Student: Rianna Aalto
Advanced SQL(2) Educator: Frederick Wennborg

# Hederlige Harrys Bilar –
# User Management Database Design Documentation

Table of Contents



---

### 1. Introduction

Hederlige Harrys Bilar is a car dealership that requires a user authentication system to manage account registration, login, and password recovery. This document outlines the design, implementation, and optimization of the database and its related components.

### 2. System Overview

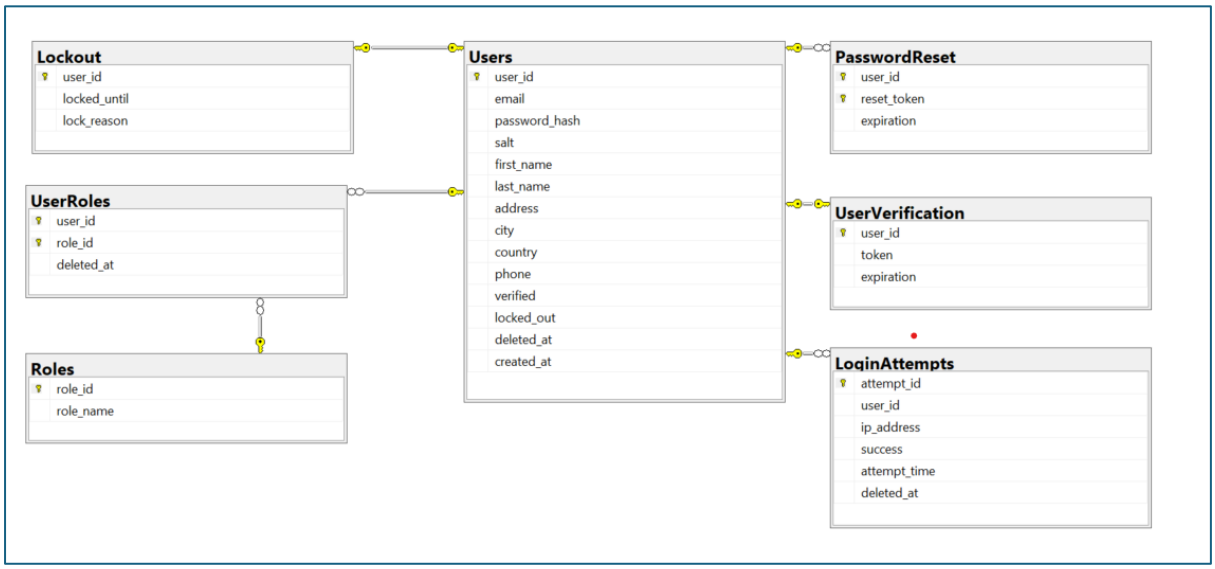
This project involves the development of a secure and optimized database that supports the following functionalities:

- User registration and verification via email token
- Role-based access control (Customer & Admin roles)
- Login attempts tracking with IP logging for security
- Password recovery mechanism with expiration-controlled tokens
- Lockout mechanism which blocks access after repeated failed logins for failed login attempts

- Optimized reporting views for login history and IP-based authentication analysis
- Performance optimizations (indexing, stored procedures, efficient queries)
- The database follows best practices in SQL development, ensuring efficiency, scalability, and security.

The system is designed using best practices in SQL development, ensuring efficiency, scalability, and security.

**ER Diagram**




## 3. Database Structure

3.1 Tables

The system consists of the following tables:

| Table Name | Description |
|---|---|
| Users | Stores user information (email, hashed password, salt, personal details, verification status, locked status). |
| Roles | Defines user roles such as Customer and Admin. |
| UserRoles | Maps users to roles in a many-to-many relationship. |
| UserVerification | Stores email verification tokens and expiration timestamps. |
| PasswordReset | Stores password reset tokens with expiration control. |
| LoginAttempts | Logs all login attempts, including IP address, timestamp, and success/failure status. |
| Lockout | Stores lockout status for users who exceed failed login attempts. |

3.2 Relationships

- Each user is assigned a role through UserRoles.
- Login attempts are recorded in LoginAttempts linked to users via user_id.
- Failed logins can result in a lockout (Lockout table).
- Password resets require a valid, non-expired token from PasswordReset.

## 4. Stored Procedures

4.1 RegisterUser

- Registers a new user with a salted and hashed password.
- Automatically assigns Customer role.

4.2 VerifyUserEmail

- Validates an email verification token.
- Marks user as verified upon success.

4.3 TryLogin

- Verifies user password (hashed + salted check).
- Logs successful & failed login attempts.
- Implements lockout after 3 failed attempts in 15 minutes.
- Stores debugging info in a temporary table for security analysis.

| Error Code | Description |
|---|---|
| -1 | Invalid Email |
| -2 | Account Locked |
| -3 | Too Many Failed Attempts (Locked for 15 Minutes) |
| -4 | Incorrect Password |
| 0 | Successful Login |

4.4 ForgotPassword

- Generates a password reset token.

- Stores the token in PasswordReset with a 24-hour expiration.
- Validates user before issuing the reset token.

4.5 SetForgottenPassword

- Validates the reset token before allowing password reset.
- Ensures token is still valid (not expired).
- Uses salting and hashing for secure password storage.

## 5. Views and Reporting

5.1 UserLoginReport

- Displays users' latest successful and failed login attempts.
- Aggregates login history for quick reporting.
- Helps track suspicious login behavior.

5.2 LoginAttemptsPerIP

- Summarizes all login attempts per IP address.
- Uses window functions for cumulative success/failure tracking.

## 6. Security Features

- Password Hashing: Uses SHA-256 with salting for secure password storage.
- Account Verification: Requires email verification before login is permitted.
- IP-based Logging: Tracks login attempts to detect suspicious activity.
- Account Lockout: Locks accounts after three failed attempts within 15 minutes.
- Token Expiration: Verification and password reset tokens have expiration timestamps.
- Enhanced TryLogin: Blocks login if account is locked and returns error codes for failures.

## 7. Performance Optimization

- Indexing: Optimized indexes on email, user_id, and attempt_time for faster queries.
- Optimized Stored Procedures:For efficient joins & transaction handling.
- Window Functions in Views (faster reporting calculations).

## 8. Testing and Validation

- Pre-inserted test accounts for demonstration.
- Stored procedures tested with different scenarios (valid/invalid login attempts, lockouts, password reset).
- SQL queries manually executed to verify the correctness of lockout policies and reporting views.

**Measure Execution Time:**

```
--measure Execution Time
SET STATISTICS TIME, IO ON;
GO
```

```
100 %
Messages
   SQL Server parse and compile time:
      CPU time = 0 ms, elapsed time = 0 ms.

    SQL Server Execution Times:
      CPU time = 0 ms,  elapsed time = 0 ms.

   Completion time: 2025-03-02T23:23:06.7930558+01:00
```

**Insert Test Users:**

This test inserts two sample users with hashed passwords and unique salts.
Expected Result: Two new user records should appear in the Users table.

```
DECLARE @salt1 VARBINARY(16) = CRYPT_GEN_RANDOM(16);
DECLARE @salt2 VARBINARY(16) = CRYPT_GEN_RANDOM(16);

INSERT INTO Users (email, password_hash, salt, first_name, last_name, address, city, country, phone, verified, locked_out, created_at)
VALUES
('erik.nilsson@gmail.com', HASHBYTES('SHA2_256', CONVERT(NVARCHAR(4000), @salt1) + 'Lösenord123@X9!b$2#'), @salt1, 'Erik', 'Nilsson', 'Storgatan 12'
('emma.svensson@hotmail.com', HASHBYTES('SHA2_256', CONVERT(NVARCHAR(4000), @salt2) + 'Lösenord456M$7&K!Q4'), @salt2, 'Emma', 'Svensson', 'Björkgata

---verifera
SELECT * FROM Users
```

sults Messages

| user_id | email | password_hash | salt | first_name | last_name | address | city | country |
|---|---|---|---|---|---|---|---|---|
| 1 | erik.nilsson@gmail.com | 0xECD6CE2639B67F7C084F0702E1D42F2CA897F6CFA6735C6... | 0x31EE031B4902EC972F6DA281AEFAE197 | Erik | Nilsson | Storgatan 12 | Stockholm | Sweden |
| 2 | emma.svensson@hotmail.com | 0xC7C0B08B97F54953004B80D99ACEEB901177EA0ED73C2D... | 0x3E39931AB834665CA0AEE69C7B3723FB | Emma | Svensson | Björkgatan 5 | Göteborg | Sweden |

**Test Successful Login:**

This test checks if the login system correctly authenticates a user with the correct password. -
Expected Result: Returns 0 (Success) and logs the attempt in the LoginAttempts table.

***EXEC TryLogin 'erik.nilsson@gmail.com', 'Lösenord123@X9!b$2#', '192.168.1.1';***

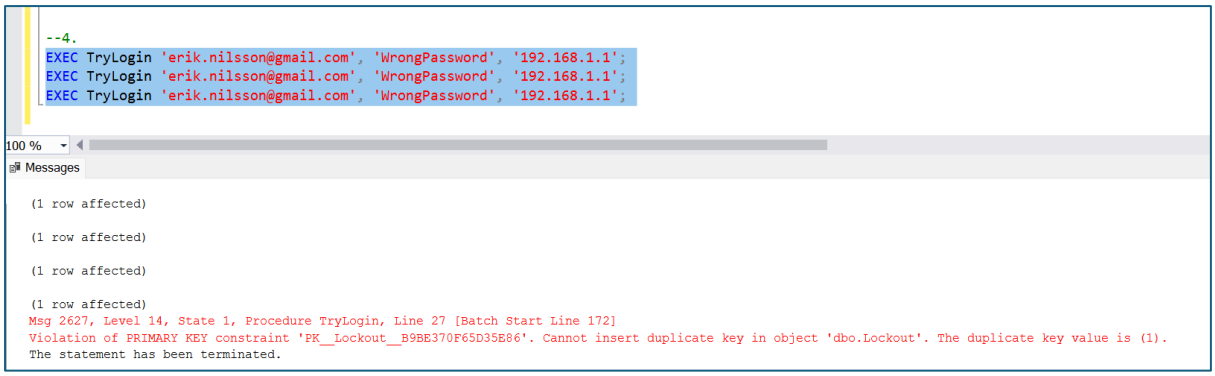
**Test Failed Login:**

This test checks if the system correctly identifies an incorrect password. Expected Result: Returns -4 (Invalid Password) and logs a failed attempt in LoginAttempts.

***EXEC TryLogin 'erik.nilsson@gmail.com', 'WrongPassword', '192.168.1.1';***

**Simulate Account Lockout (3 Failed Attempts):**

This test simulates a brute-force attack by trying an incorrect password three times.
Expected Result: After 3 failed attempts, account should be locked (Error Code: -3).



```
--4.
EXEC TryLogin 'erik.nilsson@gmail.com', 'WrongPassword', '192.168.1.1';
EXEC TryLogin 'erik.nilsson@gmail.com', 'WrongPassword', '192.168.1.1';
EXEC TryLogin 'erik.nilsson@gmail.com', 'WrongPassword', '192.168.1.1';
```

100 %

Messages

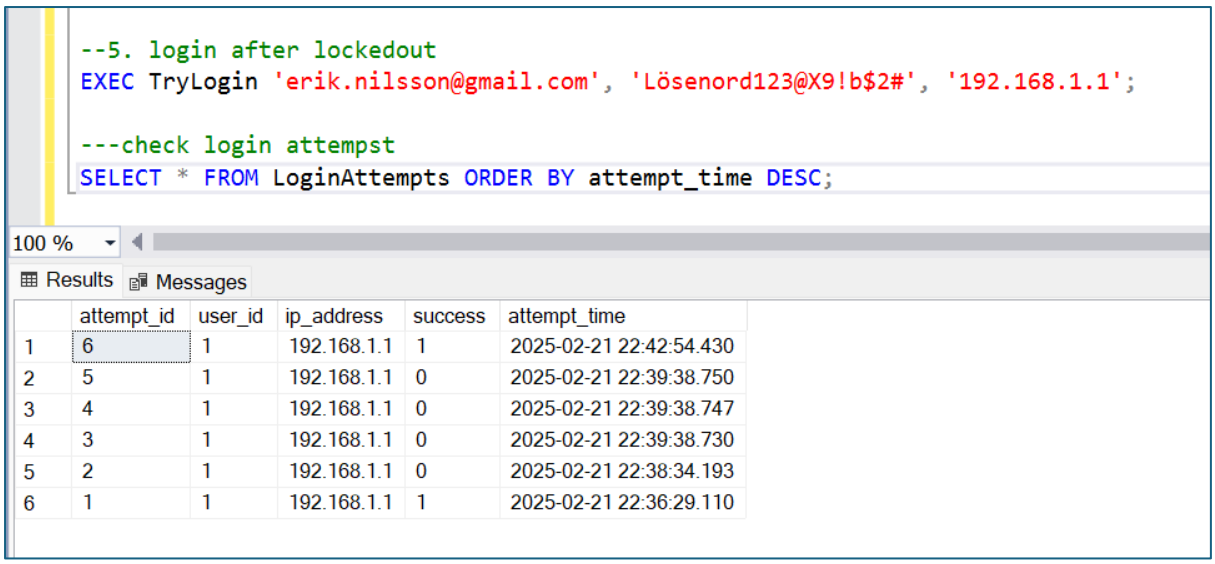
```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
Msg 2627, Level 14, State 1, Procedure TryLogin, Line 27 [Batch Start Line 172]
Violation of PRIMARY KEY constraint 'PK__Lockout__B9BE370F65D35E86'. Cannot insert duplicate key in object 'dbo.Lockout'. The duplicate key value is (1).
The statement has been terminated.
```

**Attempt Login After Lockout:**

This test checks if a locked account can still attempt login before the lockout expires.
Expected Result: Returns -2 (Account Locked) and prevents login.

**Check Login Attempts Log:**

This query retrieves all login attempts in order of attempt time.
Expected Result: Displays successful and failed login attempts.



```
--5. login after lockedout
EXEC TryLogin 'erik.nilsson@gmail.com', 'Lösenord123@X9!b$2#', '192.168.1.1';

---check login attempst
SELECT * FROM LoginAttempts ORDER BY attempt_time DESC;
```

100 %

Results | Messages

| | attempt_id | user_id | ip_address | success | attempt_time |
|---|---|---|---|---|---|
| 1 | 6 | 1 | 192.168.1.1 | 1 | 2025-02-21 22:42:54.430 |
| 2 | 5 | 1 | 192.168.1.1 | 0 | 2025-02-21 22:39:38.750 |
| 3 | 4 | 1 | 192.168.1.1 | 0 | 2025-02-21 22:39:38.747 |
| 4 | 3 | 1 | 192.168.1.1 | 0 | 2025-02-21 22:39:38.730 |
| 5 | 2 | 1 | 192.168.1.1 | 0 | 2025-02-21 22:38:34.193 |
| 6 | 1 | 1 | 192.168.1.1 | 1 | 2025-02-21 22:36:29.110 |

Check if a user is lockedout Check if User is Locked Out:
This query checks if a user is locked out. -- Expected Result: If locked, the user will have a record in the Lockout table.

```
--7 check if a user is lockedout
SELECT * FROM Lockout WHERE user_id = (SELECT user_id FROM Users WHERE email = 'erik.nilsson@gmail.com');
```

100 %

Results Messages

| | user_id | locked_until |
|---|---|---|
| 1 | 1 | 2025-02-21 22:54:38.750 |

Manually Unlock User:

This test removes the user from the Lockout table to allow login attempts again. Expected Result: User should be able to log in after running this command.

```
--8. Manually unlocked user
DELETE FROM Lockout WHERE user_id = (SELECT user_id FROM Users WHERE email = 'erik.nilsson@gmail.com');
```

100 %

Messages

```
(1 row affected)
```

Verify Login Reporting View:

This test retrieves user login history for analysis. Expected Result: Shows the last successful and failed login attempts for each user.

Check Login Attempts per IP Report:

This test retrieves statistics on login attempts per IP. Expected Result: Shows number of successful and failed login attempts per IP address.

```
--validate Performance of Views
SELECT * FROM UserLoginReport;
SELECT * FROM LoginAttemptsPerIP ORDER BY last_attempt_time ASC;
GO
```

100 %

Results Messages

| | email | first_name | last_name | senaste_lyckade | senaste_misslyckade |
|---|---|---|---|---|---|
| 1 | erik.nilsson@gmail.com | Erik | Nilsson | 2025-03-02 23:28:33.943 | 2025-03-02 23:29:48.220 |
| 2 | emma.svensson@hotmail.com | Emma | Svensson | NULL | 2025-03-02 23:29:48.233 |

| | ip_address | total_attempts | successful_attempts | failed_attempts | last_attempt_time | success_rate | cumulative_attempts | cumulative_successes | cumulative_failures |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 192.168.1.1 | 3 | 3 | 0 | 2025-03-02 23:28:33.943 | 1 | 3 | 3 | 0 |
| 2 | 192.168.1.2 | 2 | 0 | 2 | 2025-03-02 23:28:33.960 | 0 | 5 | 3 | 2 |
| 3 | 192.168.1.10 | 3 | 0 | 3 | 2025-03-02 23:29:48.220 | 0 | 8 | 3 | 5 |
| 4 | 192.168.1.11 | 2 | 0 | 2 | 2025-03-02 23:29:48.233 | 0 | 10 | 3 | 7 |

## 9. Conclusion and Further Development

### 9.1 Conclusion

This system successfully meets all authentication and security requirements. This includes secure login handling, role management, lockout mechanisms, and password recovery.

### 9.2 Further Development

To ensure scalability, security, and better performance, the system can be enhanced in several ways.

- Multi-factor authentication (MFA) for additional security.
- Enhanced role-based access control.
- Integration with a front-end web application.
- Real-time monitoring and alerting for security threats.

---

| Category | Improvement | Description | Implementation |
|---|---|---|---|
| 1. Security Enhancements | Implement Two-Factor Authentication (2FA) | Adds an extra layer of security, reducing unauthorized access. | - Implement OTP-based verification via SMS/email.<br>- Use TOTP (Google Authenticator). |
| | Improve Password Hashing | SHA-256 is good, but bcrypt, Argon2, or PBKDF2 offer better security. | - Use Argon2 for stronger hashing.<br>- Upgrade to bcrypt with a higher cost factor. |
| | Account Lockout Improvements | Attackers can bypass lockout by switching IPs. | - Implement progressive delay lockout (15 mins, 30 mins, 1 hour).<br>- Allow Admin override for unlocking accounts. |
| 2. Performance Optimizations | Indexing Optimization | Queries may slow down as the database grows. | - Create indexes on frequently searched columns:<br>CREATE INDEX idx_users_email ON Users(email); |
| | Archive Old Login Attempts | LoginAttempts table can grow very large, | - Move old login records to an archive table.<br>CREATE TABLE |

| | | | |
|---|---|---|---|
| | | affecting performance. | LoginAttempts_Archive AS SELECT * FROM LoginAttempts WHERE attempt_time < DATEADD(YEAR, -1, GETDATE()); |
| | Optimize Views & Queries | Large datasets can slow down reports. | - Use Materialized Views or Indexed Views.<br>- Optimize by creating indexes on reporting views. |
| 3. Scalability Improvements | Implement Load Balancing | Distribute traffic efficiently across multiple servers. | - Use Read-Replicas for handling queries.<br>- Implement Database Sharding (split by region). |
| | Move Authentication to a Microservice | The monolithic system may not scale well. | - Create an Authentication API using Node.js, .NET, or Python.<br>- Move authentication logic out of the database. |
| 4. User Experience & Admin Features | Improve Password Recovery | Users may mistype emails or miss reset emails. | - Show password strength indicator.<br>- Enable password reset via SMS. |
| | Admin Panel for User Management | Admins need better control over user accounts. | - Build a web interface for unlocking accounts, assigning roles, and viewing login history. |
| | Enable OAuth & Social Login | Users prefer Google, Facebook, or Microsoft login. | - Integrate OAuth 2.0 for third-party authentication.<br>- Store OAuth access tokens in the Users table. |
| 5. Logging & Monitoring | Enable Real-Time Login Monitoring | Detect brute-force attacks in real-time. | - Set up SQL Server Alerts:<br>CREATE EVENT NOTIFICATION FailedLoginAlert ON DATABASE FOR FAILED_LOGIN TO |

| | | | |
|---|---|---|---|
| | | | SERVICE 'AlertService'; |
| | Log Suspicious Activities | Detect abnormal logins (e.g., multiple country logins). | - Track IP Geolocation and store in LoginAttempts.<br>- Notify users if a login happens from a new country. |