SOFTWARE DESIGN DOCUMENT

for

Shared Password Manager System

Release 1.0

Version 1.0 approved

Prepared by The Better Team

Contents

List of Figures

List of Tables

Revision History

Date	Description	Revised by
n/a	Initial draft	n/a

1 INTRODUCTION

1.1 Purpose

The Software Design Document (SDD) for the Shared Password Manager application offers a comprehensive overview of the architecture and system design to securely manage user stored credentials.

1.2 Scope

The Shared Password manager system design will make use of a website application environment to provide functionality for three types of users to interact with the system: admin, regular, and view-only. To use the application, a user will first login to their account. The Application will then display options dependent on their privilege level. For admin users, they will have the ability to add/remove users, while regular users will have the functionality to view, add, and edit entered credentials. Additionally, View-only users will only have privilege to view credentials. The User Data will be stored securely in an SQLite database using a server.

1.3 Overview

This documents contains sections that describes the necessary components and sections that describe sub systems for the Shared password Manager system. This document should be used to understand the underlying components to communicate the intention of each systems with stakeholders. This document also serves as a troubleshooting guide to developers tasked with modifying and maintaining availability of the software.

1.4 References

[1] WT documentation, "A hands-on introduction to Wt::Dbo by Matthias, " https://www.webtoolkit.eu, 2024. (accessed Oct. 21, 2024)
https://www.webtoolkit.eu/wt/doc/tutorial/dbo.html

1.5 Definitions and Acronyms

1.6 Acronyms

Acronym	Meaning
enum	enumerated type
MVC	Model-View-Controller Architecture

2 SYSTEM OVERVIEW

The Shared Password manager system is an application with the purpose of storing user's password credentials. A user can log into their account and access the logged credentials so the user is not burdened with remembering password credentials. The system requires an admin user to control and manage user groups.

3 SYSTEM ARCHITECTURE

3.1 Architectural Design

The Shared Password manager system will use a Model-View Controller (MVC) Architecture style as presented in figure 4.1. This Model is responsible for handling the logic upon the data and interacts with the database. The View is responsible displaying. Lastly, the Controller is responsible for only handling request flows from the view and model logical units.

3.2 Decomposition Description

The Model-View Architecture chosen has been broken down into several components as presented in figure **??**. The User class component represents the model functionality of the system. The Web_interface component contains functionality for the presentation of the User Interface. Lastly, the controller is represented with the Request encryptor class which is responsible for handling the user's request between the model and view components.

3.3 Design Rationale

The Model View Architecture is a commonly used to create web based applications. No other architecture styles were considered due to chosen style has already proven viable for web applications. A design trade off with the chosen architecture, is the whitty based controller module will handle the user request to the view and model.

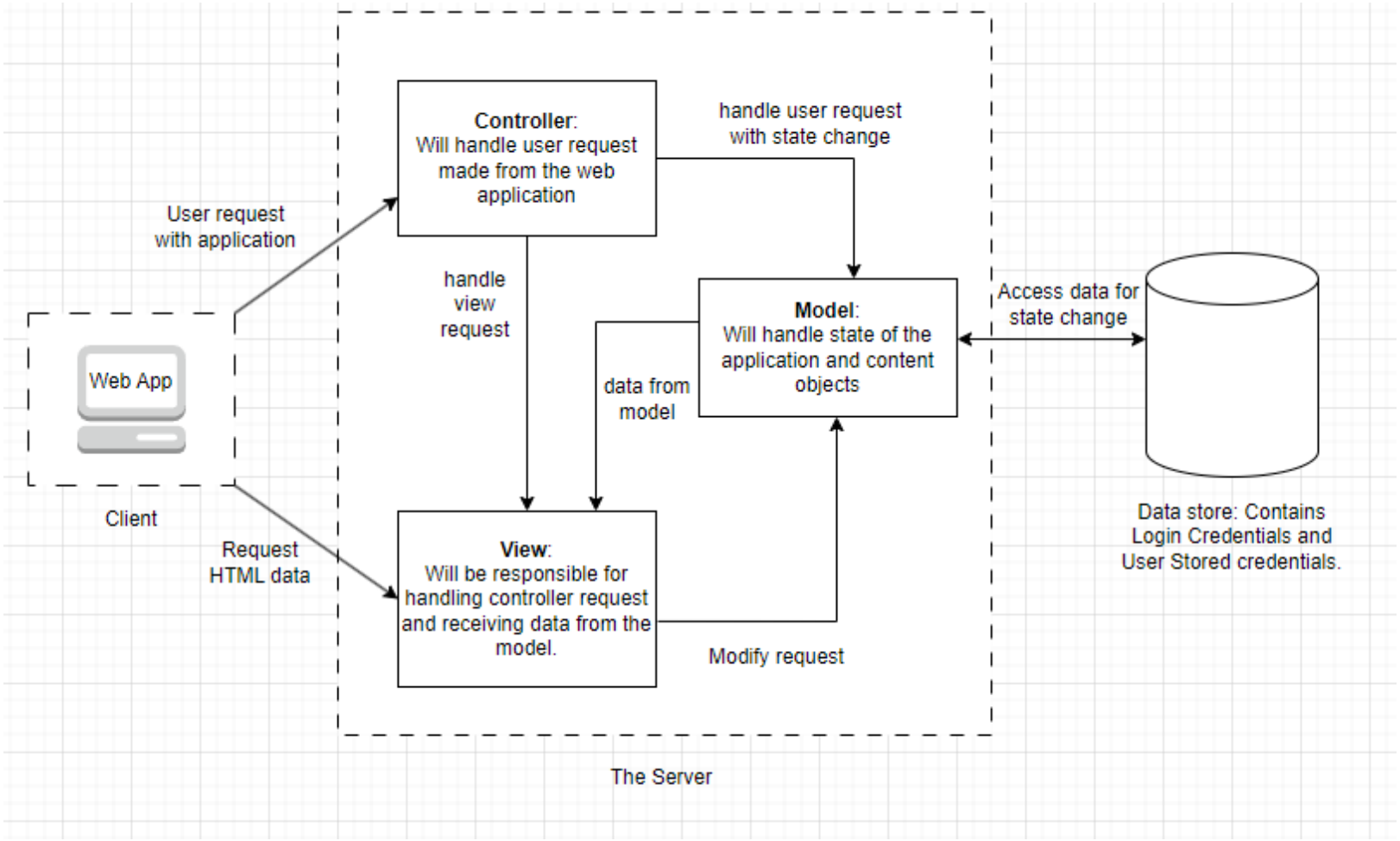

Figure 3.1: MVC Architecture diagram

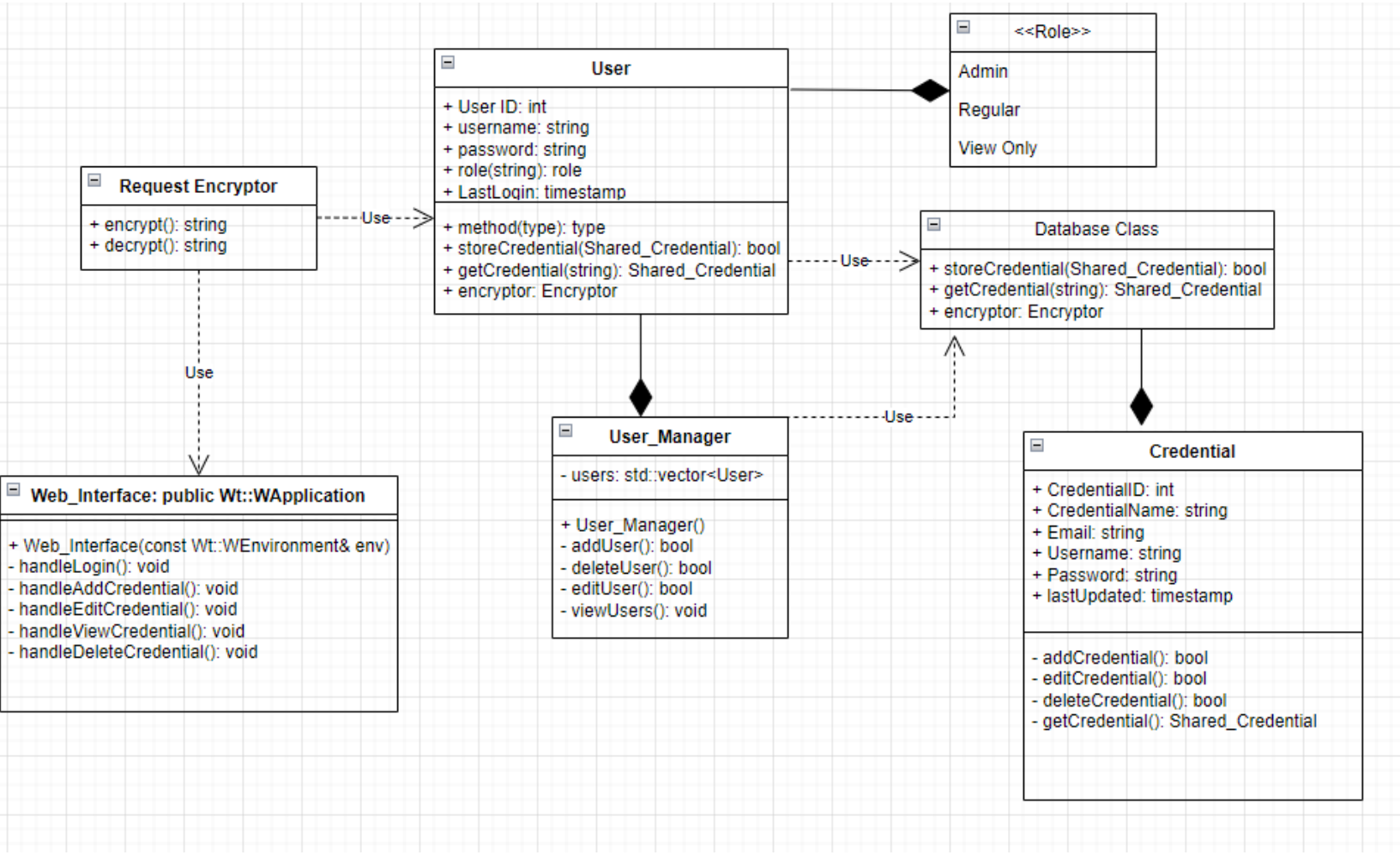

Figure 3.2: Class diagram

4 DATA DESIGN

4.1 Data Description

The data design for the Shared password manager is centered about User Data and User Credential data storage. The Entity-Relation(E/R) diagram depicts how the data is stored and logically separated between user types.

4.2 Data Dictionary

lets list out each class with parameters.

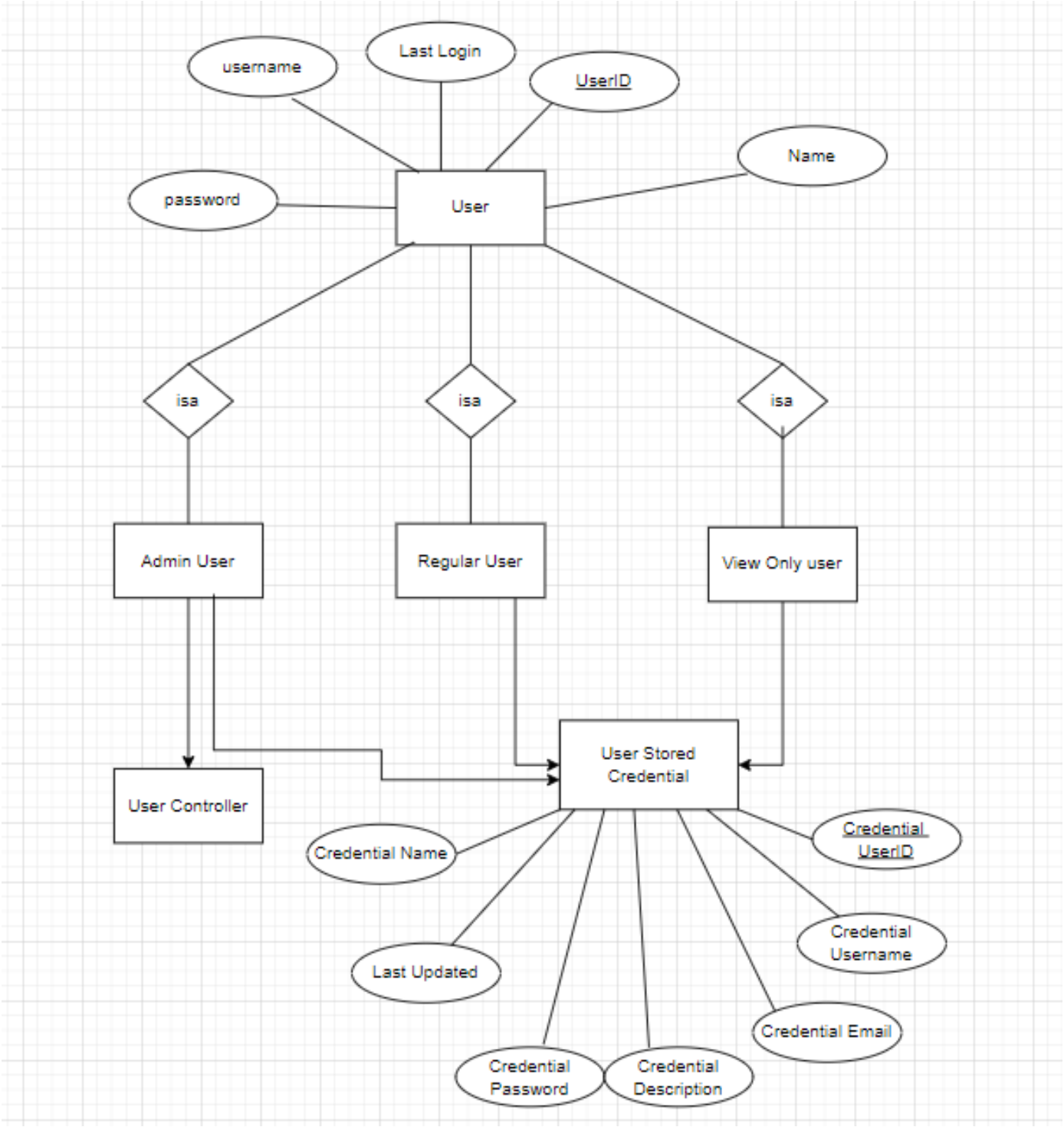

Figure 4.1: E/R diagram

5 COMPONENT DESIGN

The UML classes discussed in this section are depicted in figure **??**.

5.1 Model

The model is comprised of five classes: User, User Manager, Database, Credential, and Role.

5.1.1 User

The User class has methods for adding, editing, deleting, and viewing credentials. Adding a credential will require the instantiation of a new Credential object that will be stored in the database via the Database class's storeCredential() method. Editing will require retrieving a credential using the Database class's getCredential() method, changing the desired fields, and then storage. Deleting a credential will use the Database class's deleteCredential() method. Finally, viewing credentials will request all current credential entries. In each of these methods, the Role of the user will be checked before proceeding since each type of user has specific access as described in the first chapter.

5.1.2 User Manager

The User Manager is a helper class that will be used by Admins. Methods include adding, editing, deleting, and viewing users. Adding a user will instantiate a User class and store it in the database. Editing a user will retrieve a User from the database, change the desired fields, and store the User. Deleting a user will remove a specified User from the database. Viewing users will request all current Users from the database.

5.1.3 Database

The Database class handles interaction between the model and the SQLite database. Methods include getting, storing, viewing, and deleting a credential or user.

5.1.4 Credential

The Credential class will have getters and setters for all of the fields. All setters will have input validation.

5.1.5 Role

The Role is an enumerated class to specify the type of user: Admin, Regular, or View-Only. This is used as a field in the User class.

5.2 View

The view is handled by the Web Interface class. The constructor will set up the Web-toolkit application to appear as shown in the following chapter.

5.3 Controller

finish me

6 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

A user of the system should expect the system to behave as the use case diagram presents behavior in figure 6.1.

6.2 Screen Designs

An Illustration of the proposed system was constructed using Axure Software. The illustration of the login screen for all classes of users is displayed in Figure 6.2. The illustration of an individual credential screen for the view-only user class is displayed in Figure 6.8. The illustration of an individual credential screen for both the admin and regular user classes is displayed in Figure 6.7. An illustration of the stored credentials list for a view-only user is displayed in Figure 6.4. An illustration of the stored credentials list for a regular user is displayed in Figure 6.3. An illustration of the stored credentials list for a admin user is displayed in Figure 6.5. The illustration of the user management screen to allow an admin user to create, edit, and delete users is displayed in Figure 6.6.

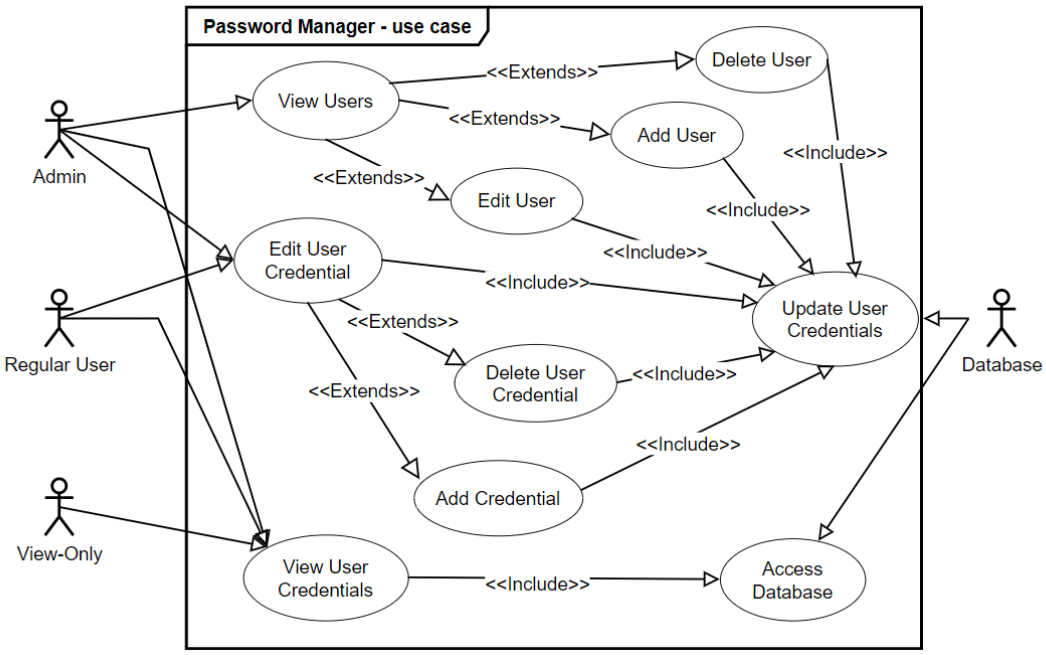

Figure 6.1: User perspective use case diagram.

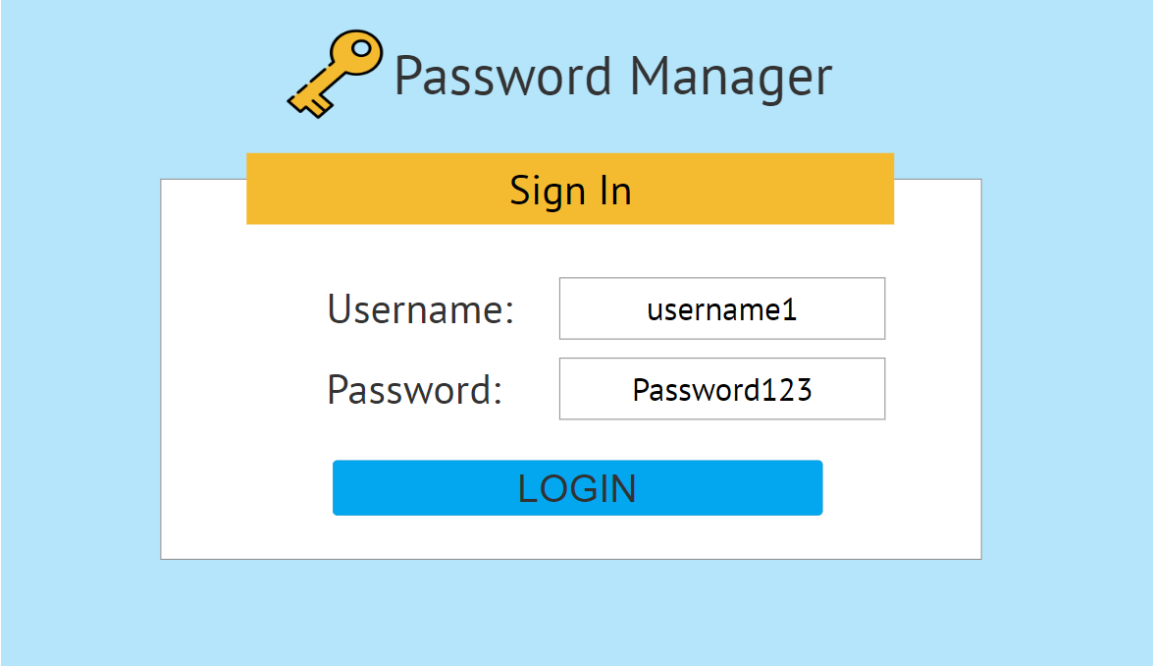

Figure 6.2: Login Illustration for All Users

Figure 6.3: Credentials List Illustration for Regular User Class

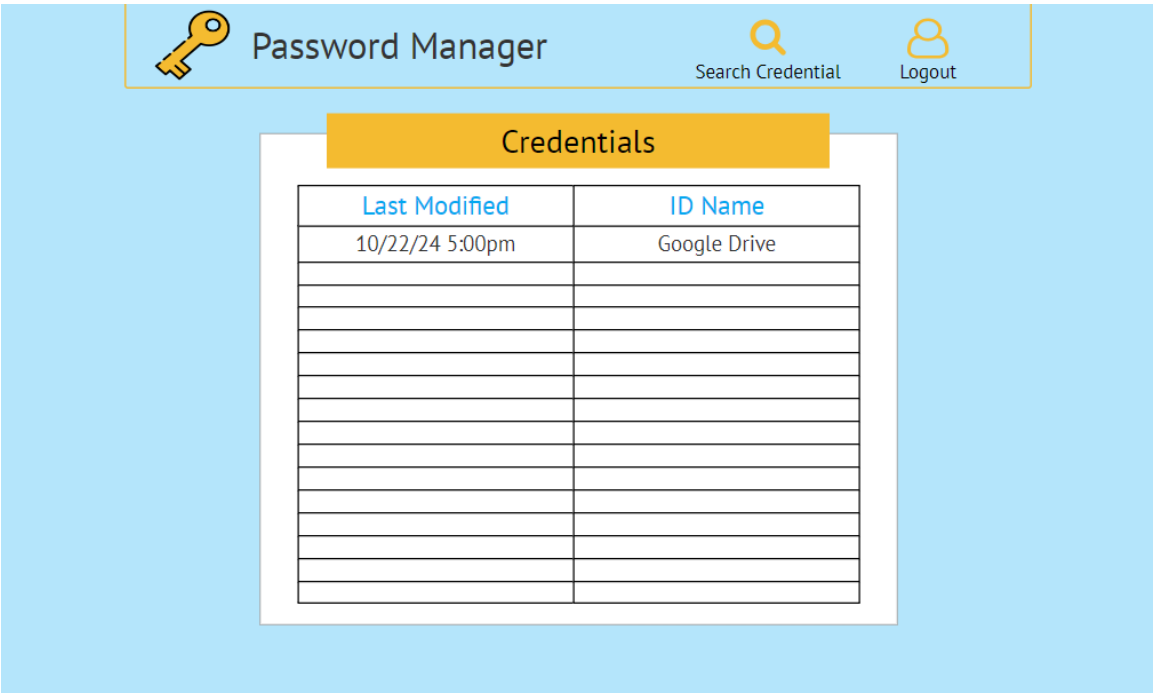

Figure 6.4: Credentials List Illustration for View User Class

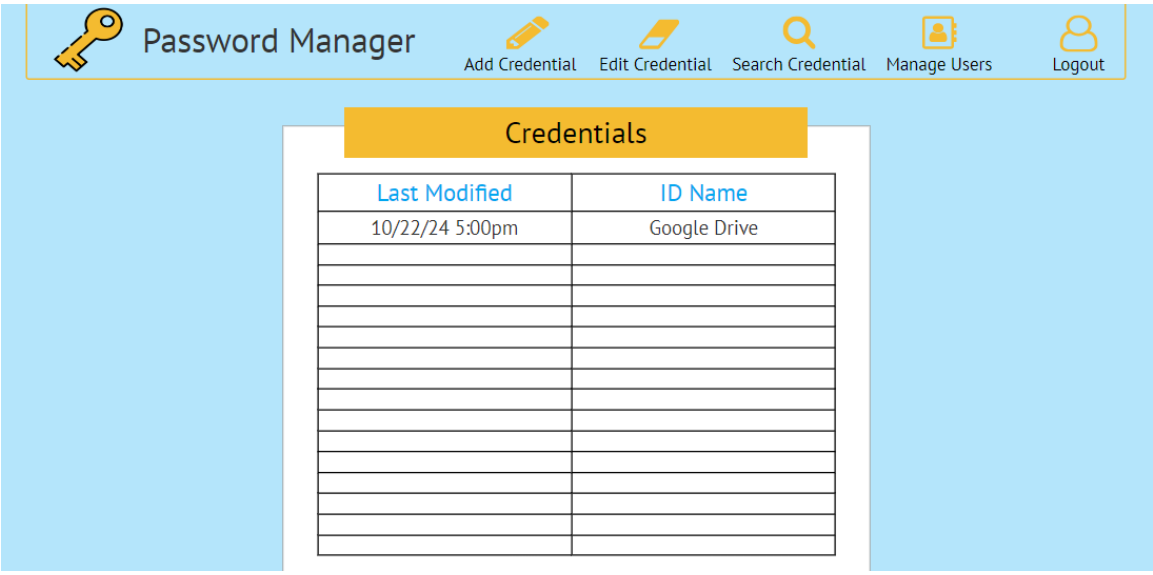

Figure 6.5: Credentials List Illustration for Admin User Class

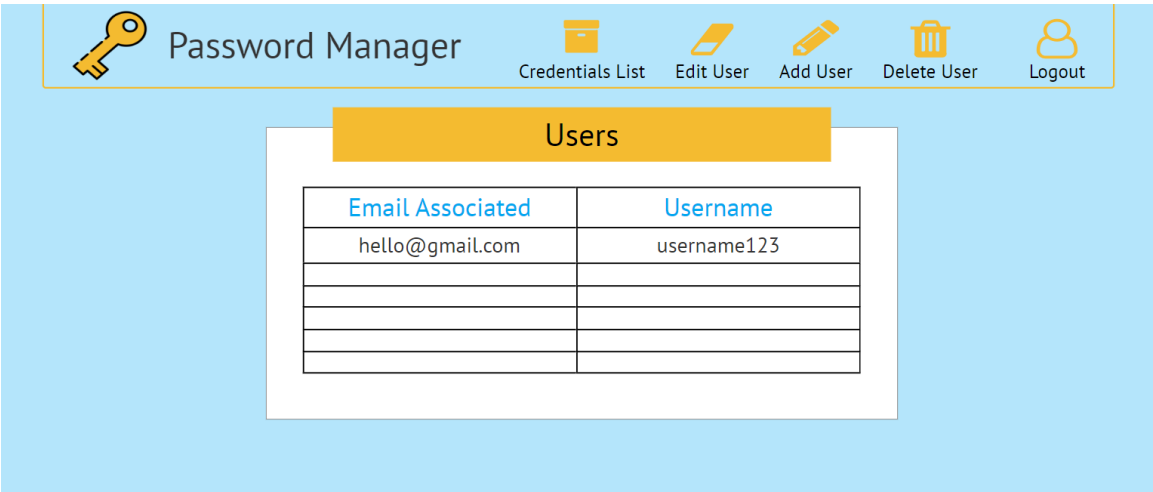

Figure 6.6: User Management Page Illustration for Admin User Class

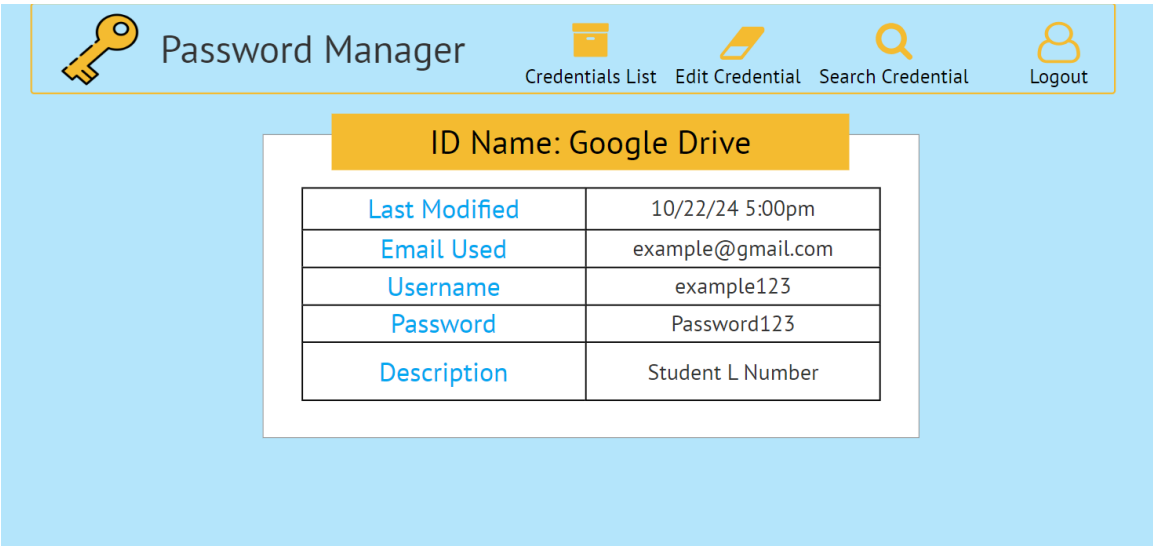

Figure 6.7: Individual Credential Illustration for Regular and Admin User Classes

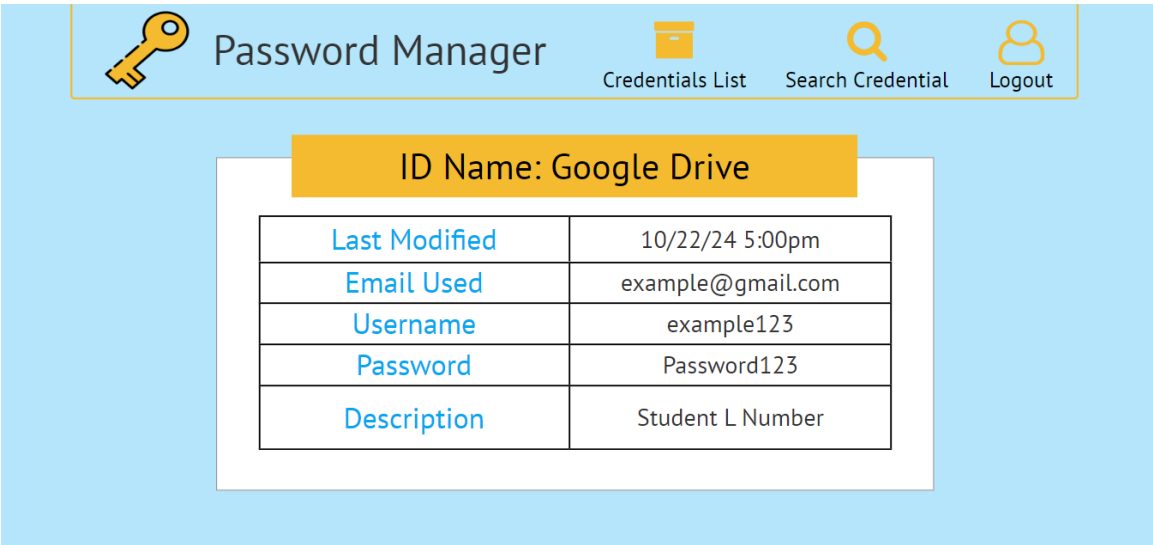

Figure 6.8: Individual Credential Illustration for View User Class

7 REQUIREMENTS MATRIX

Requirement-ID	**Requirement Description**	**Design Component**	**Test-Case #**
Performance Requirements			
R1.1	The software shall support 10 users while maintaining a maximum response time of 2 seconds.	System Architecture	T1
R1.2	The software shall exhibit response times of between 0-2 seconds after user input.	User Interface	T2
R1.3	Unplanned extended downtime shall not exceed 1 minute each week.	Server Infrastructure	T3
R1.4	The software shall support a minimum of two users altering credentials at the same time.	Authentication Module	T4

Table 7.1: Requirements Traceability Verification Matrix - Performance Requirements

Requirement-ID	Requirement Description	Design Component	Test-Case #
Security Requirements			
R3.1	Admin passwords shall not be obtainable by any user but that specific admin.	Encryption Module	T5
R3.2	Data log access shall only be available to Admin users.	Access Control	T6
R3.3	All data transfers containing user data shall be encrypted.	Encryption Module	T7
R3.4	Sessions shall expire after an inactivity of 30 minutes.	Session Manager	T8
R3.5	The software shall enforce password authentication requirements following the NIST Digital Identity Guidelines (SP 800-63B).	Compliance Module	T9
R3.6	Accounts shall lock after three repeated failed login attempts.	Authentication Module	T10
R3.7	For any database login, a minimum of AES encryption shall be used before authentication.	Database Security	T11
R3.8	The software shall prevent data loss of user credentials by completing daily backups of all files to a secondary location.	Backup System	T12
R3.9	All interactions with the database shall access the database with the least privilege possible for the least amount of time.	Database Access Control	T13

Table 7.2: Requirements Traceability Verification Matrix - Security Requirements

Requirement-ID	Requirement Description	Design Component	Test-Case #
Software Quality Attributes			
R4.1	The software shall be divided into a minimum of 2 components that can be modified individually.	Modular Design	T14
R4.2	Components shall allow the ability to update without affecting other parts of the system.	Modular Design	T15
R4.3	Software shall be accompanied by documentation.	Documentation	T16
R4.4	The software shall protect against unauthorized access and data breaches.	Security Framework	T17

Table 7.3: Requirements Traceability Verification Matrix - Software Quality Attributes