Project Design & Status Report

CMSC 495 – 7980

Group 1

Brent Harrell

Jay Bachir

Thomas Hodges

# Overview

## Background

This document describes the design details, architecture, and approach of Mellon application. Mellon is a password storage application that allow the user to store and manage passwords for various accounts.

## Design goals

The design of Melon application should have a user-friendly user interface with easy to navigate functionality. The functionalities included in the application should ensure account details security and encrypted communication channels.

## Design Approach

Mellon application uses the IEEE Secure Software Design standards (SSD)[[1]](#footnote-0) as a design baseline to ensure a flawless account details storage and secure login mechanism. The SSD standards are a set of rules to be followed during the software design process. Unlike other software standards, the SSD focuses on design flaws rather than finding bugs within the software.  Additionally, the OWASP’s methodologies, vulnerability mitigation, recommendations will be used to detect and mitigate any security flaws.

Mellon application will implement Component-based Software Engineering (CBSE) architecture.[[2]](#footnote-1) In CBSE, the software is designed into distinct sections and each section is responsible for a specific concern in order to easily maintain and enrich the software’s functionalities. At the same time, CBSE implements reuse-based approach where the various sections contain independent components and each components consists of a set of methods. As a result, the software reuses the same components repeatedly.

## Components

The main application technology stack is defined as follows:

1. User Interface (front end):
   1. JavaFX
   2. CSS to customize elements
2. Backend (Core functionalities):
   1. Java classes
   2. Internal libraries and drivers: JDK 1.8, OJDBC7
3. Database:
   1. Oracle EE 12C
   2. Amazon Web Services (AWS)

Architecture.png

Figure 1 – Application Architecture

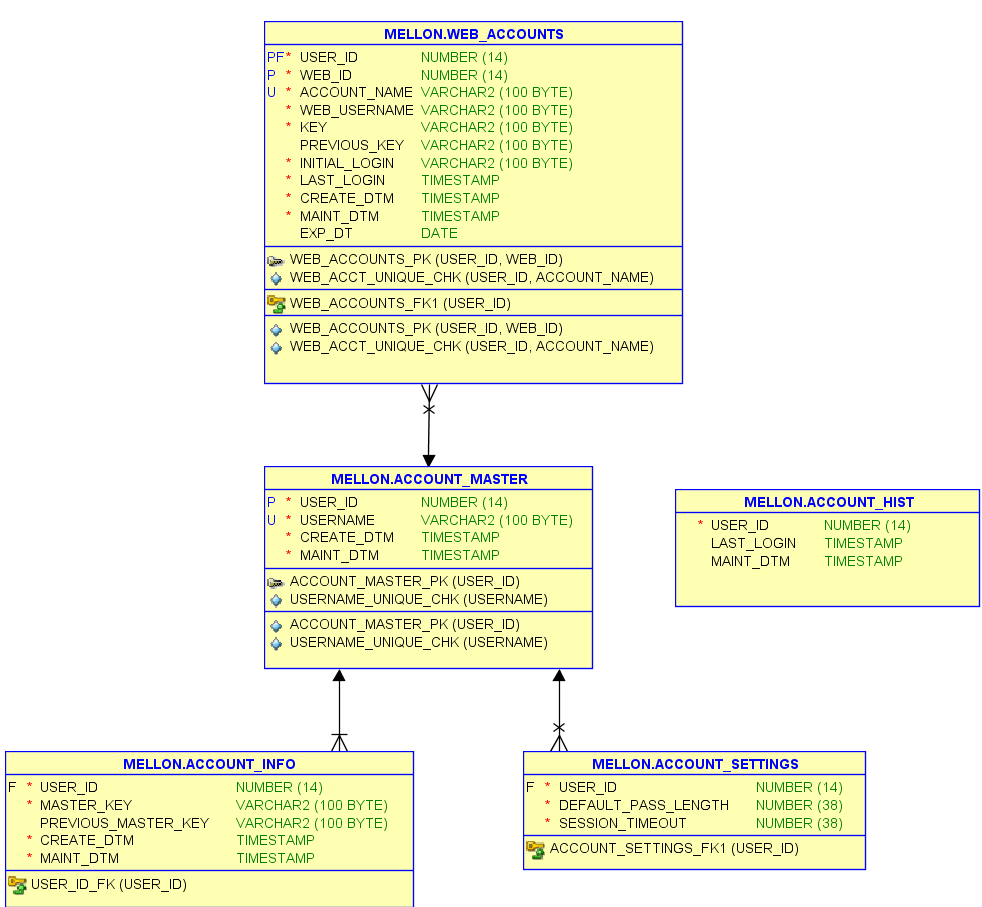


    Figure 2 – Entity Relationship Diagram

# Project Goals and Features

|  |  |  |  |
| --- | --- | --- | --- |
| # | Functionality | Brief Description | % Completed |
| i | Note on UI Progress:  Completion status for each UI element listed below is limited to functionality. Tweaks to UI appearance through CSS will be on going. | |  |
| 1 | User login UI | 1. Display user login page with username and password fields.  2.  Contains two buttons, “Login” and “Signup” | 100 |
| 2 | User login verification | Verifies that the user’s account exists within the database | 100 |
| 3 | User login authentication | Compares the user’s credentials to the stored credentials and either permits or denies access. | 100 |
| 4 | Error and data validation checks | 1. Checks if the username and password fields are populated in the login page.  2.  Displays a popup message accordingly | 100 |
| 5 | User signup UI | Displays the user’s signup form. It will consist of username, new password, and password verification fields along with signup button. Verification field is highlighted red until it matches the first input. | 100 |
| 6 | New account creation | Initializes UserInfoSingleton object and stores username and password in the database with unique generated USER\_ID. | 100 |
| 7 | Existing account verification | Checks whether the hash of the username input already exists in the database. | 100 |
| 8 | Error and data validation checks | 1. Checks if the username, password, and password verification fields are populated in the signup page.  2.  Verifies that the user’s new password and verification password are identical.  3. Displays error when the user does not exists in the database | 100 |
| 9 | User Home UI | 1. Lists all web accounts linked to the authenticated user.  2.  Each web account is clickable to view its details.  3.  Contains a clickable edit button linked to account edit page. | 100 |
| 10 | User web accounts info display | Retrieves all web accounts linked to the authenticated user and creates an array list of web account object. | 100 |
| 11 | Web Accounts View UI | 1. Displays the web accounts detail such as account name, username, and password.    a. The password field will be blurred until the user request to reveal the password. | 100 |
| 12 | User web account details | Retrieves the web account object and uses getter methods to assign each requested value. | 100 |
| 13 | Web Accounts Edit UI | 1. Displays stored account data.  2. Contains account nickname (Account name), username, and password fields for user’s edits.  3. Contains advanced button that links to password customization UI  4. Allows the user to quickly customize the account’s password through a set of checkboxes to include uppercase, lowercase, Symbols, Numbers.  5. Contains a dropdown menu to modify the desired password length (8, 16, 24, 32, and 48) or select a customized password length through a text field limited to number input.  6. Allows the user to modify the expiration date to the account password through a checkbox which in turn displays an interactive calendar.  7. Allows the user to generate a new password that matches the desired password requirements through the Generate password button.  8. Contains a “Save Account” to allow the user to save the account change and return to the accounts’ list | 100 |
| 14 | User web account details | 1. Retrieves the selected web account from the array list and uses a set of getters methods to assign each value.  2. Retrieves the user’s account modifications and uses setters methods to change the current account variables. | 100 |
| 15 | Navigation bar UI | 1. This bar will be displayed on all the application’s pages to allow the user to quickly select an application feature.  2. Has 5 different clickable icons (Home, Add an account, Account settings, help, and logout) that initiate different requests. | 100 |
| 16 | User normal logout | 1. Allows the user to securely logout of the application by nulling all objects linked the authenticated user.  2. Drops all database connection.  3. Initiates a request to display the login page. | 100 |
| 17 | User timeout logout | Tracks a predefined timeout variable (Defaulted to 10 minutes) once the timeout duration expires, a request is initiated to logout the user | 50 |
| 18 | Application Timer | Creates an internal timer that starts when the mouse becomes idle and resets 0 when the mouse moves. | 75 |
| 19 | Help UI | Shell created, content will be added closer to completion to include images of the UI’s final state for tutorials. | 25 |
| 23 | Web Account Creation UI | 1. Contains account nickname (Account name), username, and password fields for user’s input.  2. Contains advanced button that links to password customization UI  3. Allows the user to quickly customize the account’s password through a set of checkboxes to include uppercase, lowercase, Symbols, Numbers.  4. Contains a dropdown menu to select the desired password length (8, 16 (default), 24, 32, and 48) or select a customized password length through a text field,  5. Allows the user to set the expiration date to the account password through a checkbox which in turn displays an interactive calendar.  6. Allows the user to generate a new password that matches the desired password requirements through the Generate password button.  7. Contains a “Save Account” button to allow the user to save the new account and return to the accounts’ list | 100 |
| 24 | Password Customization UI | Contains a set of checkboxes which allow the user to choose the set of special characters and symbols to be included in the generated password | 100 |
| 25 | Customized password generation | Retrieves the user’s password customization input and length, then generates the corresponding password | 100 |
| 26 | Account Settings UI | 1. Contains a duration timeout field to allow the user to set the application timeout duration.  2. Contains a dropdown menu to select the desired password length (8, 16 (default), 24, 32, and 48) or select a customized password length through a text field.  3. Contains a change master password expandable box where the three text fields are displays (Current master password, new master password, and new master password re-enter) the user submits password change request through the “Save Password”.  4. Contains a button to allow the user to print all accounts details in the form of a text file. | 100 |
| 27 | Change timeout duration | Retrieves the user’s input from the account’s settings page and modifies Account Settings object to set the new timeout duration | 100 |
| 28 | Change default password length | Retrieves the user’s input from the account’s settings page and modifies Account Settings object to set the new password length | 100 |
| 29 | Master password change | 1. Verifies that the current password, new password, and password fields are populated.  2. Verifies that the user’s new password and verification password are identical.  3. Compares and verifies the current password field input to the authenticated user’s password  4. Modifies Master account object to update the new master password  5. Re-encodes all web account information using the new master password and saves to database | 100 |
| 30 | Profile printing service | 1. Retrieves the user’s default printer’s name and settings.  2. Retrieves all user’s accounts and store then in an arrayList  3. Creates a text file and writes each account’s details on a separate line.  4. Issues a print job request to the printer  5. Deletes the created file after printing is completed. | 50 |
| 31 | Database Support |  |  |
| 32 | Database encrypted connection | 1. Creates a database connection using a master username and a password, then it sets the connection’s properties to use SSL encryption for the connection parameters.  2. It issues a connection request to Oracle’s JDBC driver which in turn establishes a database connection. | 100 |
| 33 | UserInfoSingleton store/update/retrieval | 1. Requests encrypted connection to the database.  2. Accepts parameters from the UserInfoSingleton object.  3. Creates, returns, and updates UserInfoSingleton object.  4. Uses prepared statements to execute SQL statements against the targeted tables.  5. Uses dynamic variables binding on SQL statements.  6. Closes database connections. | 100 |
| 34 | Web account store/update/retrieval/deletion | 1. Requests encrypted connection to the database.  2. Accepts parameters from the Web Account object.  3. Creates, returns, updates, and deletes Web Accounts.  4. Uses prepared statements to execute SQL statements against the targeted tables.  5. Uses dynamic variables binding on SQL statements.  6. Closes database connections. | 100 |
| 35 | Account settings store/update/retrieval/ | 1. Requests encrypted connection to the database.  2. Identifies the user’s corresponding user id from the master account object  3. Updates account’s settings using SQL statements.  4. Uses prepared statements to execute SQL statements against the targeted tables.  5. Uses dynamic variables binding on SQL statements.  6. Closes database connections. | 100 |
| 36 | Relational DB model | Mellon schema uses relational model where tables are linked using a set of primary and foreign keys and will contain the following tables:  1. ACCOUNT\_MASTER: Contains the user’s master’s accounts and each account is identified by a unique user id.  2. ACCOUNT\_INFO: Contains master account’s current and previous password along with maintenance timestamps.  3. ACCOUNT\_HISTORY: Contains master account’s login attempts.  4. ACCOUNT\_SETTINGS: Holds master account’s settings such as password length and session duration.  5. WEB\_ACCOUNTS: Contain created web accounts associated with a master account. | 100 |
| 37 | Database tables triggers | 1. MAINT\_DTM field update will be triggered once an update statement is issued against database row.  2. CREATE\_DTM field will be triggered once a new row is added to the database table.  3. Master account updates triggers row insert in the ACCOUNT\_HISTORY table.  4. Master account creation triggers row insert in the ACCOUNT\_SETTINGS table. | 100 |
| 38 | Automatic sequence generation | 1. Master account user id will be generated using USER\_ID\_SEQ sequence once a row is added to the MASTER\_ACCOUNT table.  2. Web account web id will be generated using WEB\_ID\_SEQ sequence once a row is added to the WEB\_ACCOUNTS table. | 100 |
| 39 | Database backup/reboot scheduling | 1. Database will be performing automated backups once a week on Monday at 2:30 AM.  2. Backups will not impact the application’s availability in any way.  3. Database will be rebooted automatically once a week on Monday at 4:00 AM.  4. A flashback will be taken prior to database reboot to provide application’s availability during the reboot process. | 100 |
| 40 | Additional Functionalities |  |  |
| 41 | User credentials hashing | The username and password will be hashed using SHA-256 algorithm one way hashing prior sending to the database | 100 |
| 42 | User profile encryption | The user’s account name, username, and password will be encrypted using the master account’s password as the secret key before sending to the database | 100 |
| 43 | User profile decryption | The user’s account name, username, and password will be decrypted using the secret key when retrieving from the database | 100 |

Project Frontend Design

|  |  |  |
| --- | --- | --- |
| Page | Description | Wireframe |
| Login | Allows the user to enter the username and password or access the signup page |  |
| Signup | Allows the user to register a new account by completing the new account form |  |
| Main menu | Displays the user’s web accounts and allows the user to view passwords and access account edit. |  |
| Account Edit | Allows the user to edit or deletetheir account info and save the changes to the database |  |
| Account add | Allows the user to add a web account and link it to the master account/ Generate a customized password. |  |
| Settings page | Allows the user to modify their account settings. |  |
| Help menu | Contains help information such as: user manual and bug report. |  |

# Project Schedule

* 3/26/2017
  + **~~Bachir:~~** ~~Database designed and implemented~~
  + **~~Hodges:~~** ~~Connection strings and parameters finalized~~
  + **~~Harrell:~~** ~~Login UI created~~
  + **~~Hodges & Bachir:~~** ~~Login logic created~~
* 4/2/2017
  + **~~Harrell:~~** ~~Complete UI (essential elements, no design)~~
  + **~~Harrell:~~** ~~Create user guide~~
  + **~~Hodges & Bachir:~~** ~~Implement features: successful login, authentication, account creation, and password creation~~
* 4/9/2017
  + **~~Harrell:~~** ~~Front-end design -- Logo, develop theme and begin application to existing bare-bones UI, introduce animations and other graphical elements~~
  + **~~Hodges & Bachir:~~** ~~Implement features: password retrieval, customization of password options~~

Below is where we are currently, slightly ahead of schedule. Highlighted are our schedule changes, as we are already working on improving the UI and implementing the password edit, user settings, and session timeout features.

* 4/16/2017
  + **All**: Implement fixes and features based on peer feedback
  + **Harrell:** Skin the UI utilizing CSS
  + **Hodges & Bachir:** Implement features: editing of existing passwords, user settings, session timeout
* 4/22/2017
  + **All:** Implement fixes and features based on peer feedback
  + **All:** Implement additional optional features for usability as time allows: print-only report of stored passwords for physical retention, local storage of hashed passwords to bypass server, lockout after 3 password failures, option to direct-copy password to clipboard instead of displaying on screen
* 4/23/2017 - End of Class
  + **All:** Implement fixes and features based on peer feedback
  + **All**: Finishing touches on UI design and functionality, final documentation
  + **Harrell**: Update user guide for final documentation to account for all features, including optional features, add Help Page content based on completed UI and functions.

# 

# Contributions

Jay Bachir designed, created, and continuously maintains the project’s database. As changes are needed which affect the database, he writes the scripts and necessary stored procedures and triggers associated with the change. Jay also helps with the codebase, implementing the DBConnect class which is our database utility class, and the original WebAccount class which is our domain object for holding user profile information. He also wrote the Project Design and helped edit the Project Plan and Test Plan.

Brent Harrell handles all items related to the UI including insuring proper event handlers are used to submit requests to the back end. He created all of the UI-related classes and provides adjustments to fit back-end functionality and also came up with the custom logo. He will write the User Guide and helps with editing the Project Plan, Test Plan, and Design.

Thomas Hodges creates the back-end functionality of the application. He created the hashing and encoding/decoding algorithms so that no plain text information is sent to the database. He also created the UserInfoSingleton class to hold a user’s session data and Password class which generates a random password based on options set by the user. He also wrote the Project Plan and helped edit the Test Plan and Design.

1. <https://www.computer.org/cms/CYBSI/docs/Top-10-Flaws-PR.pdf> [↑](#footnote-ref-0)
2. <http://www.engr.sjsu.edu/fayad/current.courses/cmpe232-fall04/docs/CmpE232-L05-3n-CBSE-Chellenges.pdf> [↑](#footnote-ref-1)