## *Password:*

* Hashing.
* Strong password Policy.
* Avoid similarity to username, first name, middle name, or last name.
* Add salt to the password before hashing.
* Use strong hashing algorithm.
* Encrypt salt. (Use strong encryption algorithm)
* Store secret key that used for encrypting salt indifferent database.
* Limitation for trying the password.
* Use strong keys for encrypting and hashing.
* Two-Authentication.
* Use OTP (One-Time-Password).

## *SQL Injection:*

* Use parameterized queries but for more layers of security user parametrized procedures.
* Validate and sanitize all user inputs on both the client and server sides.
* Use ORM frameworks that automatically parameterize queries.

## *XSS:*

* Encode user inputs before rendering them in HTML.
* Content Security Policy.
* Validate and sanitize user inputs on both the client and server sides.
* Use secure session management procedures.
* Use security headers like X-Content-Type-Options, X-Frame-Options, and Strict-Transport-Security.

## *CSRF:*

* Anti-CSRF Tokens.
* SameSite Attribute.
* Custom headers, to confirm that requests are authentic and not the product of CSRF attacks.
* Limit the lifetime of anti-CSRF tokens. (for the sessions)

## *General Secure Coding Practices:*

* Least Privileges.
* Error Handling.
* Encrypt the sensitive data in the database.
* Avoid using any checking statements on the client side (everything must be done on the server side).
* Update Database Backup.
* Use HTTPs.
* Don’t forget the coding quality principles while you are coding.
* `Use Session - Tokens instead of usual session.
* Loging system.