Musi-Cali

Security Low-Level Design Document

Team Phoenix

https://github.com/JPJ-5/TeamPhoenix
Team Members:
Jason Jitsiripol (Team Leader)
Diego Garcia
Brandon Muhumuza
Joshua Reyes
Shane Kerr

12/16/2023

Low-Level Design Version Table:

Version	**Description**	**Date**
1.0	Initial Low-Level Design Requirements ● Feature (Authorization) ○ Sequence Diagrams ○ Classes/Interfaces ○ User Stories	**12/16/2023**

Table of Contents:

Core Components..4
Authorization Success Scenario...4
Authorization Failure Scenario...5
IAuthorization Interface... 6
Authorization Class...6
UserAuthZClass... 6-7
UserStories…………………………………………………………………………………....7-8

Core Components

Authorization

Sequence Diagrams are in a separate document.

Success Scenarios:

- **Context**: An authenticated user with the necessary permission attempts to access a feature that requires a specific permission.
  - **Success**: The IsUserAuthorized method correctly determines that the user has the required permission, and access to the restricted feature is granted.

- **Context**: An authenticated user with a specific role attempts to access content reserved for users with that role.
  - **Success**: The IsUserAuthorized method correctly identifies that the user has the required role, allowing them to access the role-specific content.

- **Context**: An unregistered user completes the registration process, verifies their email, and logs in for the first time.
  - **Success**: The IsUserAuthorized method correctly determines that the user is now a registered user, and the user successfully logs in.

- **Context**: A registered user logs in and checks their registration status.
  - **Success**: The IsUserAuthorized method correctly determines that the user's registration status is completed, indicating that the user is fully registered.

- **Context**: An authenticated user with an active session attempts to perform an action on the platform.
  - **Success**: The IsUserAuthorized method correctly identifies that the user is both logged in and has an active session, allowing them to proceed with the action.

Failure Scenarios:

- **Context:** An authenticated user without the required permission attempts to access a feature that requires a specific permission.
  - **Failure:** The IsUserAuthorized method correctly identifies that the user lacks the required permission, and access to the restricted feature is denied.

- **Context:** An authenticated user without a specific role attempts to access content reserved for users with that role.
  - **Failure:** The IsUserAuthorized method correctly identifies that the user lacks the required role, and access to the role-specific content is denied.

- **Context:** An unregistered user attempts to check their registration status without completing the registration process.
  - **Failure:** The IsUserAuthorized method correctly determines that the user's registration status is incomplete, as the user has not completed the registration process.
- **Context:** An authenticated user with a completed registration status checks if they are an unregistered user.
  - **Failure:** The IsUserAuthorized method incorrectly identifies the authenticated user as unregistered, providing inaccurate information.

- **Context:** A registered user with an expired session attempts to perform an action on the platform.
  - **Failure:** The IsUserAuthorized method correctly identifies that the user is logged in but has an inactive session, denying them access to the action.

- **Context:** An authenticated user with an expired session checks their session status.
  - **Failure:** The IsUserAuthorized method correctly identifies that the user's session is inactive, indicating that the user should log in again.

Classes/Interfaces

IAuthorization Interface (IAuthorization):

- **Methods:**
    - IsUserAuthorized(Principal userPrincipal, string resource, string action): bool:
        - Checks if a user is authorized based on their principal claims.
        - Parameters:
            - userPrincipal: The principal representing the user.
            - resource: The resource being accessed.
            - action: The action being performed on the resource.
        - Returns: True if the user is authorized; otherwise, false.

Authorization Class (Authorization):

- **Methods:**
    - IsUserAuthorized(Principal userPrincipal, string resource, string action): bool:
        - Checks if a user is authorized based on their principal claims.
        - Parameters:
            - userPrincipal: The principal representing the user.
            - resource: The resource being accessed.
            - action: The action being performed on the resource.
        - Returns: True if the user is authorized; otherwise, false.

UserAuthZ Class (UserAuthZ):

- **Attributes:**
    - Username: string?: Gets or sets the username.
    - Salt: string?: Gets or sets the salt for password hashing.
    - Password: string?: Gets or sets the hashed password.
    - Permissions: List<UserPermission>: Gets or sets the list of user permissions.
    - Roles: List<UserRole>: Gets or sets the list of user roles.
    - RegistrationTimestamp: DateTime: Gets or sets the timestamp of user registration.
    - LastLoginTimestamp: DateTime: Gets or sets the timestamp of the last user login.
    - LastActivityTimestamp: DateTime: Gets or sets the timestamp of the last user activity.
- **Methods:**

- GetUserRoles(): List<UserRole>:
  - Gets the user roles.
  - Returns: The list of user roles.
- GetUserPermissions(): List<UserPermission>:
  - Gets the user permissions.
  - Returns: The list of user permissions.
- IsAuthorize(string userIdentity, string securityContext): bool:
  - Checks if the user is authorized based on a security context (dummy implementation).
  - Parameters:
    - userIdentity: The user identity.
    - securityContext: The security context to check.
  - Returns: True if the user is authorized; otherwise, false.

User Stories

1. **As a Prospective User,**
   - I want to initiate the Musi-Cali account creation process so that I can explore and access limited features tailored to my preferences.
   - I want to provide the required registration information (email, password, name) so that I can create a Musi-Cali account and enjoy personalized features.
   - I want to receive a confirmation email for account verification so that I can verify my email address and complete the registration process.
   - I want access to limited features until my registration is finalized. so that I can begin using Musi-Cali while completing the registration steps.
2. **As a Developer,**
   - I want to implement secure authentication and registration processes so that user data is protected and the system complies with security best practices.
   - I want to define and manage user roles and permissions so that access control is organized and can be easily modified as needed.
   - I want to ensure the availability of APIs for user authentication and authorization so that other system components can integrate seamlessly with the user management system.
   - I want to implement error handling for authentication and registration scenarios so that users receive meaningful feedback in case of issues or unsuccessful attempts.
3. **As a System Admin,**
   - I want to have administrative access to user management features so that I can efficiently manage user accounts and address any issues.

- I want to receive notifications for multiple failed authentication attempts so that I can take proactive measures to secure user accounts and investigate potential security threats.
- I want to be able to disable user accounts if necessary so that I can prevent unauthorized access and maintain system security.
- I want to have a clear overview of user roles and their associated permissions so that I can ensure proper access control and make informed decisions about user roles.

Authentication

Sequence Diagrams

Success Outcomes

- User Authentication: User is able to authenticate account with proper credentials.

Failure Outcomes

- User Auth: User submits incorrect username or otp or password. Message displays "Invalid security credentials provided. Retry again or contact system administrator"
- User Auth: User submits valid information but for an account that is disabled. Message displayed "Account is disabled. Perform account recovery first or contact system administrator"
- User Auth: User was given incorrect formatted otp code. Message displays "Invalid OTP used. Please request a new OTP"
- SendOtp: email was unable to be sent to the user. Message displays " OTP email send failed"

Classes/Interface

Public Class Result

- Attributes
  - HasError: Indicates whether the operation resulted in an error.
  - ErrorMessage: Holds the error message if an error occurs during an operation.
  - StatusCode: Represents the status code of the operation.

Public Class UserAuth

- Attributes
  - Username : Username of the user trying to sign in
  - OTP: one time password given to the user
  - Password: password set up by the user after registration is complete
  - otpTimestamp: timestamp the otp was created
  - Timestamp: timestamp of first failed attempt
  - FailedAttempts: number of total railed attempts
  - LastFailedAttemptTime: time of last authentication failure
  - IsDisabled: bool value of whether account has been disabled
  - IsAuth: bool value of whether account already is authenticated from the otp
  - Salt: salt string to hash password

Public Class UserAccount

- Attributes
  - public string Username: Username of the user trying to sign in
  - public string Email: email of the user
  - public bool IsDisabled: if the account has been disabled due to failed attempts

- public string Password: password of the user
- public bool isEnabled: bool value if user has passed authentication to give access to the website.
- public string Salt: salt to encrypt passwords

Public Class Authentication

- Methods:
  - Authenticate(string username, string password, string ipAddress): Result
    - Authenticates a user based on provided credentials.
  - RecordFailedAttempt(UserAuth userA, string ipAddress): Result
    - Records a failed authentication attempt.
  - IsValidOTP(string otp): bool
    - Checks if the provided OTP meets the required parameters.
  - ValidateOTP(UserAuth userA, string otp): Result
    - Validates the provided OTP for a user.
  - ValidatePassword(UserAuth userA, string password): bool
    - Validates the provided password for a user.
  - isValidUsername(string username): bool
    - Checks if the provided username meets the specified guidelines.

User Stories

- For all users:
  - As a user of the system, I want to securely authenticate myself and access my personal resources so that I can use the application for my intended purposes.