E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\26.Client-Management-System\clientms"

Testing C:\xampp\htdocs\26.Client-Management-System\clientms ...

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

Path: admin/add-client.php, line 14

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: client/forgot-password.php, line 10

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: client/index.php, line 9

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: admin/forgot-password.php, line 10

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: client/change-password.php, line 12

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: client/change-password.php, line 13

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: admin/change-password.php, line 12

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: admin/change-password.php, line 13

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: admin/index.php, line 9

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

✗ [High] SQL Injection

Path: admin/sales-reports-detail.php, line 78

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

✗ [High] SQL Injection

Path: admin/sales-reports-detail.php, line 123

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/search-invoices.php, line 71

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/search-invoices.php, line 100

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: client/search-invoices.php, line 71

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: client/search-invoices.php, line 100

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/bwdates-reports-details.php, line 58

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/bwdates-reports-details.php, line 58

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/bwdates-reports-details.php, line 87

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\26.Client-Management-System\clientms

Summary:

18 Code issues found

9 [High] 9 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\27.Company-Visitors-Management-System\cvms"

Testing C:\xampp\htdocs\27.Company-Visitors-Management-System\cvms ...

✗ [Low] Use of Hardcoded Credentials

Path: includes/dbconnection.php, line 2

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

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

Path: index.php, line 9

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: change-password.php, line 12

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: change-password.php, line 13

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: resetpassword.php, line 11

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

✗ [High] Cross-site Scripting (XSS)

Path: search-visitor.php, line 79

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: bwdates-reports-details.php, line 80

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: bwdates-reports-details.php, line 80

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: search-visitor.php, line 97

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: bwdates-reports-details.php, line 98

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.php, line 15

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: visitor-detail.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: visitor-detail.php, line 100

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 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: index.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: visitors-form.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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\27.Company-Visitors-Management-System\cvms

Summary:

16 Code issues found

11 [High] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\28.complaint-management-system\cms"

Testing C:\xampp\htdocs\28.complaint-management-system\cms ...

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

Path: users/update-image.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: admin/index.php, line 8

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: users/index.php, line 7

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: users/index.php, line 40

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: users/registration.php, line 8

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: admin/change-password.php, line 16

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: admin/change-password.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: users/change-password.php, line 16

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: users/change-password.php, line 20

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

✗ [Low] Python 2 source code

Path: admin/assets/plugins/dynatree/doc/dynatree\_server.py, line 85

Info: This source file appears to be in Python 2. The Python 2 interpreter has been unsupported without security updates since January 2020. Consider porting this code to Python 3.

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/2512.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/html\_table.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/-complex\_header.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/two\_tables.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/js\_data.php, line 87

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/complex\_header\_2.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/deferred\_table.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/6776.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dom\_data.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dymanic\_table.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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/js\_data\_mixed\_types.php, line 87

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dom\_data\_two\_headers.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/empty\_table.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/media/unit\_testing/templates/dom\_data\_th.php, line 25

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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/dynatree/tests/qunit.js, line 545

Info: Unsanitized input from the document location 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 806

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 842

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1099

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1145

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1162

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/qunit/qunit.js, line 1254

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 806

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 842

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1099

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1145

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1162

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/summernote/test/qunit-1.11.0.js, line 1254

Info: Unsanitized input from an exception 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-mockjax/test/jquery.js, line 23

Info: Unsanitized input from the document location flows into write, 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).

✗ [Low] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/test/jquery.js, line 23

Info: Unsanitized input from the document location flows into write, 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).

✗ [Low] Path Traversal

Path: admin/assets/plugins/jquery-cookie/test/server.js, line 10

Info: Unsanitized input from the request URL flows into fs.readFile, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

✗ [Low] Cleartext Transmission of Sensitive Information

Path: admin/assets/plugins/jquery-cookie/test/server.js, line 6

Info: http.createServer uses HTTP which is an insecure protocol and should not be used in code due to cleartext transmission of information. Data in cleartext in a communication channel can be sniffed by unauthorized actors. Consider using the https module instead.

✗ [Low] Allocation of Resources Without Limits or Throttling

Path: admin/assets/plugins/jquery-cookie/test/server.js, line 6

Info: This endpoint handler performs a file system operation and does not use a rate-limiting mechanism. It may enable the attackers to perform Denial-of-service attacks. Consider using a rate-limiting middleware such as express-limit.

✗ [Medium] Cleartext Transmission of Sensitive Information

Path: admin/assets/plugins/jQuery-File-Upload/server/node/server.js, line 290

Info: http.createServer uses HTTP which is an insecure protocol and should not be used in code due to cleartext transmission of information. Data in cleartext in a communication channel can be sniffed by unauthorized actors. Consider using the https module instead.

✗ [Medium] Open Redirect

Path: users/assets/js/form-component.js, line 82

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

✗ [Medium] Cross-site Scripting (XSS)

Path: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 806

Info: Unsanitized input from an exception 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: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 842

Info: Unsanitized input from an exception 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: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1099

Info: Unsanitized input from an exception 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: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1145

Info: Unsanitized input from an exception 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: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1162

Info: Unsanitized input from an exception 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: admin/assets/plugins/bootstrap-paginator/lib/qunit-1.11.0.js, line 1254

Info: Unsanitized input from an exception 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] Debug Mode Enabled

Path: admin/assets/plugins/jQuery-File-Upload/server/gae-python/main.py, line 159

Info: Running the application in debug mode (debug flag is set to True in webapp2.WSGIApplication) is a security risk if the application is accessible by untrusted parties.

✗ [Medium] Open Redirect

Path: admin/assets/plugins/jQuery-File-Upload/server/gae-go/app/main.go, line 229

Info: Unsanitized input from a web form flows into net.http.Redirect, where it is used as an URL to redirect the user. This may result in an Open Redirect vulnerability.

✗ [Medium] Open Redirect

Path: admin/index.php, line 18

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.

✗ [Medium] Open Redirect

Path: admin/index.php, line 27

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.

✗ [Medium] Open Redirect

Path: users/index.php, line 19

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.

✗ [Medium] Open Redirect

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 755

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

✗ [Medium] Permissive Cross-domain Policy

Path: admin/assets/plugins/ckeditor/plugins/wsc/dialogs/tmp.html, line 63

Info: Setting targetOrigin to "\*" in postMessage may enable malicious parties to intercept the message. Consider using an exact target origin instead.

✗ [High] Cross-site Scripting (XSS)

Path: admin/assets/plugins/DataTables/examples/examples\_support/editable\_ajax.php, line 2

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/assets/plugins/DataTables/examples/server\_side/scripts/jsonp.php, line 190

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/assets/plugins/DataTables/examples/server\_side/scripts/objects\_jsonp.php, line 192

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/assets/plugins/DataTables/examples/server\_side/scripts/id\_jsonp.php, line 195

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/assets/plugins/jQuery-Smart-Wizard/more\_examples/services/service.php, line 62

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/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 740

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/assets/plugins/jquery-validation/demo/captcha/index.php, line 55

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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 295

Info: Unsanitized input from an HTTP header 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 308

Info: Unsanitized input from an HTTP header flows into mkdir, 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 690

Info: Unsanitized input from an HTTP header flows into mkdir, 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 330

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

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 698

Info: Unsanitized input from an HTTP header 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 708

Info: Unsanitized input from an HTTP header 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] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 700

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

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 738

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

✗ [High] Path Traversal

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 704

Info: Unsanitized input from an HTTP header 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: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 725

Info: Unsanitized input from an HTTP header 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/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 975

Info: Unsanitized input from an HTTP header 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/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 981

Info: Unsanitized input from an HTTP header 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/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 747

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

✗ [High] Path Traversal

Path: users/register-complaint.php, line 24

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: users/update-image.php, line 33

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] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 700

Info: Unsanitized input from an HTTP header flows into fopen, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 738

Info: Unsanitized input from an HTTP header flows into fopen, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 725

Info: Unsanitized input from an HTTP header flows into unlink, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 975

Info: Unsanitized input from an HTTP header flows into unlink, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 981

Info: Unsanitized input from an HTTP header flows into unlink, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Server-Side Request Forgery (SSRF)

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 747

Info: Unsanitized input from an HTTP header flows into readfile, where it is used as an URL to perform a request. This may result in a Server-Side Request Forgery vulnerability.

✗ [High] Cross-site Scripting (XSS)

Path: admin/assets/plugins/jquery-validation/lib/jquery.js, line 25

Info: Unsanitized input from the document location flows into write, 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] SQL Injection

Path: users/register-complaint.php, line 25

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/index.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: users/index.php, line 7

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: users/index.php, line 41

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: users/index.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: users/registration.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/category.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: admin/category.php, line 25

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: users/profile.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/complaint-details.php, line 61

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/complaint-details.php, line 124

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/state.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: admin/state.php, line 25

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/edit-category.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: users/check\_availability.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: admin/subcategory.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/subcategory.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: admin/manage-users.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: admin/edit-state.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: users/complaint-details.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: users/complaint-details.php, line 109

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/edit-subcategory.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: admin/updatecomplaint.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/updatecomplaint.php, line 15

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/userprofile.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: users/update-image.php, line 35

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

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

Path: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 433

Info: Unsanitized user input from an HTTP header flows into preg\_match, 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: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 719

Info: Unsanitized user input from an HTTP header flows into preg\_match, 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: admin/assets/plugins/jQuery-File-Upload/server/php/UploadHandler.php, line 852

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

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\28.complaint-management-system\cms

Summary:

115 Code issues found

58 [High] 15 [Medium] 42 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\31.Cyber-Cafe-Management-System\ccms"

Testing C:\xampp\htdocs\31.Cyber-Cafe-Management-System\ccms ...

✗ [Low] Cleartext Transmission of Sensitive Information

Path: vendors/peity/test/index.js, line 5

Info: http.createServer uses HTTP which is an insecure protocol and should not be used in code due to cleartext transmission of information. Data in cleartext in a communication channel can be sniffed by unauthorized actors. Consider using the https module instead.

✗ [Low] Cleartext Transmission of Sensitive Information

Path: vendors/peity/test/fixtures.js, line 4

Info: http.createServer uses HTTP which is an insecure protocol and should not be used in code due to cleartext transmission of information. Data in cleartext in a communication channel can be sniffed by unauthorized actors. Consider using the https module instead.

✗ [Low] Use of Hardcoded Credentials

Path: includes/dbconnection.php, line 2

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

✗ [Low] Allocation of Resources Without Limits or Throttling

Path: vendors/peity/test/app.js, line 20

Info: This endpoint handler performs a file system operation and does not use a rate-limiting mechanism. It may enable the attackers to perform Denial-of-service attacks. Consider using a rate-limiting middleware such as express-limit.

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

Path: index.php, line 9

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: change-password.php, line 12

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: change-password.php, line 13

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: resetpassword.php, line 11

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

✗ [Low] Cross-site Scripting (XSS)

Path: vendors/bootstrap/js/tests/index.html, line 19

Info: Unsanitized input from the document location flows into write, 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: vendors/jqvmap/create/jqvmap.py, line 567

Info: Unsanitized input from a command line argument flows into open, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

✗ [Medium] Path Traversal

Path: vendors/bootstrap/build/lint-vars.js, line 35

Info: Unsanitized input from a command line argument flows into glob.sync, 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: vendors/bootstrap/build/lint-vars.js, line 40

Info: Unsanitized input from a command line argument flows into fs.readFileSync, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

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

Path: vendors/bootstrap/build/change-version.js, line 56

Info: Unsanitized user input from a command line argument 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] SQL Injection

Path: view-user-detail.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: view-user-detail.php, line 110

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: add-computer.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: edit-computer-detail.php, line 15

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: edit-computer-detail.php, line 107

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 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: add-users.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: adminprofile.php, line 15

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: search.php, line 123

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: bwdates-reports-details.php, line 96

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 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] Cross-site Scripting (XSS)

Path: search.php, line 106

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: bwdates-reports-details.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: bwdates-reports-details.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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\31.Cyber-Cafe-Management-System\ccms

Summary:

27 Code issues found

15 [High] 3 [Medium] 9 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\32.Daily-Expense-Tracker\dets"

Testing C:\xampp\htdocs\32.Daily-Expense-Tracker\dets ...

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

Path: register.php, line 10

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: change-password.php, line 11

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: change-password.php, line 12

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: reset-password.php, line 11

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: index.php, line 9

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

✗ [Low] Use of Hardcoded Credentials

Path: includes/dbconnection.php, line 2

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

✗ [High] Cross-site Scripting (XSS)

Path: expense-datewise-reports-detailed.php, line 60

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: expense-datewise-reports-detailed.php, line 60

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: expense-reports-detailed.php, line 60

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: expense-reports-detailed.php, line 60

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: expense-monthwise-reports-detailed.php, line 60

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: expense-monthwise-reports-detailed.php, line 60

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: expense-yearwise-reports-detailed.php, line 60

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: expense-yearwise-reports-detailed.php, line 60

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: register.php, line 12

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: register.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: index.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: expense-datewise-reports-detailed.php, line 74

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: expense-reports-detailed.php, line 74

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: expense-monthwise-reports-detailed.php, line 74

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: expense-yearwise-reports-detailed.php, line 74

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 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: user-profile.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: add-expense.php, line 15

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\32.Daily-Expense-Tracker\dets

Summary:

24 Code issues found

18 [High] 6 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\33.Dairy-Farm-Shop-Management-System\dfsms"

Testing C:\xampp\htdocs\33.Dairy-Farm-Shop-Management-System\dfsms ...

✗ [Low] Use of Hardcoded Credentials

Path: includes/config.php, line 2

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

✗ [Low] Insufficient postMessage Validation

Path: vendors/tinymce/plugins/codesample/plugin.js, line 327

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: change-password.php, line 12

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: change-password.php, line 13

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: index.php, line 8

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

✗ [Medium] Permissive Cross-domain Policy

Path: dist/js/lightgallery-all.js, line 2253

Info: Setting targetOrigin to "\*" in postMessage may enable malicious parties to intercept the message. Consider using an exact target origin instead.

✗ [Medium] Permissive Cross-domain Policy

Path: dist/js/lightgallery-all.js, line 2261

Info: Setting targetOrigin to "\*" in postMessage may enable malicious parties to intercept the message. Consider using an exact target origin instead.

✗ [Medium] Permissive Cross-domain Policy

Path: dist/js/lightgallery-all.js, line 2303

Info: Setting targetOrigin to "\*" in postMessage may enable malicious parties to intercept the message. Consider using an exact target origin instead.

✗ [Medium] Permissive Cross-domain Policy

Path: dist/js/lightgallery-all.js, line 2311

Info: Setting targetOrigin to "\*" in postMessage may enable malicious parties to intercept the message. Consider using an exact target origin instead.

✗ [High] Cross-site Scripting (XSS)

Path: bwdate-report-details.php, line 63

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: bwdate-report-details.php, line 63

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: sales-report-details.php, line 63

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: sales-report-details.php, line 63

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: bwdate-report-details.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: bwdate-report-details.php, line 90

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: sales-report-details.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: sales-report-details.php, line 86

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 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: bwdate-report-ds.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: invoice.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: edit-category.php, line 15

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: edit-category.php, line 73

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: sales-report-ds.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: search-product.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: search-product.php, line 67

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: search-product.php, line 173

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: edit-product.php, line 17

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: edit-product.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: edit-company.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: edit-company.php, line 79

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: add-category.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: manage-companies.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: add-product.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: manage-categories.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: view-invoice.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: view-invoice.php, line 78

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: view-invoice.php, line 117

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: profile.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: invoices.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: add-company.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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\33.Dairy-Farm-Shop-Management-System\dfsms

Summary:

40 Code issues found

31 [High] 4 [Medium] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\34.Directory-Management-System\dms"

Testing C:\xampp\htdocs\34.Directory-Management-System\dms ...

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

Path: admin/index.php, line 9

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: admin/reset-password.php, line 13

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: admin/changepassword.php, line 12

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: admin/changepassword.php, line 13

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/include/dbconnection.php, line 2

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

✗ [High] SQL Injection

Path: admin/index.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: searchdata.php, line 73

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/edit-directory.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: admin/edit-directory.php, line 89

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/forget-password.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/add-directory.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: admin/search-directory.php, line 94

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/view-directory.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: admin/admin-profile.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] Cross-site Scripting (XSS)

Path: searchdata.php, line 71

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/search-directory.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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\34.Directory-Management-System\dms

Summary:

16 Code issues found

11 [High] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\35.e-Diary-Management-System\edms"

Testing C:\xampp\htdocs\35.e-Diary-Management-System\edms ...

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

Path: password-recovery.php, line 9

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 6

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: registration.php, line 8

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: change-password.php, line 11

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: change-password.php, line 12

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

✗ [High] SQL Injection

Path: password-recovery.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: password-recovery.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: login.php, line 7

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: registration.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: registration.php, line 12

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: view-note.php, line 12

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: view-note.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: view-note.php, line 66

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: view-note.php, line 104

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: search-result.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: search-result.php, line 12

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: search-result.php, line 73

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: add-notes.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: my-profile.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: edit-category.php, line 12

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: manage-categories.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: add-category.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: manage-notes.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: manage-notes.php, line 12

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: search-result.php, line 39

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\35.e-Diary-Management-System\edms

Summary:

25 Code issues found

20 [High] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\36.Employee-Leave-Management-System\elms"

Testing C:\xampp\htdocs\36.Employee-Leave-Management-System\elms ...

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

Path: index.php, line 9

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: admin/addemployee.php, line 16

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: emp-changepassword.php, line 13

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: emp-changepassword.php, line 14

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: forgot-password.php, line 9

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: admin/changepassword.php, line 13

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: admin/changepassword.php, line 14

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: admin/index.php, line 7

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

✗ [Medium] Cross-site Scripting (XSS)

Path: assets/plugins/jquery-steps/jquery.steps.js, line 723

Info: Unsanitized input from data from a remote resource flows into html, 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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\36.Employee-Leave-Management-System\elms

Summary:

9 Code issues found

1 [Medium] 8 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\37.Employee-Record-Management-System\erms"

Testing C:\xampp\htdocs\37.Employee-Record-Management-System\erms ...

✗ [Low] Use of Hardcoded Credentials

Path: front/db.php, line 3

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

✗ [Low] Use of Hardcoded Credentials

Path: includes/dbconnection.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/includes/dbconnection.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/front/db.php, line 3

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

✗ [High] SQL Injection

Path: admin/index.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: myprofile.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: myexp.php, line 27

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/editempeducation.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: admin/editempeducation.php, line 96

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/editempprofile.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/editempprofile.php, line 87

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: resetpassword.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: registererms.php, line 12

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: registererms.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: editmyeducation.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: admin/changepassword.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/changepassword.php, line 17

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: changepassword.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: changepassword.php, line 17

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: forgetpassword.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/allemployees.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/adminprofile.php, line 17

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: loginerms.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: myeducation.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: admin/editempexp.php, line 27

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/editempexp.php, line 92

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: editmyexp.php, line 27

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\37.Employee-Record-Management-System\erms

Summary:

27 Code issues found

23 [High] 4 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\38.GYM-Management-System\gym"

Testing C:\xampp\htdocs\38.GYM-Management-System\gym ...

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

Path: registration.php, line 14

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 13

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 14

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: admin/change-password.php, line 10

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: admin/change-password.php, line 11

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: admin/login.php, line 9

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 9

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/booking-history-details.php, line 215

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: registration.php, line 136

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: registration.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: registration.php, line 142

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: registration.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: registration.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: registration.php, line 151

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\38.GYM-Management-System\gym

Summary:

14 Code issues found

7 [High] 7 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\39.Hostel-management-System\hostel"

Testing C:\xampp\htdocs\39.Hostel-management-System\hostel ...

✗ [Low] Use of Hardcoded Credentials

Path: includes/config.php, line 6

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

✗ [Low] Use of Hardcoded Credentials

Path: admin/includes/config.php, line 6

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

✗ [Medium] SQL Injection

Path: index.php, line 28

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

✗ [Medium] SQL Injection

Path: login.php, line 28

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

✗ [Medium] Open Redirect

Path: admin/includes/checklogin.php, line 10

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.

✗ [Medium] Open Redirect

Path: includes/checklogin.php, line 10

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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\39.Hostel-management-System\hostel

Summary:

6 Code issues found

4 [Medium] 2 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\40.Maid-Hiring-Management-System\mhms"

Testing C:\xampp\htdocs\40.Maid-Hiring-Management-System\mhms ...

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: admin/login.php, line 24

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: admin/login.php, line 26

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: admin/login.php, line 29

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: admin/login.php, line 31

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] Use of Password Hash With Insufficient Computational Effort

Path: admin/add-maid.php, line 35

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: admin/add-maid.php, line 46

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: admin/changeidproof.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: admin/changeimage.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: admin/login.php, line 9

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: admin/change-password.php, line 12

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: admin/change-password.php, line 13

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: admin/forgot-password.php, line 10

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

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

Path: admin/login.php, line 24

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: admin/login.php, line 26

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: admin/login.php, line 29

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: admin/login.php, line 31

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.

✗ [High] SQL Injection

Path: admin/search-booking-request.php, line 102

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

✗ [High] SQL Injection

Path: admin/edit-category.php, line 85

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

✗ [High] SQL Injection

Path: admin/search-maid.php, line 114

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

✗ [High] Path Traversal

Path: admin/add-maid.php, line 36

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: admin/add-maid.php, line 47

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: admin/changeidproof.php, line 25

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: admin/changeimage.php, line 25

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/search-booking-request.php, line 84

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/search-maid.php, line 98

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/contact.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: admin/login.php, line 81

Info: Unsanitized input from cookies 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.php, line 85

Info: Unsanitized input from cookies 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\40.Maid-Hiring-Management-System\mhms

Summary:

28 Code issues found

12 [High] 16 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\41.Men-Salon-Management-System\msms"

Testing C:\xampp\htdocs\41.Men-Salon-Management-System\msms ...

✗ [Low] Use of Hardcoded Credentials

Path: admin/includes/dbconnection.php, line 2

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

✗ [Low] Use of Hardcoded Credentials

Path: includes/dbconnection.php, line 2

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

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

Path: admin/index.php, line 9

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: admin/reset-password.php, line 11

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: admin/change-password.php, line 11

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: admin/change-password.php, line 12

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

✗ [High] SQL Injection

Path: admin/view-appointment.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: admin/view-appointment.php, line 89

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: includes/footer.php, line 12

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/add-customer-services.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: admin/edit-services.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: admin/edit-services.php, line 84

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/admin-profile.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/edit-customer-detailed.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: admin/edit-customer-detailed.php, line 87

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/search-invoices.php, line 92

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/edit-appointment.php, line 68

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: appointment.php, line 17

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: appointment.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: admin/search-appointment.php, line 86

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/add-customer.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: admin/contact-us.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: admin/index.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: admin/forgot-password.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/add-services.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: admin/bwdates-reports-details.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: admin/sales-reports-detail.php, line 88

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/sales-reports-detail.php, line 128

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/about-us.php, line 15

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: admin/search-invoices.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: admin/search-appointment.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/bwdates-reports-details.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/bwdates-reports-details.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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\41.Men-Salon-Management-System\msms

Summary:

33 Code issues found

27 [High] 6 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\42.News-portal-Project\newsportal"

Testing C:\xampp\htdocs\42.News-portal-Project\newsportal ...

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

Path: admin/change-image.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: admin/add-post.php, line 34

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: admin/add-subadmins.php, line 15

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: admin/index.php, line 11

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: admin/forgot-password.php, line 10

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

✗ [Medium] SQL Injection

Path: news-details.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.

✗ [High] Path Traversal

Path: admin/change-image.php, line 28

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: admin/add-post.php, line 36

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: plugins/jquery.filer/php/remove\_file.php, line 5

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] SQL Injection

Path: admin/add-post.php, line 39

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/edit-post.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: admin/aboutus.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: admin/change-password.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: admin/add-subadmins.php, line 17

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/edit-subadmin.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/add-subcategory.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: admin/edit-subcategory.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: admin/contactus.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: news-details.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: admin/index.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: admin/edit-category.php, line 15

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/add-category.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: admin/check\_availability.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: admin/forgot-password.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/forgot-password.php, line 15

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/change-image.php, line 33

Info: Unsanitized input from an uploaded file 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: category.php, line 103

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 93

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: search.php, line 102

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: news-details.php, line 110

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: news-details.php, line 111

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: news-details.php, line 112

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: news-details.php, line 113

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\42.News-portal-Project\newsportal

Summary:

33 Code issues found

27 [High] 1 [Medium] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\43.Old-Age-Home-Management-system\oahms"

Testing C:\xampp\htdocs\43.Old-Age-Home-Management-system\oahms ...

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

Path: admin/add-scdetails.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: admin/change-image.php, line 19

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: admin/login.php, line 8

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: admin/resetpassword.php, line 11

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: admin/change-password.php, line 11

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: admin/change-password.php, line 12

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

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: admin/login.php, line 18

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: admin/login.php, line 20

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: admin/login.php, line 23

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: admin/login.php, line 25

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] Insufficient postMessage Validation

Path: admin/vendors/tinymce/plugins/codesample/plugin.js, line 544

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] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Path: admin/login.php, line 18

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: admin/login.php, line 20

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: admin/login.php, line 23

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: admin/login.php, line 25

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.

✗ [High] Path Traversal

Path: admin/add-scdetails.php, line 29

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: admin/change-image.php, line 20

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/add-scdetails.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: admin/login.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: admin/edit-scdetails.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: admin/aboutus.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: admin/forgot-password.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/add-services.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/manage-scdetails.php, line 7

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/eligibility.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: admin/bwdates-report-details.php, line 89

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/manage-services.php, line 7

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/contactus.php, line 15

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/view-enquiry.php, line 12

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/view-enquiry.php, line 86

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: contact.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/edit-services.php, line 12

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/search.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: admin/profile.php, line 15

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: search.php, line 98

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/rules.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: admin/change-image.php, line 22

Info: Unsanitized input from an uploaded file 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: admin/login.php, line 64

Info: Unsanitized input from cookies 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.php, line 67

Info: Unsanitized input from cookies 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/search.php, line 104

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: search.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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\43.Old-Age-Home-Management-system\oahms

Summary:

41 Code issues found

26 [High] 15 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\44.Online-Banquet-BooKing-System\obbs"

Testing C:\xampp\htdocs\44.Online-Banquet-BooKing-System\obbs ...

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

Path: forgot-password.php, line 10

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: signup.php, line 10

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 8

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: admin/forgot-password.php, line 10

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: change-password.php, line 11

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: change-password.php, line 12

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: admin/login.php, line 9

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: admin/change-password.php, line 12

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: admin/change-password.php, line 13

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

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

Path: admin/login.php, line 24

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: admin/login.php, line 26

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: admin/login.php, line 29

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: admin/login.php, line 31

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 Without 'HttpOnly' Flag

Path: admin/login.php, line 24

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: admin/login.php, line 26

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: admin/login.php, line 29

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: admin/login.php, line 31

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.

✗ [High] SQL Injection

Path: admin/booking-search.php, line 77

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

✗ [High] SQL Injection

Path: admin/view-user-queries.php, line 56

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

✗ [High] SQL Injection

Path: admin/booking-bwdates-reports-details.php, line 69

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/login.php, line 83

Info: Unsanitized input from cookies 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.php, line 91

Info: Unsanitized input from cookies 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/booking-search.php, line 60

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/booking-bwdates-reports-details.php, line 52

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/booking-bwdates-reports-details.php, line 52

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\44.Online-Banquet-BooKing-System\obbs

Summary:

25 Code issues found

8 [High] 17 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\45.online-dj-booking-management-system\odms"

Testing C:\xampp\htdocs\45.online-dj-booking-management-system\odms ...

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

Path: admin/login.php, line 9

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: admin/forgot-password.php, line 10

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: admin/change-password.php, line 12

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: admin/change-password.php, line 13

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

✗ [Low] Sensitive Cookie Without 'HttpOnly' Flag

Path: admin/login.php, line 24

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: admin/login.php, line 26

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: admin/login.php, line 29

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: admin/login.php, line 31

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 in HTTPS Session Without 'Secure' Attribute

Path: admin/login.php, line 24

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: admin/login.php, line 26

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: admin/login.php, line 29

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: admin/login.php, line 31

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.

✗ [High] SQL Injection

Path: admin/view-user-queries.php, line 56

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

✗ [High] SQL Injection

Path: admin/user-search.php, line 76

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

✗ [High] SQL Injection

Path: admin/booking-search.php, line 77

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

✗ [High] SQL Injection

Path: admin/booking-bwdates-reports-details.php, line 69

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/user-search.php, line 60

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/booking-search.php, line 60

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/booking-bwdates-reports-details.php, line 52

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/booking-bwdates-reports-details.php, line 52

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.php, line 83

Info: Unsanitized input from cookies 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.php, line 91

Info: Unsanitized input from cookies 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\45.online-dj-booking-management-system\odms

Summary:

22 Code issues found

10 [High] 12 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\46.Online-Library-Management-System\library"

Testing C:\xampp\htdocs\46.Online-Library-Management-System\library ...

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

Path: admin/chnage-bookimg.php, line 25

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: change-password.php, line 12

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: change-password.php, line 13

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: user-forgot-password.php, line 9

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: admin/change-bookimg.php, line 25

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: signup.php, line 19

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: index.php, line 12

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: adminlogin.php, line 11

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: admin/add-book.php, line 25

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: admin/change-password.php, line 12

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: admin/change-password.php, line 13

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

✗ [High] Path Traversal

Path: admin/chnage-bookimg.php, line 27

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: admin/change-bookimg.php, line 34

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: admin/add-book.php, line 34

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: admin/chnage-bookimg.php, line 39

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/change-bookimg.php, line 40

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] SQL Injection

Path: admin/student-history.php, line 102

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/student-history.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: admin/student-history.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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\46.Online-Library-Management-System\library

Summary:

19 Code issues found

8 [High] 11 [Low]