**Project Report for**

**Team Pseudo’s**

**Password Manager**

Dennis Tran, Kyle Parker, Brandon Pearson, Addison Burkett

Capstone Project Fall 2022

The University of West Florida

27 November 2022

CIS4595

Dr. Owsnicki-Klewe

**Executive Summary**

Password Manager is a software developed by team Pseudo, which consists of Dennis Tran, Kyler Parker, Addison Burkett, and Brandon Pearson. Privacy and security has been a forefront in today’s Internet usage. This requires many users to create and memorize various passwords and usernames for all applications that they may use. Team Pseudo’s Password manager aims at the goal of eliminating the burden of having to administer a substantial amount of credentials. With Password Manager, the user will only need to memorize one password, which would give access to their vault of credentials. With a system of credential security, Password Manager directs a wider focus on building a platform where users can safely share credentials at their will. Privacy is our top priority, thus ensuring that user data is always kept safe and secure whilst keeping the convenience of the user’s credentials at the tip of their fingers.

**Table of Contents**

| Executive Summary...................................................................................................................... | 2 |
| --- | --- |
| Table of Contents.......................................................................................................................... | 3 |
| 1 Comparison of Final and Initial Requirements......................................................................... | 4 |
| 1.1 Create Account............................................................................................................ | 4 |
| 1.2 Create Group............................................................................................................... | 4 |
| 1.3 Remove Group Member............................................................................................. | 5 |
| 1.4 Edit Group Member.................................................................................................... | 6 |
| 1.5 Create Role.................................................................................................................. | 6 |
| 1.6 Request Credentials.................................................................................................... | 7 |
| 2 Comparison of Final and Initial Timeline................................................................................. | 8 |
| 3 Project Results Compared with Expectations........................................................................... | 10 |
| 4 Software Evaluation.................................................................................................................. | 10 |
| 5 Work to be Done........................................................................................................................ | 12 |

**1 Comparison of Final and Initial Requirements**

The requirements for the Password Manager software is outlined on team Pseudo’s Trello Board. Requirements are indicated by the formation of both Use Cases and User Stories. Throughout the development of the software, these requirements have not changed. Furthermore, all software requirements have been satisfied and tested for functionality. The major components and requirements of the software system that has been developed are:

**1.1 Create Account**

Summary: Users will be able to create an account for Password Manager allowing them to set a username and password for access.

Rationale: users will need to be able to create accounts both for security and so that the username can be used for finding group members

Users: all users who want to use our application will need to make an account

Preconditions: a user has the application active and is not logged in

Basic Course of Events:

1. The user selects the “create an account” button
2. The user inputs their email address, username, and password
3. The application communicates with the database of profiles to confirm that the input email and username are not currently in use
4. The application calls for a new profile entry to be added to the database
5. A message confirming that the account was created successfully
6. The user is brought to the main screen of the application

Postconditions: The user now has a valid account and can proceed to create groups, be added to groups, or customize their profile.

**1.2 Create Group**

Summary - Users will be able to form a group by sending an invite to other users. The creator of the group will have owner privileges and will be able to grant admin roles to other group members. The owner/admins will be able to provide the credentials they wish to share in addition to creating more roles within the group.

Rationale - When sharing credentials, the user must be an owner or admin in a group to ensure that credentials are only shared to others that they specify.

Users - Any user that wishes to share credentials. Typically a manager of some kind.

Preconditions - The software has been started up and the user is logged in.

Basic course of events:

1. The user presses on the “create a group” button
2. The system will prompt the user to enter a group name, etc.
3. The user enters in the group information and presses the “add member” button
4. The system asks the group owner to enter in usernames to add
5. The user enters in a username
6. The system adds the user to the group

Postconditions - The group owner will be able to assign the new member a role. The role determines what credentials the member has access to.

**1.3 Remove Group member**

Summary - Users with admin privileges will be able to remove group members.

Rationale - If a group member were ever to lose their permission to access the credentials within their organization it will be important that the group admins be able to remove them and change the credentials in case they may have recorded them outside of Password Manager

Users - Group Admins

Preconditions - The user has navigated to the Edit Group dialog and is an owner/admin of the group

Basic Course of Events:

1. The User clicks the group they wish to edit
2. The system displays members of the group
3. The user clicks the remove member button
4. The system responds with an updated view of the current members of the group
5. The user confirms and submits
6. The application calls for the group credential database to scramble the groups credentials into a random string of letters, numbers, and symbols.
7. The application displays a message informing the user that the group member was successfully removed and reminding them to change their credentials as soon as possible

Postconditions - The affected user’s profile will be updated to display changes made by the owner/admin

**1.4 Edit Group Member’s Role**

Summary - Users will be able to edit the admin privileges, credentials, and group members of an existing group.

Rationale - If a group member’s role within their team changed and their level of clearance needed to be updated the application would need to be able to accommodate those changes conveniently.

Users - Group Admins

Preconditions - The user has navigated to the Edit Group dialog and is an owner/admin of the group

Basic Course of Events:

1. The User clicks the group they wish to edit
2. The system displays members of the group
3. The user clicks the edit member button
4. The system responds with a dialog of role options for that member
5. The user makes the changes, confirms, and submits
6. The application calls for the group database to update the group member’s info accordingly
7. The application calls for the group credential database to scramble the groups credentials into a random string of letters, numbers, and symbols.
8. The application displays a message informing the user that the group member was successfully updated and reminding them to change their credentials as soon as possible

Postconditions - The affected user’s profile will be updated to display changes made by the owner/admin

**1.5 Create Role**

Summary - Users with admin privileges will be able to create and assign roles with varying levels of clearance.

Rationale - the role system allows users to assign clusters of privilege based on their role within their wider team or organization.

Users - admins, team leaders, and group owners

Preconditions - an admin wants to make changes to the role structure of a group

Basic Course of Events:

1. The User navigates to the edit roles page
2. The user selects the Create Role button
3. The user customizes the role, confirms, and submits
4. The application makes a call to the group database and request the changes be made in the database
5. The application displays a message informing the user that the role was successfully created and prompts them to assign the new role
6. The user selects the “assign role now” button and is navigated to the group members page
7. Alternate paths - the user selects the “do not assign” button is navigated back to the role page

Postconditions - The new role is now created and can be assigned, edited, or deleted at a later point

**1.6 Request Credentials**

Summary - Users will be able to request access to credentials that their role has access to, displaying the credentials and logging the request in the audit log.

Rationale - This is the primary function of our entire application.

Users - users who do not have admin privileges

Preconditions - the user wants or needs their credentials

Basic Course of Events:

1. The user navigates to the group page for the credential they need
2. The user selects the “Request Credential” Button
3. The application makes a call to the group database to ensure the user has the correct clearance
4. The application makes a call to the group database to call down the credential
5. The application displays the credential to the user so they can gain access to the protected resource

Alternate Paths - in step three the user’s access may be denied

Postconditions - the user has access to resources and is free to go about their business

**2 Comparison of Final and Initial Timeline**

Some changes have been made to the initial timeline. These changes have occurred due to time constraints and time lapses with other group requirements that had to be completed. For example, some sprints needed to be reduced by one week as required documentation took longer than expected to complete. Furthermore, on week 15 the requirements of completing the final presentation and final report/repository was added. The following two tables show our timelines. The first table is the initial time and the second table is the final timeline with color coded red sections to show what adjustments were needed.

**Initial Timeline**

| **Week** | **Days** | **Development Milestone** |
| --- | --- | --- |
| 1 | 08/22 - 08/28 | Formation of group / Determining group name |
| 2 | 08/29 - 09/4 | Determine project idea / Complete Project proposal |
| 3 | 09/05 - 09/11 | Complete project plan and presentation |
| 4 | 09/12 - 09/18 | Presentation 1 / Sprint 1 planning / Sprint 1 start |
| 5 | 09/19 - 09/25 | Sprint 1 cont. |
| 6 | 09/26 - 10/2 | Group Meeting 1 / SCRUM review 1 at end of week |
| 7 | 10/03 - 10/09 | Sprint 2 planning / Sprint 2 start / Group Report, Individual Report, Individual Presentation |
| 8 | 10/10 - 10/16 | Sprint 2 cont. |
| 9 | 10/17 - 10/23 | Group Meeting 2 / SCRUM review 2 at end of week |
| 10 | 10/24 - 10/30 | Presentation 2 / Sprint 3 planning / Sprint 3 start |
| 11 | 10/31 - 11/06 | Sprint 3 cont. |
| 12 | 11/07 - 11/13 | Group Meeting 3 / SCRUM review 3 at end of week |
| 13 | 11/14 - 11/20 | Sprint 4 planning / Sprint 4 start |
| 14 | 11/21 - 11/27 | Sprint 4 cont. / Software deployment at end of week |
| 15 | 11/28 - 12/04 | Final presentation |
| 16 | 12/05 - 12/11 | Finals Week |

**Final Timeline**

| **Week** | **Days** | **Development Milestone** |
| --- | --- | --- |
| 1 | 08/22 - 08/28 | Formation of group / Determining group name |
| 2 | 08/29 - 09/4 | Determine project idea / Complete Project proposal |
| 3 | 09/05 - 09/11 | Complete project plan and presentation |
| **4** | 09/12 - 09/18 | Presentation 1 / Sprint 1 planning |
| **5** | 09/19 - 09/25 | Sprint 1 start |
| 6 | 09/26 - 10/2 | Group Meeting 1 / SCRUM review 1 at end of week |
| 7 | 10/03 - 10/09 | Sprint 2 planning / Sprint 2 start / Group Report, Individual Report, Individual Presentation |
| 8 | 10/10 - 10/16 | Sprint 2 cont. |
| 9 | 10/17 - 10/23 | Group Meeting 2 / SCRUM review 2 at end of week |
| 10 | 10/24 - 10/30 | Presentation 2 / Sprint 3 planning / Sprint 3 start |
| 11 | 10/31 - 11/06 | Sprint 3 cont. |
| 12 | 11/07 - 11/13 | Group Meeting 3 / SCRUM review 3 at end of week |
| 13 | 11/14 - 11/20 | Sprint 4 planning / Sprint 4 start |
| **14** | 11/21 - 11/27 | Sprint 4 complete / Prepare final report and manual |
| 15 | 11/28 - 12/04 | Final presentation |
| 16 | 12/05 - 12/11 | Finals Week |

**3 Project Results Compared with Expectations**

**3.1 Which of the initial use cases are functional?**

The use cases that are relevant to our project have been listed in section 1 of this document. The initial use cases include the ability to create an account and login, store credentials, access credentials, create teams, create roles within teams, have an audit log, and encrypt credentials on the database. Throughout the four sprints the development team have been able to implement all use cases that have been listed. Furthermore, tests have been developed to ensure that each use case is functional. Thus, it can be concluded that from the initial use cases, all have been implemented and are functional in the software system.

**3.2 Which use cases have been removed/added/modified?**

The use cases that have been showcased in section 1 are the initial use cases that were developed for our software. Since their creation, no use cases have been removed, added, nor modified.

**4 Software Evaluation**

**4.1 Functionality**

The team utilized several testing capabilities to ensure that the software was functioning to user expectations. Tests were mainly implemented with the utilization of JUnit which is a framework for Java. This has allowed the team to follow a test-driven development process. These JUnit tests are used to check the methods in the JavaFX controller class and interface classes for functionality. By localizing our testing through the JUnit framework, we can quickly identify errors and pinpoint exactly where they occurred. Testing our system also had many difficulties. For instance, our software system has multiple classes that have to interact with other classes. Therefore, testing one method in a class requires the testing of other classes/methods that are utilized. In order to combat this issue, the team performed mocking for isolation by hard coding test objects and creating dummy objects to test on. To top it off, end-to-end testing was utilized to simulate an actual user going through the software and seeing if the software breaks at any point.

As for the testing timeline, it depends on what was implemented during that sprint. For example, for test driven development, if during the sprint planning phase, the team decided to implement User Story 3 (logging into account), then tests would be made for that.

There is currently only one functionality issue that is still open. The software is unable to grab website favicons to display with the credential. The website favicon is basically the website logo when the user enters in the website URL for their saved credentials. There was difficulty in getting the JavaFX libraries to be able to download web icons.

**4.2 Security**

Several security methods and tools were utilized to keep our software secure. Encryption of user information was implemented through the JCA (Java Cryptography Architecture). JCA provides us with an API for symmetric and asymmetric encryption, key generation and management. Rather than needing to implement security algorithms, our application can simply request services from the Java platform. Therefore, our software has a separate class that encrypts and decrypts database queries. Furthermore, a keystore is used for key management.

As for observations and open issues, the current issue would be within our database class. Our software does SQL injections through user input. Although user input is checked for validity and safety purposes before queries are made, SQL injection within code is usually still unsafe.

**5 Work to be done**

A majority, if not all of our planned functionalities are completed. Though, the Graphical User Interface could still have adjustments. Since the development team were the only people who tested the GUI, further user evaluation could help us make adjustments to the GUI that would make it better for other users. Additionally, the GUI does not look too great, therefore improvements could definitely be made. Though, due to time constraints, the GUI had to be quickly done. Other work to be done would be to fix the functionality of being able to download website favicons to display next to the user credentials.