SocialHive: Technical Document

By Elisea Mizzi

Introduction

SocialHive is a social media platform built using PHP and MySQL. It is designed to help people connect, share content, and engage in online communities. This technical document explains how I set up the database, built the application, configured a local virtual server, and ensured everything worked smoothly through thorough testing.

Database Setup

Structure and Design

The database is the backbone of SocialHive, and it's important to set it up correctly. I started by designing it with multiple tables, each focused on a specific function:

1. Users Table: Stores user information like username, email, password, bio, and profile picture. user_id is the primary key.
2. Posts Table: Manages posts created by users, with content, image, and timestamp. Each post is linked to a user via user_id.
3. Comments Table: Contains all comments, each linked to a post and a user.
4. Groups Table: Stores group details including name, description, and admin user.
5. Other Tables: These include Group_Members, Messages, Group_Messages, Follows, Notifications, and Settings, each with appropriate foreign keys to maintain relationships and ensure data integrity.

Referential Integrity and Constraints

To maintain accurate and consistent data, I applied foreign key constraints. For instance, each post references a valid user. Comments, messages, and group memberships are all tied to existing users or groups through foreign keys to avoid orphaned records.

Setting Up a Virtual Server Locally

To test SocialHive in a real environment, I used XAMPP, which bundles Apache, PHP, and MySQL.

Steps:

1. Installation: Installed XAMPP to get an integrated development environment.
2. Configuration: Edited Apache config files to serve my project directory and set up a virtual host.

3. Starting Services: Used XAMPP's control panel to start Apache and MySQL.
4. Database Setup: Created the MySQL database using phpMyAdmin.
5. Environment Variables: Stored sensitive settings like database credentials in an .env file for security and easy updates.

Building a Dynamic Web Application

To make the app dynamic and responsive, I used several key techniques:

1. Session Management: Users remain logged in across pages using PHP sessions, which store user IDs and related data.
2. Form Validation: Implemented both client-side (JavaScript) and server-side (PHP) validation to protect against incorrect or harmful input.
3. Bootstrap Framework: Designed mobile-friendly, clean UIs using Bootstrap components.
4. JavaScript Interaction: Used JavaScript to update parts of the page (e.g., adding a post or comment) without needing a full page reload.

Example: Creating a Post

- HTML Form: Collects the post content and any image.
- PHP Script: Validates and stores the post.
- JavaScript: Updates the page immediately to show the new post.

Data Manipulation Techniques

All CRUD operations are built with PHP and MySQL:

- Inserting Data: e.g., User registration adds a new row to the Users table using prepared statements.
- Reading Data: Used for user login and loading posts or comments.
- Updating Data: For example, users editing their profile info via an SQL UPDATE query.
- Deleting Data: Like removing a post or comment using DELETE queries.

I also used:

- Prepared Statements to prevent SQL injection.
- Transactions for multi-step operations that must succeed or fail together.
- Stored Procedures to wrap more complex actions in reusable SQL scripts.

Evaluation and Testing

This section reflects on the development process, summarising the key steps and how I ensured the application worked reliably.

Database Setup

I started by designing the database to support the core features of SocialHive, like user accounts, posts, comments, groups, and messaging. I made sure each table had proper relationships and foreign key constraints to prevent broken connections and keep everything organised. phpMyAdmin helped me visualise the structure and make changes easily during development.

Local Server Configuration

To test SocialHive on my machine, I installed XAMPP which includes Apache, PHP, and MySQL. I placed my files in the htdocs folder and set up Apache to serve my project as a local site. Using phpMyAdmin, I created the database and linked it to the app securely using a .env file for sensitive settings.

Building a Dynamic Web App

I focused on making the app feel seamless and interactive. I used sessions to keep users logged in, Bootstrap for a polished layout, and JavaScript to add interactivity — like liking posts or commenting — without full page reloads. I also added form validation to protect the database and help users avoid mistakes when entering data.

Data Manipulation

Every key feature, from signing up to sending a message, interacts with the database. I used prepared statements to prevent SQL injection, and SQL queries to handle data creation, retrieval, updates, and deletions. In more complex cases, I used transactions to ensure that multiple database changes either all happen or none do.

Test Cases

To ensure everything worked properly, I created a test plan covering every interactive element. Each test was based on the process described in my IPO chart and included what input was used, what process occurred, and what the expected output was. If a test didn't pass initially, I revised the code and retested until it did.

Test Cases

TEST NUMBER	TEST CASE	INPUT	PROCESS	OUTPUT	PASS/FAIL
1	Click to sign up link	Click sign up	Go to sign up page	Sign up page appears	Pass
2	Click to login link	Click login	Go to login page	Login page appears	Pass

3	Click Login button	Click login	Directed to home page	User authenticated and redirected	Pass
4	Click Signup button	Click signup	Create new user	User saved to database and directed to login	Pass
5	Click Like button	Click like	Like post	Like saved to database	Pass
6	Submit comment	Click submit	Save comment	Comment saved to DB and displayed	Pass
7	Edit comment	Click edit	Update comment	New comment version saved and shown	Pass
8	Delete comment	Click delete	Remove comment	Comment removed from DB and page	Pass
9	Add post	Click add post	Navigate to post form	New post form loads	Pass
10	Choose file	Click choose file	Select image	File name appears in form	Pass
11	Submit post	Click post	Save post info	Post appears in profile and DB	Pass
12	Send message	Press send	Save message	Message added to chat and DB	Pass
13	Save settings	Press save	Update settings	Settings stored and reflected on screen	Pass
14	Follow user	Click follow	Add follow record	Follower count updates	Pass
15	Unfollow user	Click unfollow	Remove follow record	Follower count decreases	Pass
16	Delete post	Click delete	Remove post	Post removed from profile and DB	Pass
17	Create new group	Click create	Load group form	Group creation page opens	Pass
18	Create group	Press create	Save group	Group saved and listed	Pass
19	Join group	Click join	Enter group chat	User added to group	Pass
20	Send in group chat	Click send	Post group message	Message saved and shown	Pass

21	Logout	Click logout	End session	User redirected to login page	Pass

Conclusion

Setting up SocialHive was a great experience that brought together several essential web development skills — from structuring the database and managing user data to designing a responsive UI and configuring a local environment. By applying best practices like prepared statements, session management, and responsive design with Bootstrap, I created a secure and interactive platform. Thorough testing helped ensure the app was reliable, user-friendly, and ready for future improvements.