E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\11.Blood-Bank-&-Donor-Management-System\bbdms"

Testing C:\xampp\htdocs\11.Blood-Bank-&-Donor-Management-System\bbdms ...

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

Path: sign-up.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 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 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/forgot-password.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 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: change-password.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: login.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: admin/index.php, line 7

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/request-received-bydonar.php, line 105

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

✗ [High] SQL Injection

Path: admin/request-received-bydonar.php, line 132

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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\11.Blood-Bank-&-Donor-Management-System\bbdms

Summary:

10 Code issues found

2 [High] 8 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\12.COVID19-Testing-Management-System\covid-tms"

Testing C:\xampp\htdocs\12.COVID19-Testing-Management-System\covid-tms ...

✗ [Low] SQL Injection

Path: new-user-testing.php, line 21

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

✗ [Low] SQL Injection

Path: registered-user-testing.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.

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

Path: vendor/fontawesome-free/js/conflict-detection.js, line 521

Info: MD5 hash (used in rawMD5) is insecure. Consider changing it to a secure hashing algorithm.

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

Path: vendor/fontawesome-free/js/conflict-detection.js, line 565

Info: MD5 hash (used in rawMD5) is insecure. Consider changing it to a secure hashing algorithm.

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

Path: vendor/fontawesome-free/js/conflict-detection.js, line 562

Info: MD5 hash (used in hexMD5) is insecure. Consider changing it to a secure hashing algorithm.

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

Path: vendor/fontawesome-free/js/conflict-detection.js, line 587

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: vendor/fontawesome-free/js/conflict-detection.js, line 589

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: vendor/fontawesome-free/js/conflict-detection.js, line 592

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

✗ [Low] Use of Hardcoded Credentials

Path: includes/config.php, line 5

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: 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: test-details.php, line 48

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: 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: 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: change-password.php, line 14

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

✗ [Low] Cross-site Scripting (XSS)

Path: registered-user-testing.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] SQL Injection

Path: patient-report.php, line 99

Info: Unsanitized input from an HTTP parameter 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 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-phlebotomist.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: bwdates-report-ds.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: add-phlebotomist.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: 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: search-report-result.php, line 102

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

✗ [High] SQL Injection

Path: test-details.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: test-details.php, line 51

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

✗ [High] SQL Injection

Path: test-details.php, line 59

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

✗ [High] SQL Injection

Path: test-details.php, line 60

Info: Unsanitized input from an HTTP parameter 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 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: check\_availability.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: check\_availability.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: bwdates-report-result.php, line 103

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

✗ [High] SQL Injection

Path: test-details.php, line 52

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: patient-report.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: search-report-result.php, line 65

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-report-result.php, line 66

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-report-result.php, line 66

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

✗ [High] Path Traversal

Path: test-details.php, line 50

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

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\12.COVID19-Testing-Management-System\covid-tms

Summary:

37 Code issues found

22 [High] 15 [Low]

E:\Thesis\web app PT\Static-tools>

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\13.Doctor-Appointment-System\dams"

Testing C:\xampp\htdocs\13.Doctor-Appointment-System\dams ...

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

Path: doctor/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: doctor/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: doctor/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: doctor/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: doctor/signup.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: doctor/search.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: doctor/appointment-bwdates-reports-details.php, line 61

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: doctor/appointment-bwdates-reports-details.php, line 61

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: check-appointment.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] SQL Injection

Path: doctor/search.php, line 92

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: doctor/appointment-bwdates-reports-details.php, line 84

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: check-appointment.php, line 106

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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\13.Doctor-Appointment-System\dams

Summary:

12 Code issues found

7 [High] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\14.Hospital-Management-System\hospital"

Testing C:\xampp\htdocs\14.Hospital-Management-System\hospital ...

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

Path: hms/admin/registration.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: hms/doctor/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: hms/admin/add-doctor.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: hms/reset-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: hms/doctor/reset-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: hms/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: hms/change-password.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: hms/user-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: hms/doctor/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: hms/doctor/change-password.php, line 18

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: hms/registration.php, line 10

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

✗ [Medium] Open Redirect

Path: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1120

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] Open Redirect

Path: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1120

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] Open Redirect

Path: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1120

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] Open Redirect

Path: hms/include/checklogin.php, line 9

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: hms/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: hms/admin/vendor/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: hms/doctor/vendor/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: hms/vendor/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] Cleartext Transmission of Sensitive Information

Path: hms/admin/vendor/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] Cleartext Transmission of Sensitive Information

Path: hms/doctor/vendor/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] Cleartext Transmission of Sensitive Information

Path: hms/vendor/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] Debug Mode Enabled

Path: hms/vendor/jquery-file-upload/server/gae-python/main.py, line 164

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] Debug Mode Enabled

Path: hms/admin/vendor/jquery-file-upload/server/gae-python/main.py, line 164

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] Debug Mode Enabled

Path: hms/doctor/vendor/jquery-file-upload/server/gae-python/main.py, line 164

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.

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

Path: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1067

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1103

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1067

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1103

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1067

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1103

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1090

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1359

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1365

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1090

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1359

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1365

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1090

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1359

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1365

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1112

Info: Unsanitized input from an uploaded file 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] Server-Side Request Forgery (SSRF)

Path: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1112

Info: Unsanitized input from an uploaded file 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] Server-Side Request Forgery (SSRF)

Path: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1112

Info: Unsanitized input from an uploaded file 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: hms/doctor/search.php, line 75

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: hms/admin/patient-search.php, line 75

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: hms/admin/betweendates-detailsreports.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: hms/admin/betweendates-detailsreports.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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1105

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

✗ [High] Cross-site Scripting (XSS)

Path: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1105

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

✗ [High] Cross-site Scripting (XSS)

Path: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1105

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

✗ [High] Command Injection

Path: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 952

Info: Unsanitized input from an HTTP header flows into exec, where it is used to build a shell command. This may result in a Command Injection vulnerability.

✗ [High] Command Injection

Path: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 978

Info: Unsanitized input from an HTTP header flows into exec, where it is used to build a shell command. This may result in a Command Injection vulnerability.

✗ [High] Command Injection

Path: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 952

Info: Unsanitized input from an HTTP header flows into exec, where it is used to build a shell command. This may result in a Command Injection vulnerability.

✗ [High] Command Injection

Path: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 978

Info: Unsanitized input from an HTTP header flows into exec, where it is used to build a shell command. This may result in a Command Injection vulnerability.

✗ [High] Command Injection

Path: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 952

Info: Unsanitized input from an HTTP header flows into exec, where it is used to build a shell command. This may result in a Command Injection vulnerability.

✗ [High] Command Injection

Path: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 978

Info: Unsanitized input from an HTTP header flows into exec, where it is used to build a shell command. This may result in a Command Injection vulnerability.

✗ [High] SQL Injection

Path: hms/doctor/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: hms/doctor/index.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: hms/doctor/index.php, line 28

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

✗ [High] SQL Injection

Path: hms/admin/add-doctor.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: hms/user-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: hms/user-login.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: hms/user-login.php, line 29

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

✗ [High] SQL Injection

Path: hms/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: hms/doctor/search.php, line 91

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

✗ [High] SQL Injection

Path: hms/admin/patient-search.php, line 91

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

✗ [High] SQL Injection

Path: hms/admin/betweendates-detailsreports.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: hms/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: hms/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] SQL Injection

Path: hms/admin/edit-doctor-specialization.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: hms/doctor/forgot-password.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: hms/doctor/check\_availability.php, line 5

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

✗ [High] SQL Injection

Path: hms/change-emaild.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: hms/edit-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: hms/doctor/edit-patient.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: hms/doctor/edit-patient.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: hms/admin/change-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: hms/admin/change-password.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: hms/admin/view-patient.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: hms/admin/view-patient.php, line 82

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

✗ [High] SQL Injection

Path: hms/admin/view-patient.php, line 121

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

✗ [High] SQL Injection

Path: hms/forgot-password.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: hms/appointment-history.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: hms/view-medhistory.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: hms/view-medhistory.php, line 81

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

✗ [High] SQL Injection

Path: hms/view-medhistory.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: hms/doctor/edit-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: hms/admin/query-details.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: hms/admin/manage-users.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: 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: hms/doctor/add-patient.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: hms/doctor/appointment-history.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: hms/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: hms/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: hms/admin/contact.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: hms/admin/doctor-specilization.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: hms/admin/doctor-specilization.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: hms/admin/manage-doctors.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: hms/doctor/view-patient.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: hms/doctor/view-patient.php, line 82

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

✗ [High] SQL Injection

Path: hms/doctor/view-patient.php, line 121

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

✗ [High] SQL Injection

Path: hms/admin/edit-doctor.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: hms/book-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: hms/get\_doctor.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: hms/get\_doctor.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: hms/admin/registration.php, line 11

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

✗ [High] Path Traversal

Path: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1065

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1075

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1065

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1075

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1065

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1075

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1067

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1103

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1067

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1103

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1067

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1103

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1071

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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1071

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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1071

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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1090

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1359

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1365

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1090

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1359

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1365

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1090

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1359

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1365

Info: Unsanitized input from an uploaded file 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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1112

Info: Unsanitized input from an uploaded file 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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1112

Info: Unsanitized input from an uploaded file 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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1112

Info: Unsanitized input from an uploaded file 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] Regular Expression Denial of Service (ReDoS)

Path: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 368

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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 401

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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1012

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: hms/admin/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1207

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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 368

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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 401

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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1012

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: hms/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1207

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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 368

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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 401

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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1012

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: hms/doctor/vendor/jquery-file-upload/server/php/UploadHandler.php, line 1207

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\14.Hospital-Management-System\hospital

Summary:

145 Code issues found

120 [High] 14 [Medium] 11 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\15.Online-Birth-Certificate-System\obcs"

Testing C:\xampp\htdocs\15.Online-Birth-Certificate-System\obcs ...

✗ [Low] Use of Hardcoded Credentials

Path: admin/download-certificate.php, line 5

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

✗ [Low] Use of Hardcoded Credentials

Path: user/download-certificate.php, line 5

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/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/dompdf/vendor/dompdf/dompdf/lib/Cpdf.php, line 2958

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/dompdf/vendor/dompdf/dompdf/lib/Cpdf.php, line 2991

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/dompdf/vendor/dompdf/dompdf/lib/Cpdf.php, line 3189

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 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: user/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/dompdf/vendor/phenx/php-svg-lib/src/Svg/Surface/CPdf.php, line 2899

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/dompdf/vendor/phenx/php-svg-lib/src/Svg/Surface/CPdf.php, line 2932

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/dompdf/vendor/phenx/php-svg-lib/src/Svg/Surface/CPdf.php, line 3130

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/register.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/dompdf/vendor/dompdf/dompdf/src/FontMetrics.php, line 186

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/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: user/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/dompdf/vendor/dompdf/dompdf/lib/Cpdf.php, line 2944

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

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

Path: admin/dompdf/vendor/phenx/php-svg-lib/src/Svg/Surface/CPdf.php, line 2885

Info: MD5 hash (used in md5\_file) 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.

✗ [Low] XML External Entity (XXE) Injection

Path: admin/dompdf/vendor/phenx/php-svg-lib/src/Svg/Document.php, line 128

Info: Unsanitized input from data from a remote resource flows to xml\_parse. This may result in an XXE vulnerability. You may be vulnerable if using an old version of PHP (<8.0)

✗ [Low] XML External Entity (XXE) Injection

Path: admin/dompdf/vendor/phenx/php-svg-lib/src/Svg/Document.php, line 225

Info: Unsanitized input from data from a remote resource flows to xml\_parse. This may result in an XXE vulnerability. You may be vulnerable if using an old version of PHP (<8.0)

✗ [Medium] Path Traversal

Path: admin/dompdf/vendor/masterminds/html5/src/HTML5.php, line 208

Info: Unsanitized input from data from a remote resource 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] Cross-site Scripting (XSS)

Path: admin/login.php, line 127

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 140

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: user/search.php, line 156

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: user/search.php, line 156

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/users-applications.php, line 99

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.php, line 147

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/bwdates-reports-details.php, line 121

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 121

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

✗ [High] SQL Injection

Path: admin/users-applications.php, line 135

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.php, line 168

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/bwdates-reports-details.php, line 142

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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\15.Online-Birth-Certificate-System\obcs

Summary:

42 Code issues found

11 [High] 1 [Medium] 30 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\16.student-study-center\sscms"

Testing C:\xampp\htdocs\16.student-study-center\sscms ...

✗ [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/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/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] Cross-site Scripting (XSS)

Path: admin/report.php, line 96

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/report.php, line 96

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\16.student-study-center\sscms

Summary:

6 Code issues found

2 [High] 4 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\17.Teacher-Subject-Allocation-System\tsas"

Testing C:\xampp\htdocs\17.Teacher-Subject-Allocation-System\tsas ...

✗ [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] 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/changeimage.php, line 22

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/edit-teacher-info.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/add-teacher.php, line 33

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 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/teacher-info.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] SQL Injection

Path: admin/search.php, line 129

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/changeimage.php, line 89

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-course.php, line 84

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: index.php, line 83

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-teacher-info.php, line 99

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.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: admin/search.php, line 238

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 67

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

✗ [High] Cross-site Scripting (XSS)

Path: admin/manage-teacher.php, line 191

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 72

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 76

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] Path Traversal

Path: admin/changeimage.php, line 23

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-teacher.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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\17.Teacher-Subject-Allocation-System\tsas

Summary:

29 Code issues found

14 [High] 15 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\18.Teachers-Record-Management-System\trms"

Testing C:\xampp\htdocs\18.Teachers-Record-Management-System\trms ...

✗ [Low] Cross-site Scripting (XSS)

Path: admin/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).

✗ [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).

✗ [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] Allocation of Resources Without Limits or Throttling

Path: admin/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: admin/changeimage.php, line 21

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

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

Path: admin/add-teacher.php, line 41

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: teacher/changeimage.php, line 21

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

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

Path: teacher/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] Use of Password Hash With Insufficient Computational Effort

Path: teacher/signup.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: teacher/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: teacher/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: teacher/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.

✗ [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] 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] 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: admin/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] Cleartext Transmission of Sensitive Information

Path: admin/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.

✗ [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: admin/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.

✗ [Medium] Path Traversal

Path: admin/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.

✗ [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: admin/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.

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

Path: admin/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] 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] Cross-site Scripting (XSS)

Path: admin/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: admin/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: search-result.php, line 49

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/queries.php, line 46

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.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] SQL Injection

Path: admin/bwdates-reports-details.php, line 98

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: search-result.php, line 57

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/queries.php, line 89

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.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] SQL Injection

Path: admin/changeimage.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] SQL Injection

Path: admin/edit-subjects-detail.php, line 92

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/add-teacher.php, line 24

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-teacher-detail.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/changeimage.php, line 22

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-teacher.php, line 42

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: teacher/changeimage.php, line 22

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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\18.Teachers-Record-Management-System\trms

Summary:

44 Code issues found

18 [High] 6 [Medium] 20 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\19.Art-Gallery-MS\agms"

Testing C:\xampp\htdocs\19.Art-Gallery-MS\agms ...

✗ [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 Password Hash With Insufficient Computational Effort

Path: admin/assets/morris.js-0.4.3/spec/vendor/sinon-1.5.0.js, line 189

Info: sha1 hash (used in crypto.createHash) is insecure. Consider changing it to a secure hashing algorithm.

✗ [Low] Insufficient postMessage Validation

Path: admin/assets/morris.js-0.4.3/spec/vendor/mocha-1.6.0.js, line 4825

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] Insufficient postMessage Validation

Path: js/minicart.js, line 1009

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: 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] Use of Password Hash With Insufficient Computational Effort

Path: admin/changeimage4.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/changepropic.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/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/changeimage3.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/changeimage1.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/changeimage2.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/add-art-product.php, line 65

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/changeimage.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/add-artist.php, line 30

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

✗ [Medium] Permissive Cross-domain Policy

Path: admin/assets/morris.js-0.4.3/spec/vendor/mocha-1.6.0.js, line 4834

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: js/minicart.js, line 1022

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

✗ [Medium] Open Redirect

Path: admin/js/form-component.js, line 81

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.

✗ [High] Cross-site Scripting (XSS)

Path: product.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: admin/search.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: search.php, line 64

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

✗ [High] Path Traversal

Path: admin/changeimage4.php, line 26

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/changepropic.php, line 26

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/changeimage3.php, line 26

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/changeimage1.php, line 26

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/changeimage2.php, line 26

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-art-product.php, line 70

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 26

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-artist.php, line 31

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: product.php, line 101

Info: Unsanitized input from an HTTP parameter 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 102

Info: Unsanitized input from an HTTP parameter 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 102

Info: Unsanitized input from an HTTP parameter 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/changeimage4.php, line 106

Info: Unsanitized input from an HTTP parameter 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/changepropic.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: admin/login.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/changeimage3.php, line 106

Info: Unsanitized input from an HTTP parameter 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/changeimage1.php, line 106

Info: Unsanitized input from an HTTP parameter 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/changeimage2.php, line 106

Info: Unsanitized input from an HTTP parameter 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-art-product.php, line 75

Info: Unsanitized input from an HTTP parameter 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/changeimage.php, line 106

Info: Unsanitized input from an HTTP parameter 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-artist.php, line 32

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

✗ [High] SQL Injection

Path: admin/edit-art-type-detail.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-art-type-detail.php, line 95

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

✗ [High] SQL Injection

Path: single-product.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: admin/edit-art-product-detail.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: admin/edit-art-product-detail.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: admin/contactus.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: art-enquiry.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/view-enquiry-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: admin/view-enquiry-detail.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-art-type.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/aboutus.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/edit-art-medium-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: admin/edit-art-medium-detail.php, line 91

Info: Unsanitized input from an HTTP parameter 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/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/add-art-medium.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/edit-artist-detail.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: admin/edit-artist-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: admin/changeimage4.php, line 27

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] SQL Injection

Path: admin/changepropic.php, line 27

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] SQL Injection

Path: admin/changeimage3.php, line 27

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] SQL Injection

Path: admin/changeimage1.php, line 27

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] SQL Injection

Path: admin/changeimage2.php, line 27

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] SQL Injection

Path: admin/changeimage.php, line 27

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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\19.Art-Gallery-MS\agms

Summary:

67 Code issues found

47 [High] 3 [Medium] 17 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\20.Auto-Taxi-Stand-Management-System-project\atsms"

Testing C:\xampp\htdocs\20.Auto-Taxi-Stand-Management-System-project\atsms ...

✗ [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/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/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: admin/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: admin/search-autoortaxi.php, line 74

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 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/bwdates-reports-details.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: search.php, line 73

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

✗ [High] SQL Injection

Path: admin/search-autoortaxi.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/bwdates-reports-details.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: search.php, line 91

Info: Unsanitized input from an HTTP parameter 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/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: print.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/print.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/auto-taxi-entry-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: admin/auto-taxi-entry-detail.php, line 95

Info: Unsanitized input from an HTTP parameter 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/new-autoortaxi-entry-form.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/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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\20.Auto-Taxi-Stand-Management-System-project\atsms

Summary:

20 Code issues found

15 [High] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\21.Apartment-Visitors-Management-System\avms"

Testing C:\xampp\htdocs\21.Apartment-Visitors-Management-System\avms ...

✗ [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] SQL Injection

Path: visitors-form.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: 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: create-pass.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: pass-details.php, line 76

Info: Unsanitized input from an HTTP parameter 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: bwdates-passreports-details.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: 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: category.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] 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] Cross-site Scripting (XSS)

Path: bwdates-passreports-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-passreports-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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\21.Apartment-Visitors-Management-System\avms

Summary:

22 Code issues found

17 [High] 5 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\22.Bank-locker-Managament-System\blms"

Testing C:\xampp\htdocs\22.Bank-locker-Managament-System\blms ...

✗ [Low] Use of Hardcoded Credentials

Path: banker/includes/config.php, line 5

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: banker/changeimage1.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: banker/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: banker/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: banker/password-recovery.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: banker/add-locker-form.php, line 43

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: banker/add-locker-form.php, line 44

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: banker/add-subadmin.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: banker/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: banker/change-password.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: banker/reset-subadmin-pwd.php, line 13

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

✗ [High] SQL Injection

Path: banker/contact-us.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: banker/aboutus.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: banker/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: search-locker-details.php, line 58

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

✗ [High] SQL Injection

Path: banker/edit-locker.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: banker/changeimage1.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: banker/changeidproof.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: banker/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: banker/password-recovery.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: banker/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: banker/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: banker/add-locker-form.php, line 46

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

✗ [High] SQL Injection

Path: banker/add-locker-form.php, line 54

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

✗ [High] SQL Injection

Path: banker/search-report-details.php, line 93

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

✗ [High] SQL Injection

Path: banker/edit-subadmin.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: banker/edit-assign-locker.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: banker/add-subadmin.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: banker/add-lockertype.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: banker/changeimage1.php, line 26

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] SQL Injection

Path: banker/changeidproof.php, line 26

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] Path Traversal

Path: banker/changeimage1.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: banker/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: banker/add-locker-form.php, line 45

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

✗ [High] Path Traversal

Path: banker/add-locker-form.php, line 53

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: search-locker-details.php, line 38

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

✗ [High] Cross-site Scripting (XSS)

Path: banker/search-report-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\22.Bank-locker-Managament-System\blms

Summary:

37 Code issues found

26 [High] 11 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\23.BP-Monitoring-Management-System\bpmms"

Testing C:\xampp\htdocs\23.BP-Monitoring-Management-System\bpmms ...

✗ [Low] Use of Hardcoded Credentials

Path: includes/config.php, line 5

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: vendor/fontawesome-free/js/conflict-detection.js, line 521

Info: MD5 hash (used in rawMD5) is insecure. Consider changing it to a secure hashing algorithm.

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

Path: vendor/fontawesome-free/js/conflict-detection.js, line 565

Info: MD5 hash (used in rawMD5) is insecure. Consider changing it to a secure hashing algorithm.

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

Path: vendor/fontawesome-free/js/conflict-detection.js, line 562

Info: MD5 hash (used in hexMD5) is insecure. Consider changing it to a secure hashing algorithm.

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

Path: vendor/fontawesome-free/js/conflict-detection.js, line 587

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: vendor/fontawesome-free/js/conflict-detection.js, line 589

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: vendor/fontawesome-free/js/conflict-detection.js, line 592

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

✗ [High] Cross-site Scripting (XSS)

Path: bwdates-report-result.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: bwdates-report-result.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: bwdates-report-result.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] SQL Injection

Path: bwdates-report-result.php, line 85

Info: Unsanitized input from an HTTP parameter 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-report-result.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: edit-family-member.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: login.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: change-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: change-password.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: add-bp-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: 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 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: registration.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: check\_availability.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: add-family-member.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: profile.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: bwdates-report-ds.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: registration.php, line 17

Info: Unsanitized input from an HTTP parameter flows into mysqli\_multi\_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\23.BP-Monitoring-Management-System\bpmms

Summary:

25 Code issues found

18 [High] 7 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\24.Bus-Pass-Management-System\buspassms"

Testing C:\xampp\htdocs\24.Bus-Pass-Management-System\buspassms ...

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

Path: admin/index.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/index.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/index.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/index.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] 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/changeimage.php, line 22

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-pass.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/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/index.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/index.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/index.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/index.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] Cross-site Scripting (XSS)

Path: admin/index.php, line 76

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/index.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: download-pass.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/pass-bwdates-reports-details.php, line 65

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/pass-bwdates-reports-details.php, line 65

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-pass.php, line 74

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

✗ [High] Path Traversal

Path: admin/changeimage.php, line 23

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-pass.php, line 35

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

✗ [High] SQL Injection

Path: download-pass.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/pass-bwdates-reports-details.php, line 81

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-pass.php, line 90

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/changeimage.php, line 83

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-detail.php, line 90

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: view-pass-detail.php, line 63

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-pass-detail.php, line 98

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-pass-detail.php, line 66

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-enquiry.php, line 66

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.

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\24.Bus-Pass-Management-System\buspassms

Summary:

31 Code issues found

17 [High] 14 [Low]

E:\Thesis\web app PT\Static-tools>snyk.exe code test "C:\xampp\htdocs\25.Car-Rental-Portal\carrental"

Testing C:\xampp\htdocs\25.Car-Rental-Portal\carrental ...

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

Path: includes/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 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 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: includes/login.php, line 5

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: update-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: update-password.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.

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

Path: includes/forgotpassword.php, line 6

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

✗ [High] Path Traversal

Path: admin/changeimage2.php, line 15

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/changeimage5.php, line 15

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/changeimage3.php, line 15

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/changeimage4.php, line 15

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/post-avehical.php, line 37

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/post-avehical.php, line 38

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/post-avehical.php, line 39

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

✗ [High] Path Traversal

Path: admin/post-avehical.php, line 40

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

✗ [High] Path Traversal

Path: admin/post-avehical.php, line 41

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

✗ [High] Path Traversal

Path: admin/changeimage1.php, line 15

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: includes/login.php, line 17

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: search.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).

✔ Test completed

Organization: daisy2310

Test type: Static code analysis

Project path: C:\xampp\htdocs\25.Car-Rental-Portal\carrental

Summary:

20 Code issues found

12 [High] 8 [Low]