user can like a post successfully.	The user is logged in and viewing a post.	Click the "Star" button on a post.	The star count for the post increments by one.	The star count for the post increments by one.	Pass
TC09	Unliking a Post	Verify that a user can unlike a post successfully.	The user is logged in and has already liked the post.	Click the "Star" button on a post.	The star count for the post lowers by one.	The star count for the post lowers by one.	Pass
TC10	View Profile	Verify that a user can view their profile information.	The user is logged in.	Click on the "Profile" link.	The user's profile information is displayed.	The user's profile information is displayed.	Pass
TC11	Edit Profile	Verify that a user can edit their profile information.	The user is logged in and viewing their profile.	1.Click the "Edit Profile" button. 2.Change the username and bio. 3.Click the "Save" button.	The profile information is updated and displayed.	The profile information is updated and displayed.	Pass
TC12	Change Password	Verify that a user can change their password.	The user is logged in and viewing their profile.	1.Click the "Change Password" button. 2.Enter the current password. 3.Enter the new password. 4.Confirm the new password. 5.Click the "Save" button.	The password is changed successfully, and a confirmation message is displayed.	The password is changed successfully , and a confirmation message is displayed.	Pass

TC13	Delete Profile	Verify that a user can delete their profile.	The user is logged in and viewing their profile.	1.Click the "Delete Profile" button. 2.Confirm the deletion.	The profile is deleted, and the user is logged out and redirected to the login page.	The profile is deleted, and the user is logged out and redirected to the login page.	Pass
TC14	View FAQ	Verify that a user can view the FAQ page.	The user is logged in.	Click on the "FAQ" link.	The FAQ page is displayed fully,	The FAQ page is displayed fully	Pass
TC15	Contact Us Form Validation	Verify that the Contact Us form validates input fields correctly.	The Contact Us page is accessible.	1.Navigate to the Contact Us page. 2.Enter a valid email address. 3.Enter a message. 4.Click the "Send" button.	The message is sent successfully, and a confirmation message is displayed.	The message is sent successfully , and a confirmation message is displayed.	Pass
TC16	Contact Us Form Error Handling	Verify that the Contact Us form displays error messages for missing fields.	The Contact Us page is accessible.	1.Leave the email field empty and click "Send". 2.Leave the message field empty and click "Send".	Appropriate error messages are displayed for each missing field.	Appropriate error messages are displayed for each missing field.	Pass