**CS411 Project:**

**ChillPass – A Password Manager**

Chia-Shen Lin – U52960602

Batyr Issabekov – U69145032

**Problem Definition:**

In today’s world an average person uses multiple services that require an account registration. As more tools and services are created, the number of accounts a person has also increases accordingly. Therefore, it might be challenging for users to keep track of their usernames and passwords and remembering all of them. To tackle this problem, we have decided to create a simple and reliable password manager that is easy to use.

**Project Objective:**

The objective of our project is to create a tool that can help people keep track and manage of all their accounts in one place, by only remembering a single master login and password. The application must allow the addition and storage of new account logins anytime to accommodate the ever-increasing number of web services in today’s world. For the purpose of preventing online theft, the logins stored must also be editable and removable.

**Stakeholder 1:** Chia-Shen Lin (Student programmer)

Developer that will build and directly influence the software. The key interest is to gain experience in collaborative building of an executable software that can be potentially used by the public.

**Stakeholder 2:** Batyr Issabekov (Student programmer)

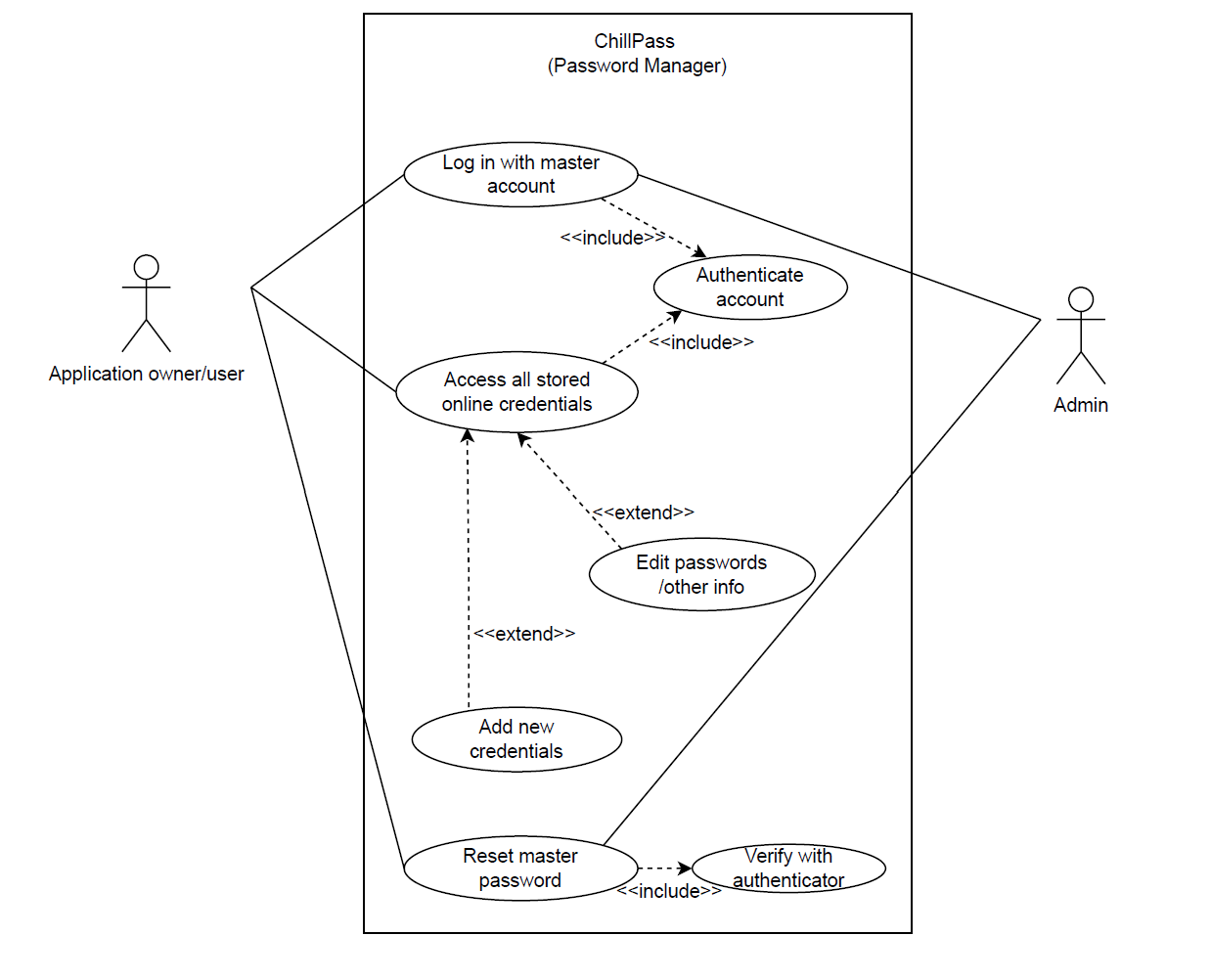
Developer that will build and directly influence the software. The key interest is to gain experience in collaborative building of an executable software that can be potentially used by the public.

**Stakeholder 3:** Professor Ahmed Ibrahim (Supervisor/sponsor/client)

This stakeholder’s goal for sponsoring this project is to provide the student programmers an opportunity to practice engineering a software using the software development lifecycle. His biggest impact would be familiarizing the student programmers on how to analyze requirements, design application architecture, implement source code, and validate application functions. Potential user.

|  |  |
| --- | --- |
| Stakeholders | Success/Acceptance Criteria |
| Chia-Shen Lin | An executable password manager software that meets all essential requirements stated in the planning phase.  All deliverables such as diagrams and PPT completed on time. |
| Batyr Issabekov | An executable software that achieves the main goals set by the developers team and is open to public use. |
| Professor Ahmed Ibrahim | Working password manager program that satisfies all stated objectives and requirements.  Documented source code.  A project report that depicts software planning and implementation process in depth.  Less than 5 minute video presentation of software in the form of PPT.  All deliverables submitted on time before 5pm April 29th. |

**Use case diagram:**



Use Case Descriptions

1.

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Add new account credentials | |
| Scenario: | Insert and save the user’s newly created account credentials in the app database. | |
| Triggering Event: | The user successfully logs into the password manager using the correct master account credentials. | |
| Brief Description: | After the password manager app verifies the user’s identity via the master login, the app displays all the securely stored online or web credentials. Upon viewing these credentials, the user then has the option to edit stored account passwords, add new credentials, or reset the master login. | |
| Actors: | Application user(accounts owner) | |
| Related Use Cases: | Extends: Access stored account credentials. | |
| Stakeholders: | CS411 Project Team 1: Engineer a working password manager program by following the software development lifecycle.  Professor Ahmed Ibrahim: Sponsors and supervises the entire password manager development project. | |
| Preconditions: | Users must exist and have at least one or more personal accounts.  Users must have a master login for the password manager application.  The admin must have power to help verify identity and reset the master account in case the user ever forgets the login info. | |
| Postconditions: | New credentials must be added and stored securely into the password manager database.  The account credentials must only be accessed by the user with the unique master login.  The master login must be updated correspondingly whenever the user triggers a reset. | |
| Flow of Activities: | Actor | System |
| 1. User enters the master username and password in the app login frame.  2. User edits the password of a personal online or web account stored in the app.  3. User adds and stores credentials of a newly created personal account in the app.  4. User resets the master login. | 1.1 Authenticates master account.  1.4 Displays frame listing all the stored account credentials.  2.1 Displays frame with the updated credentials.  3.1 Displays frame with the newly saved credentials.  4.1 Verify user identity.  4.2 Updates the master login. |
| Exception Conditions: | 1.2 If the user enters the incorrect master username or password, then authentication fails and system will deny access to the stored account credentials until the correct master login is entered.  1.3 If user forgets the master login, he or she can contact the admin. The admin verifies the user’s identity and brings he or she directly to step 4. | |

2.

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Reset Master Login | |
| Scenario: | Reset the password associated with the master account that grants access to the stored credentials. | |
| Triggering Events: | Application grants user access to the stored credentials upon verifying the master login.  User forgets master account and contacts admin. | |
| Brief Description: | After the password manager app verifies the user’s identity via the master login, the app displays all the securely stored online or web credentials. Upon viewing these credentials, the user then has the option to edit stored account passwords, add new credentials, or reset the master login. | |
| Actors: | Application user(accounts owner) and admin. | |
| Related Use Cases: | Includes: Verify with authenticator. | |
| Stakeholders: | CS411 Project Team 1: Engineer a working password manager program by following the software development lifecycle.  Professor Ahmed Ibrahim: Sponsors and supervises the entire password manager development project. | |
| Preconditions: | Users must exist and have at least one or more personal accounts.  Users must have a master login for the password manager application.  The admin must have power to help verify identity and reset the master account in case the user ever forgets the login info. | |
| Postconditions: | New credentials must be added and stored securely into the password manager database.  The account credentials must only be accessed by the user with the unique master login.  The master login must be updated whenever the user triggers a reset. | |
| Flow of Activities: | Actor | System |
| 1. User enters the master username and password in the app login frame.  2. User edits the password of a personal online or web account stored in the app.  3. User adds and stores credentials of a newly created personal account in the app.  4. User resets the master login. | 1.1 Authenticates master account.  1.4 Displays frame listing all the stored account credentials.  2.1 Displays frame with the updated credentials.  3.1 Displays frame with the newly saved credentials.  4.1 Verify user identity.  4.2 Updates the master login. |
| Exception Conditions: | 1.2 If the user enters the incorrect master username or password, then authentication fails and system will deny access to the stored account credentials until the correct master login is entered.  1.3 If user forgets the master login, he or she can contact the admin. The admin verifies the user’s identity and brings he or she directly to step 4. | |

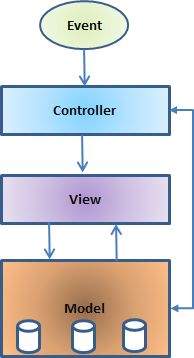
**Sequence diagram:**

Diagram

Description automatically generated

**System Architecture:**

We are using the MVC (Model View Controller) architecture, where the Model is out MySQL database that holds the user information and their credentials. The View is the table that is on the main client page, that fetches the data from out database and presents it to the user. The controller is the add credentials and delete credentials frames that are accessible to the user from the client.



**Class diagram:**

Diagram

Description automatically generated

**ER Diagram:**

Diagram

Description automatically generated

**GitHub Link to Source Code:**

[**https://github.com/Chia-Shen-Lin/ChillPass**](https://github.com/Chia-Shen-Lin/ChillPass)

**Conclusion:**

Through this project, we have learned how to use an engineering approach to create software. We have actively practiced working and develop as software engineers and communicate accordingly. In the development process, we have learned skills of utilizing MySQL by creating databases and learned aspects of JavaFX for creating an attractive software UI that is also simple to use. We further utilized Java and JavaFX to connect the front-end and back-end of our program. Through testing and research, we have managed to figure out ways to let the user manipulate their password data by connecting to our database and modify, delete, and add passwords as they wish. Most importantly, we have learned how to collaborate and work in a team as software engineers to create a successful program.

**References:**

System Architecture, MVC model diagram: [**https://www.tutorialspoint.com/struts\_2/basic\_mvc\_architecture.htm**](https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm)

Appendix

**Work Breakdown Structure (WBS):**

Diagram

Description automatically generated

**Task Assignment Matrix:**

|  |  |  |
| --- | --- | --- |
| **Task** | **Task Owner** | **Support** |
| **Create Login Page,**  **Create front end (text boxes,**  **Buttons)** | **Jason** |  |
| **Create database for user master credentials** | **Batyr** | **Jason** |
| **Connect the login page to the database** | **Batyr** | **Jason** |
| **Create Client Page** | **Jason** |  |
| **Create a table to display credentials** | **Jason** |  |
| **Create buttons for adding/deleting credentials** | **Jason** |  |
| **Connect the buttons to the database and implement their functionality** | **Batyr** | **Jason** |
| **Create add credentials page** | **Jason** |  |
| **Create front end for accepting input** | **Jason** | **Batyr** |
| **Connect the credentials page input to the database and update** | **Batyr** | **Jason** |
| **Create delete credentials page** | **Jason** | **Batyr** |
| **Create front end for accepting input** | **Jason** |  |
| **Connect the input to the database to remove credentials accordingly** | **Batyr** | **Jason** |