Task-2 Report
Security Implementation

Prepared by: Abdullah Akram
ID: DHC-3606

Date: July 9, 2025

A Comprehensive Report on Implementing Security Measures for a PHP-Based Web Application

Home Rental System

1 Introduction

This report outlines the implementation of security measures for a PHP-based web application as part of Task-2. The focus was on enhancing the security of the login, registration, and HTTP headers to protect user data and prevent vulnerabilities.

2 Security Measures Implemented

2.1 Input Validation and Sanitization

- **File Modified**: `register.php`
- **Action**: Added input sanitization using `filter_var(trim($email), FILTER_SANITIZE_EMAIL` and validation with `filter_var($email, FILTER_VALIDATE_EMAIL)` to ensure valid email formats.
- **Purpose**: Prevents injection attacks and ensures clean input data.

2.2 Password Hashing

- **Files Modified**: `register.php`, `login.php`
- **Action**: Replaced insecure `md5()` with `password_hash($password, PASSWORD_BCRYPT)` for storing passwords and `password_verify()` for login verification.
- **Purpose**: Ensures passwords are securely hashed, making them resistant to brute-force attacks.

2.3 Secure HTTP Headers

- **File Modified**: `header.php`
- **Action**: Added headers such as:
  - `X-Content-Type-Options: nosniff`
  - `X-Frame-Options: SAMEORIGIN`
  - `X-XSS-Protection: 1; mode=block`
  - `Strict-Transport-Security: max-age=31536000; includeSubDomains`
- **Purpose**: Protects against MIME-type sniffing, clickjacking, XSS attacks, and enforces HTTPS.

3 Challenges Faced

- **Issue**: Existing users' passwords stored in `md5()` format were incompatible with `password_verify()`.
- **Solution**: Recommended manual or automated password upgrades using `password_hash()`.
- **Issue**: Blank page on `register.php`.

- **Solution**: Ensured all required files (`config.php`, `PHPMailer`) were correctly included and added error reporting.

4 Conclusion

The implemented security measures significantly enhance the application's security by ensuring proper input handling, secure password storage, and robust HTTP headers. Future improvements could include JWT-based authentication for API endpoints.