Carina Liu

12/2/25

Web Systems Quiz 2 pt 1

1.1. Explain three possible features of a web application that require (or, at least, made easier by) a server-side component written in a language such as PHP. Don't just mention the feature, explain in detail what it involves.

One important feature that requires a server-side component is **user authentication and session management**, since HTTP is a stateless protocol and cannot remember anything about a user from one request to the next. PHP handles this by calling session_start() to create or restore a session and storing user information in the $_SESSION array, while a session ID is kept in a cookie so the server can recognize the same user across page loads. Another feature made easier by server-side PHP is secure database access through **PDO**, where the server connects using new PDO(...) and executes SQL queries. Prepared statements (prepare() and execute()) allow variables to be bound safely and help prevent SQL injection, while the server ensures that actions respect relational rules such as foreign key constraints that require matching rows to exist before inserts. A third feature is **server-side validation and controlled updates** to stored data, where PHP checks user input, applies required logic, and interacts with tables that enforce NOT NULL, UNIQUE, or referential integrity. These server-driven steps ensure that operations like inserting or modifying records are performed securely and consistently in a way that client-side code alone cannot guarantee.

1.2. Explain two actions that can be taken to secure a web application. These may be related to user-authentication & authorization, server configuration, codebase, and/or network infrastructure. Don’t just mention the feature, explain in detail what it involves.

One action to secure a web application is to store passwords using **salted one-way hashes** rather than saving them in plaintext or as simple hashes. A unique random salt—such as one generated with uniqid(mt_rand(), true)—is stored with each user, and the server hashes the password combined with the salt using a secure function like sha256 or sha512. This prevents attackers from using rainbow tables and makes brute-force attempts far more expensive, because every user has a different hash even if they share the same password. Another key action is to **protect the authentication and session process,** since identity and access control rely on this mechanism. Only the user’s identity should be stored in $_SESSION, while permissions are rechecked on each request so that access can be revoked instantly. When logging out, the server should destroy the session, remove the session cookie, and create a new session ID to prevent session hijacking. Using HTTPS is also essential, because HTTP transmits credentials and session IDs in plaintext. Together, these measures strengthen both login security and ongoing access control throughout the application.