CraftVerify

Security Low-Level Design

Team Natural Selection

GitHub Link: <https://github.com/Natural-Selection491/CraftVerify>

Members:

Kameron Ferrer

Aiden Hocks

Jacob Phillips

Parth Thanki

An Nguyen

Khuong Nguyen

12/03/2023

| **Version** | **Who Edited** | **Date** | **What Is Edited/Changed?** |
| --- | --- | --- | --- |
| **1.0** | **Aiden and Kameron** | **12/03/2023** | **Created Document** |
|  |  |  |  |

**Security Library**

Low-Level Design Document Draft for Authentication

**1. Introduction**

This section of the document describes the purpose, scope, and functionality of the Authentication component in the Security System. It provides an overview of the feature's capabilities, such as validating a user’s identity.

1.1 Purpose

The Authentication component allows users to verify their identity before giving them access to registered user-specific content.

1.2 Scope

Any unauthenticated user can attempt to authenticate with a username and the OTP they received inside their email.

**2. Authentication Class Design**

This section outlines the structure of the Authentication class.

2.1 Attributes

* None

2.2 Methods

* validateLogin(username : string, OTP : string) : string
* hashOTP(OTP : string, userHash : string) : string
* validateUserOTPHash(username : string) : string
* LogInsert(UserHash : string, IsSuccessful : bool, DateCreate: date) : bool
* GetUserRole(UserHash : string) : string

2.2 v2 Methods (Also trying to make the class more extensible)

* hashPassword(password : string, identifier : string) : string
* validateUserOTPHash(username : string) : string
* LogInsert(UserHash : string, IsSuccessful : bool, DateCreate: date) : bool
* createUserRoleObject(userRole : string, UserHash : string) : RoleObject

**3. Sequence Diagrams for Authentication**

3.1 Success Scenario

* Attached Separately (for now)

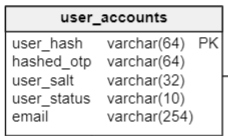
3.2 Failure Scenario

(We will submit failure scenarios when the success scenario diagram gets approval)

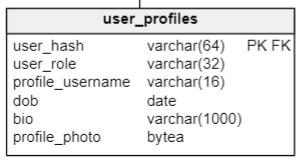
**4. Database Design**

This section details the database schema required to support profile updates, including modifications to existing tables and the introduction of new tables.

4.1 User Account Table

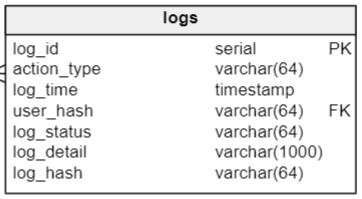


4.2 User Profile Table



4.3 Logging Table

The Logging table is used to track each authentication attempt.



**5. Error Handling and Logging**

This section outlines the strategies for handling errors and logging during the Authentication process.

5.1 Error Handling

Error handling methods include:

* Validation Errors: Before updating, validate data formats (e.g., date formats, string lengths). Reject invalid updates with descriptive error messages.
* Database Errors: Catch exceptions during database operations. Log these errors with details for troubleshooting.
* Access Control Errors: Ensure the user has permission to update the profile. Unauthorized attempts should be logged and rejected.

5.2 Logging

Approach for logging:

* Successful Updates: Log all successful authentication attempts with timestamps and user hash.
* Failed Attempts: Log failed update attempts, including error details and user hash. Use this data to identify common issues or potential security concerns.

**6. Test Cases**

This section lists the test cases for the Authentication feature.

6.1 Unit Tests

Unit tests for the Authentication class:

* Test validate login: Verify that the validateLogin() method correctly validates the registered account login credentials.
* Test hashing OTP: Test the hashOTP() method and confirm that the method correctly outputs a hash string when given a One-time Password and the User’s random salt string
* Test validate user OTP hash: Test validateUserOTPHash() for comparing user-inputted OTP to the hashed OTP stored in the database.
* Test log insert: Validate that logInsert() correctly inserts into the log table.
* Test reading user role: Validate that the getUserRole() method correctly returns the User’s user role when given the User’s hash value

6.2 Integration Tests

Integration tests cover the complete authentication process:

* Authentication Flow: Test the entire flow from the user login request to the database update and logging.
* Error Handling: Simulate various error conditions (e.g., database unavailability, invalid input) and verify that they are handled appropriately.
* Security and Permissions: Verify that only authorized users can authenticate into an existing registered account.