|  |  |  |
| --- | --- | --- |
| RAZDA Co. | | |
| **Filename: [user\_management.py]** | | |
| **Summary***:*  The user\_management.py file is designed to handle all aspects of user account management for the Razda Market platform. This includes creating, updating, and deleting user accounts, managing roles and privileges, enforcing password security, and handling account status (active/inactive). Additionally, it integrates authentication measures for admin-level access and provides functionality for password resets. With its extensive validation mechanisms, this module ensures compliance with account security policies, helping maintain a secure and orderly user environment. | | |
| ***Processes*** | | |
| * **Admin Authentication** | * **Authenticates admin credentials before permitting access to sensitive functions like adding, deleting, or updating user accounts.** | |
| * **Password Hashing and Verification** | * **Uses bcrypt to hash passwords before storage and verify passwords during login, ensuring password security and integrity.** | |
| * **User Account Creation** | * **Adds a new user account to the database after validating username, email, and password format, ensuring data integrity and security standards.** | |
| * **User Account Deletion** | * **Removes a specified user from the system based on their unique user ID, facilitating user lifecycle management and data privacy.** | |
| * **User Account Update** | * **Modifies account information (e.g., username, email, password, status) based on specified parameters, maintaining data accuracy.** | |
| * **Listing all Users** | * **Retrieves and displays a list of all user accounts, including essential details such as status, created date, and last login, enhancing admin visibility.** | |
| * **View Specific User Details** | * **Displays detailed information for a specified user, assisting administrators in user account oversight and decision-making.** | |
| * **Password Reset Initiation** | * **Generates a password reset token for users who have requested a reset, storing it with an expiration timestamp for security.** | |
| * **Perform Password Reset** | * **Resets the user's password using a provided reset token, ensuring that only valid tokens are accepted to prevent unauthorized resets.** | |
| * **Input Validation and Sanitization** | * **Enforces format standards for usernames, emails, and passwords, reducing risks of invalid data and enhancing data consistency.** | |
| **Files it Gets Information From:** | | **Files it Sends too:** |
| * **Environment Variables** (.env): For credentials and configuration (ADMIN\_USERNAME, ADMIN\_PASSWORD, etc.). | * **Logging File**: Logs events to user\_management.log for accountability and tracking changes in user accounts. | |
| * **Database (MySQL)**: Accesses the Razda Market database to store and retrieve user data. | * **Alert/Email System** (hypothetical): In a full implementation, password reset tokens would be sent via email. | |
| * **CLI Input**: Accepts input through the command line for various user management operations. |  | |
| **Expected input into file:** | | **Expected output from file:** |
| **Environment Variables**: For authentication and DB connection.  ‘’’python  DB\_HOST, DB\_USER, DB\_PASSWORD, DB\_NAME, ADMIN\_USERNAME, ADMIN\_PASSWORD  ‘’’  **CLI Input**: User details (e.g., username, email) and commands for account operations. | | **Log Records**: Each action logs success or failure in the user\_management.log.  ‘’’python  logging.info("User ID {user\_id} updated successfully.")  ‘’’  **Database Records**: Adds, updates, or deletes entries in the users table. |
| **Things that need to be taking place:** | | |
| |  | | --- | | **● Admin-Only Access Enforcement: Ensures only authorized administrators can manage user accounts, preventing unauthorized modifications.** |  |  | | --- | | **● Password Security: Enforces strong password policies and securely hashes passwords to protect user data.** |  |  | | --- | | **● Comprehensive User Validation: Validates usernames, emails, and passwords before database operations, enhancing data reliability and preventing errors.** |  |  | | --- | | **● User Lifecycle Management: Handles all stages of user accounts from creation to deletion, allowing streamlined user management.** |  |  | | --- | | **● Detailed Logging: Tracks every user management action (e.g., creation, deletion) to provide accountability and assist in troubleshooting.** |  |  | | --- | | **● Secure Password Reset: Generates time-sensitive reset tokens, protecting against unauthorized password changes and enhancing user security.** |  |  | | --- | | **● Real-Time User Overview: The ability to list all users and view individual details aids in overall user account administration and monitoring.** |  |  | | --- | |  | | | |
| Edit log (update each time you make changes to doc or file). | | |
| Contribution to the entire Project:  The **user\_management.py** file is integral to the security and functionality of the Razda Market platform. By enforcing strict user account controls and maintaining high standards for password security, this module strengthens the platform’s resilience against unauthorized access. The admin authentication requirement safeguards sensitive operations, while the detailed logging functionality provides a clear audit trail of all user management activities.  Additionally, the password reset functionality and validation protocols further ensure that the platform adheres to best practices in user data security. This module not only supports a robust account management system but also aligns with data privacy standards, positioning it as a cornerstone in the Razda Market’s user-focused security infrastructure. | | |
| * - Oliver Smith (Razda Admin) Nov 8, 2024: Added new functionality for user account suspension and password reset initiation. | | |