Project Plan

Group 1: Ammar Bachir, Brent Harrell, Thomas Hodges

Mellon

CMSC 495 - 7980

03/25/2017

[**1. Overview**](#_tba1bwm1wtnn) **3**

[1.1 Project Summary](#_80u8brkouczz) 3

[1.1.1 Purpose, scope and objectives](#_cmr514dqats7) 3

[1.1.2 Assumptions and constraints](#_jgifk2qqrpy) 3

[1.1.3 Project deliverables](#_y057fqkyp3uo) 4

[**2. Schedule**](#_lhlm3wkzbf53) **4**

[**3. Project Requirements**](#_u93raueqnt65) **5**

[3.1 Required general functionality](#_hgfdyjzg3mpc) 5

[3.2 Required security features](#_s7asaiyox0ek) 5

[3.3 Optional functional requirements](#_jvx3rgojbgwk) 6

[3.4 Out of scope requirements](#_a2i9h5spi1hs) 6

[3.6 Classes](#_8vsba4nhaz7f) 6

[**4. References**](#_mn74ttce93h) **8**

[**5. Definitions**](#_9epmy0ab1g5d) **8**

[**6. Project organization**](#_o5rpnps0y7r3) **8**

[6.1 External interfaces](#_z0i03utjrv7o) 8

[6.2 Internal structure](#_kjh85zgwcdy3) 8

[6.3 Roles and responsibilities](#_rdul4takmip) 8

[**7. Managerial process plans**](#_dywdijoowzdm) **9**

[7.1 Work plan](#_v369gwfzp3ow) 9

[**Appendix A: Class structure**](#_vadyxd3ngfsc) **10**

# 

# 

# 1. Overview

Mellon is a password management application which is intended to be run as a client-side application. Internally, it will connect to a remote database in order to download the user’s information after the user successfully authenticate with their master credentials. the client-side utility will provide options for retrieving passwords, editing current passwords/accounts, and creating new accounts by utilizing either a user-provided password or generating a custom, strong password given a list of character options.

## 1.1 Project Summary

### 1.1.1 Purpose, scope and objectives

The purpose of this project is to create a secure means of storing a person’s credentials (usernames and passwords) for multiple external accounts. The scope of this project includes only the input, storage, and retrieval of these credentials. This project will not attempt to prevent “hacking” of an individual’s account using outside means, or enforce security policies. It is possible for secure passwords to be generated, but the option to override and input a custom password is always an option. Objectives of this project include, but are not limited to:

* Create a desktop application as a front-end for the program.
* Establish connectivity between the desktop application and a networked database.
* Store only hashed credentials on the database (compromising the database does not give an attacker access to raw data).
* Decrypt information only on the client application (only transmit hashed information over the network).
* Provide the ability to generate secure passwords.
* Remind users to change passwords after they are a set number of days old.

Technologies used in this project include:

* Java SE Runtime Environment (version 1.8.0 and higher)
* Oracle 12C Business
* Active internet connection

The project will be written utilizing Java (JDK 8), JavaFX, and CSS. Code will be hosted in a Github repository which all members of the group have access to view and change. The current URL for the project repository is: <https://github.com/brenther/Mellon>.

### 1.1.2 Assumptions and constraints

Assumptions of this project are that the user has a willingness to increase their security awareness through good password policies. These policies include using different passwords for each account, enforcing complex password requirements, and changing passwords often. Additional assumptions are that the client machine (PC) has the Java Runtime Environment (JRE) version 8 or higher installed.

Constraints of this project include the limited time for development. A completed project must be submitted in less than two months. As a result, not all intended features may make it into the final product, but continued work is expected after the class ends to fully realize this functionality.

### 1.1.3 Project deliverables

The project will deliver the following:

1. Software package consisting of:
   1. Secure password storage in an internet accessible database for retrieval across multiple systems
   2. Password creation tools to enhance a user’s security through strong and randomized passwords
   3. Password retrieval through the use of a master password known only to the user
2. User guide

# 2. Schedule

The primary deliverables for the project will be based on the class requirements, as given below:

* 4/9/2017 – Phase I Source
* 4/16/2017 – Phase II Source
* 4/23/2017 – Phase III Source

In addition to the class deliverables, an internal schedule of weekly milestones has been set as follows to maximize time in the classroom and maintain a substantial headstart on the class requirements. The team member responsible for each item is given for each requirement:

* 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 utilizing CSS
  + **Hodges & Bachir:** Implement features: password retrieval, customization of password options
* 4/16/2017
  + **All**: Implement fixes and features based on peer feedback
  + **Harrell:** Continue to improve UI design through subtle animations, positioning, and skinning through 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

Some flexibility may exist in the schedule to allow for supporting back end development on complex subjects.

# 3. Project Requirements

## 3.1 Required general functionality

* Allow new master account creation
* Allow the user to log in with existing master account
* Utilize user-provided nicknames for profile for retrieval
* Provide capability to create and store both username and password in user generated profiles
* Provide password generation with:
  + Specified length
  + Optional uppercase characters
  + Optional lowercase characters
  + Optional special characters
    - The ability to specify allowed special characters
  + Optional numbers
* Allow for user-defined password and generated password editing
* View all user profiles within the account

## 3.2 Required security features

* Allow a user to create a master account which contains the key for all associated profiles
* Securely store credential information with hashed and encoded values
* Securely communicate over SSL with the database
* Retrieve encoded credential information from the database and decode in the client application
* Display the plain text version of the web accounts associated with the master account

## 3.3 Optional functional requirements

* Password expiration reminder
* Frequently used account priority
* Export web account information to file

## 3.4 Out of scope requirements

* Syncing changed passwords with their associated web site
* Retrieving or storing the master account password
* Customer service (excluding bug reports)

3.5 User Interface

* Log-in Page
  + Sub menu: account creation
* Main menu
  + Contains profile retrieval menu, references to other menus
* User password creation menu
  + Advanced character selection menu
  + Password creation menu also used for editing of existing profiles
* User settings menu
* Help page with FAQ, general operation hints

## 3.6 Classes

The following classes encompass the program functionalities as described above. Methods and classes are subject to change as optional functionalities are added.

**General operation classes:**

* Password
  + setPassword() - saves the generated password as a string
  + checkPassword(ArrayList<Character>) - ensures that the password conforms to the given options
  + fixPassword(ArrayList<Character>, ArrayList<Character>) - replaces extra characters with missing characters from the UI options
  + getPasswordString() - returns the generated password as a string
* PasswordBuilder
  + includeSpecialCharacters(boolean) - indicates whether the password should be created with special characters
  + includeNumbers(boolean) - indicates whether the password should be created with numbers
  + includeCapitals(boolean) - indicates whether the password should be created with capitals
  + includeLowers(boolean) - indicates whether the password should be created with lowers
* DBConnect
  + getAccountFound() - returns the accountFound boolean variable
  + getConnect() - creates the connection to the networked database
  + getCredentials(String, String) - returns the user’s associated web accounts
  + checkUser(String, String) - checks if the user’s master account exists
  + registerUser(String, String) - adds a new master account to the database
* MasterAccount
  + hashString(String) - creates a hash of a given input
  + getUserAccounts() - returns the associated web accounts
  + getAuthenticated() - returns if the user has successfully authenticated
  + getUsernameHash() - returns the hash of the master username
  + getPasswordHash() - returns the hash of the master password
* WebAccount
  + getID() - returns the user’s ID from the database
  + setID() - sets the user’s ID
  + decode(String, String) - decodes an encrypted string
  + encode(String, String) - encodes a plain string

**UI classes:**

* MellonFramework
  + getScene() - returns the Scene object
* MainMenu
  + addItems() - adds the control items and user account information
* LoginPage
  + addItems() - adds the control items
* SignUpPage
  + addItems() - adds the control items
* CreationPage
  + addItems() - adds the control items
  + setAllowable(ArrayList<Character>) - stores the selected characters for password creation
  + getSelections() - returns selected characters for password creation and menu population
* AdvancedMenu
  + addItems() - adds the control items
  + getAllowable() - retrieves the selected characters for use in password creation from the advanced selection menu
* SettingsMenu
  + addItems() - adds the control items
* AccountInfoPane
  + addItems() - adds the control items
* HelpPage
  + addItems() - adds the control items

# 4. References

IEEE Standard for Software Project Management Plans. (1998). doi:10.1109/ieeestd.1988.121942

# 5. Definitions

GUI: Graphical User Interface

Master account: This the the account which credentials are provided in order to access the program

Profile: contains detailed information on the different user-defined accounts (for example, Gmail will contain both the email address and password). A master account contains a profile for each associated user-defined account.

Repository: An archive of all changes and modifications to the source code of a project

UI: User Interface

# 6. Project organization

## 6.1 External interfaces

External entities to this project include peer reviewers, which are students in the CMSC 495- 7980 course and the professor.

## 6.2 Internal structure

Internal entities to this project include the members of the group, listed below.

* Ammar Bachir
* Brent Harrell
* Thomas Hodges

## 6.3 Roles and responsibilities

From the group membership above, specific roles and responsibilities have been assigned to each member, they are:

* Ammar Bachir
  + Database and back end
    - Ensures database schema and associated triggers and constraints are created and functional
* Brent Harrell
  + Front end (UI)
    - Create each GUI element which the user of the application will interact with. Responsible for design elements.
* Thomas Hodges
  + Back end
    - Creates the logic between UI elements and database connections.

All members of the group will be responsible for testing all aspects of the project.

# 7. Managerial process plans

## 7.1 Work plan

All members of the team are responsible for completing their assigned task for the week. Updates should be posted to the Google Group periodically, and any commits to the repository should be accompanied by comments describing the changes.

7.1.1 Work activities

The work activities shall follow the above defined responsibility areas. Some overlap may occur, for instance adding event listeners to the UI as the back end is completed.

7.1.2 Schedule allocation

The schedule was set and agreed upon. Adjustments, if necessary, shall be discussed by all members of the team.

7.1.3 Resource allocation

The scope of work is such that there is flexibility for members of the team to assist others when a particularly challenging task is encountered.

# Appendix A: Class structure

