**Prompt Question**

**Design a mobile responsive chat box using column fields sender and receiver. Insert chat record into the tbl\_chats table. Load users username from the tbl\_users table. Add login, logout and dashboard pages. Give me html, css, ajax, php codes. Include SQL code to create tbl\_chats and tbl\_users tables. $servername = 127.0.0.1"; $username = "root"; $password = ""; $database = "userchatdb2"**

**Chat Application Overview**

I've designed a responsive chat application with all the features you requested. Here's what's included:

**Database Structure**

* **tbl\_users**: Stores user information including username, password (hashed), email, and online status
* **tbl\_chats**: Stores chat messages with sender, receiver, message content, and read status

**Key Features**

1. **User Authentication**
   * Login/registration system with password hashing
   * Session management for user authentication
   * Online/offline status tracking
2. **Chat Interface**
   * Real-time messaging using AJAX (polls every 5 seconds)
   * Mobile-responsive design that works on all screen sizes
   * Message history loading with proper formatting
3. **Dashboard**
   * User list with search functionality
   * Online status indicators
   * User profile information

**Implementation Details**

**Backend (PHP)**

* Secure database connection and queries using prepared statements
* Proper session handling and user authentication
* AJAX endpoints for sending and retrieving messages
* Input sanitization to prevent XSS attacks

**Frontend**

* Responsive design using CSS media queries
* Real-time updates using jQuery AJAX
* User-friendly interface with status indicators
* Mobile optimization with toggle menu

**How to Use the Application**

1. Set up your database using the SQL code provided
2. Upload all files to your PHP server
3. Access the application through your browser

The application uses a simple folder structure with separate files for different functionalities, making it easy to maintain and extend.

**Security Features**

* Password hashing using PHP's password\_hash()
* Protection against SQL injection using prepared statements
* Input sanitization to prevent XSS attacks
* Session-based authentication