E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\69.baby-daycare-management"

Testing C:\xampp\htdocs\69.baby-daycare-management ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: updatemenu.php, line 28

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: updatewelcome.php, line 27

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: changepassword.php, line 26

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: changepassword.php, line 27

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: editpost.php, line 31

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: register.php, line 20

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: addpost.php, line 31

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: login.php, line 21

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: addmenu.php, line 26

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [High] Path Traversal

Path: updatemenu.php, line 50

Info: Unsanitized input from an HTTP parameter flows into unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.

✗ [High] Path Traversal

Path: updatewelcome.php, line 58

Info: Unsanitized input from an HTTP parameter flows into unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.

✗ [High] Path Traversal

Path: editpost.php, line 65

Info: Unsanitized input from an HTTP parameter flows into unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.

✗ [High] Path Traversal

Path: admin/pagerole.php, line 44

Info: Unsanitized input from an HTTP parameter flows into unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.

✗ [High] Path Traversal

Path: admin/posts.php, line 53

Info: Unsanitized input from an HTTP parameter flows into unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.

✗ [High] Path Traversal

Path: updatemenu.php, line 40

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: updatewelcome.php, line 45

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: editpost.php, line 49

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: addpost.php, line 41

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: addmenu.php, line 39

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Cross-site Scripting (XSS)

Path: admin/pagerole.php, line 67

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/pagerole.php, line 68

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/posts.php, line 88

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/contentsectionpage.php, line 54

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: admin/pagerole.php, line 73

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/posts.php, line 82

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/posts.php, line 233

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: inc/contentsectionpage.php, line 16

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: inc/contentsectionpage.php, line 20

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: inc/contentsectionpage.php, line 35

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: inc/contentsectionpage.php, line 50

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: inc/contentsectionpage.php, line 71

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inbox.php, line 119

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/siteoptions.php, line 184

Info: Unsanitized input from an HTTP parameter flows into select, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\69.baby-daycare-management

Summary:

33 Code issues found

24 [High] 9 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\php-trs"

Testing C:\xampp\htdocs\php-trs ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: inc/packages.php, line 30

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Use of a Broken or Risky Cryptographic Algorithm

Path: classes/SystemSettings.php, line 239

Info: Electronic Code Book (AES-128-ECB) mode should not be used (in openssl\_encrypt), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Cipher-Block Chaining).

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 239

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_encrypt.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 243

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_decrypt.

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 111

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 91

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: inc/navigation.php, line 92

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: admin/reminders/manage\_reminder.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/reminders/view\_reminder.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/user/manage\_user.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 47

Info: Unsanitized input from an HTTP parameter flows into file\_put\_contents, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

✗ [High] File Inclusion

Path: index.php, line 26

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 28

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 27

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 29

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\php-trs

Summary:

16 Code issues found

13 [High] 2 [Medium] 1 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\php-spms"

Testing C:\xampp\htdocs\php-spms ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: inc/packages.php, line 30

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Login.php, line 52

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Insufficient postMessage Validation

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 609

Info: The origin of the received message is not checked. This means any site (even malicious) can send message to this window. If you don't expect this, consider checking the origin of sender.

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Use of a Broken or Risky Cryptographic Algorithm

Path: classes/SystemSettings.php, line 239

Info: Electronic Code Book (AES-128-ECB) mode should not be used (in openssl\_encrypt), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Cipher-Block Chaining).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/php-email-form/validate.js, line 81

Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 326

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 239

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_encrypt.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 243

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_decrypt.

✗ [High] File Inclusion

Path: admin/index.php, line 40

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 42

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 79

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 81

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 47

Info: Unsanitized input from an HTTP parameter flows into file\_put\_contents, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 478

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/index.php, line 18

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/index.php, line 23

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: admin/user/manage\_user.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/services/manage\_service.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/services/view\_service.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inquiries/view\_inquiry.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: services/view.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\php-spms

Summary:

32 Code issues found

18 [High] 11 [Medium] 3 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\php-acrss"

Testing C:\xampp\htdocs\php-acrss ...

✗ [Low] Insufficient postMessage Validation

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 609

Info: The origin of the received message is not checked. This means any site (even malicious) can send message to this window. If you don't expect this, consider checking the origin of sender.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: inc/packages.php, line 30

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Login.php, line 52

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 326

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/php-email-form/validate.js, line 81

Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] Use of a Broken or Risky Cryptographic Algorithm

Path: classes/SystemSettings.php, line 239

Info: Electronic Code Book (AES-128-ECB) mode should not be used (in openssl\_encrypt), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Cipher-Block Chaining).

✗ [Medium] SQL Injection

Path: admin/bookings/index.php, line 45

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 239

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_encrypt.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 243

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_decrypt.

✗ [High] Cross-site Scripting (XSS)

Path: admin/index.php, line 18

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/index.php, line 23

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 478

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 47

Info: Unsanitized input from an HTTP parameter flows into file\_put\_contents, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

✗ [High] File Inclusion

Path: admin/index.php, line 40

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 42

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 52

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 54

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] SQL Injection

Path: admin/services/manage\_service.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: services/view.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inquiries/view\_inquiry.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/user/manage\_user.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/bookings/view\_booking.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/bookings/manage\_booking.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/services/view\_service.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\php-acrss

Summary:

35 Code issues found

20 [High] 12 [Medium] 3 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\php-lfis"

Testing C:\xampp\htdocs\php-lfis ...

✗ [Low] Insufficient postMessage Validation

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 609

Info: The origin of the received message is not checked. This means any site (even malicious) can send message to this window. If you don't expect this, consider checking the origin of sender.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: inc/packages.php, line 30

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Login.php, line 52

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Prototype Pollution

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into a member access and is used to access a property of this object by name. This may allow a malicious user to pollute the Object.prototype and cause a crash, remote code execution or logic bypasses.

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 326

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized input from an event listener flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/vendor/php-email-form/validate.js, line 81

Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Scripting attack (DOMXSS).

✗ [Medium] Use of a Broken or Risky Cryptographic Algorithm

Path: classes/SystemSettings.php, line 239

Info: Electronic Code Book (AES-128-ECB) mode should not be used (in openssl\_encrypt), because it exposes frequency of symbols in your plaintext. Consider using other modes (e.g. Cipher-Block Chaining).

✗ [High] Cross-site Scripting (XSS)

Path: admin/index.php, line 18

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/index.php, line 23

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 370

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 478

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 539

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Regular Expression Denial of Service (ReDoS)

Path: assets/vendor/tinymce/plugins/codesample/plugin.js, line 614

Info: Unsanitized user input from an event listener flows into RegExp, where it is used to build a regular expression. This may result in a Regular expression Denial of Service attack (reDOS).

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 47

Info: Unsanitized input from an HTTP parameter flows into file\_put\_contents, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 239

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_encrypt.

✗ [High] Hardcoded Secret

Path: classes/SystemSettings.php, line 243

Info: Avoid hardcoding values that are meant to be secret. Found hardcoded secret used in openssl\_decrypt.

✗ [High] SQL Injection

Path: admin/items/view\_item.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/items/manage\_item.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: items/index.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/categories/manage\_category.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/user/manage\_user.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: items/view.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/categories/view\_category.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inquiries/view\_inquiry.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] File Inclusion

Path: admin/index.php, line 40

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 42

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 52

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 54

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\php-lfis

Summary:

35 Code issues found

21 [High] 11 [Medium] 3 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\vservice"

Testing C:\xampp\htdocs\vservice ...

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: inc/packages.php, line 30

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Users.php, line 28

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Users.php, line 87

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Users.php, line 94

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Login.php, line 20

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: classes/Login.php, line 45

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Code Injection

Path: classes/Master.php, line 473

Info: Unsanitized input from a database flows into eval, where it is executed as php code. This may result in a Code Injection vulnerability.

✗ [Medium] Code Injection

Path: classes/Master.php, line 477

Info: Unsanitized input from a database flows into eval, where it is executed as php code. This may result in a Code Injection vulnerability.

✗ [Medium] Path Traversal

Path: inc/packages.php, line 13

Info: Unsanitized input from a database flows into scandir, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to list arbitrary directories.

✗ [Medium] Path Traversal

Path: classes/Users.php, line 78

Info: Unsanitized input from a database flows into unlink, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to delete arbitrary files.

✗ [Medium] SQL Injection

Path: classes/Master.php, line 463

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: classes/Master.php, line 464

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: classes/Master.php, line 530

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: admin/service\_requests/index.php, line 40

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: admin/service\_requests/index.php, line 41

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: admin/inventory/index.php, line 40

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Medium] SQL Injection

Path: admin/inventory/index.php, line 41

Info: Unsanitized input from a database flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] File Inclusion

Path: admin/index.php, line 27

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: admin/index.php, line 29

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 18

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] File Inclusion

Path: index.php, line 20

Info: Unsanitized input from an HTTP parameter flows into include, where it is included dynamically. Allowing unvalidated user input to control files that are included dynamically in PHP can lead to malicious code execution.

✗ [High] Cross-site Scripting (XSS)

Path: inc/topBarNav.php, line 11

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/report/index.php, line 37

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/report/index.php, line 37

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 164

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inc/navigation.php, line 165

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 70

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 145

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 145

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 160

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 160

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 174

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: products/index.php, line 174

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/report/orders.php, line 38

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/report/orders.php, line 38

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/report/service\_requests.php, line 37

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/report/service\_requests.php, line 37

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: classes/Zone.php, line 68

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/inventory/manage\_stock.php, line 18

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 50

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: classes/SystemSettings.php, line 60

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: classes/Users.php, line 35

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] SQL Injection

Path: admin/report/index.php, line 69

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/report/index.php, line 71

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/report/index.php, line 75

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products/index.php, line 90

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/report/orders.php, line 67

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/report/service\_requests.php, line 71

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/report/service\_requests.php, line 73

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/report/service\_requests.php, line 77

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inventory/manage\_stock.php, line 6

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inventory/manage\_stock.php, line 25

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_request.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_request.php, line 6

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/products/view\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/user/manage\_user.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage\_service.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_service.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage\_brand.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/clients/manage\_client.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: products/view\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/mechanics/manage\_mechanic.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/inventory/view\_stock.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/maintenance/manage\_category.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/products/manage\_product.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/service\_requests/view\_request.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/service\_requests/manage\_request.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/view\_order.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: view\_order.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/orders/update\_status.php, line 4

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/service\_requests/manage\_inventory.php, line 3

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\vservice

Summary:

72 Code issues found

55 [High] 11 [Medium] 6 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\75.online-examination-system-for-mcq"

Testing C:\xampp\htdocs\75.online-examination-system-for-mcq ...

✗ [Low] Use of Hardcoded Credentials

Path: dbConnection.php, line 3

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: forgot-password.php, line 24

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: sign.php, line 31

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Use of Password Hash With Insufficient Computational Effort

Path: login.php, line 15

Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Medium] Open Redirect

Path: login.php, line 27

Info: Unsanitized input from an HTTP parameter flows into header, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [Medium] Open Redirect

Path: feed.php, line 12

Info: Unsanitized input from an HTTP parameter flows into header, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [Medium] Open Redirect

Path: admin.php, line 22

Info: Unsanitized input from an HTTP parameter flows into header, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [Medium] Open Redirect

Path: logout.php, line 6

Info: Unsanitized input from an HTTP parameter flows into header, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: dash.php, line 323

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: forgot-password.php, line 135

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: account.php, line 20

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: account.php, line 144

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: account.php, line 148

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: account.php, line 157

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: feedback.php, line 17

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: feedback.php, line 95

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: index.php, line 18

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: index.php, line 229

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: dash.php, line 213

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: forgot-password.php, line 8

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: forgot-password.php, line 26

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: forgot-password.php, line 37

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: account.php, line 139

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: account.php, line 147

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: account.php, line 168

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: sign.php, line 34

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: sign.php, line 38

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: sign.php, line 46

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: login.php, line 16

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: feed.php, line 11

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 9

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 19

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 20

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 28

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 31

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 32

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 34

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 35

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 36

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 54

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 71

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 80

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 81

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 82

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 83

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 212

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 219

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 226

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 228

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 237

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 242

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 250

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 252

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 260

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 269

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 278

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 287

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 302

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: update.php, line 307

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin.php, line 11

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\75.online-examination-system-for-mcq

Summary:

61 Code issues found

53 [High] 4 [Medium] 4 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\76.theme-park-management-system"

Testing C:\xampp\htdocs\76.theme-park-management-system ...

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: util/logout.php, line 9

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: util/login.php, line 68

Info: setcookie misses the HttpOnly attribute (it is false by default). Set it to true to protect the cookie from possible malicious code on client side.

✗ [Low] SQL Injection

Path: util/test.php, line 21

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: util/logout.php, line 9

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: util/login.php, line 68

Info: setcookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

✗ [Medium] Privacy Leak

Path: util/userlist\_update.php, line 30

Info: Sensitive data from a password flows into the echo statement, where it is leaked.

✗ [High] SQL Injection

Path: util/login.php, line 30

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: util/load-rides-list.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: util/ticketlist\_delete.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: util/loadmore-data.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: util/deletestaff.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: util/deleteride.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: util/userlist\_edit.php, line 6

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: util/ridelist\_edit.php, line 6

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: ticketlist.php, line 120

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: ticketlist.php, line 136

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: index.php, line 15

Info: Unsanitized input from cookies flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: index.php, line 77

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: ticketlist.php, line 168

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: addticket.php, line 117

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: addticket.php, line 120

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: pricing.php, line 91

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: pricing.php, line 94

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: rideslist.php, line 144

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: rideslist.php, line 147

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: util/userlist\_update.php, line 30

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: addrides.php, line 48

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: addrides.php, line 51

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: util/ridelist\_update.php, line 26

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: stafflist.php, line 170

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: stafflist.php, line 173

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: addstaff.php, line 89

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: addstaff.php, line 92

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: ticketlist.php, line 166

Info: Unsanitized input from an HTTP header flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: ticketlist.php, line 171

Info: Unsanitized input from an HTTP header flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\76.theme-park-management-system

Summary:

35 Code issues found

29 [High] 1 [Medium] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\77.coffee-shop-store"

Testing C:\xampp\htdocs\77.coffee-shop-store ...

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/fetchcombo.php, line 37

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/index.php, line 24

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/combo%20offers1.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/dessert.php, line 30

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/frosteals.php, line 35

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/big%20eats.php, line 29

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/sundaes.php, line 29

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/fetchfood.php, line 37

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/frappe.php, line 34

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/hot%20coffee.php, line 34

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/comboedit.php, line 4

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/cake%20away.php, line 32

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/small%20eats.php, line 29

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/cold%20coffee.php, line 30

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/updatefood.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/food1.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/foodedit.php, line 15

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/fetchbeverages.php, line 37

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/new1.php, line 8

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/beverages1.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/combo%20offers.php, line 20

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: cafecoffee/admin/beveragesedit.php, line 4

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysql\_connect.

✗ [High] Path Traversal

Path: cafecoffee/admin/combo%20offers1.php, line 11

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: cafecoffee/admin/food1.php, line 11

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] Path Traversal

Path: cafecoffee/admin/beverages1.php, line 11

Info: Unsanitized input from an uploaded file flows into move\_uploaded\_file, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to move arbitrary files.

✗ [High] SQL Injection

Path: cafecoffee/admin/index.php, line 26

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/combo%20offers1.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/comboedit.php, line 7

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/updatefood.php, line 8

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/food1.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/foodedit.php, line 18

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/new1.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/beverages1.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cafecoffee/admin/beveragesedit.php, line 7

Info: Unsanitized input from an HTTP parameter flows into mysql\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: cafecoffee/admin/combo%20offers1.php, line 17

Info: Unsanitized input from an uploaded file flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: cafecoffee/admin/food1.php, line 17

Info: Unsanitized input from an uploaded file flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: cafecoffee/admin/beverages1.php, line 17

Info: Unsanitized input from an uploaded file flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\77.coffee-shop-store

Summary:

37 Code issues found

15 [High] 22 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\78.crime-record-management-system\ghpolice"

Testing C:\xampp\htdocs\78.crime-record-management-system\ghpolice ...

✗ [Low] Use of Hardcoded Credentials

Path: cid/dbconnect.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: officer/dbconnect.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: admin/dbconnect2.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: admin/dbconnect.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli\_connect.

✗ [Low] Use of Hardcoded Credentials

Path: dbconnect.php, line 2

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli\_connect.

✗ [High] SQL Injection

Path: cid/investigation.php, line 52

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cid/investigation.php, line 83

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/casedetails.php, line 46

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/casedetails.php, line 136

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/profile\_update.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/profile\_update.php, line 22

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/investigation.php, line 52

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/investigation.php, line 83

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/assigncase.php, line 23

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cid/profile\_update.php, line 13

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cid/profile\_update.php, line 21

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/editcase.php, line 10

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/profile\_update.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: officer/profile\_update.php, line 22

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/assigncase.php, line 23

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cid/cidstatement.php, line 51

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cid/cidstatement.php, line 82

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/savestaffedit.php, line 11

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/savestaffedit.php, line 14

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cid/casedetails.php, line 45

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: cid/casedetails.php, line 129

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/saveuserlogin.php, line 34

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/investigation.php, line 52

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/investigation.php, line 83

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/casedetails.php, line 46

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: admin/casedetails.php, line 136

Info: Unsanitized input from an HTTP parameter flows into mysqli\_query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: cid/investigation.php, line 57

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/casedetails.php, line 27

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/casedetails.php, line 29

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/casedetails.php, line 51

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/investigation.php, line 57

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/assigncase.php, line 67

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/assigncase.php, line 67

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: cid/cidstatement.php, line 56

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: cid/casedetails.php, line 26

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: cid/casedetails.php, line 50

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/investigation.php, line 57

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/casedetails.php, line 27

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/casedetails.php, line 29

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/casedetails.php, line 51

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/addcase.php, line 53

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/addcase.php, line 54

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/addcase.php, line 64

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/addcase.php, line 81

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: officer/addcase.php, line 82

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: admin/login\_details.php, line 43

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\78.crime-record-management-system\ghpolice

Summary:

51 Code issues found

46 [High] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\79.yoga-classes-registration-system\Yoga-Website-master"

Testing C:\xampp\htdocs\79.yoga-classes-registration-system\Yoga-Website-master ...

✗ [Low] Use of Hardcoded Credentials

Path: PHP%20Files/connection/connection.php, line 3

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli.

✗ [Low] Use of Hardcoded Credentials

Path: PHP%20Files/connection/connection.php, line 7

Info: Do not hardcode credentials in code. Found a hardcoded credential used in mysqli.

✗ [Medium] Open Redirect

Path: PHP%20Files/connection/index.php, line 8

Info: Unsanitized input from an HTTP header flows into header, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: PHP%20Files/register.php, line 107

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: PHP%20Files/register.php, line 138

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: PHP%20Files/register.php, line 139

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: PHP%20Files/contact.php, line 89

Info: Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: PHP%20Files/register.php, line 148

Info: Unsanitized input from an HTTP header flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] Cross-site Scripting (XSS)

Path: PHP%20Files/contact.php, line 97

Info: Unsanitized input from an HTTP header flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

✗ [High] SQL Injection

Path: PHP%20Files/register.php, line 106

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✗ [High] SQL Injection

Path: PHP%20Files/register.php, line 109

Info: Unsanitized input from an HTTP parameter flows into query, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\79.yoga-classes-registration-system\Yoga-Website-master

Summary:

11 Code issues found

8 [High] 1 [Medium] 2 [Low]