Secure Programming Project Documentation Report

1. Cover Page

PROJECT REPORT SECURE PROGRAMMING (COMP6695001)

“WareHouse”

Prepared By:

1. Nova Adji Abimanyu – 2502036533
2. Maghfirli Alif Al Ayubi – 2502022925
3. Shearen Berlian Eshar – 2702253906
4. Steve Manuel – 2702321775
5. Claudia Trisna Jaya – 2702329954
6. Janssen Bryant Ardi Tanyid – 2440030046

Lecturer: Christopher Limawan, S.Kom., M.Kom. (D6954 – Primary Instructor)

Course: COMP6695001 – Secure Programming

Program: Bachelor of Cyber Security

School of Computer Science

BINUS University Submission Date: 19 Desember 2025

Group: 2

2. Executive Summary

Website Warehouse dikembangkan sebagai solusi modern untuk mempermudah proses pencatatan, pengelolaan, dan pengawasan barang di dalam gudang. Selama ini, proses manajemen gudang sering dilakukan secara manual, sehingga rentan terhadap kesalahan pencatatan, keterlambatan informasi, dan kesulitan dalam memantau ketersediaan stok secara akurat. Melalui aplikasi ini, manajemen inventaris diharapkan dapat berjalan lebih efektif, efisien, dan terstruktur dengan baik.

Aplikasi ini menyediakan fitur-fitur penting seperti pengelolaan data barang, pembaruan stok, pemantauan aktivitas, serta sistem akses berbasis peran untuk memastikan setiap pengguna hanya dapat mengakses fungsi sesuai kewenangannya. Sejalan dengan tujuan mata kuliah Secure Programming, pengembangan aplikasi ini juga menerapkan berbagai praktik keamanan, seperti hashing password, validasi input, pencegahan SQL Injection dengan prepared statements, perlindungan session dan cookie, serta mekanisme penanganan error yang aman.

Secara keseluruhan, proyek ini menunjukkan penerapan prinsip secure software engineering dengan menggabungkan kebutuhan fungsional pengelolaan gudang dan penerapan kontrol keamanan yang kuat, guna meningkatkan akurasi operasional, efisiensi proses, serta menjaga integritas data.

3. Background & Objectives

3.1 Background

Pengelolaan gudang merupakan salah satu aktivitas penting dalam operasional bisnis, terutama bagi perusahaan yang bergantung pada ketersediaan barang dan kelancaran distribusi. Namun, dalam banyak kasus, proses pencatatan serta pemantauan stok masih dilakukan secara manual menggunakan kertas atau spreadsheet sederhana. Pendekatan tersebut memiliki berbagai keterbatasan, seperti:

- Tingginya risiko kesalahan pencatatan.
- Kesulitan memantau perubahan stok secara real-time.
- Ketidakefisienan dalam pencarian data barang.
- Tidak adanya kontrol akses yang jelas antar pengguna.
- Sulitnya melakukan audit atau pelacakan aktivitas.

Masalah-masalah tersebut dapat mengganggu produktivitas, memperlambat proses operasional, dan berpotensi menyebabkan kerugian akibat kesalahan manajemen stok. Oleh karena itu, dibutuhkan sebuah solusi digital yang mampu mengotomatisasi proses, meningkatkan akurasi data, serta memberikan pengawasan yang lebih baik.

Website Warehouse dikembangkan sebagai jawaban atas kebutuhan tersebut. Selain menyediakan fitur pengelolaan inventaris yang terstruktur, aplikasi ini juga dirancang dengan menerapkan prinsip Secure Programming untuk memastikan keamanan data dan mencegah potensi kerentanan yang umum ditemukan pada aplikasi web.

3.2 Objectives

A. Functional Objectives (Tujuan Fungsional)

Aplikasi Warehouse bertujuan untuk menyediakan fitur-fitur berikut:

1. Pencatatan dan pengelolaan data barang secara terpusat.
2. Pembaruan stok (stock in / stock out) dengan mudah dan akurat.
3. Sistem login dan autentikasi pengguna untuk akses ke sistem.
4. Role-based access control (RBAC) agar setiap pengguna memiliki hak akses sesuai kewenangannya.

5. Pemantauan aktivitas agar perubahan terkait stok maupun data barang dapat dilacak.
6. Penyajian informasi stok secara real-time untuk mendukung pengambilan keputusan.

B. Non-Functional Objectives (Tujuan Non-Fungsional)

Untuk mendukung kinerja aplikasi, tujuan non-fungsional yang diimplementasikan adalah:

1. Keamanan (Security):
    a. Hashing password
    b. Validasi input
    c. Pencegahan SQL Injection
    d. Proteksi session dan cookie
    e. Penanganan error yang aman
2. Reliability:
   Sistem harus mampu berjalan stabil dalam penggunaan jangka panjang.
3. Usability:
   Tampilan antarmuka dibuat sederhana, mudah digunakan, dan mudah dipahami.
4. Efficiency:
   Proses pemanggilan data dan pengelolaan stok dilakukan secara cepat dan efisien.
5. Maintainability:
   Struktur kode dibuat rapi dan modular agar mudah diperbarui di masa depan.

4. Scope & User Roles

4.1 Scope (Ruang Lingkup Proyek)

Proyek Warehouse Web Application berfungsi sebagai sistem manajemen gudang yang mendukung proses registrasi pengguna, verifikasi email, approval admin, manajemen stok barang, pelacakan aktivitas, dan keamanan berbasis OTP. Ruang lingkup sistem ini meliputi fitur-fitur berikut:

A. Fitur-Fitur Sistem

1. Registrasi & Verifikasi Pengguna

1. User melakukan registrasi dengan memasukkan:
   Nama, Email, Password, Confirm Password
2. Sistem mengirimkan OTP ke email user untuk verifikasi email.
3. Setelah email terverifikasi, akun user masih dalam status inactive.
4. Admin harus menyetujui/approve akun agar user dapat login.
5. Setelah aktif, user dapat menggunakan aplikasi.

2. Login dengan OTP

- User login menggunakan email + password.
- Sistem mengirimkan OTP login ke email untuk keamanan tambahan.
- User hanya dapat login jika:
  - Email sudah diverifikasi
  - Akun sudah di-approve admin

3. Manajemen Barang & Stok

Untuk User:

- Barang Masuk:
  User hanya bisa memasukkan barang yang sudah tersedia di kategori.
  Jika kategori belum ada → user harus membuat request barang baru.
- Barang Keluar:
  User mengajukan permintaan (request) untuk mengeluarkan barang.
  Permintaan ini harus disetujui admin sebelum barang keluar.
- Request Barang:
  User dapat request barang baru / kategori baru / barang lama yang stoknya habis.

Untuk Admin:

- Menambah barang
- Mengedit barang
- Menghapus barang
- Memproses barang masuk dan barang keluar
- Melihat seluruh stok dan status barang

4. Approvals (Proses Persetujuan oleh Admin)

Admin memiliki panel khusus untuk memverifikasi:

1. Request barang baru
2. Request barang masuk
3. Request barang keluar
4. Aktivasi akun user baru

Semua tindakan sensitif harus melalui admin untuk alasan keamanan dan untuk memastikan tracking yang jelas.

5. User Profile

1. User dapat mengubah:
2. Foto profil
3. Username / nama tampilan
4. Email & role tidak bisa diubah demi keamanan.

6. Lupa Password (Forgot Password)

1. User memasukkan email
2. Sistem mengirimkan OTP reset password
3. Setelah OTP valid, user dapat mengatur password baru

7. Dashboard User

Menampilkan:

1. Informasi stok
2. Barang kritis (stok minimum)
3. Aksi cepat:

4. Barang masuk
5. Barang keluar
6. Cek stok
7. Request barang
8. Riwayat request

8. Dashboard Admin

Admin memiliki dashboard yang lebih lengkap:

1. Total barang
2. Barang kritis
3. Stok per kategori
4. Notifikasi barang keluar hari ini
5. Request pending dari user
6. Statistik aktivitas

9. Laporan

1. Admin dapat melihat laporan:
2. Barang masuk (history)
3. Barang keluar (history)
4. Dengan kolom:
5. Nomor
6. Nama barang
7. Jumlah
8. Tanggal
9. User yang memproses

4.2 User Roles (Peran Pengguna)

Aplikasi Warehouse menggunakan sistem akses berbasis peran (Role-Based Access Control) untuk membatasi tindakan pengguna sesuai tanggung jawab mereka di dalam sistem.

Berikut peran pengguna yang terdapat dalam aplikasi:

1. Admin

Hak & Akses:

- Mengelola seluruh user dalam sistem
- Menambah, mengubah, menghapus, dan melihat seluruh barang
- Melakukan stock in dan stock out
- Melihat seluruh riwayat aktivitas
- Mengakses seluruh fitur tanpa batasan

Fungsi Utama:

- Mengawasi jalannya sistem
- Memastikan data barang dan stok selalu akurat
- Menjaga integritas data

2. User

Hak & Akses:

- Melihat daftar barang
- Menambah dan mengurangi stok
- Mengedit data barang tertentu (jika diizinkan)
- Melihat riwayat aktivitas yang relevan

Fungsi Utama:

- Melakukan operasi harian gudang
- Memastikan stok tercatat dengan benar
- Membantu admin dalam pemutakhiran data

4.3 Batasan Ruang Lingkup

Aplikasi tidak mencakup hal-hal berikut:

- Integrasi dengan hardware gudang (barcode scanner, RFID)
- Otomasi logistik atau perhitungan pengiriman
- Mobile application
- Multi-gudang (hanya 1 gudang dalam scope project)
- Sistem laporan PDF otomatis (jika tidak dibuat)

5. Technology Stack

Pengembangan Website Warehouse dilakukan menggunakan teknologi web yang umum digunakan dan stabil, dengan fokus pada kemudahan pengembangan serta penerapan aspek keamanan (*secure programming*). Adapun teknologi yang digunakan dalam pengembangan aplikasi ini adalah sebagai berikut:

5.1 Backend

Aplikasi ini dikembangkan menggunakan PHP Native sebagai bahasa pemrograman sisi server.

PHP digunakan untuk:

- Mengelola proses autentikasi dan otorisasi pengguna
- Mengelola logika bisnis aplikasi
- Memproses input pengguna
- Berinteraksi dengan database
- Mengimplementasikan mekanisme keamanan seperti hashing password, OTP, dan validasi data

Penggunaan PHP Native memungkinkan pengembang untuk memiliki kontrol penuh terhadap struktur kode dan penerapan keamanan secara manual sesuai kebutuhan sistem.

5.2 Database

Aplikasi Warehouse menggunakan MySQL sebagai sistem manajemen basis data.

MySQL digunakan untuk:

- Menyimpan data pengguna
- Menyimpan data barang dan kategori
- Menyimpan data stok masuk dan keluar
- Menyimpan data request barang
- Menyimpan log aktivitas dan laporan

Database dirancang menggunakan relasi antar tabel untuk menjaga konsistensi dan integritas data.

5.3 Frontend

Antarmuka pengguna dikembangkan menggunakan:

- HTML sebagai struktur tampilan
- CSS untuk pengaturan tampilan dan layout
- JavaScript untuk interaksi dasar dan validasi sisi klien

Frontend dirancang dengan tampilan sederhana dan mudah digunakan agar mendukung kenyamanan pengguna dalam mengoperasikan sistem.

5.4 Email Service

Untuk keperluan keamanan dan verifikasi pengguna, aplikasi menggunakan layanan SMTP Email.

SMTP digunakan untuk:

- Pengiriman OTP verifikasi email saat registrasi
- OTP login
- OTP reset password

Penggunaan SMTP memastikan bahwa proses autentikasi berbasis email berjalan dengan aman dan andal.

5.5 Security Libraries & Tools

Beberapa fitur keamanan yang diimplementasikan menggunakan:

- Password hashing menggunakan fungsi bawaan PHP
- Prepared statements untuk mencegah SQL Injection
- Session management PHP untuk pengelolaan sesi login
- Input validation di sisi server dan klien

6. System Flow & Architecture

6.1 System Architecture Overview

Website Warehouse menggunakan arsitektur web-based client–server, di mana pengguna mengakses sistem melalui browser, kemudian permintaan diproses oleh server menggunakan PHP Native, dan data disimpan pada database MySQL.

Secara umum, arsitektur sistem terdiri dari tiga lapisan utama:

1. **Presentation Layer (Frontend)**
    a. Dibangun menggunakan HTML, CSS, dan JavaScript
    b. Menyediakan antarmuka bagi user dan admin
    c. Menampilkan data hasil proses dari server
2. **Application Layer (Backend)**
    a. Menggunakan PHP Native
    b. Mengelola logika bisnis aplikasi
    c. Menangani autentikasi, otorisasi, OTP, dan approval
    d. Mengelola request barang, stok masuk, dan stok keluar
3. **Data Layer (Database)**
    a. Menggunakan MySQL
    b. Menyimpan seluruh data pengguna, barang, stok, request, dan log aktivitas

Selain itu, sistem terhubung dengan SMTP Email Server untuk pengiriman OTP dan notifikasi keamanan.

6.2 Alur Sistem (System Flow)

6.2.1 Alur Registrasi dan Aktivasi Akun

1. User melakukan registrasi dengan mengisi nama, email, password, dan konfirmasi password.
2. Sistem melakukan validasi input dan hashing password.
3. Sistem mengirimkan OTP verifikasi email ke email user melalui SMTP.
4. User memasukkan OTP untuk memverifikasi email.
5. Setelah email terverifikasi, akun user berstatus menunggu persetujuan admin.
6. Admin melakukan approval akun user melalui menu user management.
7. Setelah akun diaktifkan oleh admin, user dapat login ke sistem.

6.2.2 Alur Login dengan OTP

1. User memasukkan email dan password.
2. Sistem memverifikasi kredensial user.
3. Jika valid, sistem mengirimkan OTP login ke email user.
4. User memasukkan OTP login.
5. Sistem memverifikasi OTP dan membuat session login.
6. User diarahkan ke dashboard sesuai dengan role (admin atau user).

6.2.3 Alur Barang Masuk

User:

1. User memilih menu barang masuk.
2. User hanya dapat memasukkan barang yang sudah tersedia dalam daftar kategori.
3. Jika kategori barang belum tersedia, user harus melakukan request barang baru.

Admin:

1. Admin menerima notifikasi barang masuk atau request barang.
2. Admin melakukan pengecekan dan menyetujui proses barang masuk.
3. Sistem memperbarui stok barang di database.

6.2.4 Alur Barang Keluar

1. User mengajukan permintaan barang keluar melalui sistem.
2. Permintaan masuk ke tabel verifikasi admin.
3. Admin melakukan pengecekan permintaan.
4. Jika disetujui, admin memproses barang keluar.
5. Sistem mencatat transaksi barang keluar beserta identitas user.

6.2.5 Alur Request Barang

1. User mengajukan request barang baru atau request barang lama yang stoknya habis.
2. Request masuk ke sistem admin.
3. Admin melakukan evaluasi request.
4. Jika disetujui, admin membuat kategori atau menambahkan barang ke sistem.
5. User dapat melanjutkan proses barang masuk setelah request disetujui.

6.2.6 Alur Lupa Password

1. User memilih fitur lupa password.
2. User memasukkan email terdaftar.
3. Sistem mengirimkan OTP reset password ke email user.
4. User memasukkan OTP dan membuat password baru.
5. Sistem menyimpan password baru dalam bentuk hash.

6.3 Alur Dashboard Admin

Dashboard admin menampilkan:

- Total barang
- Stok barang per kategori
- Barang kritis (stok di bawah minimum)
- Jumlah barang keluar hari ini
- Request user yang menunggu persetujuan

Dashboard ini membantu admin dalam melakukan pengawasan dan pengambilan keputusan.

6.4 Keamanan dalam Arsitektur Sistem

Keamanan diterapkan pada setiap alur sistem, antara lain:

- OTP untuk registrasi, login, dan reset password
- Approval admin untuk akun user dan transaksi sensitif
- Session management yang aman
- Validasi input di setiap form
- Prepared statements untuk akses database

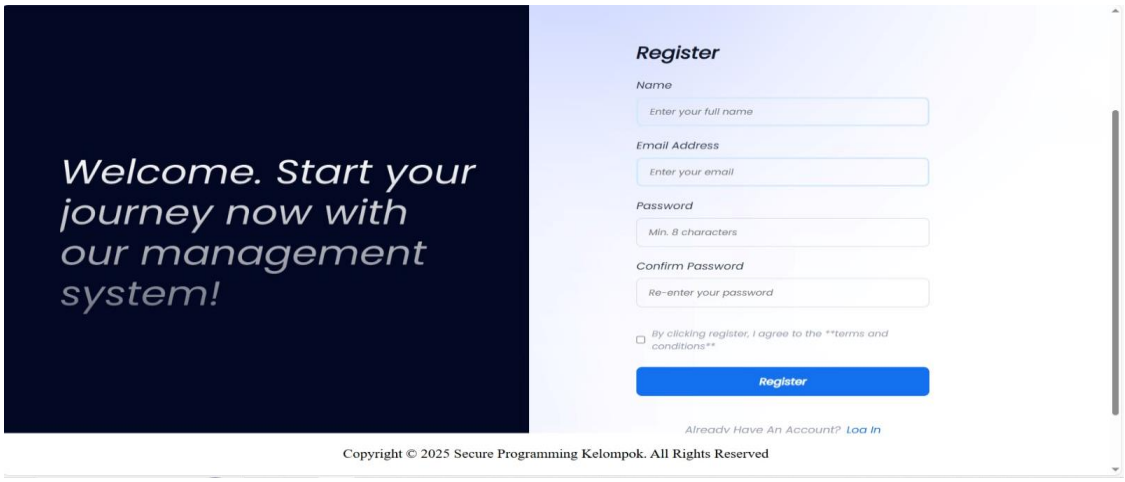
7. User Flow (Step-by-step)

Bagian ini menjelaskan alur penggunaan sistem dari sudut pandang User dan Admin, mulai dari proses login hingga penggunaan fitur utama.

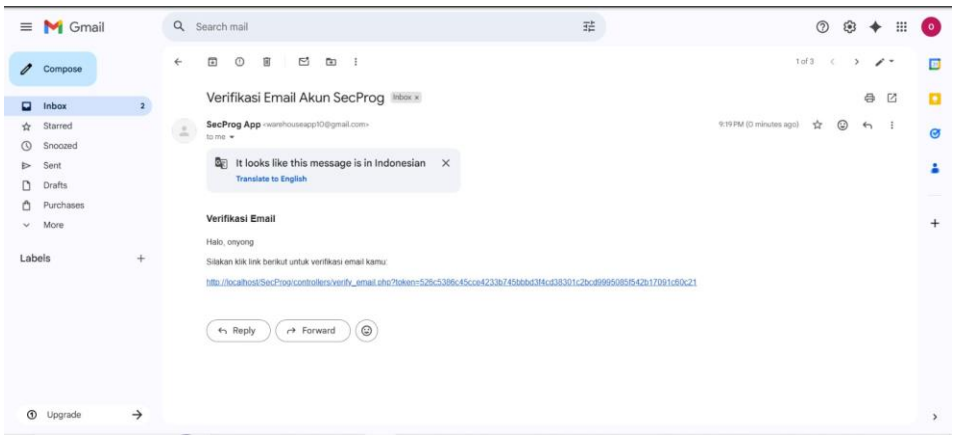
7.1 User Flow – Registrasi & Login

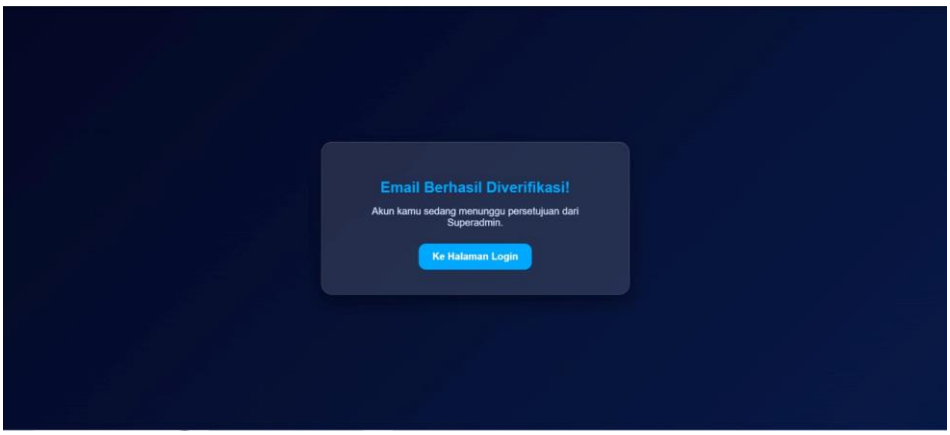
7.1.1 Registrasi Akun

1. User membuka halaman registrasi.
2. User mengisi form registrasi yang terdiri dari:
    a. Nama
    b. Email
    c. Password
    d. Confirm Password
3. Sistem melakukan validasi input dan menyimpan password dalam bentuk hash.
4. Sistem mengirimkan OTP verifikasi email ke alamat email user.



(Screenshot 7.1: Halaman Registrasi)

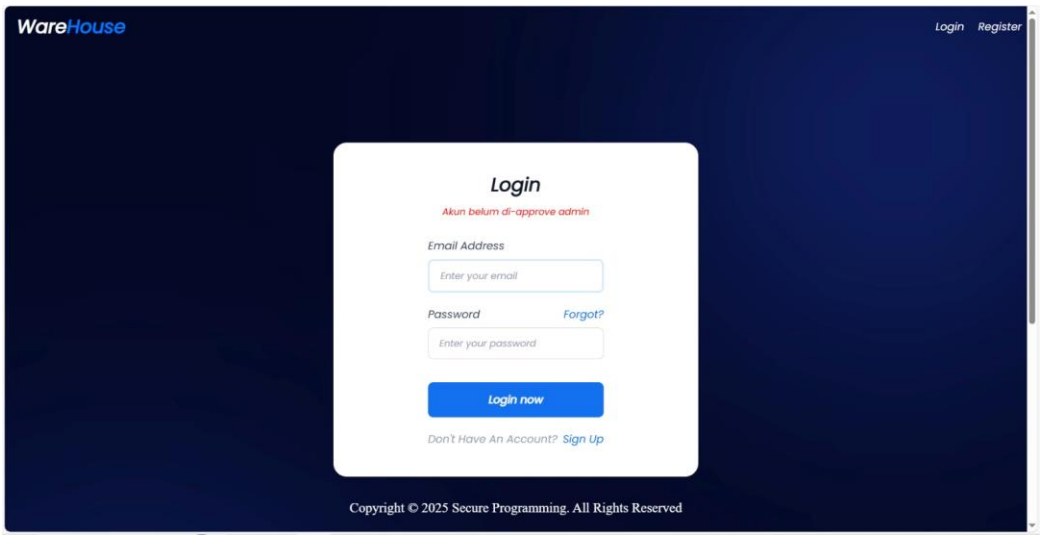




(Screenshot 7.2: Email OTP Verifikasi)

7.1.2 Menunggu Persetujuan Admin

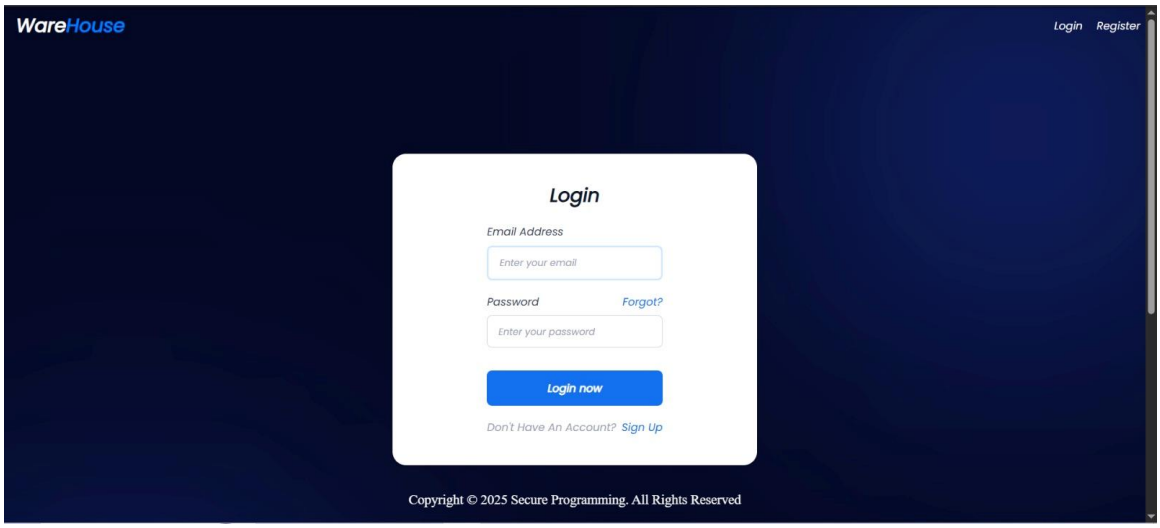
1. Setelah user berhasil melakukan verifikasi email menggunakan OTP, akun user belum dapat digunakan untuk login.
2. User mencoba melakukan login menggunakan email dan password yang telah terdaftar.
3. Sistem menolak proses login dan menampilkan pesan “Akun belum di-approve admin”.
4. Akun user akan berada dalam status inactive hingga admin melakukan persetujuan akun.
5. Setelah akun disetujui oleh admin, user dapat melanjutkan proses login ke tahap OTP.



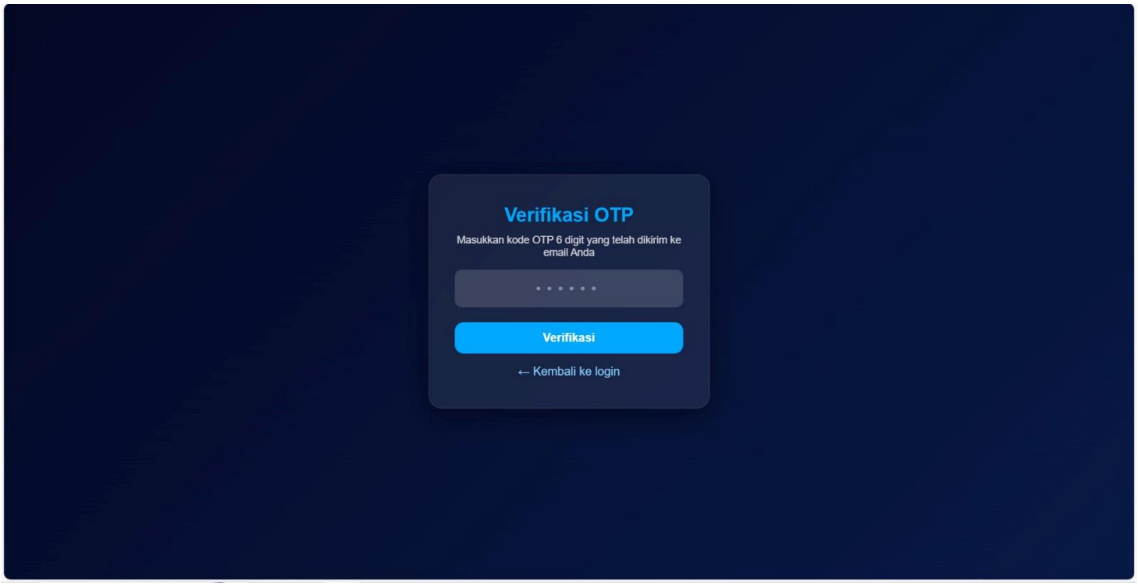
(Screenshot 7.3: Pesan akun belum di-approve admin)

7.1.3 Login dengan OTP

1. User memasukkan email dan password pada halaman login.
2. Sistem memverifikasi kredensial user.
3. Sistem mengirimkan OTP login ke email user.
4. User memasukkan OTP login.
5. Jika OTP valid dan akun sudah di-approve admin, user berhasil login.



(Screenshot 7.4: Halaman Login)



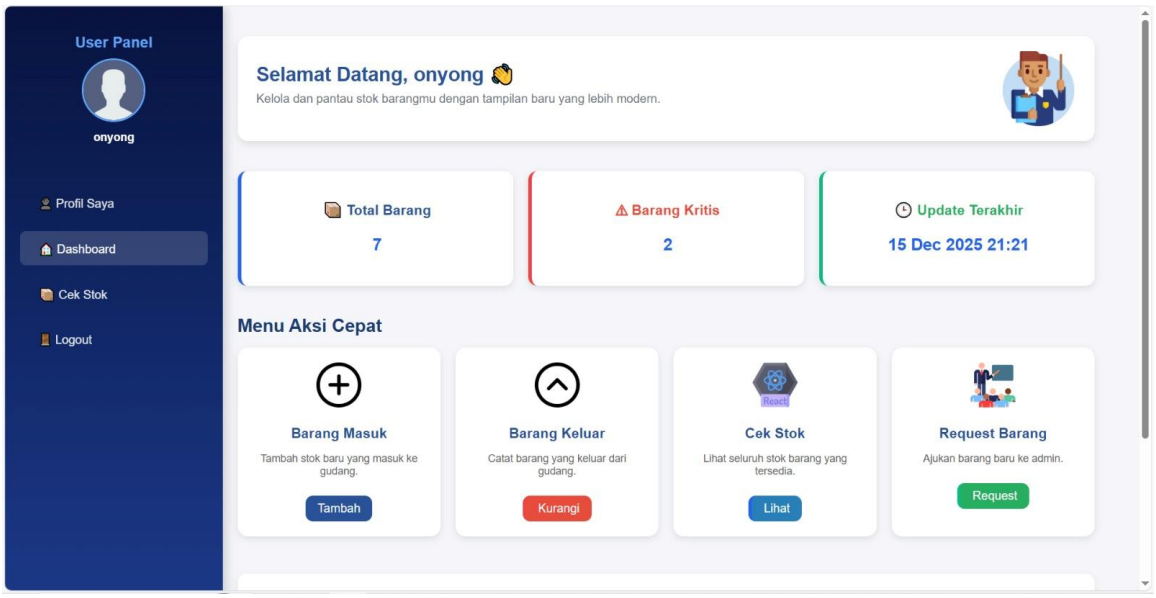
(Screenshot 7.5: OTP Login)

7.2 User Flow – Dashboard User

7.2.1 Dashboard User

Setelah login, user diarahkan ke dashboard yang menampilkan:

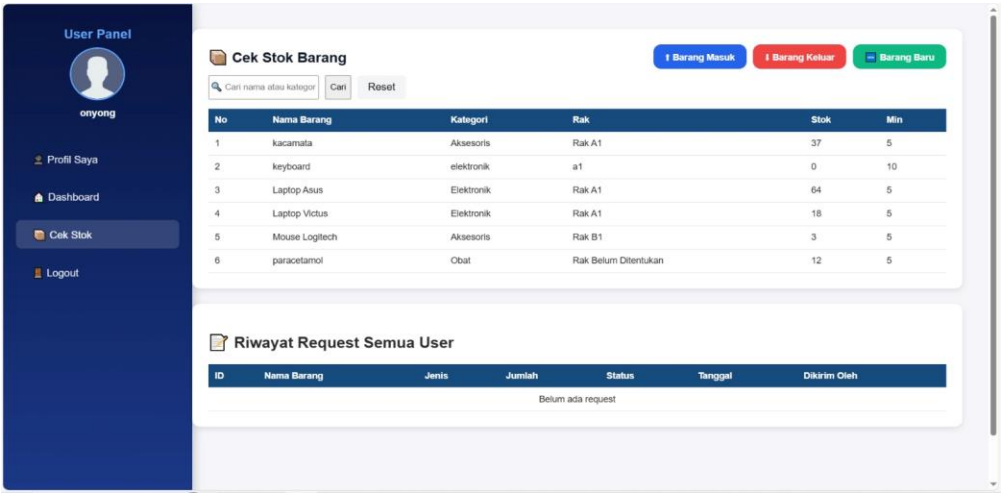
- Informasi stok barang
- Barang kritis (stok di bawah minimum)
- Menu aksi cepat:
    - Barang masuk
    - Barang keluar
    - Cek stok
    - Request barang



(Screenshot 7.6: Dashboard User)

7.2.2 Cek Stok Barang

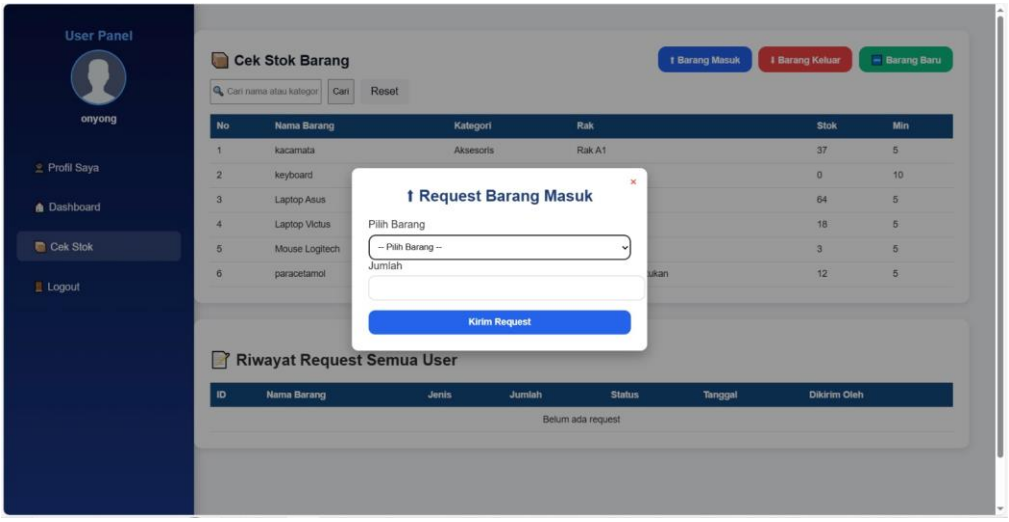
1. User memilih menu Cek Stok.
2. Sistem menampilkan tabel stok barang yang berisi:
    a. Nomor
    b. Nama barang
    c. Kategori
    d. Lokasi rak
    e. Jumlah stok
    f. Stok minimum



(Screenshot 7.7: Halaman Cek Stok)

7.2.3 Barang Masuk (User)

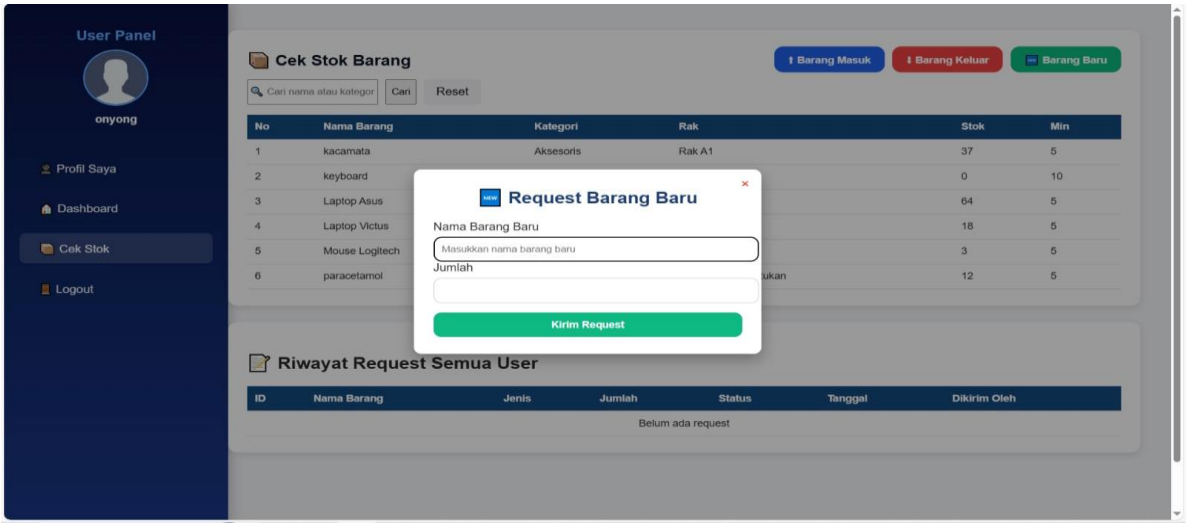
1. User memilih menu Barang Masuk.
2. User memilih barang yang sudah tersedia dalam daftar kategori.
3. User mengisi jumlah barang masuk.
4. Sistem mengirimkan permintaan ke admin untuk diproses.



(Screenshot 7.8: Form Barang Masuk)

7.2.4 Request Barang Baru

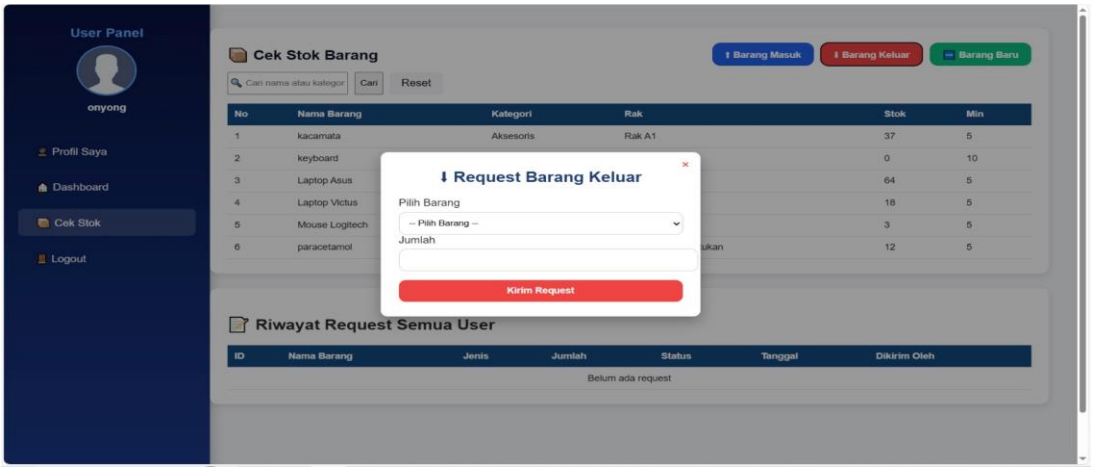
1. User memilih menu Request Barang.
2. User mengisi data barang yang belum tersedia.
3. Sistem menyimpan request dan menunggu persetujuan admin.



(Screenshot 7.9: Form Request Barang)

7.2.5 Barang Keluar (User)

1. User memilih menu Barang Keluar.
2. User mengajukan permintaan barang keluar beserta jumlahnya.
3. Permintaan masuk ke sistem admin untuk diverifikasi.



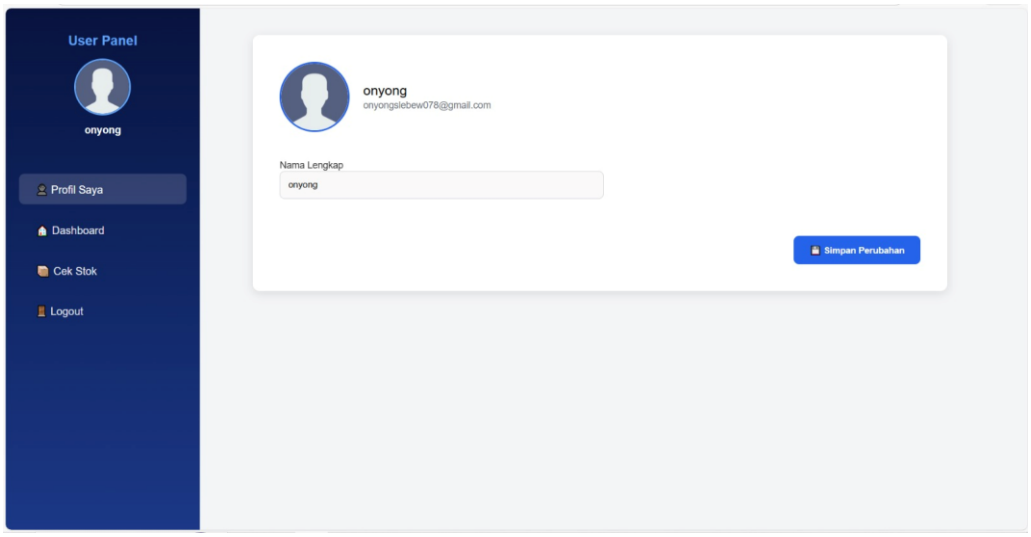
(Screenshot 7.10: Form Barang Keluar)

7.2.6 User Profile

1. User membuka menu Profile pada dashboard.
2. Sistem menampilkan data profil user.
3. User dapat:
    a. Mengubah foto profil

b. Mengubah username

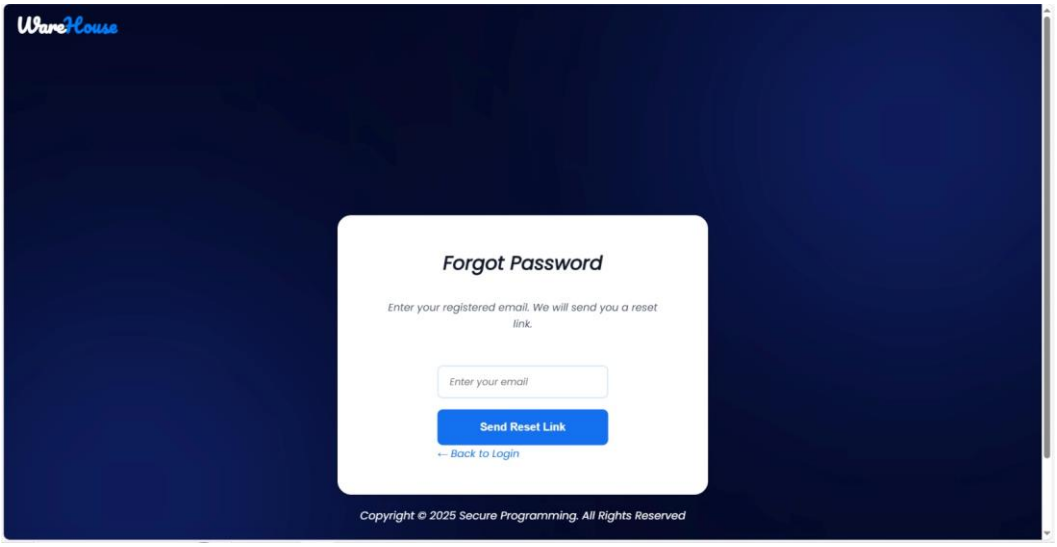
4. Sistem melakukan validasi input.
5. Perubahan data profil disimpan ke database.



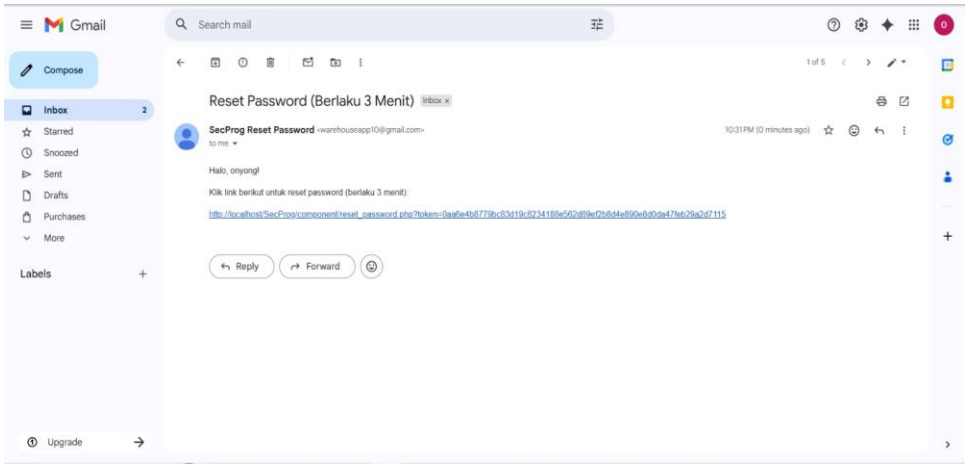
(Screenshot 7.11: Halaman User Profile)

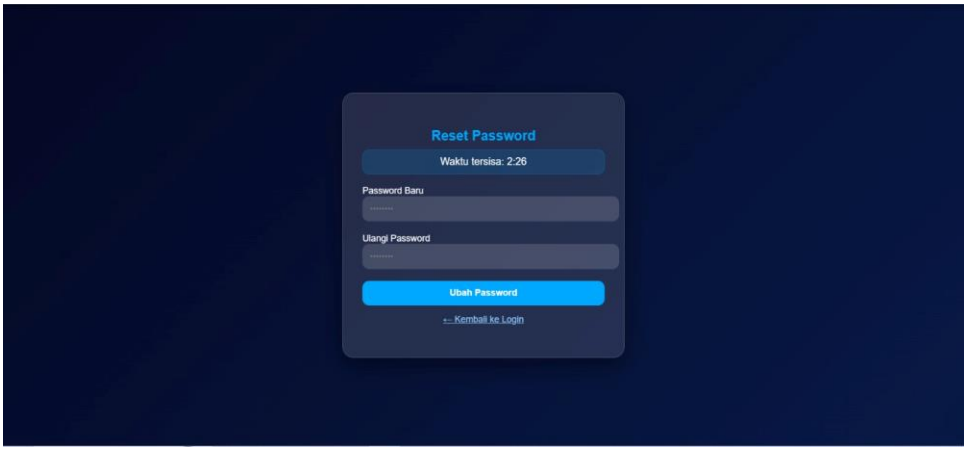
7.2.7 Forgot Password

1. User menekan tombol Forgot Password pada halaman login.
2. User memasukkan email yang telah terdaftar dan tervalidasi.
3. Sistem mengirimkan OTP reset password ke email user.
4. User memasukkan OTP yang diterima.
5. Sistem memverifikasi OTP.
6. User memasukkan password baru.
7. Sistem menyimpan password baru dalam bentuk hash.
8. User diarahkan kembali ke halaman login.



(Screenshot 7.12: Halaman Forgot Password)





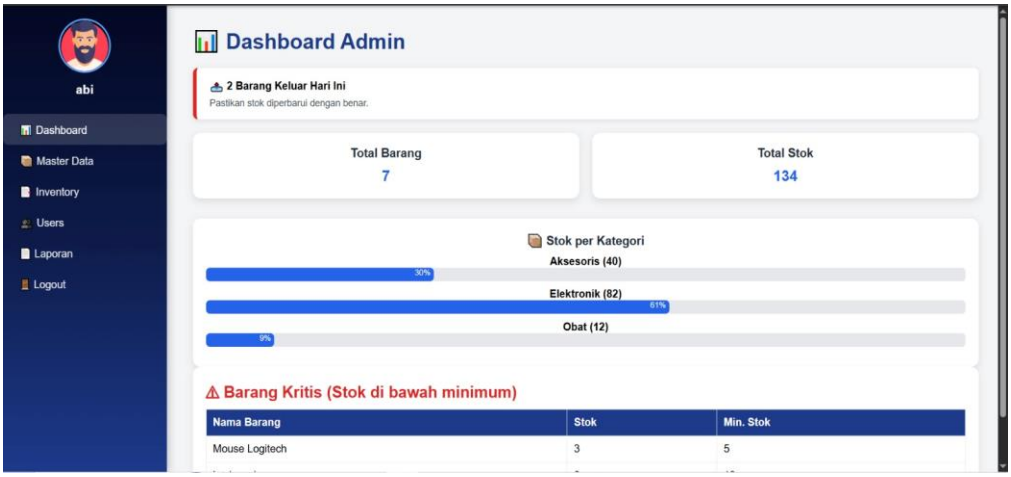
(Screenshot 7.13: OTP Reset Password Email)

7.3 User Flow – Admin

7.3.1 Dashboard Admin

Admin login dan melihat dashboard yang menampilkan:

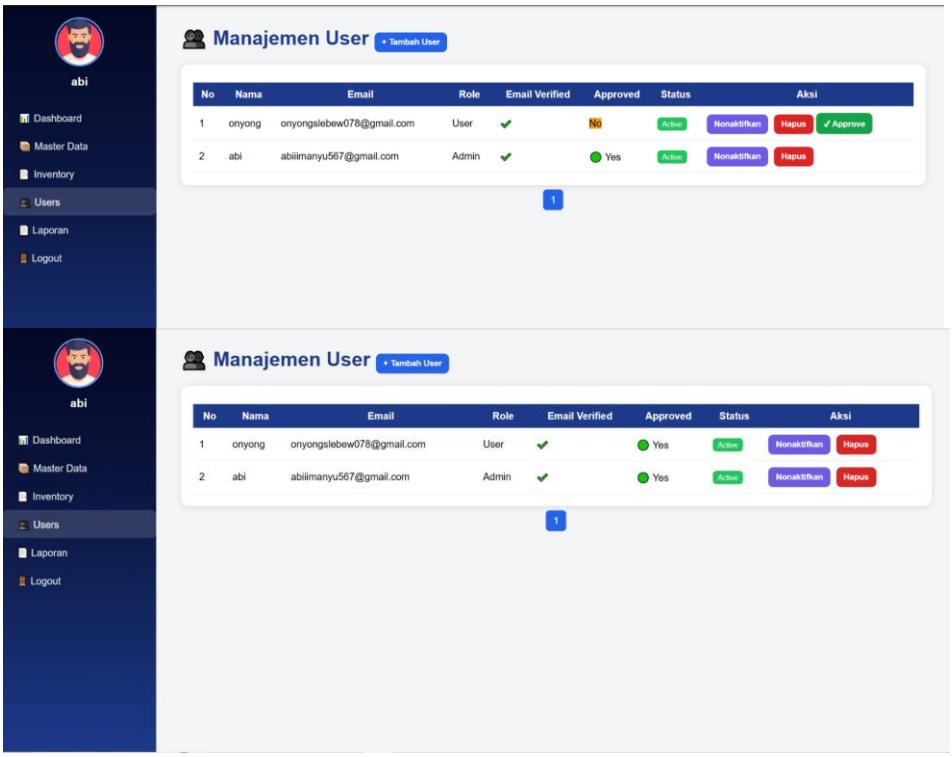
- Total barang
- Barang keluar hari ini
- Stok per kategori
- Barang kritis
- Notifikasi request user



(Screenshot 7.14: Dashboard Admin)

7.3.2 Approval Akun User

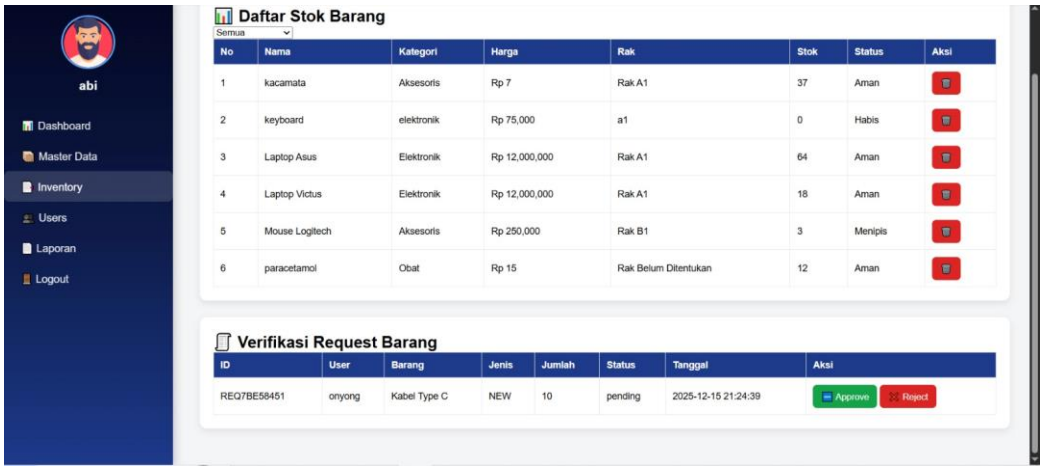
1. Admin membuka menu User Management.
2. Admin melihat daftar akun user yang belum aktif.
3. Admin melakukan approve atau reject akun user.

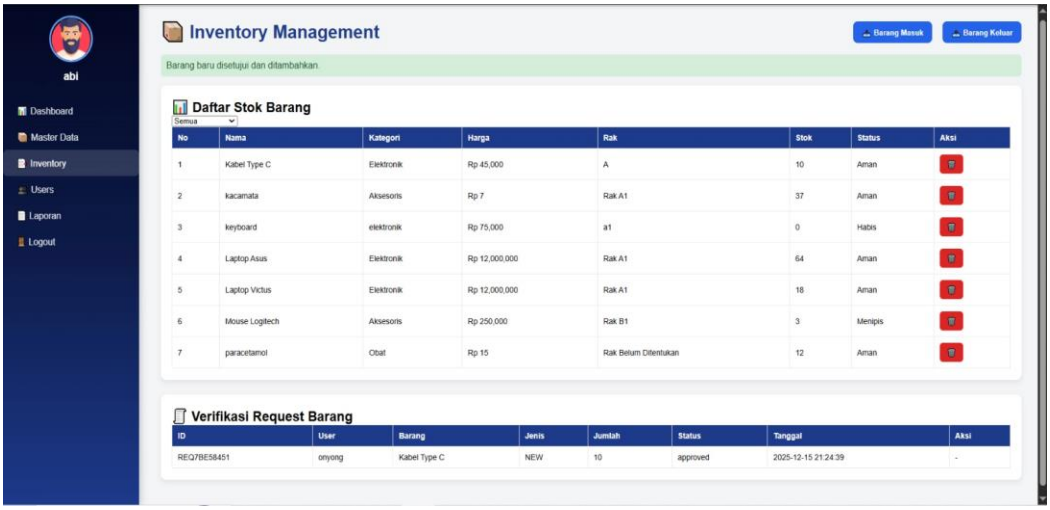


(Screenshot 7.15: User Management – Approval Akun)

7.3.3 Verifikasi Request Barang

1. Admin membuka menu Verifikasi Request.
2. Admin meninjau request barang masuk, barang keluar, atau barang baru.
3. Admin menyetujui atau menolak request.
4. Jika disetujui, sistem memperbarui data stok.



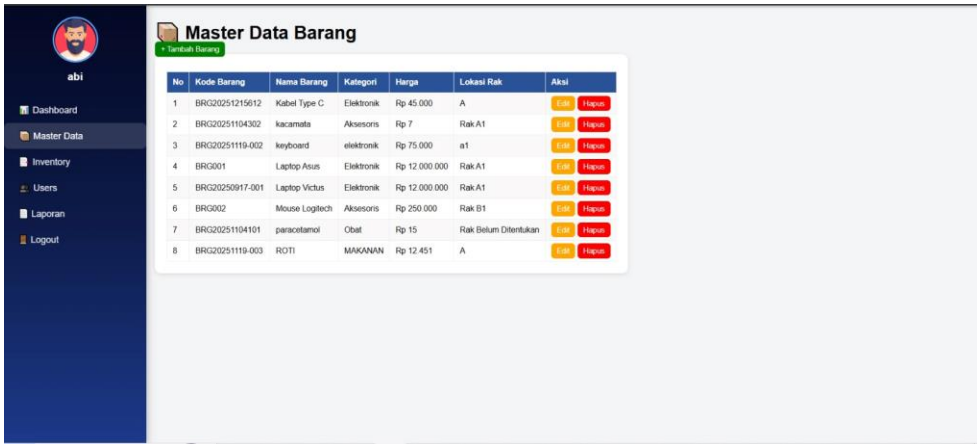


(Screenshot 7.16: Verifikasi Request Barang)

7.3.4 Master Data Barang

1. Admin membuka menu Master Data.
2. Admin dapat:
    a. Menambah barang
    b. Mengedit barang

c. Menghapus barang



(Screenshot 7.17: Master Data Barang)

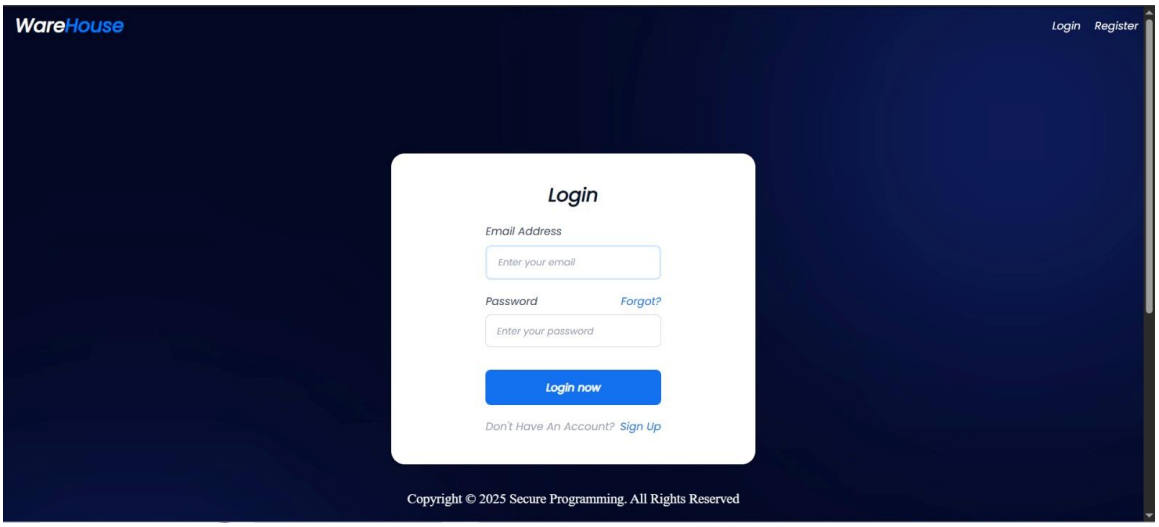
7.3.5 Laporan

1. Admin membuka menu Laporan.
2. Sistem menampilkan riwayat barang masuk dan barang keluar.

(Screenshot 7.18: Halaman Laporan)

7.4 Logout

1. User atau admin menekan menu Logout.
2. Sistem menghapus session login yang sedang aktif.
3. User langsung diarahkan kembali ke halaman login tanpa menampilkan konfirmasi tambahan.



(Screenshot 7.19: Logout)

8. Security Implementations

Website Warehouse menerapkan berbagai mekanisme keamanan untuk melindungi data, mencegah akses tidak sah, serta meminimalkan risiko kerentanan keamanan. Implementasi keamanan dilakukan pada sisi autentikasi, otorisasi, validasi input, pengelolaan sesi, serta perlindungan terhadap serangan umum pada aplikasi web.

8.1 Authentication & Account Security

Sub-bab ini membahas mekanisme autentikasi dan perlindungan akun pengguna, termasuk pengamanan kredensial, verifikasi identitas, dan validasi status akun sebelum login.

8.1.1 Password Hashing

Penjelasan

Aplikasi Warehouse menerapkan mekanisme password hashing untuk melindungi kredensial pengguna. Password yang dimasukkan oleh user tidak disimpan dalam bentuk teks asli (plaintext), melainkan diubah terlebih dahulu menjadi bentuk hash menggunakan fungsi bawaan PHP sebelum disimpan ke dalam database.

Fungsi password_hash() digunakan karena telah menerapkan algoritma hashing yang aman serta *salt* secara otomatis. Dengan demikian, apabila terjadi kebocoran database, password asli pengguna tetap tidak dapat diketahui.

Implementasi pada Kode

Proses hashing password dilakukan pada saat user melakukan registrasi akun. Password yang telah di-hash kemudian disimpan ke database sebagai nilai userPassword.

```
$hash = password_hash(password: $pass, algo: PASSWORD_DEFAULT);
```

Nilai hash tersebut kemudian digunakan saat menyimpan data user ke database:

```
48  $stmt = $con->prepare(query: "
49      INSERT INTO msuser (userName, userEmail, userPassword, userRole, userStatus, isVerified, isApproved, verifyToken, status)
50      VALUES (?, ?, ?, 'user', 'active', 0, 0, ?, 'pending')
51  ");
52  $stmt->bind_param(types: "ssss", var: &$name, vars: &$email, $hash, $verifyToken);
```

Tujuan Keamanan

- Mencegah penyimpanan password dalam bentuk plaintext
- Mengurangi risiko kebocoran kredensial pengguna
- Menerapkan praktik autentikasi yang aman sesuai standar secure programming

8.1.2 OTP-based Authentication

Penjelasan

Aplikasi Warehouse menerapkan mekanisme One-Time Password (OTP) sebagai lapisan keamanan tambahan untuk memastikan bahwa proses registrasi dilakukan oleh pemilik sah dari alamat email yang didaftarkan. OTP diimplementasikan dalam bentuk token verifikasi unik yang dikirimkan ke email pengguna melalui layanan SMTP.

Token OTP dihasilkan secara acak menggunakan fungsi kriptografis yang aman, disimpan ke dalam database, dan hanya dapat digunakan satu kali. Proses verifikasi email ini merupakan bagian dari mekanisme defense in depth, sebelum akun dapat diaktifkan oleh admin.

Implementasi pada Kode

Generate OTP / Verification Token

Sistem menghasilkan token verifikasi secara acak menggunakan fungsi random_bytes() dan mengonversinya ke format heksadesimal.

```php
$verifyToken = bin2hex(string: random_bytes(length: 32));
```

Token ini bersifat unik dan sulit ditebak sehingga aman untuk digunakan sebagai OTP berbasis email.

Penyimpanan OTP ke Database

Token OTP disimpan ke dalam database bersamaan dengan data user saat proses registrasi.

```php
$stmt = $con->prepare(query: "
    INSERT INTO msuser (userName, userEmail, userPassword, userRole, userStatus, isVerified, isApproved, verifyToken, status)
    VALUES (?, ?, ?, 'user', 'active', 0, 0, ?, 'pending')
");
$stmt->bind_param(types: "ssss", var: &$name, vars: &$email, $hash, $verifyToken);
```

Pengiriman OTP melalui Email (SMTP)

Token OTP dikirim ke email user dalam bentuk link verifikasi menggunakan PHPMailer dan protokol SMTP.

```php
$link = "http://localhost/SecProg/controllers/verify_email.php?token=$verifyToken";

$mail->Body = "
    <h3>Verifikasi Email</h3>
    <p>Halo, $name</p>
    <p>Silakan klik link berikut untuk verifikasi email kamu:</p>
    <p><a href='$link'>$link</a></p>
";
```

Konfigurasi SMTP dengan Enkripsi TLS

Pengiriman OTP dilakukan melalui protokol SMTP dengan enkripsi TLS untuk menjaga keamanan data selama proses transmisi email.

```php
$mail->isSMTP();
$mail->Host       = "smtp.gmail.com";
$mail->SMTPAuth   = true;
$mail->SMTPSecure = 'tls';
$mail->Port       = 587;
```

Tujuan Keamanan

- Memastikan kepemilikan email pengguna
- Mencegah pembuatan akun palsu
- Mengurangi risiko penyalahgunaan akun
- Menambah lapisan keamanan pada proses autentikasi

Kesimpulan

Dengan menerapkan OTP berbasis email, aplikasi Warehouse memastikan bahwa hanya pengguna yang memiliki akses ke email terdaftar yang dapat melanjutkan proses registrasi. Mekanisme ini meningkatkan keamanan sistem dan mendukung praktik secure programming yang baik.

8.1.3 Admin Account Approval

Penjelasan

Aplikasi Warehouse menerapkan mekanisme Admin Account Approval sebagai lapisan keamanan tambahan setelah proses registrasi dan verifikasi email selesai. Meskipun user telah berhasil memverifikasi email menggunakan OTP, akun tidak dapat langsung digunakan sebelum mendapatkan persetujuan dari admin.

Mekanisme ini bertujuan untuk memastikan bahwa hanya user yang benar-benar sah dan berwenang yang dapat mengakses sistem, serta mencegah penyalahgunaan akun oleh pihak yang tidak bertanggung jawab.

Implementasi pada Kode

Setelah user melakukan registrasi dan verifikasi email:

- Status akun user masih dalam kondisi inactive / pending
- User akan ditolak saat mencoba login
- Sistem menampilkan pesan bahwa akun belum di-approve oleh admin

Akun baru akan aktif setelah admin melakukan persetujuan melalui menu User Management.

Validasi Status Akun

Sistem melakukan pengecekan status akun sebelum mengizinkan proses login.

```
30  if ($user['userRole'] === 'user') {
31      if ((int)$user['isVerified'] === 0) {
32          header(header: "Location: ../component/login.php?error=Email belum diverifikasi");
33          exit;
34      }
35      if ((int)$user['isApproved'] === 0) {
36          header(header: "Location: ../component/login.php?error=Akun belum di-approve admin");
37          exit;
38      }
39  }
```

Proses Approval oleh Admin

1. Admin login ke sistem.
2. Admin membuka menu User Management.
3. Admin melihat daftar user dengan status pending.
4. Admin melakukan approve atau reject akun user.
5. Jika disetujui, status akun user berubah menjadi aktif dan user dapat login.

Tujuan Keamanan

- Mencegah akses akun yang belum diverifikasi sepenuhnya
- Menghindari pembuatan akun tidak sah
- Memberikan kontrol penuh kepada admin terhadap pengguna sistem
- Menambah lapisan authorization sebelum akses sistem diberikan

8.2 Authorization (Role-Based Access Control)

Penjelasan

Aplikasi Warehouse menerapkan mekanisme *Role-Based Access Control (RBAC)* untuk memastikan bahwa setiap pengguna hanya dapat mengakses fitur sesuai dengan perannya. Sistem membedakan peran admin dan user, di mana admin memiliki hak akses penuh terhadap manajemen data, sedangkan user hanya dapat mengakses fitur operasional yang telah dibatasi. Penerapan RBAC ini bertujuan untuk mencegah akses tidak sah terhadap fitur administratif dan data sensitif.

Implementasi pada Kode

8.2.1 – Validasi Role User Setelah Login

```
30  if ($user['userRole'] === 'user') {
31      if ((int)$user['isVerified'] === 0) {
32          header(header: "Location: ../component/login.php?error=Email belum diverifikasi");
33          exit;
34      }
35      if ((int)$user['isApproved'] === 0) {
36          header(header: "Location: ../component/login.php?error=Akun belum di-approve admin");
37          exit;
38      }
39  }
```

Penjelasan Kode:

Kode ini memastikan bahwa akun dengan role user harus memenuhi syarat verifikasi email dan persetujuan admin sebelum dapat melanjutkan proses login. Admin tidak dikenakan validasi ini karena memiliki hak akses yang berbeda.

8.2.2 – Pembatasan Akses Halaman Admin

```php
if (!isset($_SESSION['is_login']) || $_SESSION['userRole'] !== 'admin') {
    header(header: "Location: ../login.php");
    exit;
}
```

Penjelasan Kode:
Kode ini mencegah user biasa mengakses halaman admin secara langsung melalui URL. Hanya akun dengan role admin yang diizinkan mengakses halaman dan fitur administratif.

Tujuan Keamanan

Penerapan mekanisme *Role-Based Access Control (RBAC)* pada aplikasi Warehouse bertujuan untuk:

- Membatasi akses fitur administratif agar hanya dapat digunakan oleh pengguna dengan peran admin.
- Mencegah pengguna tanpa otorisasi melakukan manipulasi terhadap data dan fungsi sensitif.
- Menerapkan prinsip *Least Privilege* dengan memberikan hak akses sesuai kebutuhan masing-masing peran.
- Melindungi sistem dari upaya akses langsung ke halaman administratif melalui URL tanpa hak akses yang sah.

8.3 SQL Injection Prevention (Prepared Statements)

Penjelasan

Aplikasi Warehouse menerapkan mekanisme pencegahan *SQL Injection* dengan menggunakan prepared statements dan parameter binding pada setiap interaksi dengan database. Pendekatan ini memastikan bahwa input pengguna tidak dieksekusi sebagai bagian dari perintah SQL, sehingga mencegah manipulasi query oleh pihak yang tidak berwenang.

Implementasi pada Kode

8.3.1 – Prepared Statement pada Query SELECT

```php
$stmt = $con->prepare(query: "SELECT * FROM msuser WHERE userEmail = ? LIMIT 1");
$stmt->bind_param(types: "s", var: &$email);
$stmt->execute();
```

Penjelasan Kode:
Query database menggunakan placeholder (?) dan bind_param() untuk memastikan input email diperlakukan sebagai data, bukan sebagai bagian dari perintah SQL.

8.3.2 – Prepared Statement pada Query INSERT

```
$stmt = $con->prepare(query: "
    INSERT INTO msuser (userName, userEmail, userPassword, userRole, userStatus, isVerified, isApproved, verifyToken, status)
    VALUES (?, ?, ?, 'user', 'active', 0, 0, ?, 'pending')
");
$stmt->bind_param(types: "ssss", var: &$name, vars: &$email, $hash, $verifyToken);
```

Penjelasan Kode:
Data yang berasal dari input user disisipkan ke dalam database menggunakan parameter binding, sehingga mencegah injeksi perintah SQL berbahaya.

8.3.3 – Prepared Statement pada Query UPDATE

```
$upd = $con->prepare(query: "UPDATE msuser SET otpCode = ?, otpExpiry = ? WHERE UserID = ?");
$upd->bind_param(types: "ssi", var: &$otp, vars: &$expire, $user['UserID']);
$upd->execute();
```

Penjelasan Kode:
Query UPDATE juga menggunakan prepared statement untuk memastikan parameter OTP dan ID user tidak dapat dimanipulasi melalui input berbahaya.

Tujuan Keamanan

Penerapan prepared statements pada aplikasi Warehouse bertujuan untuk:

- Mencegah serangan *SQL Injection* melalui input pengguna.
- Melindungi integritas dan konsistensi data dalam database.
- Memastikan seluruh operasi database dijalankan secara aman.
- Mengurangi risiko kebocoran atau manipulasi data sensitif.

8.4 Session Management & Session Security

8.4.1 Penggunaan Session pada Halaman Dashboard

Penjelasan

Aplikasi Warehouse menggunakan mekanisme session PHP untuk menyimpan dan mengelola identitas pengguna yang sedang aktif login. Informasi session ini dimanfaatkan pada halaman dashboard untuk menampilkan data pengguna serta memastikan bahwa akses ke fitur utama aplikasi hanya dilakukan oleh pengguna yang telah terautentikasi.

Implementasi pada Kode

```
$userName = $_SESSION['userName'] ?? 'User';
$userID = $_SESSION['userID'] ?? null;
```

Penjelasan Kode:
Kode di atas mengambil data identitas pengguna dari session, yaitu `userName` dan

`userID`. Jika session tidak tersedia, sistem akan memberikan nilai default. Mekanisme ini menunjukkan bahwa dashboard bergantung pada session untuk mengenali pengguna yang sedang login.

Tujuan Keamanan

Penggunaan session pada halaman dashboard bertujuan untuk:

- Memastikan identitas pengguna tetap terjaga selama sesi penggunaan aplikasi.
- Mencegah penggunaan identitas anonim saat mengakses fitur utama sistem.
- Menjaga konsistensi akses pengguna terhadap data dan fitur yang tersedia.

Catatan
Implementasi session pada aplikasi ini masih bersifat dasar dan belum mencakup pengamanan lanjutan seperti regenerasi session ID atau pengaturan cookie security flags, yang dapat menjadi pengembangan di masa mendatang.

8.4.2 Validasi Session pada Halaman Terproteksi

Penjelasan

Aplikasi Warehouse menerapkan validasi session pada halaman terproteksi untuk memastikan bahwa hanya pengguna yang telah login dan memiliki hak akses yang sesuai yang dapat mengakses sistem. Validasi session dilakukan baik secara langsung pada halaman admin maupun melalui mekanisme pengecekan terpusat (*middleware*) pada halaman user.

Implementasi pada Kode

8.4.2.1 – Validasi Session & Role pada Admin Dashboard

```
5   if (!isset($_SESSION['is_login']) || $_SESSION['userRole'] !== 'admin') {
6       header(header: "Location: ../login.php");
7       exit;
8   }
```

Penjelasan Kode:
Kode ini memastikan bahwa hanya pengguna yang telah login dan memiliki role admin yang dapat mengakses dashboard admin. Jika session tidak valid atau role tidak sesuai, pengguna akan diarahkan kembali ke halaman login.

8.4.2.2 – Validasi Session User Menggunakan Middleware

```
2   require "../../controllers/session_check.php";
3   require_login(role: "user");
```

Penjelasan Kode:
Validasi session untuk user dilakukan melalui fungsi require_login() yang berada pada file session_check.php. Pendekatan ini memusatkan logika autentikasi sehingga mengurangi duplikasi kode dan memastikan konsistensi keamanan pada seluruh halaman user.

Tujuan Keamanan

Validasi session pada halaman terproteksi bertujuan untuk:

- Membatasi akses halaman sistem hanya untuk pengguna yang telah login.
- Memastikan pengguna hanya dapat mengakses halaman sesuai dengan perannya.
- Mencegah akses langsung ke halaman internal melalui URL tanpa autentikasi.
- Menjaga konsistensi keamanan session di seluruh bagian aplikasi.

8.4.3 Session Termination (Logout)

Penjelasan

Aplikasi Warehouse menyediakan mekanisme logout untuk mengakhiri sesi pengguna yang sedang aktif. Proses logout dilakukan dengan menghapus seluruh data session sehingga pengguna tidak lagi memiliki akses ke halaman terproteksi dan harus melakukan login ulang untuk dapat mengakses sistem kembali.

Implementasi pada Kode

```
2   session_start();
3   session_unset();
4   session_destroy();
5
6   header(header: "Location: login.php");
7   exit;
```

Penjelasan Kode:
Kode di atas menghapus seluruh data session pengguna dan menghentikan sesi yang sedang aktif. Setelah session dihancurkan, pengguna diarahkan kembali ke halaman login, sehingga session tidak dapat digunakan kembali.

Meskipun logout tidak menampilkan konfirmasi dan langsung melakukan redirect, mekanisme ini tetap valid dari sisi keamanan.

Tujuan Keamanan

Penghentian session saat logout bertujuan untuk:

- Mengakhiri sesi autentikasi pengguna secara aman.
- Mencegah penyalahgunaan session setelah pengguna keluar dari sistem.
- Melindungi akun pengguna pada perangkat bersama atau publik.

8.5 Cross-Site Request Forgery (CSRF) Protection

Penjelasan

Cross-Site Request Forgery (CSRF) merupakan serangan yang memanfaatkan sesi autentikasi pengguna untuk menjalankan aksi tanpa sepengetahuan pengguna. Pada aplikasi Warehouse, sebagian besar fitur sensitif hanya dapat diakses oleh pengguna yang telah login dan bergantung pada validasi session untuk memproses setiap permintaan.

Implementasi pada Kode

```
2   require "../../controllers/session_check.php";
3   require_login(role: "user");
```

Penjelasan Kode:
Kode di atas memastikan bahwa setiap request yang memicu aksi dalam sistem hanya dapat dijalankan oleh pengguna yang memiliki session aktif. Dengan demikian, request yang tidak memiliki konteks session tidak akan diproses oleh sistem.

Catatan

Saat ini, aplikasi Warehouse belum menerapkan CSRF token secara eksplisit pada setiap form atau request sensitif. Perlindungan terhadap CSRF masih mengandalkan:

- Session autentikasi pengguna
- Pembatasan akses halaman dan controller berdasarkan status login

Pendekatan ini memberikan perlindungan dasar, namun belum sepenuhnya mencegah serangan CSRF pada skenario tertentu.

Tujuan Keamanan

Pembahasan dan penerapan dasar mekanisme terkait CSRF pada aplikasi Warehouse bertujuan untuk:

- Mengurangi risiko eksekusi aksi tanpa autentikasi pengguna.
- Memastikan setiap request berasal dari konteks session pengguna yang sah.
- Menjadi dasar evaluasi untuk pengembangan mekanisme CSRF token di masa mendatang.

8.6 Input Validation & Sanitization

Penjelasan

Aplikasi Warehouse menerapkan validasi input untuk memastikan bahwa data yang diterima dari pengguna sesuai dengan format dan aturan yang telah ditentukan. Validasi ini bertujuan untuk mencegah masuknya data tidak valid, mengurangi risiko kesalahan sistem, serta meminimalkan potensi serangan keamanan seperti *SQL Injection* dan *malicious input*.

Implementasi pada Kode

8.6.1 – Validasi Format Email

```
18  if (!preg_match(pattern: "/@gmail\.com$/", subject: $email)) {
19      header(header: "Location: ../component/register.php?error=Gunakan email Gmail!");
20      exit;
21  }
```

Penjelasan Kode:
Validasi ini memastikan bahwa email yang didaftarkan memiliki format yang sesuai dengan aturan sistem, sehingga mencegah penggunaan email dengan format tidak valid.

8.6.2 – Validasi Kekuatan Password

```
24  if (strlen(string: $pass) < 8 ||
25      !preg_match(pattern: '/[A-Z]/', subject: $pass) ||
26      !preg_match(pattern: '/[a-z]/', subject: $pass) ||
27      !preg_match(pattern: '/[0-9]/', subject: $pass) ||
28      !preg_match(pattern: '/[\W]/', subject: $pass)) {
29
30      header(header: "Location: ../component/register.php?error=Password harus minimal 8 karakter, mengandung huruf besar,
31      huruf kecil, angka, dan simbol!&name=$name&email=$email");
32      exit;
33  }
```

Penjelasan Kode:
Kode ini memvalidasi kompleksitas password untuk memastikan keamanan kredensial pengguna dan mencegah penggunaan password yang lemah.

8.6.3 – Sanitasi Input dengan trim()

```
13    $name  = trim(string: $_POST['name']);
14    $email = trim(string: $_POST['email']);
```

Penjelasan Kode:
Fungsi `trim()` digunakan untuk menghapus spasi berlebih pada input pengguna, sehingga mengurangi risiko kesalahan input dan manipulasi sederhana.

Tujuan Keamanan

Penerapan validasi dan sanitasi input pada aplikasi Warehouse bertujuan untuk:

- Memastikan data yang diproses sistem sesuai dengan format yang diharapkan.
- Mencegah masuknya data berbahaya atau tidak valid.
- Mengurangi risiko kesalahan logika dan potensi serangan berbasis input.
- Meningkatkan keandalan dan keamanan sistem secara keseluruhan.

8.7 Error Handling & Logging

Penjelasan

Aplikasi Warehouse menerapkan mekanisme *error handling* untuk menangani kondisi kesalahan yang terjadi selama proses sistem, seperti input tidak valid, kegagalan autentikasi, atau status akun yang tidak memenuhi syarat. Penanganan error dilakukan dengan memberikan pesan kesalahan yang terkontrol kepada pengguna tanpa menampilkan detail teknis internal sistem. Pendekatan ini bertujuan untuk mencegah kebocoran informasi sensitif dan menjaga stabilitas aplikasi.

Implementasi pada Kode

8.7.1 – Penanganan Error Autentikasi Login

```
20  if (!$user) {
21      header(header: "Location: ../component/login.php?error=Email tidak terdaftar");
22      exit;
23  }
24
25  if (!password_verify(password: $pass, hash: $user['userPassword'])) {
26      header(header: "Location: ../component/login.php?error=Password salah");
27      exit;
28  }
```

Penjelasan Kode:
Sistem menangani kesalahan login dengan mengembalikan pesan error yang umum kepada pengguna tanpa mengungkapkan detail teknis seperti struktur database atau query yang digunakan.

8.7.2 – Penanganan Error Status Akun

```
31      if ((int)$user['isVerified'] === 0) {
32          header(header: "Location: ../component/login.php?error=Email belum diverifikasi");
33          exit;
34      }
35      if ((int)$user['isApproved'] === 0) {
36          header(header: "Location: ../component/login.php?error=Akun belum di-approve admin");
37          exit;
38      }
```

Penjelasan Kode:
Error ditangani dengan pesan yang jelas namun tidak mengungkapkan detail internal sistem, sehingga tetap informatif bagi pengguna tanpa menimbulkan risiko keamanan.

8.7.3 – Penanganan Error Proses Registrasi

```
38  if ($cek->get_result()->num_rows > 0) {
39      header(header: "Location: ../component/register.php?error=Email sudah terdaftar!");
40      exit;
41  }
```

Penjelasan Kode:
Sistem mencegah duplikasi data dengan melakukan pengecekan terlebih dahulu dan memberikan respon error yang terkontrol.

Tujuan Keamanan

Penerapan error handling pada aplikasi Warehouse bertujuan untuk:

- Mencegah kebocoran informasi sensitif melalui pesan error.
- Memberikan feedback yang jelas namun aman kepada pengguna.
- Menjaga stabilitas sistem saat terjadi kesalahan.
- Mengurangi potensi eksploitasi melalui error message.

Catatan

Saat ini, aplikasi belum menerapkan mekanisme *logging* terpusat untuk pencatatan error ke file atau sistem monitoring. Penanganan error masih difokuskan pada kontrol alur aplikasi dan pesan kepada pengguna. Mekanisme logging dapat menjadi pengembangan lanjutan untuk meningkatkan kemampuan audit dan monitoring sistem.

8.8 File Upload Security (Foto Profil Pengguna)

Penjelasan

Aplikasi Warehouse menyediakan fitur unggah foto profil yang memungkinkan pengguna untuk memperbarui informasi akun mereka. Fitur unggah file berpotensi menjadi celah keamanan apabila tidak dikontrol dengan baik, karena dapat dimanfaatkan untuk mengunggah file berbahaya atau menyebabkan gangguan pada sistem.

Oleh karena itu, aplikasi ini menerapkan mekanisme keamanan pada proses unggah foto profil dengan melakukan pembatasan akses pengguna, validasi tipe file, validasi ukuran file, penamaan file secara unik, serta pengelolaan penyimpanan file yang aman.

Implementasi pada Kode

Keamanan unggah file diimplementasikan pada halaman `user_profile.php`. Sistem memastikan hanya pengguna yang telah login dan memiliki session aktif yang dapat mengakses fitur unggah foto profil. File yang diunggah akan divalidasi sebelum disimpan ke server dan direferensikan ke database.

8.8.1 Validasi Akses Pengguna

```php
require "../../controllers/session_check.php";
require_login(role: "user");

$userID = $_SESSION['userID'] ?? null;
if (!$userID) {
    header(header: "Location: ../login.php");
    exit;
}
```

Sistem melakukan pengecekan session untuk memastikan bahwa hanya pengguna yang telah terautentikasi yang dapat mengakses halaman profil dan melakukan unggah foto. Apabila session tidak ditemukan, pengguna akan diarahkan kembali ke halaman login. Mekanisme ini mencegah akses tidak sah terhadap fitur unggah file.

8.8.2 Validasi Tipe File

```php
$fileExt = strtolower(string: pathinfo(path: $fileName, flags: PATHINFO_EXTENSION));
$allowedExt = ['jpg', 'jpeg', 'png', 'gif'];

if (in_array(needle: $fileExt, haystack: $allowedExt)) {
```

Penjelasan Kode

Sistem membatasi tipe file yang dapat diunggah dengan memeriksa ekstensi file. Hanya file gambar dengan ekstensi tertentu yang diperbolehkan. Validasi ini bertujuan untuk mencegah unggahan file berbahaya seperti script atau file executable yang dapat mengancam keamanan sistem.

8.8.3 Validasi Ukuran File

```php
if ($_FILES['userPhoto']['size'] <= 2 * 1024 * 1024) {
```

Penjelasan Kode

Ukuran file foto profil dibatasi maksimal sebesar 2MB. Pembatasan ini diterapkan untuk mengurangi risiko penyalahgunaan sistem dan menjaga performa server dari unggahan file berukuran besar.

8.8.4 Penamaan File Secara Unik

```php
$newName = uniqid(prefix: "photo_") . "." . $fileExt;
move_uploaded_file(from: $fileTmp, to: $targetDir . $newName);
```

Penjelasan Kode

Sistem mengganti nama file asli dengan nama baru yang dihasilkan secara unik sebelum disimpan ke server. Penamaan file secara unik ini mencegah konflik nama file, menghindari penimpaan file yang sudah ada, serta mengurangi kemungkinan penyerang menebak nama atau lokasi file di server.

8.8.5 Penyimpanan File pada Direktori Khusus

```
26      $targetDir = "../../uploads/user_photos/";
27      if (!file_exists(filename: $targetDir)) mkdir(directory: $targetDir, permissions: 0777, recursive…true);
```

Penjelasan Kode

File foto profil disimpan pada direktori khusus yang terpisah dari file aplikasi utama. Pemisahan direktori ini bertujuan untuk mengurangi risiko eksekusi file berbahaya serta menjaga struktur penyimpanan file tetap aman dan terorganisir.

8.8.6 Penyimpanan Referensi File ke Database

```
43      $stmt = $con->prepare(query: "UPDATE MsUser SET userName=?, Photo=? WHERE UserID=?");
44      $stmt->bind_param(types: "ssi", var: &$name, vars: &$photoName, $userID);
```

Penjelasan Kode

Sistem hanya menyimpan nama file foto profil ke dalam database sebagai referensi. Sistem tidak menyimpan path lengkap atau isi file ke database, sehingga mengurangi risiko manipulasi path dan menjaga integritas data pengguna.

Tujuan Keamanan

Penerapan mekanisme keamanan pada fitur unggah foto profil bertujuan untuk:

- Mencegah unggahan file berbahaya ke dalam sistem.
- Membatasi akses unggah file hanya untuk pengguna yang telah terautentikasi.
- Mengurangi risiko penyalahgunaan sumber daya server.
- Menjaga keamanan dan stabilitas aplikasi.

Catatan

Validasi MIME type dan pemindaian konten file belum diterapkan pada sistem ini. Fitur tersebut dapat menjadi pengembangan lanjutan untuk meningkatkan keamanan unggahan file di masa mendatang.

9. Testing & Validation

9.1 Testing Environment

Pengujian dilakukan pada lingkungan pengembangan lokal dengan spesifikasi sebagai berikut:

- Sistem Operasi: Windows
- Web Server: Apache (XAMPP)
- Bahasa Pemrograman: PHP Native
- Database: MySQL
- Browser: Google Chrome

Pengujian difokuskan pada pengujian fungsionalitas sistem serta validasi terhadap mekanisme keamanan yang telah diimplementasikan pada aplikasi.

9.2 Functional & Security Testing

9.2.1 Login dan Autentikasi Pengguna

Pengujian dilakukan untuk memastikan bahwa hanya pengguna dengan kredensial yang valid, email terverifikasi, dan akun yang telah di-approve oleh admin yang dapat mengakses sistem.

Skenario	Input	Hasil
Email belum terdaftar	Email acak	Sistem menolak login
Password salah	Password salah	Sistem menampilkan pesan error
Email belum diverifikasi	Email valid	Login ditolak
Akun belum di-approve	Email valid	Login ditolak
Login valid	Email, password, OTP benar	Login berhasil

9.2.2 Role-Based Access Testing

Pengujian dilakukan untuk memastikan bahwa pengguna dengan peran user tidak dapat mengakses fitur atau halaman admin secara langsung melalui URL.

Skenario	Aksi	Hasil
User akses dashboard admin	Akses URL admin	Dialihkan / ditolak
Admin akses fitur user	Akses berhasil	Berjalan normal

9.2.3 File Upload Testing (Foto Profil)

Pengujian dilakukan pada fitur unggah foto profil untuk memastikan sistem hanya menerima file yang valid.

Skenario	Input	Hasil
Upload file gambar valid	JPG < 2MB	Berhasil
Upload file > 2MB	JPG > 2MB	Ditolak
Upload file non-gambar	.php, .exe	Ditolak

9.2.4 Business Logic Testing (Inventory)

Pengujian dilakukan pada fitur manajemen inventori untuk memastikan alur bisnis berjalan sesuai aturan.

Skenario	Aksi	Hasil
User menambah barang tanpa kategori	Request barang	Menunggu approval admin
User mengeluarkan barang	Request keluar	Menunggu verifikasi admin
Admin approve request	Approve	Stok berubah

9.2.5 Error Handling Testing

Pengujian dilakukan dengan memberikan input yang tidak valid pada beberapa fitur seperti login, registrasi, dan upload file.

Hasil pengujian menunjukkan bahwa sistem menampilkan pesan kesalahan yang terkontrol tanpa menampilkan detail teknis sistem atau query database.

9.3 Testing Summary

Berdasarkan hasil pengujian yang dilakukan, seluruh fitur utama dan mekanisme keamanan yang diimplementasikan pada aplikasi Warehouse berjalan sesuai dengan yang diharapkan. Tidak ditemukan bug kritis pada fitur autentikasi, otorisasi, maupun manajemen inventori selama proses pengujian.

9.4 Limitations of Testing

Pengujian keamanan pada aplikasi ini masih terbatas pada pengujian fungsional dan validasi dasar. Pengujian lanjutan seperti pengujian session hijacking, session fixation, dan penetration testing menggunakan tools khusus belum dilakukan dan dapat menjadi pengembangan selanjutnya.

10. Conclusion & Future Improvements

10.1 Conclusion

Berdasarkan proses perancangan, pengembangan, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi Warehouse berhasil diimplementasikan dengan memperhatikan prinsip-prinsip secure programming yang telah dipelajari pada mata kuliah Secure Programming.

Aplikasi ini telah menerapkan mekanisme keamanan utama seperti autentikasi pengguna berbasis password dan OTP, verifikasi dan persetujuan akun oleh admin, pengelolaan hak akses berbasis peran (Role-Based Access Control), validasi input, penggunaan prepared statement untuk interaksi database, serta pengamanan fitur unggah file.

Selain itu, sistem juga dirancang dengan alur bisnis yang terkontrol, khususnya pada proses manajemen inventori, sehingga setiap aktivitas penting seperti penambahan dan pengeluaran barang harus melalui proses verifikasi dan persetujuan yang sesuai. Hal ini membantu mengurangi risiko penyalahgunaan sistem serta menjaga integritas data inventori.

Secara keseluruhan, aplikasi Warehouse telah memenuhi tujuan utama pengembangan, yaitu menyediakan sistem manajemen gudang yang fungsional, terorganisir, dan memiliki tingkat keamanan yang memadai sesuai dengan standar yang diajarkan dalam mata kuliah Secure Programming.

10.2 Future Improvements

Meskipun aplikasi telah menerapkan berbagai mekanisme keamanan, masih terdapat beberapa aspek yang dapat dikembangkan lebih lanjut untuk meningkatkan keamanan dan kualitas sistem, antara lain:

1. **Implementasi Logging dan Monitoring**
   Menambahkan mekanisme pencatatan aktivitas dan kesalahan sistem untuk memudahkan proses audit dan deteksi insiden keamanan.
2. **Peningkatan Keamanan Session dan Cookie**
   Mengatur atribut cookie seperti HttpOnly, Secure, dan SameSite untuk meningkatkan perlindungan terhadap serangan berbasis session.
3. **Validasi File Upload Lanjutan**
   Menambahkan validasi MIME type dan pemindaian konten file untuk meningkatkan keamanan fitur unggah file.
4. **Rate Limiting dan Brute Force Protection**
   Menerapkan pembatasan percobaan login untuk mengurangi risiko serangan brute force.
5. **Pengujian Keamanan Lanjutan**
   Melakukan penetration testing dan pengujian keamanan lanjutan menggunakan tools khusus untuk mengidentifikasi potensi kerentanan yang belum terdeteksi.
6. **Peningkatan User Experience dan Security Awareness**
   Menambahkan notifikasi keamanan dan panduan penggunaan sistem untuk meningkatkan kesadaran pengguna terhadap keamanan akun.

11. Deployment / Running Instructions

11.1 System Requirements

Untuk menjalankan aplikasi Warehouse, diperlukan lingkungan sistem dengan spesifikasi sebagai berikut:

- Sistem Operasi: Windows / Linux / macOS
- Web Server: Apache
- PHP: Versi 8.x atau lebih baru
- Database: MySQL / MariaDB
- Web Browser: Google Chrome atau browser modern lainnya
- Internet Connection: Diperlukan untuk pengiriman email OTP dan verifikasi email (SMTP)

11.2 Software Tools

Perangkat lunak yang digunakan dalam proses deployment dan pengujian aplikasi adalah:

- XAMPP (Apache, PHP, MySQL)
- Visual Studio Code (Text Editor)
- Google Chrome (Browser)
- PHPMailer Library (Email Service)

11.3 Database Setup

Langkah-langkah untuk menyiapkan database:

1. Jalankan Apache dan MySQL melalui XAMPP Control Panel.
2. Buka phpMyAdmin melalui browser.
3. Buat database baru dengan nama sesuai konfigurasi aplikasi
4. Import file SQL yang berisi struktur tabel aplikasi ke dalam database.
5. Pastikan tabel seperti MsUser, Barang, Kategori, dan tabel transaksi lainnya berhasil dibuat.

11.4 Application Configuration

1. Letakkan folder project aplikasi Warehouse ke dalam direktori:

   htdocs/

2. Konfigurasi koneksi database pada file connection.php:

```
2    $config = [
3        'server' => 'localhost',
4        'username' => 'root',
5        'password' => '',
6        'database' => 'secprogasg',
7    ];
```

3. Konfigurasi email SMTP pada file yang menggunakan PHPMailer:
   a. SMTP Host: smtp.gmail.com
   b. Port: 587
   c. Encryption: TLS

    d. Email pengirim: akun Gmail yang telah dikonfigurasi
4. Pastikan fitur “Less Secure App” atau App Password telah dikonfigurasi pada akun email yang digunakan.

11.5 Running the Application

Langkah-langkah menjalankan aplikasi:

1. Aktifkan Apache dan MySQL.
2. Buka browser dan akses aplikasi melalui:

http://localhost/SECPROG

3. Registrasi akun baru melalui halaman registrasi.
4. Lakukan verifikasi email melalui link yang dikirimkan ke email pengguna.
5. Admin melakukan approval akun melalui dashboard admin.
6. Pengguna dapat login dan menggunakan fitur aplikasi sesuai dengan peran masing-masing.

11.6 User Roles and Access

Aplikasi memiliki dua peran utama:

- **Admin**: Mengelola data barang, kategori, pengguna, dan approval request.
- **User**: Mengelola permintaan barang, melihat stok, dan memperbarui profil.

Hak akses dibatasi berdasarkan peran untuk menjaga keamanan dan integritas sistem.

11.7 Notes

- Aplikasi dijalankan pada lingkungan pengembangan lokal.
- Deployment ke server publik memerlukan konfigurasi tambahan seperti HTTPS dan pengamanan server.
- Data login admin awal disesuaikan dengan data yang tersedia pada database.

12. Appendices (Lampiran)

12.1 Source Code Excerpts

Lampiran ini berisi cuplikan kode sumber utama yang berkaitan dengan implementasi keamanan pada aplikasi Warehouse. Cuplikan kode ditujukan untuk mendukung penjelasan pada Bab 8 mengenai secure programming.

Contoh kode yang disertakan dalam lampiran meliputi:

- Proses registrasi pengguna dan hashing password.
- Mekanisme verifikasi email dan OTP.
- Validasi login dan status akun.
- Penerapan Role-Based Access Control (RBAC).
- Validasi input dan prepared statement.
- Pengamanan unggah file foto profil pengguna.

12.2 Database Schema

Aplikasi Warehouse menggunakan beberapa tabel utama untuk mendukung proses autentikasi pengguna, manajemen inventori, serta pencatatan transaksi barang masuk dan keluar. Struktur tabel database dirancang untuk menjaga integritas data dan mendukung implementasi fitur keamanan yang telah dijelaskan pada bab sebelumnya.

Tabel-tabel utama yang digunakan dalam sistem Warehouse adalah sebagai berikut:

1. msuser

Tabel `msuser` digunakan untuk menyimpan data pengguna sistem.

Fungsi utama:

- Menyimpan informasi akun user dan admin
- Mendukung autentikasi dan otorisasi
- Menyimpan status verifikasi dan approval akun

2. msbarang

Tabel `msbarang` digunakan untuk menyimpan data master barang.

Fungsi utama:

- Menyimpan informasi barang yang tersedia di gudang
- Menjadi referensi pada transaksi barang masuk dan keluar

3. mscategory

Tabel `mscategory` digunakan untuk mengelompokkan barang berdasarkan kategori.

Fungsi utama:

- Memudahkan pengelolaan dan pencarian barang
- Menjaga konsistensi klasifikasi barang

4. msinventory

Tabel `msinventory` digunakan untuk mencatat stok barang yang tersedia.

Fungsi utama:

- Menyimpan jumlah stok terkini
- Menghubungkan data barang dengan kondisi inventori

5. barangmasuk

Tabel `barangmasuk` digunakan untuk mencatat transaksi barang masuk ke gudang.

Fungsi utama:

- Mencatat penambahan stok barang
- Menyimpan histori transaksi barang masuk

6. barangkeluar

Tabel `barangkeluar` digunakan untuk mencatat transaksi barang keluar dari gudang.

Fungsi utama:

- Mencatat pengurangan stok barang
- Menyimpan histori transaksi barang keluar

7. msrequest

Tabel `msrequest` digunakan untuk mencatat permintaan barang oleh user.

Fungsi utama:

- Mengelola proses request barang

- Mendukung mekanisme approval oleh admin
- Mengontrol distribusi barang

Kesimpulan Database

Struktur database Warehouse dirancang secara terpisah antara data master, transaksi, dan pengguna. Pemisahan ini bertujuan untuk:

- Menjaga integritas data
- Memudahkan pengelolaan sistem
- Mendukung penerapan kontrol akses dan validasi proses bisnis

12.3 Screenshots

Lampiran ini berisi tangkapan layar (screenshot) dari:

- Halaman login dan registrasi
- Proses verifikasi email dan OTP
- Dashboard admin dan user
- Proses request barang dan approval
- Halaman profil dan unggah foto

Screenshot ini bertujuan untuk memperjelas alur penggunaan aplikasi dan mendukung dokumentasi fitur.

12.4 Testing Evidence

Lampiran ini menyertakan bukti pengujian berupa tangkapan layar atau tabel hasil pengujian yang telah dijelaskan pada Bab 9. Bukti ini menunjukkan bahwa pengujian fungsional dan validasi keamanan dasar telah dilakukan.

12.5 References

Referensi yang digunakan dalam pengembangan dan penulisan laporan ini antara lain:

- OWASP Top 10 Web Application Security Risks
- Dokumentasi resmi PHP
- Dokumentasi PHPMailer
- Materi perkuliahan Secure Programming