Documentation for Hederlige Harrys Bilar – User Management & Login System

1. Introduction

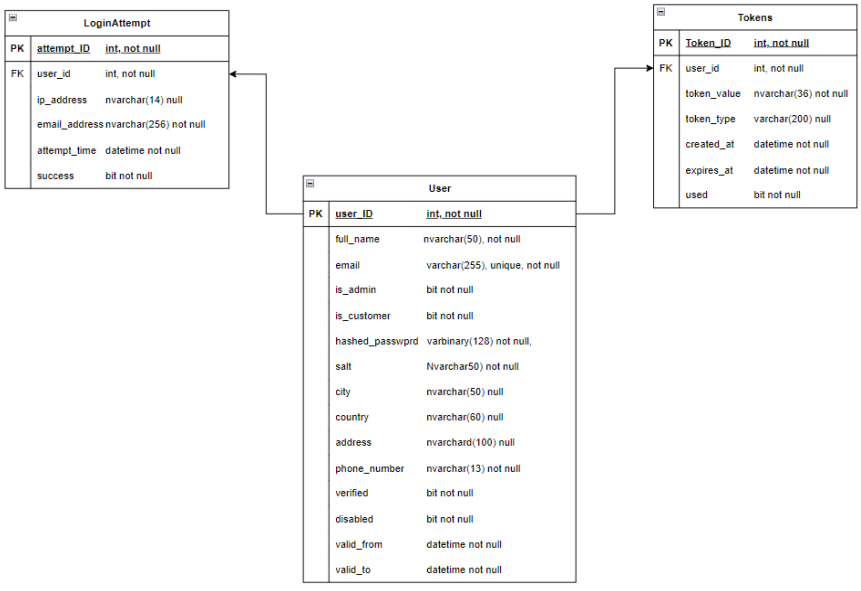
This document describes the design, implementation, and functionality of the user management and login system developed for Hederlige Harrys Bilar. The system provides functionality for user registration, email verification, password reset, and secure login—including features such as account lockout after repeated failed attempts, role assignment, and comprehensive logging and reporting. The document is structured to detail the database design, stored procedures, security measures, logging/reporting, and future optimizations.

2. Database Structure & ER Diagram

The database is designed with optimization and security best practices in mind. The main entities include:

- **Users:** Stores user information such as full name, email, hashed password, salt, address, phone number, verification status, and role (indicated by Is_admin and Is_customer flags).
- **Tokens:** Stores tokens used for email verification and password resets. Each record includes a unique token value (a GUID stored in Token_value), token type (e.g., 'Password Reset'), creation and expiration dates, and a flag indicating whether the token has been used.
- **LoginAttempt:** Logs every login attempt with details including the user ID, IP address, email, timestamp, and a success indicator.

An ER diagram was prepared to show the relationships:



Relationships:

- **Users** has a one-to-many relationship with **LoginAttempt** (one user can have many login attempts).
- **Users** has a one-to-many relationship with **Tokens**.

3. Tables and Column Descriptions

3.1 Users Table

- **User_id INT):** Primary key, auto-incremented (IDENTITY). Unique identifier for each user.
- **Full_name VARCHAR(50):** The user's full name.
- **Email VARCHAR(256):** Unique email address used for login and verification.
- **Hashed_password VARBINARY(128):** Stores the hashed version of the user's password. Hashing is performed externally on the web server.
- **Salt NVARCHAR(50):** A unique salt used in conjunction with the password hash. (The salt is stored as a hex string converted from a binary value.)
- **City, Country, Address:** Optional columns (NULL allowed) for user address details.
- **Phone_number NVARCHAR(13):** User's phone number; required for contact or multi-factor authentication.
- **Verified BIT:** Indicates whether the user’s email has been verified.
- **Disabled BIT:** Indicates if the account is locked or disabled.
- **Valid_from DATETIME & Valid_to DATETIME:** Define the account's active period.

3.2 Tokens Table

- **Token_id INT:** Primary key, auto-incremented (IDENTITY).
- **User_id INT:** Foreign key referencing Users.
- **Token_value NVARCHAR(36):** The unique token code generated.
- **Token_type VARCHAR(200):** Describes the purpose of the token (e.g., "Password Reset").
- **Created_at DATETIME:** Timestamp when the token was generated.
- **Expires_at DATETIME:** Expiration time of the token (24 hours after creation).
- **Used (IT:** Indicates whether the token has been used.

3.3 LoginAttempt Table

- **Attempt_id INT:** Primary key, auto-incremented (IDENTITY). Unique identifier for each login attempt.
- **User_id INT:** Foreign key referencing Users. Stores the ID of the user attempting to log in.
- **Ip_address NVARCHAR(14):** IP address from which the login attempt was made.
- **Email_address NVARCHAR(256):** The email provided during the login attempt.

- **Attempt_time DATETIME:** The timestamp when the login attempt occurred.
- **Success BIT:** Indicates whether the login attempt was successful (1) or not (0).

4. Stored Procedures

4.1 TryLogin Stored Procedure

This procedure manages the login process:

- **Parameters:** Accepts email, a hashed password, and IP address.
- **Functionality:**
    - Retrieves the stored hashed password from the Users table.
    - Compares the provided password with the stored one.
    - Logs a failed attempt if the passwords do not match.
    - Counts failed attempts within the last 15 minutes:
        - If there are 3 or more failed attempts, the account is locked (Users.Disabled is set to 1), and an appropriate log is recorded.
        - If there is 1 failed attempt or multiple (but fewer than 3), an alternative log is recorded.
    - Logs a successful login if the passwords match.

- **Logging:**
  The procedure logs the outcome in a permanent table (LoginAttempt) and also returns a log via a global temporary table (##LoginLog) for testing purposes.
- **Error Codes:** Returns -1 for account lockout, -2 for a wrong password, and 0 for a successful login.

4.2 ForgotPassword Stored Procedure

This procedure implements the "forgot password" functionality:

- **Parameters:** Accepts an email.
- **Functionality:**
    - Looks up the user by email.
    - Generates a unique token using NEWID()).
    - Sets the token expiration to 24 hours from the current time.
    - Inserts the token into the Tokens table with Token_value and Token_type (e.g., "Password Reset").
- **Error Handling:**
  Returns an error code if the user is not found.
- **Note:**
  The token is generated on the server and stored, it will later be sent to the user for verification during the password reset process.

4.3 SetForgottenPassword Stored Procedure

This procedure updates the user's password after verifying the reset token:

- **Parameters:** Accepts email, a new hashed password, and the token.
- **Functionality:**
    - Verifies that the token provided is valid (i.e., exists, is not expired, and has not been used).
    - If valid, generates a new salt, hashes the new password (combining the password with the new salt), updates the Users table, and marks the token as used.
- **Error Handling:**
  Returns specific error codes for conditions such as user not found or token invalid/expired.

5. Security Considerations

- **Password Storage:**
  Passwords are not stored in plain text. Instead, the system stores a hashed version of the password using SHA2_256 along with a unique salt. Importantly, the password hashing is performed on the web server during account creation for security reasons. The already hashed password is then imported into the SQL database, so no plain text or hashing occurs directly within SQL.
- **Token Management:**
  Password reset tokens are generated using GUIDs and stored in the Tokens table (in the Token_value column) with an expiration time of 24 hours. Tokens are marked as used once the password reset process is completed.
- **Login Lockout:**
  If a user fails to log in three times within 15 minutes, the system locks the account (or prevents further login attempts), even if the correct password is eventually provided.

6. Logging and Reporting

- **LoginAttempt Logging:**
  Every login attempt (successful or failed) is logged with details such as the user’s ID, IP address, email, timestamp, and success status. This data is used for auditing, security analysis, and reporting purposes.
- **Views and CTE Reports:**
  Additional views have been created that:
    - Report the most recent successful and failed login attempts per user.
    - Summarize the cumulative login attempts per IP address using window functions.

- **Temporary Table Logging (for SP Testing):**
  Stored procedures (e.g., TryLogin) use a global temporary table to log login attempts for testing and debugging, providing immediate feedback on procedure execution.

7. Optimization and Future Improvements

- **Indexing:**
  We have added non-clustered indexes on key columns that are frequently queried—such as Email in the Users table, Attempt_time in the LoginAttempt table, and User_id (and in some cases a composite index with Token_value) in the Tokens table—to improve query performance. Although these indexes do not show a noticeable performance difference with our current dataset of just five users, they will become critical as the database grows. Once the system scales to a few million rows, the optimized indexes will significantly reduce query execution times and maintain efficient performance.
- **Execution Plans:**
  To ensure our optimizations are effective, we have reviewed execution plans on key queries. For example, analysis of the query filtering by **Email** in the Users table confirmed that an index on the Email column significantly improves performance. This process helps us identify any potential bottlenecks early and ensures that our indexing strategy is well-targeted for future growth.
- **Security Enhancements:**
  In addition to securely hashing and salting passwords, further security can be achieved by encrypting sensitive data at rest and in transit.

8. Conclusion

This documentation outlines the design and functionality of the user management system for Hederlige Harrys Bilar. It covers the key database tables, stored procedures, security measures, logging, and optimization strategies. Although the current dataset is small, the design has been built with scalability in mind—ensuring that as the database grows, performance will remain efficient and security will be maintained. Future improvements include additional query optimizations, encryption enhancements, and advanced monitoring to ensure that the system can handle a larger user base and increased login activity.