Vulnerability-Free Coding

Essential Guidelines

Database

1. Define Type of field
2. Define maximum character length for character fields
3. Add validators to check for the malicious code
4. Add validator to check for special characters allowed - allow only the characters required
5. If File field - allow only required file extensions ,
6. Check for file content and header content for malicious code in file
7. If decimal field mention the number of digits maximum and minimum

API

1. API is open or not
2. Encrypt the url and content if it is open API
3. Check for authentication
4. Check for authorization - if user is authorized then only give/post the data otherwise restrict
5. Error messages should be generic . It should not be directly from database
6. Exception Handling must be there for every API
7. There should be a set of http responses
    a. 200 OK: The request was successful.
    b. 201 Created: The request was successful, and a new resource was created.
    c. 403 Forbidden - permission denied
    d. 500- internal server error
    e. 404 Not found- page not found
    f. 401 - unauthorized
    g. 301 Moved Permanently: The resource has been permanently moved to a new URL.
    h. 400 Bad Request: due to invalid syntax.
    i. 429 Too Many Requests: The user has sent too many requests
    j. 503 Service Unavailable
8. API structure should be same for all portal backends
9. Commented code, print statements should be removed ,
10. Setup confidential parameters in separate file
11. Maintain logs

Login & Logout

1. Captcha should be generated & validated at server side and auto refresh on every request.
2. Payload should be encrypted
3. Password policy should be implemented
4. Error message should be generic - It must be Invalid credentials even though the password policy does not meet
5. Account lockout feature must be there if the valid user gives three invalid passwords - ban the account if it is more than 5 - admin should unlock
6. Multiple login - map the login with ip to restrict any attack
7. Logout should be done properly - it should redirect to login page - destroy local storage and destroy the tokens or session properly
8. Make sure random text is used to send authorization details like role

Server

1. Define security headers
2. Block Directory listing - server-status
3. Serversignature off
4. Restrict the access to .git folders like /.gitignore and .git/HEAD in both http and https
5. Error page if user tries to access any other urls
6. No.of requests should be defined to avoid DOS
7. Check TLS version and SSL version
8. Host header validation
9. Maintain logs