PassUSB: Logins Made Easier

Team Number 10: Sonali Benni, Rey Jairus Marasigan, Chidiebere Otuonye, Kaiya Roberts, Gentman Tan

Professor Kianoosh G. Boroojeni, Ph.D.

CEN4010 U02: Software Engineering I

April 29, 2022

Abstract

The principles of password security and personal convenience has been a growing dilemma between organizations and individuals. An effect of such has been the increasing in complexity that password of online accounts has been required to be, leading to forgotten passwords or the reuse of passwords across multiple accounts, the latter of which has proven to lead to many instances of unauthorized account access. Our hardware and software solution aims to solve this issue in providing a convenient way to store, retrieve and modify stored secrets (i.e. passwords, payment information) across most modern devices. Our PassUSB application will ask an end user to only remember a single master password to be used to log into the application. The application will then present the user with the list of stored entires that are encrypted within the PassUSB application. Users will then be able to retrieve their information through either copying to the system clipboard or "injecting" it into an input field on a seperate digital device via the PassUSB dongle. The PassUSB dongle wirelessly interfaces with the mobile device that the PassUSB app is installed onto after a pairing procedure, and interfaces with the ancilliary machine that the PassUSB dongle is connected to in order to send keystrokes to said machine. For the purposes of security, the application can also clear the system clipboard after a certain period of time. The design of the addition and modification of account entries' subsystem within the PassUSB app allows for the inclusion of random password generators which encourage the user to use differing passwords across between multiple accounts, with the passwords themselves being generated to be highly entropic for increased security. From this project, we hope to create a password management system that is truly multi-platform and offers a more secure and convenient way for individuals to use password-based authentication.

Contents

1 Introduction

In the prevalence of large scale information technologies, there exists an ever-increasing need for individuals to secure one's own access to online accounts. The typical method of doing so requires the user to create a secret passphrase that they would then be responsible for memorizing in order to access a given system. However, several factors make such a task difficult and unsafe; first, the exponential rise in computational power has led to the feasibility of "brute force attacks", in turn forcing IT administrators to enforce increased password length and complexity. Another effect that the increase in password length has is making the memorization of multiple different passwords difficult, thereby incentivizing individuals to unsafely reuse their own passwords. With these shortcomings in mind, we propose a system that would solve all of these issues in a single package.

1.1 Purpose of system

Solve the security issues facing PC end users in the realm of password authentication

1.2 Scope of the system

In scope: Multi-platform mobile application, password management, multi-platform USB keyboard emulation, mitigation against man-in-the-middle, replay and spoofing attacks Out of scope: Application data security, side-channel attacks

1.3 Objectives and success criteria of the project

- Provide a mobile app for users to create and store passwords
- Provide a USB hardware dongle that can be paired with the mobile app which can type in passwords in lieu of keyboard input

1.4 Definitions, acronyms, and abbreviations

- PassUSB: the project's USB dongle solution that emulates a USB keyboard
- Password manager: a computer program that generates, stores and retrieves passswords for its users
- HID (Human Interface Device): a computer device that facilitates communications between a computer user and a computer
- App: a computer application
- Pairing: the process of recognition and acknowledgement between the mobile device and USB dongle

1.5 Overview of document

The project will consist of two types of coding assignments, one for frontend development i.e. mobile app development, and the other for backend development i.e. microcontroller programming. The mobile app will prompt the user to create a new password database, in which he/she will then enter a master password that is to be used to secure the database. The user will be given an option to pair the PassUSB with the app. Should the user choose to or not choose to pair the PassUSB, the user is then able to utilize the app's password generation, management and storage features.

2 Current System

Without password managers, users are succeptable to password leaks which can lead to the exploitation of other accounts that the user owns if the same password is used. Also, the memorization of increasingly entropic passwords for many accounts is becoming an increasingly difficult challenge. Password managers presently exist to solve these issues; however, the login process can be tedious if the application or website that is to be logged into is on a device that has not been set up with a user's password manager and account database. Existing solutions rely on cloud services that require a user to either (1) log in and unlock their account database on any devices that choose to log in, or (2) have the user deal with the meticulous process of manually typing out a password, referring to their mobile device's password manager, onto the device that they wish to log into. The first existing solution has major security implications due to the user's entire database being encrypted on a system which they might not trust the fully (i.e. a public computer). The second existing solution avoids this problem; however, the user may find it cumbersome especially with passwords consisting of many non-alphanumeric characters. As a whole, current systems are limited in functionality and multi-platform capabilities, which leads to greater frustration from the end-user's perspective.

3 Project Plan

The project will span a 7 week timeline and will be made possible via the use of various hardware and software components. Due to the PassUSB system's design specification requiring the usage of various forms of hardware communicating via the interference-prone protocol Bluetooth, regular testing must be performed to ensure the minimization of software faults.

3.1 Project Organization

1. Sonali Benni: Architecture liason, Tester
2. Rey Jairus Marasigan: API Engineer, Implementor
3. Chidiebere Otuonye: Configuration Manager, Implementor
4. Kaiya Roberts: Configuration manager, Implementor
5. Gentman Tan: Project Manager, Implementor

3.2 Hardware and software requirements

1. Hardware:
   (a) Multi-core workstations for code building and continuous testing
   (b) Programmable microcontroller with HID-capable USB controller and Bluetooth transceiver (STM32 based)
   (c) Mobile devices of varying make and model
2. Software:
   (a) Flutter cross-platform mobile software development kit
   (b) FlutterBlue third-party library for Bluetooth interface
   (c) Arduino IDE
   (d) Arduino USBHID and ArdunioBLE libraries
   (e) Usage of OS-level virtualization containers for reproducable build environments

3.3 Table with Project schedule

PassUSB

Gantt Chart Template © 2006-2018 by Vertex42.com

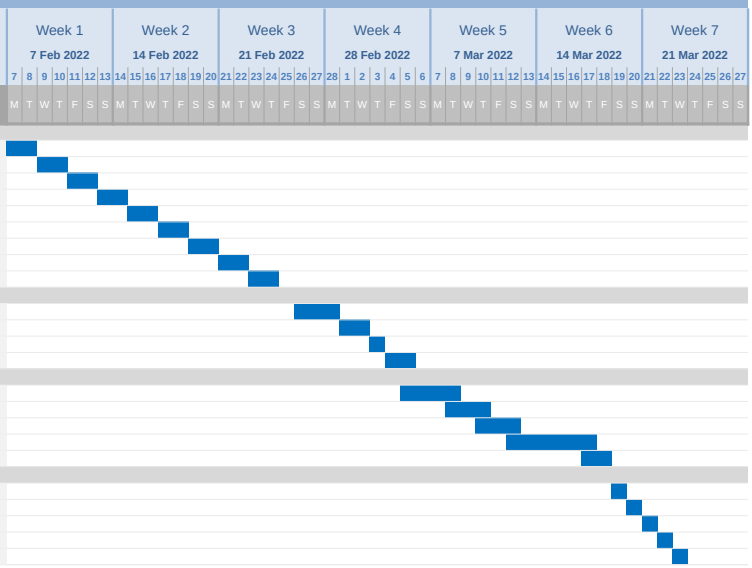
Learn about the Pro version >

Bonobo CO.

Project Start Date 2/7/2022 (Monday) | Display Week 1

Project Lead Gentman Tan

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Requirement Elicitation**						
0.1	Client meeting	Rey/GT	Mon 2/07/22	Tue 2/08/22	2	0%	2
0.1	Case Analysis	Rey/GT	Wed 2/09/22	Thu 2/10/22	2	0%	2
0.1	Case Analysis	Rey/GT	Fri 2/11/22	Sat 2/12/22	2	0%	1
0.1	User Analysis	Rey/GT	Sun 2/13/22	Mon 2/14/22	2	0%	1
0.0.1	Requirement	Rey/GT	Tue 2/15/22	Wed 2/16/22	2	0%	2
0.0.1	nonfunctional?	Rey/GT	Thu 2/17/22	Fri 2/18/22	2	0%	2
0.1	functional?	Rey/GT	Sat 2/19/22	Sun 2/20/22	2	0%	0
0.1	Prototype	Rey/GT	Mon 2/21/22	Tue 2/22/22	2	0%	2
0.1	Video Mockup Demo	Rey/GT	Wed 2/23/22	Thu 2/24/22	2	0%	2
1	**Design**						
0.1	Sketch User Interface System	Kaiya	Sat 2/26/22	Mon 2/28/22	3	0%	1
0.1	Design Use Cases	Kaiya	Tue 3/01/22	Wed 3/02/22	2	0%	2
0.1	Draw Use Case Diagrams	Kaiya	Thu 3/03/22	Thu 3/03/22	1	0%	1
0.1	Create Sequence Diagram	Kaiya	Fri 3/04/22	Sat 3/05/22	2	0%	1
1	**System Development**						
0.1	Application Login Development	Chidie	Sat 3/05/22	Tue 3/08/22	4	0%	2
0.1	Password generating backend	Chidie	Tue 3/08/22	Thu 3/10/22	3	0%	3
0.1	Information storage and encryption	Chidie	Thu 3/10/22	Sat 3/12/22	3	0%	2
0.1	UI/UX development for dongle	Chidie	Sat 3/12/22	Thu 3/17/22	6	0%	4
0.1	Backend finalization	Chidie	Thu 3/17/22	Fri 3/18/22	2	0%	2
1	**Testing**						
0.1	Build Application	Sonali	Sat 3/19/22	Sat 3/19/22	1	0%	0
0.1	Install Application on test devices	Sonali	Sun 3/20/22	Sun 3/20/22	1	0%	0
0.1	Test iOS Deivces	Sonali	Mon 3/21/22	Mon 3/21/22	1	0%	1
0.1	Test Andoird devices	Sonali	Tue 3/22/22	Tue 3/22/22	1	0%	1
0.1	Test PasUSB dongle	Sonali	Wed 3/23/22	Wed 3/23/22	1	0%	1



4 Requirements Elicitation

4.1 Use Cases

Use Case ID:	UC-1-AA
Use Case Name:	Add account
Created By: Rey Jairus Marasigan	Last Updated By: Rey Jairus Marasigan
Date Created: February 21, 2022	Last Revision Date: February 21, 2022
Actors:	• User
Description:	This use case is used for specifying the different elements of our system that the DeviceUser goes through to add an entry to a list of account entries within our app. The ideal outcome of this use case is that the account is added to the database without any adverse effects.
Trigger:	The user presses the plus icon on the top right of the UI.
Preconditions:	• The User has created a database • The User has opened the app and unlocked their database • The User is in the correct directory to be able to see and touch the plus icon.
Postconditions:	• At minimum: the system retains consistency and integrity in case of an error or deviation from the flow of events i.e. the user can "add account" again without repercussions from previous try. • Everything in harmony and works without error: Account is added to the database. • The account added can be accessed after being saved to the database.

Normal Flow:	(assumes the user wants to generate password when adding the account) 1. The DeviceUser presses the plus icon on the top right 2. The view switches to a display that shows 3 text fields: a website field, a username and a password field. The first two will be required to be filled using the pop-up keyboard. The password field can either be manually filled or automatically generated using the associated button located to its left. <generate password use case> 3. The user presses a button that opens to another view that shows multiple fields, boxes, and sliders to be filled, checked, and adjusted, respectively, to meet a certain criteria associated with the password. 4. The user enters the field with a pop-up keyboard on their phone. 5. Below the fields, the users checks various boxes or switches that configures the password generated by the apps such as: • contains special characters- an option that includes special characters in the password • contains numbers- an option that includes number • etc. 6. Below the switches, there exists a slider that controls how long the password will be. When it is adjusted towards the right, it increases the length. When it is adjusted towards the left, it decreases the length. The maximum length of the slider occupies the entire width of the screen with some deadzone. The minimum length is 8 characters and the maximum is 32 characters. Initially the slider is set to 12. 7. As the user changes the length of the desired password, a text box that shows the password itself changes. The password is shown in its pure form i.e. the actual password to be used. Here, the password is changeable by pressing on the box and using the pop-up keyboard. Initially, the password generated here is from the combination of the slider and checkboxes. 8. The user then presses the "save password" button below to generate the password. (Includes generate password use case) 9. The user finally then presses the "add account" button below to add the account to the list. 10. A notification pops up that lets the user know that the account has been added. 11. The state of the app then returns to its default view.

Alternative Flows:	2b. In step 2 of the normal flow, if the user chooses to manually enter the password rather than generate a new password, 1. The entire normal flow can skip to step 9 7b. In step 7 of the normal flow, if the user changes the generated password and then slides the password slider, 1. An entirely different password is generated. 2. The new password is instantly displayed within the text box. 3. The normal use case continues thereafter on step 7 where the user can edit the newly generated password. 9b. In step 9 of the normal flow, if the website field and username field is not filled 1. A notification pops up to let the user know about the required fields 2. The app is scrolled the field needed to be filled 3. The fields are highlighted red to indicate requirement. 4. The keyboard pops up automatically 5. Once the fields are filled, the use case enters step 8 of the normal flow.
Exceptions:	Exceptions to adding account to list 9b. In step 9 of the normal flow, if the user does not enter a website or username.
Includes:	editAccount- steps 3 to 7 would be required to edit an entry within the list and can be separated in its own use case
Frequency of Use:	On-demand
Special Requirements:	• Users should not wait >20 seconds for an action to be performed • Animations should last <250ms • Colors should be made to be accessible to colorblind • Application should be made available on all major mobile platforms
Assumptions:	• The Customer has installed and is using the application • The application is readable to the Customer • The client has the ability to use the keyboard and interact with the screen.

Notes and Issues:	• Is the minimum and maximum possible length of the password generator feasible for the users? • Should there be a limit to the amount of accounts added to the list? • What are the different switches that we can include in the forum? • should the app request for the master password (the password required to enter the app)or other biometrics for the ability to add an entry to the list? • Should we use other characters outside of the printable ascii characters?

Use Case ID:	UC-2-AB
Use Case Name:	Login to Database
Created By: Sonali Benni	Last Updated By: Sonali Benni
Date Created: February 22, 2022	Last Revision Date: February 22, 2022
Actors:	• DeviceUser
Description:	This use case is for logging into the app to access passwords. Ideally, the actor would have the app on their phone. They would log into the app and have access to their home screen/ main menu.
Trigger:	The DeviceUser clicks on the app on their personal device.
Preconditions:	• The DeviceUser is logged in on their personal device such as a cellphone where the app is located. • The DeviceUser opens the app
Postconditions:	• The DeviceUser is able to get into the app and is brought to a screen that allows them to enter login information. • The DeviceUser has successfully entered their information and is brought to a screen that shows their home screen.
Normal Flow:	1. The user turns on their personal device and is logged in. 2. The user then locates the app on their device. 3. The user clicks on the app. 4. The app then opens and displays a login screen. 5. The user then enters a username and password into the app. 6. The information is successfully processed and the user can see their home screen. 7. From this point, the user is free to update, add, or delete any accounts.
Alternative Flows:	6a. In step 6 of the normal flow, the login information is not successfully processed and the user can't see their home screen. 1. The app will prompt the user with an error, try again message 2. Once the information is correct and processed, the user is able to log in 3. Use Case resumes on step 6 7a. Once the user is successfully in the app, if the personal device were to turn off/shut down/force closed: 1. The user would be automatically signed out of the app. 2. Use Case resumes on step 4

Exceptions:	4a. In step 4 of the normal flow, the login screen is not displaying 1. If the app is down for maintenance or needs to be updated, a message will be displayed when the app is clicked on. 5a. The user forgets their password 1. The user hits forget password 2. Then a recovery key could be entered or security questions could be answered to login 3. User is prompted to create a new password
Includes:	Included in any Login use case scenarios
Frequency of Use:	On-demand: Depends on how many accounts the user needs to login into on that particular day. This use case will most likely be used multiple times a day.
Special Requirements:	The app should be organized and users should be able to quickly find what they are looking for. Color themes matter, color affects readability.
Assumptions:	• The user understands English and knows their username and password for the app.
Notes and Issues:	• Should the app have the ability to use Face ID or any other form of biometric authentication?

Use Case ID:	UC-232-AB
Use Case Name:	Pair Dongle
Created By: Gentman Tan	Last Updated By: Gentman Tan
Date Created: February 22, 2022	Last Revision Date: February 22, 2022
Actors:	• DeviceUser • Mobile Device • PassUSB
Description:	Initialize a secure connection between the DeviceUser's MobileDevice to the PassUSB for future device communication.
Trigger:	The DeviceUser initializes the pairing subroutine
Preconditions:	• The DeviceUser is logged in on their personal device such as a cellphone where the app is located • The DeviceUser has the app opened • The DeviceUser has created a database • The DeviceUser has logged into a database
Postconditions:	• The DeviceUser has successfully paired the PassUSB dongle to their database
Normal Flow:	1. The DeviceUser selects the "Pair Dongle" option in the drop down menu 2. AccountHolder is prompted to enable bluetooth discovery mode 3. The app searches for a PassUSB dongle 4. Once found, the Bluetooth dongle's name, serial number and PIN number is displayed 5. AccountHolder is prompted to either allow or decline the pairing process 6. Once the AccoutHolder presses "Allow", they are prompted to enter the PIN number displayed on the PassUSB 7. Once the PIN number is entered and confirmed, the devices are subsequently paired
Alternative Flows:	1a. In step 1 of the normal flow, a Bluetooth dongle is already paired 1. The app will prompt the user to either unpair the already paired dongle or cancel the pairing process 2. If the unpair button is pressed, the currently paired dongle is unpaired and PairDongle resumes on step 2 3. If the cancel button is pressed, the pairing dialog box is closed and the PairDongle use case exits

Exceptions:	3a. In step 3 of the normal flow, a Bluetooth dongle is not found 1. The app will prompt the user to either retry or cancel the pairing process 2. If the retry button is selected, PairDongle resumes on step 3 3. If the cancel button is pressed, the pairing dialog box is closed and the PairDongle use case exits
Includes:	N/A
Frequency of Use:	On-demand
Special Requirements:	• The app should filter out other unrelated Bluetooth devices that are discovered
Assumptions:	• The DeviceUser understands English and knows their username and password for the app. • The DeviceUser has a funcitonal and powered PassUSB dongle
Notes and Issues:	• Should the user have the ability to pair multiple different PassUSB dongles simultaneously?

Use Case ID:	UC-3-AC
Use Case Name:	Generate Password
Created By: Kaiya Roberts	Last Updated By: Kaiya Roberts
Date Created: February 22, 2022	Last Revision Date: February 22, 2022
Actors:	• DeviceUser
Description:	The reason behind this use case is to prevent the user from having to re use the same password or a variation of one combination of password. The ideal outcome of this use case will be the creation of a unique password with the requested character amount, letter amount, number amount, and special character amount.
Trigger:	The user clicks on the create new password button on their personal device
Preconditions:	• The DeviceUser is logged in on their personal device such as a cellphone where the app is located • The DeviceUser has the app opened • The DeviceUser has created a database • The DeviceUser has logged into a database
Postconditions:	• The account holder has decided to not generate a password, but include their own password instead. Their own password will be stored in the account, and the account holder will be alerted that their password was stored • The password is generated with the conditions specifed by the accountHolder, and the accountHolder will be notified that the password was successfully created and it is stored into the system.

Normal Flow:	1. The user has added the desired account that they would like to generate a password for (mentioned in the Add Account use case). 2. The user has pressed the create a password button for the desired account. 3. The user device switches to a view where they have the option to click an eye icon that will show the password as it generates. 4. On the same view mentioned above the user has the option to choose between certain parameters that they can adjust based on the parameters of the password they would like to generate. 5. The user clicks the parameter they would like to add to the password. The examples are provided below. • length of password • upperCase letters • lowerCase letters • digits • Minus • underscore • space • special characters • bracket variations 6. The user adjusts the desired length of password by moving the slider icon to their desired amount. • The slider starts at 12 characters and it can be adjusted to another number. • The maximum number of characters in the password is 32. • When the slider is moved towards the right it increases the desired number • When the slider moves towards the left it decreases the desired number 7. As the parameters are updated, the user will see the changes in the password box where the new password will be generated. 8. The user updates the password with the desired parameters and has created a final password 9. The user presses save password 10. The password is saved into the application's database
Alternative Flows:	• At any time, the user can cancel the Generate Password dialog box and exit the Generate Password use case
Exceptions:	N/A

Includes:	N/A
Frequency of Use:	On-demand
Special Requirements:	• Options chosen should reflect immediately in the password (the DeviceUser should not need to wait >500ms for a password to be generated)
Assumptions:	• The DeviceUser understands English and knows their username and password for the app • The DeviceUser is logged into a database • The DeviceUser has elected to either add a new account entry or edit an existing account entry
Notes and Issues:	• Which kinds of cryptographic algorithms should be used in the process of password generation? • Should passwords be shown in plaintext to the DeviceUser by default?

Use Case ID:	UC-5-AC
Use Case Name:	Delete Account
Created By: Chidiebere Otuonye	Last Updated By: Chidiebere Otuonye
Date Created: February 21, 2022	Last Revision Date: February 22, 2022
Actors:	• DeviceUser
Description:	This use case is for outlining the behavior of DeviceUser when they're deleting an account from the USB dongle, via the app.
Trigger:	The user presses the "edit" button.
Preconditions:	• The DeviceUser is logged in on their personal device such as a cellphone where the app is located • The DeviceUser has the app opened • The DeviceUser has created a database • The DeviceUser has logged into a database
Postconditions:	• The system retains all data from accounts that user intends to keep and integrity of control flow is preserved • The correct account is deleted and the other accounts are preserved, control flow is preserved and the user can execute other actions and navigate to other parts of app
Normal Flow:	1. The accountHolder presses the 'edit' icon. 2. Selection bubbles appear next to all accounts. 3. The accountholder can select multiple accounts or just 1. 4. The accountholder selects the trash icon in the top right-hand corner. 5. A dialogue box pops up asking if user is sure they want to delete selected accounts. 6. App authenticates identity before deleting account.

Alternative Flows:	3a. In step 3 the user decides not to delete any accounts after pressing 'edit'. 1. User selects 'cancel' in top left-hand corner where the 'edit' button previously was. 2. Selection bubbles disappear from next to account fields. 3b. In step 3 the user decides not to delete any accounts after pressing 'edit' and selecting account(s). 1. User selects 'cancel' in top left-hand corner where the 'edit' button previously was. 2. Selection bubbles disappear from next to account fields. 3c. In step 3 the user decides to delete all accounts after pressing 'edit' button. 1. User presses the 'select all' button. 2. User presses the trash icon. 3. User is prompted to confirm if they're sure they want to delete 'ALL accounts'. 4. User is prompted to authenticate using physical password. 5. User is then prompted once more if they'd like to delete 'ALL accounts as the action is irreversible' 6. User selects 'delete' and all accounts are deleted. 5a. In step 5 user decides not to delete accounts after selecting the trash icon. 1. User selects the 'Never mind' option when presented with the options to delete or not delete selected accounts.

Exceptions:	6a. In step 6 user isn't able or chooses not to authenticate their identity to complete deletion process. 1. User fails authentication and is prompted to authenticate biometrically 2 additional times. 2. User is then asked to enter physical password. 3. User enters physical password incorrectly 3 times and is locked out of account for 30 minutes. 4. User enters physical password incorrectly 1 more time and is locked out for 45 minutes. 5. User enters physical password incorrectly 1 more time and is locked out for 1 hour. 6. User enters physical password incorrectly 1 more time and is locked out permanently until they recover account. N/A
Includes:	N/A
Frequency of Use:	On-demand
Special Requirements:	• Customer can organize accounts with different methods to make deleting quicker.
Assumptions:	• The user is able to use the application without issue on their device(s). • The user understands which button to choose to select and delete accounts
Notes and Issues:	• Should account objects that are to be deleted undergo lazy deletion or completely dereferenced?

Use Case ID:	UC-6-AC
Use Case Name:	Backup Database
Created By: Gentman Tan	Last Updated By: Gentman Tan
Date Created: April 9, 2022	Last Revision Date: April 9, 2022
Actors:	• DeviceUser • FlashDrive
Description:	This use case specifies the process of which a database is backed up onto an external storage device
Trigger:	The DeviceUser presses the 'backup' button.
Preconditions:	• The DeviceUser is logged in on their personal device such as a cellphone where the app is located • The DeviceUser has the app opened • The DeviceUser has created a database • The DeviceUser has logged into a database
Postconditions:	• An encrypted container containing the database is created on a user selected external storage device

Normal Flow:	1. The DeviceUser presses the 'backup' menu entry. 2. The DeviceUser is shown a system directory chooser and is prompted to select a directory to be used to save the database backup 3. The DeviceUser chooses a directory 4. The DeviceUser is prompted to enter a password twice to be used to secure the backup file and enters a password 5. The DeviceUser receives confirmation that the account database is backed up onto the specified directory with the filename passusb-<date>.bak, where <date> would be in the format MMDDYYYY, for example 01012001 for January 1, 2001.
Alternative Flows:	3a. In step 3, a file of the same name as the backup file already exists 1. The DeviceUser is prompted in a dialog box that a file of the same name as the new backup 2. In the same dialog box DeviceUser is presented with the button options 'overwrite file', 'rename backup file' or 'cancel' 3. If the DeviceUser chooses the 'overwrite file' button, the file existing on the external storage device is overwritten. If the DeviceUser chooses the 'rename backup file' option, a string "(¡num¿)" is appended to the new file, where <num> starts from 1 and is incremented if a new filename already exists. If the DeviceUser chooses the 'cancel' option, the Backup Database subroutine exits. 4a. In step 4, the DeviceUser does not enter a password 1. The DeviceUser is prompted that storing an unencrypted database can be unsafe, and that it is recommended to enter a password. The user can choose the 'Back' button to return to the beginning of step 4, or the 'Continue Anyways' button to continue saving the unencrypted backup of the account database

Exceptions:	3b. In step 3, the directory is unwriteable 1. The DeviceUser is prompted that the directory that has been chosen is invalid and is presented the corresponding error (for example, directory is read only) 2. The DeviceUser is presented with button options to either choose a new directory or cancel the backup procedure 3b. In step 3, the directory does not have enough free space to store the database backup 1. The DeviceUser is prompted with an error dialog notifying that the chosen directory does not have enough free space, and is presented with a 'choose another directory' button to restart step 3 or a 'cancel' button to exit the backup
Includes:	N/A
Frequency of Use:	On-demand
Special Requirements:	• The backup file should incorporate checksumming in order to ensure data consistency
Assumptions:	• The DeviceUser is able to use the application without issue on their device(s). • The DeviceUser understands which button to choose to backup the database • The DeviceUser uses a storage medium with a filesystem that is supported by the mobile operating system.
Notes and Issues:	N/A

Use Case ID:	UC-6-AC
Use Case Name:	Create Account Entry Group
Created By: Gentman Tan	Last Updated By: Gentman Tan
Date Created: April 9, 2022	Last Revision Date: April 9, 2022
Actors:	• DeviceUser
Description:	This use case specifies the process of which an account entry is grouped
Trigger:	The DeviceUser presses the 'add group' button.
Preconditions:	• The DeviceUser is logged in on their personal device such as a cellphone where the app is located • The DeviceUser has the app opened • The DeviceUser has created a database • The DeviceUser has logged into a database
Postconditions:	• An account entry group is created

Normal Flow:	1. The DeviceUser presses the 'add group' button 2. A new list entry is presented in the account list view section of the app with a folder icon and text entry field for the user to create a new name for the account group 3. The DeviceUser types in an appropriate name for the account group 4. The DeviceUser presses return on their keyboard and the account group is created
Alternative Flows:	3a. In step 3, the user does not input any characters for a name for the folder 1. The DeviceUser presses retur no ntheir keyboard and an account group with the name "New Folder" is created. If a folder exists with this name, a number will be appended to the name i.e. "New Folder (¡number¿)"
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	On-demand
Special Requirements:	• The backup file should be resilient to data corruption
Assumptions:	• The DeviceUser is able to use the application without issue on their device(s). • The DeviceUser understands which button to choose to create an account group
Notes and Issues:	N/A

Use Case ID:	UC-4.6.22.2
Use Case Name:	Check Database for Account Data Compromise
Created By: Rey Jairus Marasigan	Last Updated By: Rey Jairus Marasigan
Date Created: April 6, 2022	Last Revision Date: April 6, 2022
Actors:	• DeviceUser • HIBP (haveibeenpwned.com) • MobileDevice
Description:	This use case is used when the DeviceUser wants to check if their passwords are safe to be used periodically using the email associated with their accounts. The ideal outcome for the use case is that the DeviceUser is notified of one of two events, a breach or not a breach.
Trigger:	The event that initiates this use case could either be on demand of the DeviceUser through a specific button on the app or be from timely checks every so often passively. This use case will be focusing on the latter.

Preconditions:	• The device has an internet connection. • There is an established connection between HIBP with an API key set. (from agree to breach checks use case) • There is at least one entry(account) in the database with a recovery email associated. • The DeviceUser has checked the option to receive checks in the settings.
Postconditions:	• The DeviceUser is notified of either events of being compromised or being safe. • In the event of some accounts being compromised, the breach accounts are highlighted until they are modified within the app. • At minimum, the data in the database is not compromised to the third party password verifier.
Normal Flow:	1. After pressing the switch to agree for periodic checks of their accounts, the DeviceUser closes the app and turns off the screen of their phone (not entirely where their phone is powered off). 2. Depending on the amount of time they specified for the checks, the app sends the DeviceUser a notification on the status of their accounts every so often. 3. During a check, the app passively sends a query to HIBP for all the accounts retrieved from the database through database queries. 4. In the case of accounts being compromised, a notification appears on the MobileDevice that shows a red X symbol alongside with a message indicating which accounts may have been compromised. 5. Upon pressing the notification, the app opens up to the log-in page that asks for the master password or other biometrics to get access into the list/database. 6. After logging in, the device opens the app to the main/usual UI that shows the entire list of accounts with certain entries highlighted in red indicating a possible breach. 7. Pressing on one of the highlighted accounts brings up the usual editAccount use case. However, it will display a banner on the top to indicate urgency to change the password of the email associated with the account. 8. The DeviceUser selects all highlighted accounts and modifies them. The state of the app then reverts back to its usual state.

Alternative Flows:	4a. In step 4 of the normal flow, if the accounts are not breached, 1. The notification shows a green check mark alongside a reassuring message. 2. Pressing on it opens the app to a pop-up screen that shows a more specific description based on the aforementioned message. 3. The screen has a button on the top right to close the pop-up message and reverts to the log-in screen to access the database. 8a. In step 8 of the normal flow, if the customer does not change the highlighted accounts, 1. The state of the app will remain with the accounts still highlighted. 2. The DeviceUser closes the app. 3. The DeviceUser logins into the app later on, the state of the list will be the same as in step 6 after logging in.
Exceptions:	3a. In step 3 of the normal flow, if the app is unable to reach HIBP, 1. The app will go through the connect to HIBP use case. 2. If successful the use case will continue back on step 3
Includes:	Edit Account: step 7 of the normal flow for the user to edit the account that needs to be modified. Logging into app: step 5 in the normal flow for the user to login into the app. Query HIBP: step 3 in the normal flow requires the specification of how the app interacts with HIBP. Agree to Breach Checks: a use case used for the preconditions. Connect to HIBP: a use case included within agree to breach checks use case. Lock App - step 8 in the normal flow where the DeviceUser is done modifying the account and exits.
Frequency of Use:	Depends on the frequency placed set by use case Agree to breach checks, can be from daily, weekly, biweekly, monthly, bimonthly, or yearly
Special Requirements:	• Users should not wait >20 seconds for an action to be performed • Animations should last <250ms • Colors should be made to be accessible to colorblind • Application should be made available on all major mobile platforms

Assumptions:	• The Customer has installed and is using the application • The application is readable to the Customer • The client has the ability to use the keyboard and interact with the screen.
Notes and Issues:	None

Use Case ID:	UC-4.6.22.3
Use Case Name:	Copy password to MobileDevice clipboard
Created By: Rey Jairus Marasigan	Last Updated By: Rey Jairus Marasigan
Date Created: April 6, 2022	Last Revision Date: April 6, 2022
Actors:	• DeviceUser • MobileDevice
Description:	This use case is used when the DeviceUser wants to copy one of the passwords within the list to their phone's clipboard. The outcome of this use case is having the entry in the clipboard
Trigger:	The event that initiates this use case is from a simple press on the button next to the text field within the account's screen that displays the website, account name, password, and email.
Preconditions:	• The DeviceUser is logged in on their personal device such as a cellphone where the app is located • The DeviceUser has the app opened • The DeviceUser has created a database • The DeviceUser has logged into a database • The screen displayed is on the account where the user wants to copy the account/password from.
Postconditions:	• The text for pasting is cleared from the clipboard or the desired text stays on the clipboard. • At minimum, the text copied from is unaltered or not deleted.
Normal Flow:	1. While at the account screen, the DeviceUser sees a button next to each of the 4 text fields present. The button has a clipboard symbol. 2. The DeviceUser presses the password clipboard button. 3. The app then decrypts the password from its encrypted hash form. 4. After decryption, the text is placed in the MobileDevice's clipboard. 5. The MobileDevice then clears the text from the clipboard through the clear clipboard automatically use case.

Alternative Flows:	2a. In step 2 of the normal flow, if the user presses instead presses any the other clipboard buttons, 1. The text is directly copied into the clipboard without having to go through decryption. 2. The flow then continues in step 5 of the normal flow. 5a. In step 5 of the normal flow, if the customer did not go through the clear clipboard automatically use case, 1. The text stays in the clipboard instead of being deleted. 3a. In step 3 of the normal flow, if the app is unable decrypt the password, 1. An error will show on the screen in the form of a drop-down banner that will advise the DeviceUser to change the password. 2. The DeviceUser will then lead the use case to enter the editAccount use case. 3. After the editAccount use case, the flow will continue on step 2
Exceptions:	3a. In step 3 of the normal flow, if the app is unable decrypt the password, 1. An error will show on the screen in the form of a drop-down banner that will advise the DeviceUser to change the password. 2. The DeviceUser will then lead the use case to enter the editAccount use case. 3. After the editAccount use case, the flow will continue on step 2
Includes:	Edit Account - step 3a of the exceptions for the user to edit the password. Clear Clipboard Automatically- step 5 of the normal flow Decrypt Password - step 3 of the normal flow
Frequency of Use:	On-demand
Special Requirements:	• Users should not wait >20 seconds for an action to be performed • Application should be made available on all major mobile platforms • App should have an interface to MobileDevice • text copied into clipboard should not be modified

Assumptions:	• The DeviceUser is able to use the application without issue on their device(s). • The DeviceUser understands which button to choose to create an account group
Notes and Issues:	• Password could be cached within the clipboard undesirably • How long will the clipboard hold on to the copied text? • Where should the button be placed when faced with varying screen sizes?

Use Case ID:	UC-2-BC
Use Case Name:	Modifying Account Details
Created By: Sonali Benni	Last Updated By: Sonali Benni
Date Created: April 1, 2022	Last Revision Date: April 1, 2022
Actors: •	DeviceUser
Description:	This use case is for changing any information to an account that has been already added to the database. It changes the username, password, and other information within an account.
Trigger:	The actor clicks the manage accounts button within Settings.
Preconditions:	• The user is logged into the app on their personal device and goes to Settings.
Postconditions:	• Minimal Guarantee: The user is able to click on manage accounts within the app's Settings. They are then brought to a page that allows them to tap on an account they would like to modify or edit. • Success Guarantee: The user is able to select the account they would like to change and are successfully able to make changes.
Normal Flow:	1. The user logins into the app. 2. The user then locates the manage accounts button under Settings and clicks it. 3. The user is brought to a screen displaying a list of their accounts. 4. The user clicks onto the account they wish to modify. 5. The user then clicks on what they want to edit and makes the necessary changes. 6. The user then decides whether to click save changes or cancel. 7. Whichever option is chosen, the user is brought back to the same account page where they can make more changes to the account's information or click the return to home screen button.

Alternative Flows:	5a. In step 5 of the normal flow, the user doesn't hit save or cancel after inputting new information. The phone shuts off or the app closes. 1. The new inputted information is not saved. The old information is kept. 2. The user gets back into the app and returns to the manage accounts section. 3. The user clicks onto the account they want to change 4. Use Case resumes on step 5
Exceptions:	In step 7 of the normal flow, the new password was evaluated but not accepted because it's invalid due to requirements such as capitalized letters, numbers, and special characters. 1. The app keeps the old data and nothing is changed.
Includes:	The login to the app use case is included because the user must be logged in to the app to execute this use case.
Frequency of Use:	On-demand: Depends on when a user wants to update or change information. This use case may be used once a month or every couple of months.
Special Requirements:	• The app's manage accounts section should be fairly easy to locate and use. • Accounts should be organized and easily found.
Assumptions:	• The user understands English • The user is logged into the database • The user wishes to modify account information
Notes and Issues:	• Is there a limit to how often accounts can be modified within a time span? For example, how many times can you change one account within a week?

Use Case ID:	UC-2-BB
Use Case Name:	Changing Database Password
Created By: Sonali Benni	Last Updated By: Sonali Benni
Date Created: April 1, 2022	Last Revision Date: April 1, 2022
Actors:	• DeviceUser
Description:	This use case is for changing the database password.
Trigger:	The actor clicks on Settings and locates and clicks change database password.
Preconditions:	• The user is logged into the app on their personal device and clicks Settings.
Postconditions:	• Minimal Guarantee: The user is able to click on Settings within the app. They are then brought to a screen that allows them to tap on change database password. • Success Guarantee: The user is able to successfully change their database password.

Normal Flow:	1. The user logins into the app. 2. The user then locates Settings and clicks it. 3. The user is brought to a screen that has the option to change the database password. 4. The user clicks change database password. 5. The user is prompted for their original password. 6. Once the original password is accepted, the new password is able to be inputted. 7. The new password is asked to be confirmed by typing it in again. 8. Once the new passwords match, a button is able to be clicked that confirms the user wants to change the password. 9. Once the button is clicked, the new password is set.
Alternative Flows:	5a. In step 5 of the normal flow, the original password is wrong. 1. Password incorrect message is displayed 2. The user has 2 more tries to input the correct password 3. If correct within the given amount of tries, use case resumes on step 6
Exceptions:	7a. In step 7 of the normal flow, the passwords don't match. 1. An error is displayed 2. The user is prompted to try to correctly match the passwords 3. Use Case resumes on step 8 once the passwords match
Includes:	5a. In step 5 of the normal flow, the original password is wrong. 1. Password incorrect message is displayed 2. The user has 2 more tries to input the correct password 3. If wrong, the system forbids a database password update for the next 24 hours
Frequency of Use:	On-demand: Depends on how many times the user wants to change or update their database password. This use case is seldom used.
Special Requirements:	• The app's Settings should be fairly easy to locate and navigate. • Users should be able to find the change database password section easily and it should be simple to know how to change it.

Assumptions:	• The user understands English • The user is logged into the database • The user wishes to modify their database password
Notes and Issues:	• Is there a limit to how often the database password can be modified within a time span? Is it controlled by a date and time stamp?

Use Case ID:	UC-23-43		
Use Case Name:	Lock Out		
Created By:	Kaiya Roberts	**Last Updated By:**	Kaiya Roberts
Date Created:	4/7/22	**Last Revision Date:**	4/14/22
Actors:	Primary Actor: DeviceUser Secondary Actor: MobileDevice		
Description:	This use case is used when the DeviceUser has been locked out of their account due to the fact that there have been multiple inputs of an incorrect master password in order to initially log into the app.		
Trigger:	The event that initiates this use case are 5 incorrect inputs of the master password to get into the web application		
Preconditions:	1. The device has an internet connection. 2. The DeviceUser has already created an account (including a master password) for the web application. 3. The DeviceUser has a known recovery key in order to be able to reset their password or login under different circumstances.		
Postconditions:	1. The DeviceUser is notified of either events of being compromised or being safe. 2. The DeviceUser is locked out of their account and can not login until a certain period of time 3. The DeviceUser received a message on the screen telling them that they have been locked out and must log in again later		
Normal Flow:	1. The DeviceUser is prompted to input a username and password 2. The DeviceUser has entered a 5th incorrect master password attempt into the mobile app 3. The DeviceUser is switched to a screen asking them to now input the recoveryKey/securityKey for the master account at the beginning of the account creation process 4. The mobileDevice checks to see if the securityKey is correct with the one assigned to the user 5. The united securityKey is deemed as incorrect and the DeviceUser is alerted 6. The screen switches to a message saying “Lockdown incorrect password input” and is required to wait a specific amount of time (suggested 30 minutes) before the next login attempt 7. The account Owner is alerted to know there has been a breach of password, with the number of attempts, times, dates, and source 8. The account owner is asked if they would like to reset their master password		
Alternative Flows:	1a. The DeviceUser takes too long to input a password		

	1. The allotted time to input login information runs out and is indicated by the timer on the screen going to 0:00 time left 2. The time is then communicated with the mobileDevice 3. A message appears saying “you have run out of time” 4. This counts as one login attempt that was not successful and alerts the DeviceUser that they have 4 attempts left 5. The mobile app automatically exits out of the screen 1b. The device user goes into another screen during the login attempt 1. The DeviceUser attempts to login to the mobileDevice 2. The DeviceUser switches onto another application 3. While switching to the other app the screen for the web app goes black 4. The mobileDevice alerts the app that there has been a move onto another screen 5. The DeviceUser switches back onto the screen for the app 6. The DeviceUser is brought back to the login screen for the mobile app 7. The DeviceUser is prompted to input login information and is alerted that they have used a,ogin attempt and have 4 more attempts left before itis considered a password breach 8b. The DeviceUser has had a collective of 30 incorrect password inputs 1. The mobileDevice detects 30 incorrect password inputs 2. The account owner is alerted that they have had 30 incorrect password inputs 3. The account owner is alerted that they are required change their master password to continue logging into the account 4. The account owner is required to enter their recoveryKey to continue 5. The account owner inputs the correct recoveryKey as detected by the mobileDevice 6. The account owner changes their password to enter 7. The password is successfully changed 8. The account owner is asked to login to the app to verify the password
Exceptions:	2a. The DeviceUser forgot their password 1. The DeviceUser inputs the correct securityKey 2. The mobile app determines that the securityKey is correct 3. The DeviceUser is alerted on the screen that the securityKey is correct 4. The DeviceUser is brought to the home screen of the mobile app
Includes:	● Changing database password ● Check database for password breach ● Backup database ● Logging into app

Frequency of Use:	Depends on the amount of incorrect attempts to get into the account before a password reset. After 30 cumulative incorrect inputs, the password automatically has to be reset for the DeviceUser and then the process resets
Special Requirements:	• Users should not wait >20 seconds for an action to be performed • Animations should last <250ms • Colors should be made to be accessible to colorblind • Application should be made available on all major mobile & web platforms
Assumptions:	• The Customer has installed and is using the application • The application is readable to the Customer • The client has the ability to use the keyboard and interact with the screen. • In the normal flow, the user is a person trying to log into the app that is not their own account. They are attempting a breach.

Use Case ID:	UC-25-68		
Use Case Name:	searchDatabase		
Created By:	Kaiya Roberts	**Last Updated By:**	Kaiya Roberts
Date Created:	4/7/22	**Last Revision Date:**	4/14/22
Actors:	Primary Actor: DeviceUser		
Description:	This use case is used when the DeviceUser would like to search up a specific account in the database to either edit their password, add an account, or generate a new password.		
Trigger:	The event that initiates this use case are 5 incorrect inputs of the master password to get into the web application		
Preconditions:	1. The device has an internet connection. 2. The DeviceUser has already created an account (including a master password) for the web application. 3. The DeviceUser has a known recovery key in order to be able to reset their password or login under different circumstances. 4. Completion of the Logging into app use case 5. Completion of the add account use case		
Postconditions:	• The DeviceUser has entered the AccountView mode of the app where they have found the intended account		
Normal Flow:	1. The DeviceUser chooses the search bar on the application		

	2. The DeviceUser locates and clicks the search database button 3. The DeviceUser types in the account that they are looking for 4. The DeviceUser clicks on that wanted account 5. The DeviceUser is prompted to a screen with the information for the accountView
Alternative Flows:	3a. The DeviceUser does not find the account they are looking for 1. The DeviceUser clicks account not found 2. The DeviceUser clicks add information manually 3. The device user adds the information manually 4. The information is saved in the database
Exceptions:	N/A
Includes:	● Backup database ● Logging into app ● Add Account
Frequency of Use:	Depends on the frequency that the DeviceUsersearches up a website in the search bar
Special Requirements:	● Users should not wait >20 seconds for an action to be performed ● Animations should last <250ms ● Colors should be made to be accessible to colorblind ● Application should be made available on all major mobile & web platforms
Assumptions:	● The Customer has installed and is using the application ● The application is readable to the Customer ● The client has the ability to use the keyboard and interact with the screen. ● The DeviceUser has created a master account to access the app in its entirety

Use Case ID:	UC-3.0.0		
Use Case Name:	PassNuke		
Created By:	Chidiebere Otuonye	**Last Updated By:**	Chidiebere Otuonye
Date Created:	April 6, 2022	**Last Revision Date:**	April 6, 2022

Actors:	Primary Actor: accountHolder
Description:	This use case is for outlining the behavior of accountholder when the "Password Nuke" setting has been enabled and the password has been entered incorrectly 4 times, all account data will be erased from device.
Trigger:	User enters password incorrectly 4 times.
Preconditions:	1. App and USB stick are paired. 2. The setting is enabled. 3. The password has been entered incorrectly 3 times.
Postconditions:	1. *Minimal Guarantee:* The system destroys all data from accounts and integrity of control flow is preserved. 2. *Success Guarantee:* All account information is deleted and the flow is preserved.

Normal Flow:	1. The user is on the login screen and enters the password incorrectly 1 time and a message is displayed indicating that the data will be deleted with 3 more failed attempts. 2. The user enters the password incorrectly a second time and a message is displayed indicating that the data will be deleted with 2 more failed attempts. 3. The user enters the password incorrectly a third time and a message is displayed indicating that the data will be deleted with 1 more failed attempt. 4. The user enters the password incorrectly a fourth time and a message is displayed indicating that all the account data has been erased.
Alternative Flows:	1a, 2a, 3a. The user enters the correct password at any one of these steps and is allowed access to account.
Exceptions:	Exceptions to the PassNuke use case 1a, 2a, 3a. The user Enters the password incorrect then enters it correct after the first, second, or third attempts.
Includes:	Login use case.
Frequency of Use:	On-demand: Shouldn't be used often, if at all.
Special Requirements:	• Data should not be recoverable once it's destroyed.
Assumptions:	• The user's USB is connected to their mobile device.
Notes and Issues:	1. In the future, functionality can be added to disable biometric authentication after incorrectly entering password twice.

Use Case ID:	UC-2.0.0		
Use Case Name:	Reset Password		
Created By:	Chidiebere Otuonye	**Last Updated By:**	Chidiebere Otuonye
Date Created:	April 6, 2022	**Last Revision Date:**	April 6, 2022
Actors:	Primary Actor: accountHolder		
Description:	This use case is for outlining the behavior of the system when the password is forgotten and the user needs to enter the recovery code in order to login and set new password.		
Trigger:	User forgets password and clicks “Forgot Password”.		
Preconditions:	1. The app and USB stick are paired. 2. The user is on the login screen		
Postconditions:	1. *Minimal Guarantee:* The system retains all data from accounts that user intends to keep and integrity of control flow is preserved. 2. *Success Guarantee:* The user’s password is changed successfully and they’re able to log in.		

Normal Flow:	1. The user is on the login screen and presses the “Forgot Password” button. 2. The user enters the correct recovery key. 3. The user is brought to a page that allows them to reset their password. 4. The user enters a new password with the correct specifications twice for verification and confirms. 5. The user sees a message indicating that the password was changed successfully and is brought to their “Accounts” page.
Alternative Flows:	1a. In step 1 the user decides not to enter their recovery key and hits the “cancel” button. 1. User is brought back to login page. 2a. The user enters an incorrect recovery key. 1. The user is shown a message saying their recovery key is incorrect. 4a. In step 4 the user enters a password that’s incorrect or doesn’t meet the specified requirements. 1. The user is shown a message saying that the passwords must match and meet the specified requirements.
Exceptions:	Exceptions to the Reset Password use case None
Includes:	Login use case.
Frequency of Use:	On-demand: Shouldn’t be used often, only if user forgets password.
Special Requirements:	• None
Assumptions:	• The user is able to use the application without issue on their device. • Device and USB stick are paired.
Notes and Issues:	1. The password must be at least 6 letters long, have 1 capital letter, and one special character.

4.2 Use Case Diagrams

Figure 1: Backup Database

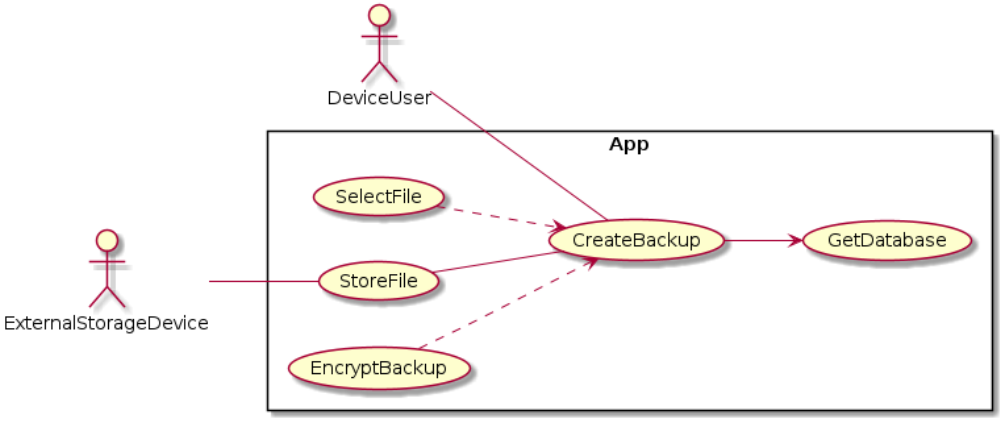


Figure 2: Create Account Entry Group

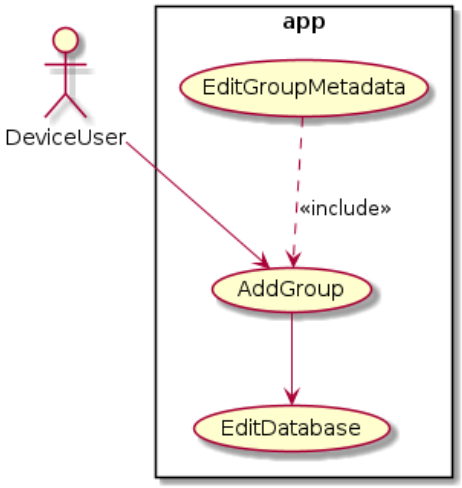


Figure 3: Check Database for Account Data Compromise

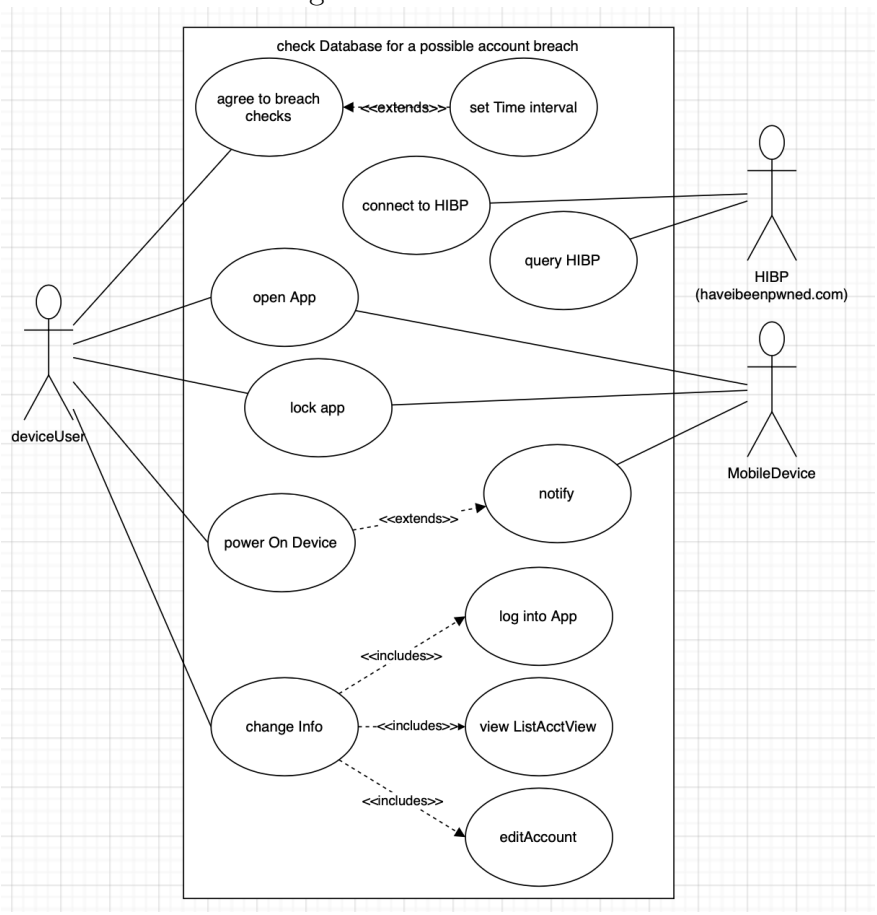


Figure 4: Copy Password to MobileDevice Clipboard

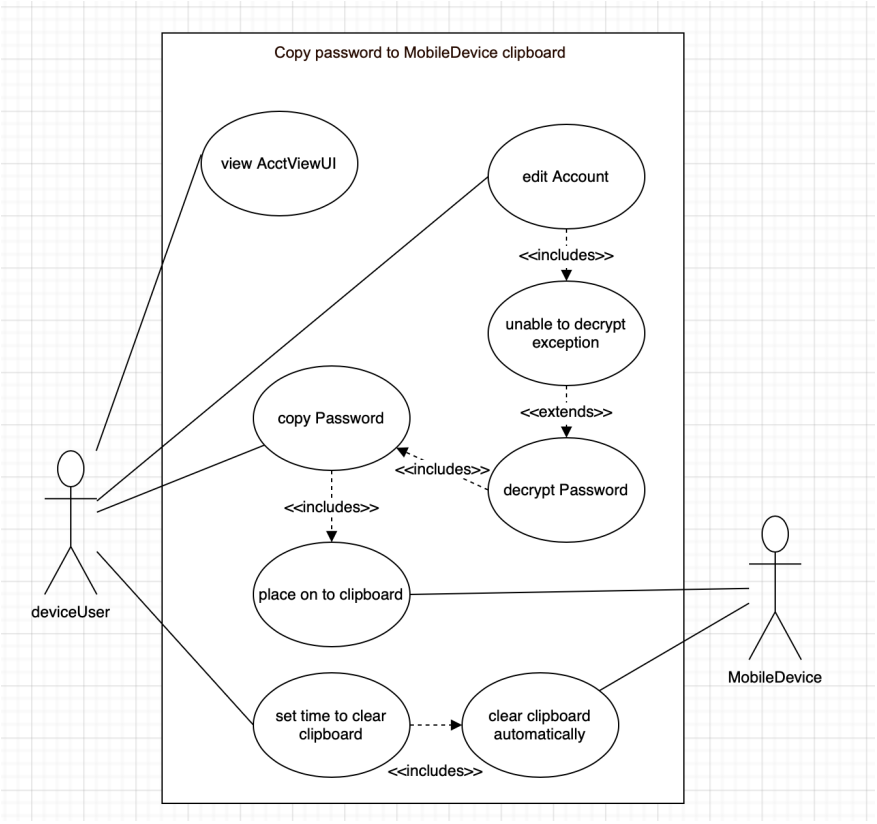


Figure 5: Changing database password

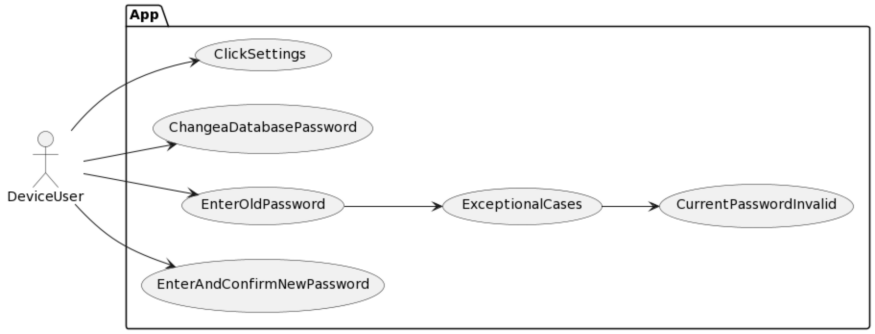


Figure 6: Changing Account Password

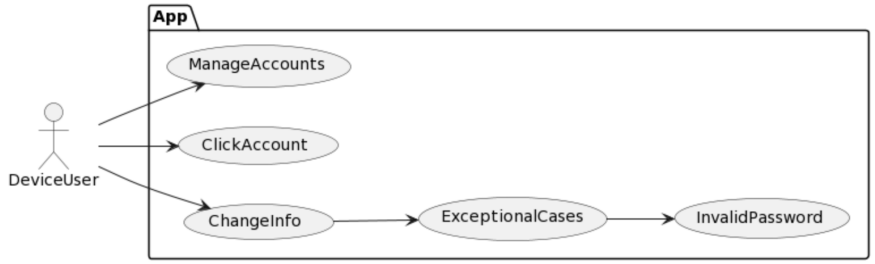


Figure 7: Change Database Login Password

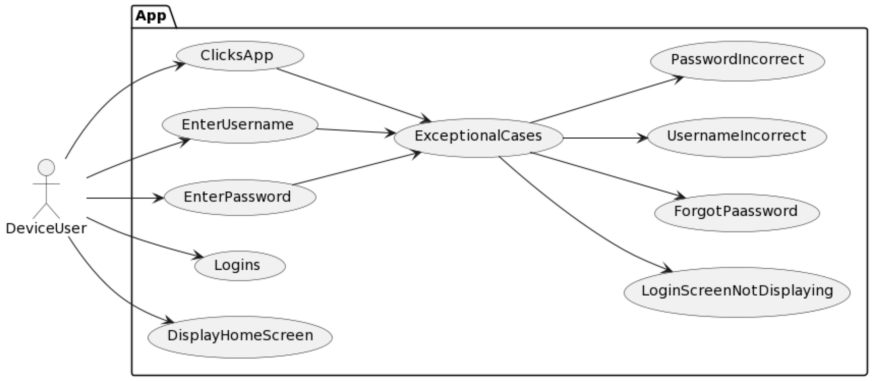


Figure 8: Lock App

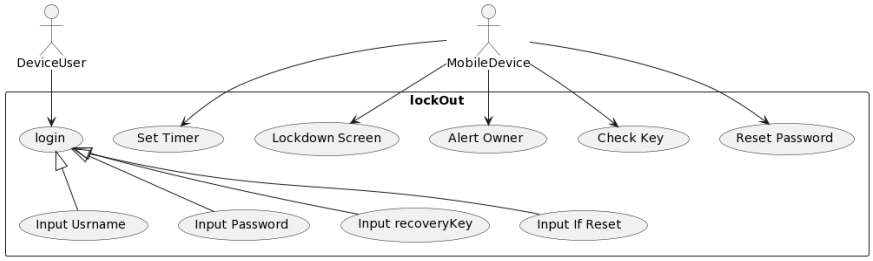


Figure 9: Search Database

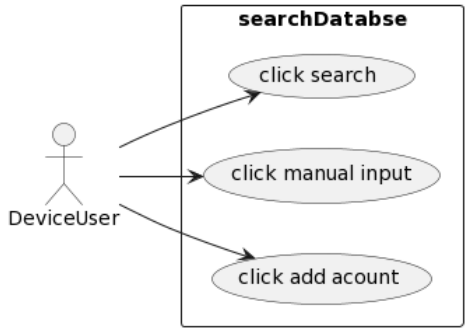


Figure 10: Nuke Database

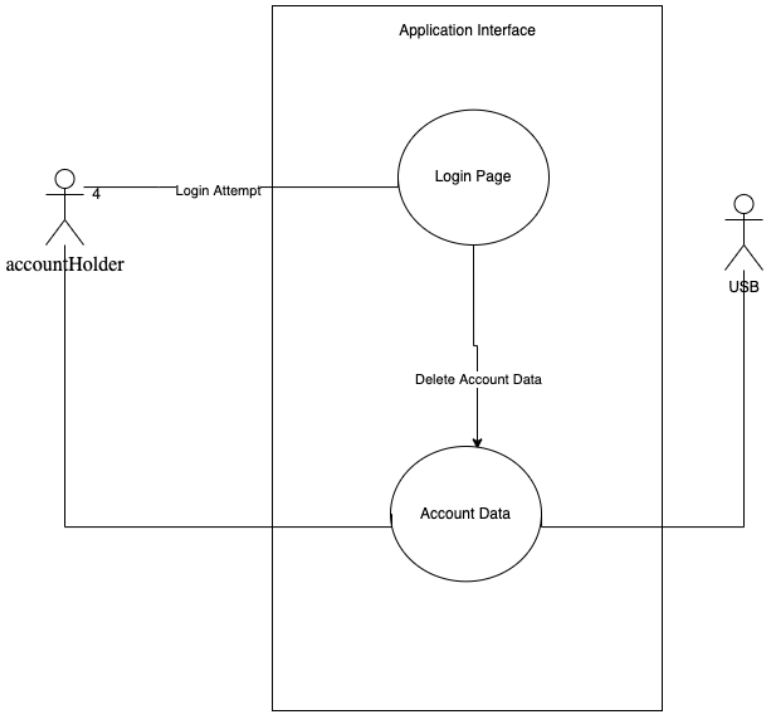
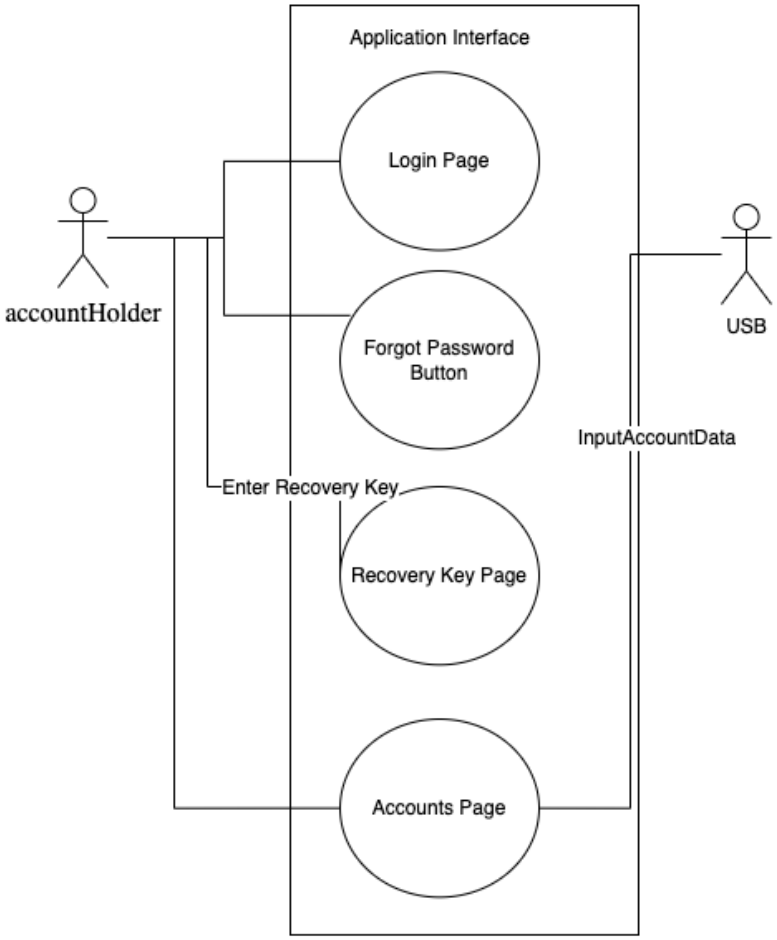


Figure 11: Reset Password



5 Reqirements Analysis

5.1 Scenarios

Backup Database: The DeviceUser decides they would like to create an external backup of their account database. To do so, they open the PassUSB app, open their existing account database, press the drop-down menu button and press the menu entry titled "Backup Database". In doing so, a system directory chooser menu will appear prompting the user to choose a suitable directory to store the database backup. The DeviceUser then navigates to the path of their external storage device, and presses the "Backup To This Directory" button to choose the directory that will store the database backup. Then, the user is asked to enter a password twice to be used to encrypt the backup file. Once the user confirms the password to be used to backup the file, the backup procedure begins and the user is shown a throbber during the process. After the backup process is complete, the User is notified that the backup has been successfully saved, and is given the name of the saved backup file. The User is brought back to the list of accounts menu.

Create Account Entry Group: After opening their existing account database, the DeviceUser decides they would like to create a group to organize their account entries better. To do so, they press the button labeled with the + symbol and then press the "Add group" button. The DeviceUser is then presented a dialog box prompting them to enter a name for the group as well as the option to edit the symbol emoji associated with the group which by default is that of a folder. Once the DeviceUser is satisfied with their editing they can then press the "Add" button. After the folder has been added to the database, the DeviceUser is brought back to the account list view.

Check Database for Account Data Compromise: After pressing the switch to agree for periodic checks of their accounts, Jon closes the app and turns off the screen of their phone (not entirely where their phone is powered off). While Jon's iPhone is turned off, the app passively checks the accounts stored for possible breaches through haveibeenpwned.com. The app sends Jon's iPhone a notification on the status of their accounts every so often. After some time, a notification appears on Jon's iPhone that shows a red X symbol alongside with a message indicating which accounts may have been compromised. Upon pressing the notification, the app opens up to the log-in page that asks for the password or a scan of Jon's face to get into the list of accounts. After Jon logs in using a master password, the app enters the main/usual page that shows the entire list of accounts but with certain account/entries highlighted in red indicating a possible breach. Jon presses on one of the highlighted accounts. This action brings up a page that allows Jon to change the password or other info with a banner on the top that indicates urgency. Jon then changes the password for the account. Jon saves the changes and exits to the list of accounts by pressing a button on the top left. Jon changes the password of all the highlighted accounts through the previously mentioned steps and the state of the list of accounts reverts back to normal.

Copy Password to MobileDevice Clipboard: While at the account screen, Jon sees a button next to each of the 4 text fields present. The button has a clipboard symbol. Jon presses the password clipboard button. The password is placed in the MobileDevice's clipboard. Jon's iPhone then clears the text from the clipboard after a certain amount of time.

Scenario for Modifying Account Details: Bob wishes to change his password for Facebook. He decides to first log into the app PassUSB. Once in the app, Bob locates the app's Settings, from

there he clicks the Manage Accounts button. Bob is then brought to a list of his accounts that are saved within the app. Bob finds Facebook and clicks on it. Once he is within the Facebook account, he selects change password. As he is entering his new password his phone shuts down due to no more battery life left. Bob charges his phone to 80% and decides to try again. Bob then logs back into the app, finds Settings, clicks into Manage Accounts, finds Facebook, and modifies the password. He finds that his old password remains and the one he had entered before his phone shut down didn't save. Once Bob finishes entering his new password, he hits the Save button and is brought back to the page that lists his accounts. Bob then decides to click the return to home screen button and logs out of the app.

Scenario for Logging into the App: Miles wants to add his YouTube account information into the app PassUSB. First Miles has to login into PassUSB. He turns on his phone and locates the app. Miles then clicks onto PassUSB. He is brought to a screen that prompts him for his username and password. Miles enters his username, he then moves to password. Miles can't recall his password and after searching can't find it written anywhere. Miles then decides to click forgot password. Miles then sees an option that allows him to answer a security question. He clicks on this option, and the question "Who is your childhood best friend?" displays. Miles answers "James", the answer is accepted, and Miles is prompted to enter his new password for the app. Miles decides to select the option that randomly generates a password. He saves this new password and hits return to login. Miles then enters his username and password into PassUSB and hits enter. His information is successfully processed and he is brought to the main screen on the app.

PassNuke: John loses his USB stick and is unable to find it; someone finds the stick and attempts to pair it with their device and login, not knowing that John enabled the PassNuke function. They attempt to login 4 times and after the fourth time, all the information on the USB stick is erased and the individual is unable to retrieve any of the information previously on it.

Reset Password: Brenna forgot the password to her USB stick. She attempts to enter it 3 times to no avail. She finds her recovery key and hits the "Forgot Password" button. She enters her recovery key incorrectly the first time, then correctly the second time. She then creates a new password that fulfills all the requirements and is able to access her information.

Lock App: Dave is a bad hacker trying to get the passwords from everyone else's devices. He connects his computer to a man's device he stole and attempts to login to the password generator/storing app to get his information and sell it online for free. When he connects his tools to get the passwords he inputs the wrong password five times. After that fifth attempt he is alerted that he sees that the screen switches to a different screen asking him to input a recovery key. Dave puts in some random numbers, and after the mobileDevice checks its database it discovers that the securityKey is wrong. The screen changes to a solid background with the message "Lockdown incorrect password input" and starts a countdown of 30 minutes on the screen. The account owner John is alerted about each login attempt (specifically the time, source, date, and number of attempts), and is asked if he would like to change his master password. John thinks he is secure and chooses to say no and not change his password.

Search Database: John logs into the app and wants to check the available accounts that he can add information for and generate a password for it. He types the search database button and looks up CrunchRoll. He notices that there is no CrunchyRoll available, so he clicks input manually

button. It is there he goes through the same process in the addAccount view to create his new account.

5.2 Static model

5.2.1 Object Diagrams

Figure 12: An example of opening an account in read-only mode

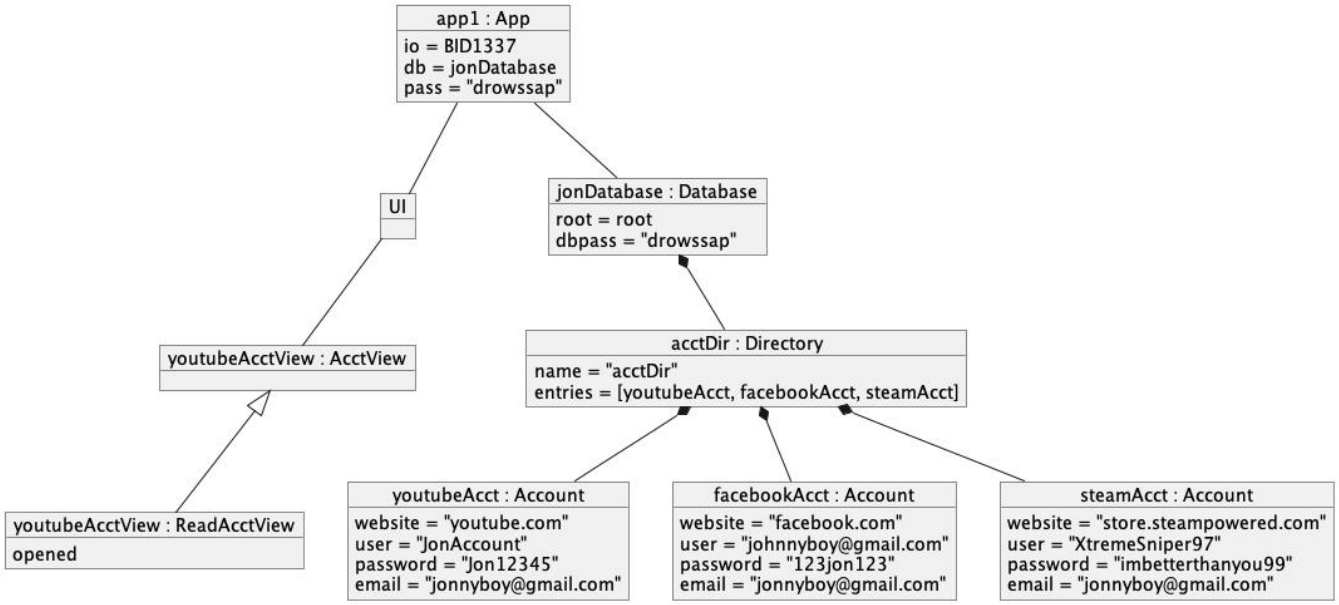


Figure 13: Demonstrating the storage topology of the database

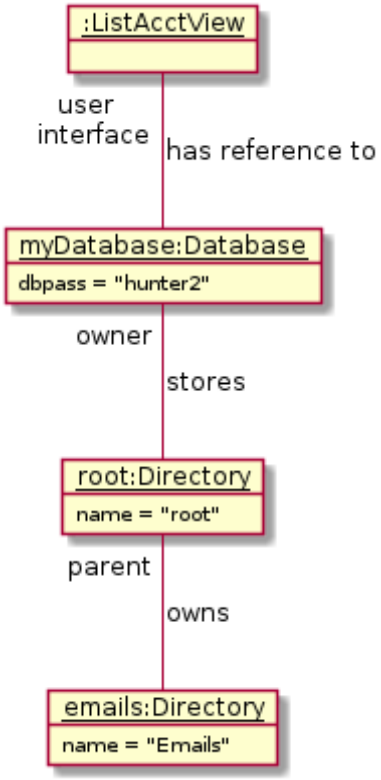


Figure 14: Demonstrating the use of subfolders

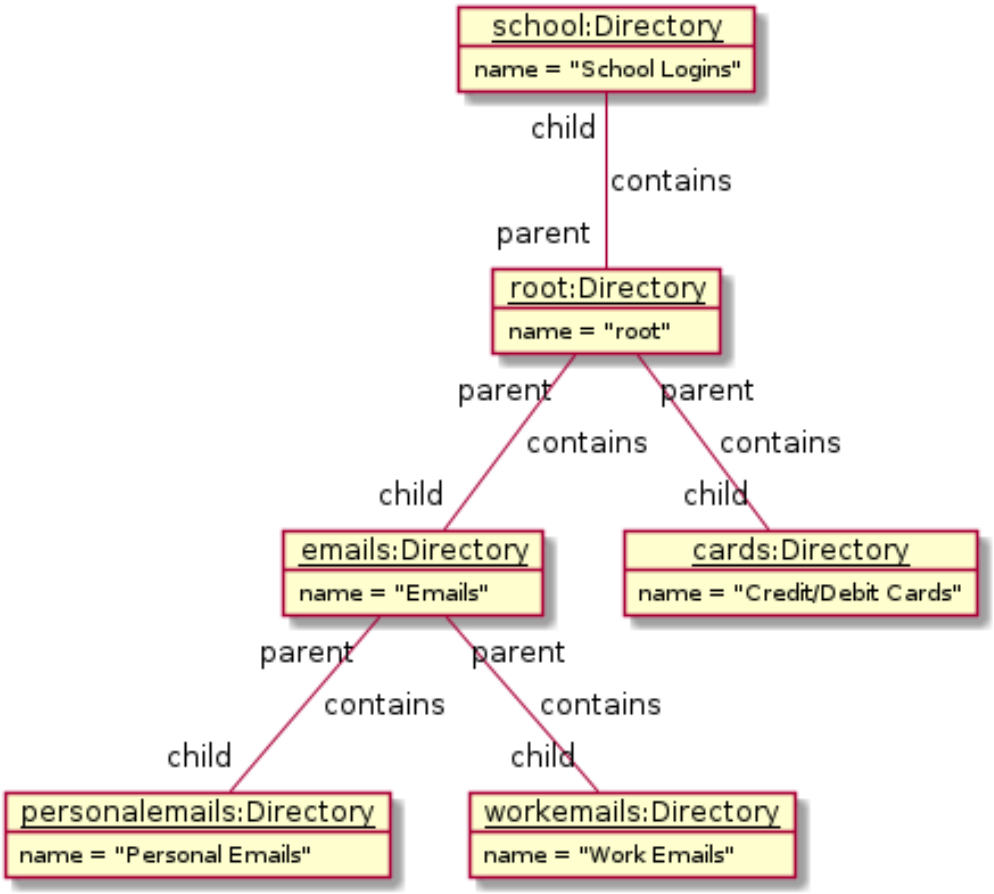


Figure 15: Demonstrating keystroke transmission to PassUSB dongle

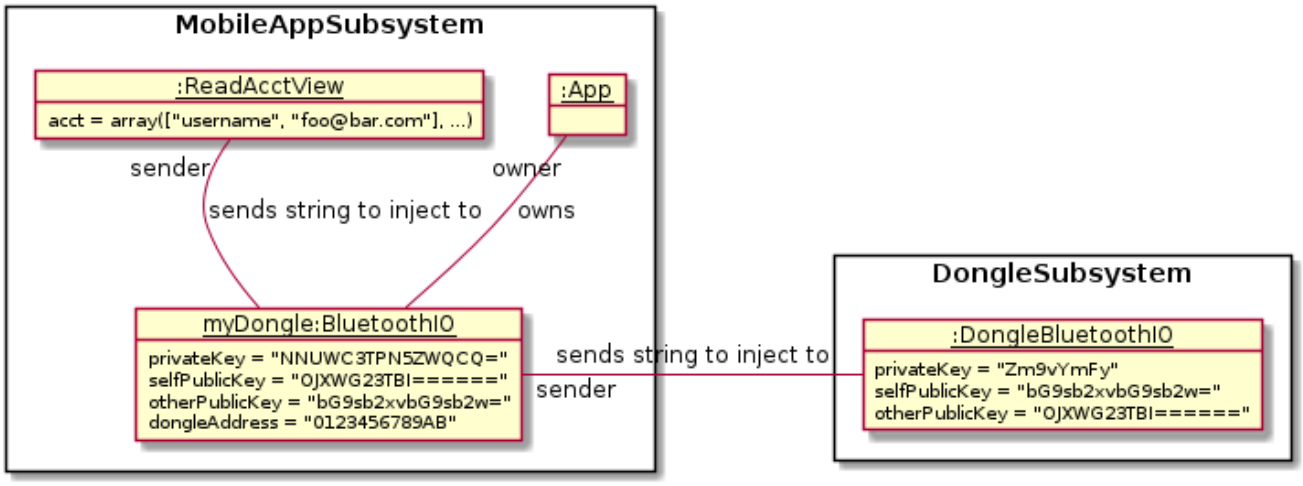


Figure 16: Demonstrating app unlocking datapaths and logic

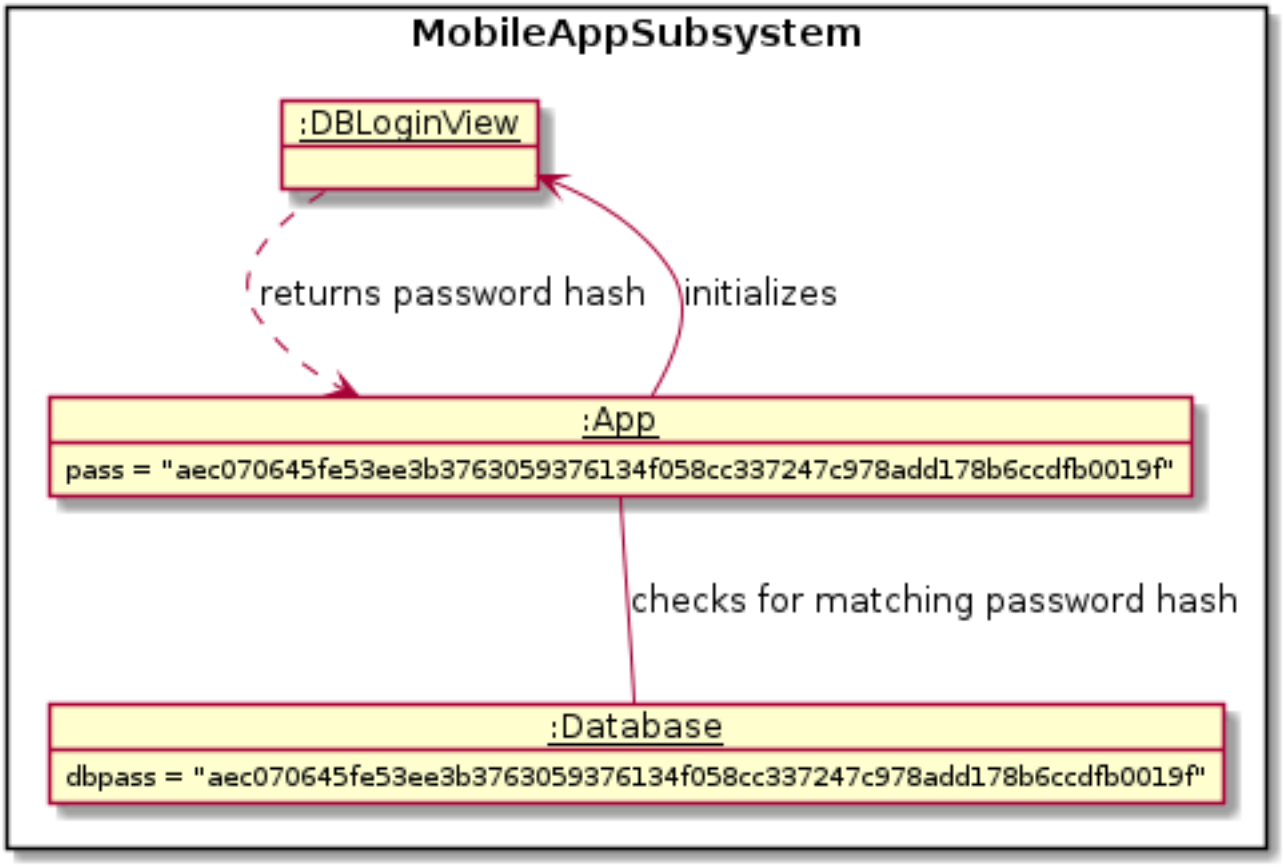
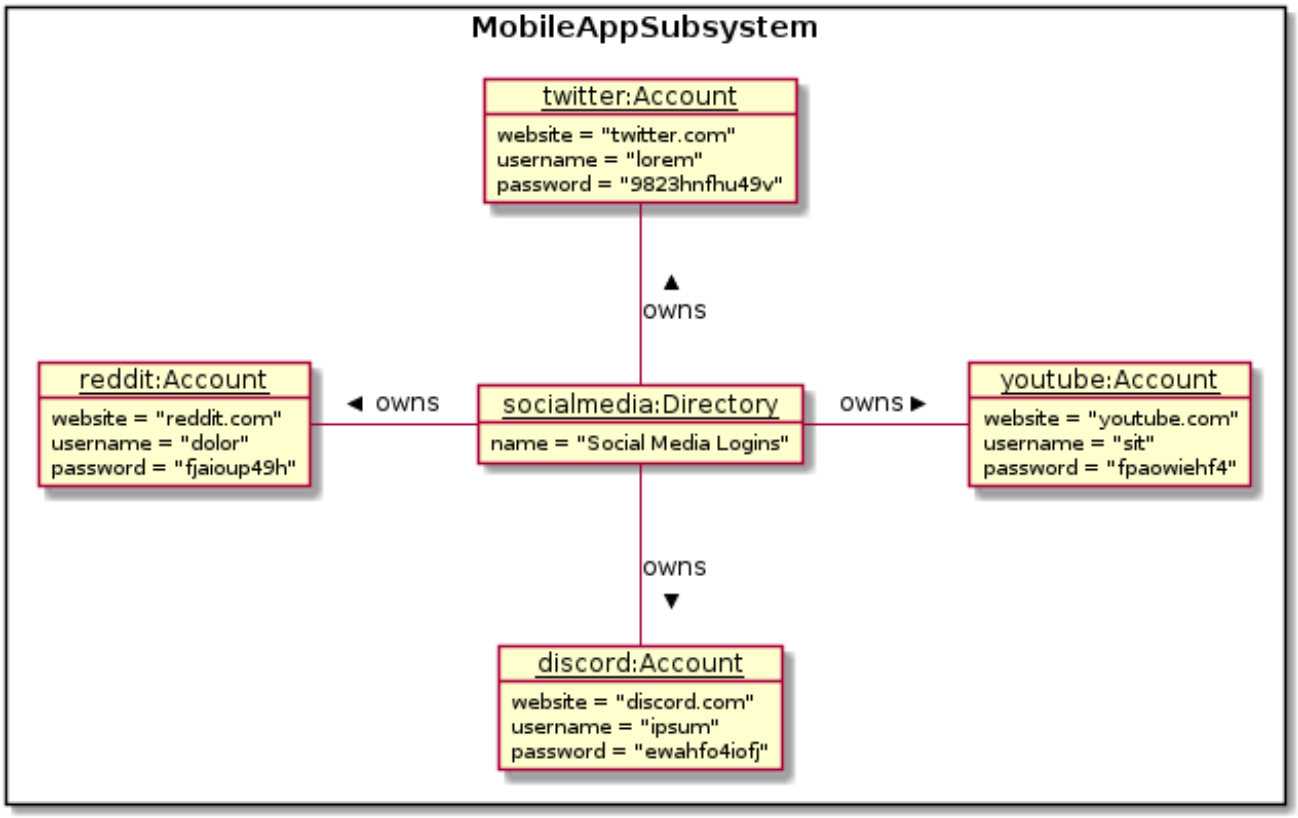
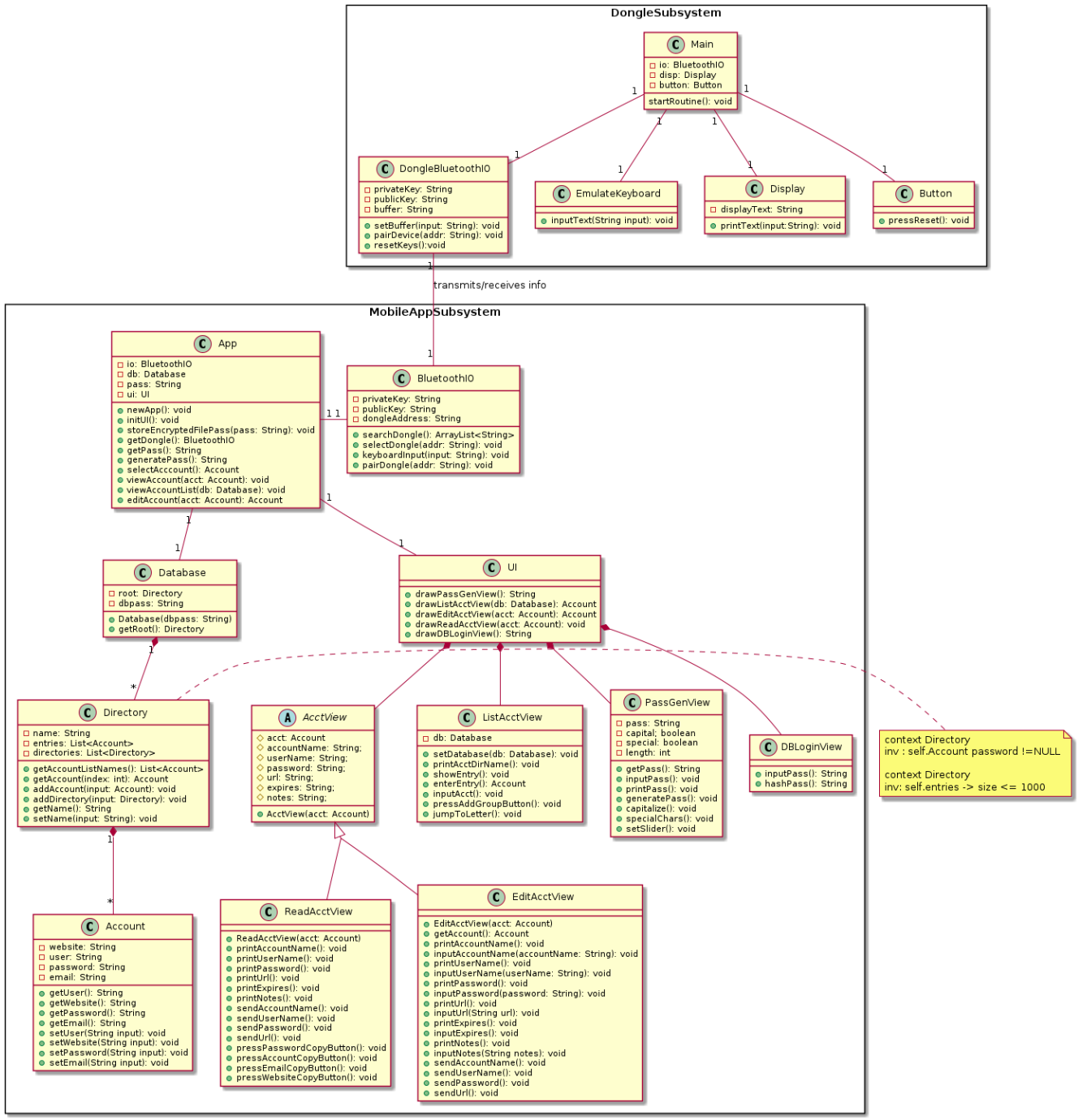


Figure 17: An example of account entry storage



5.2.2 Class Diagram



5.3 Dynamic Model

Figure 18: Create Account Entry Group

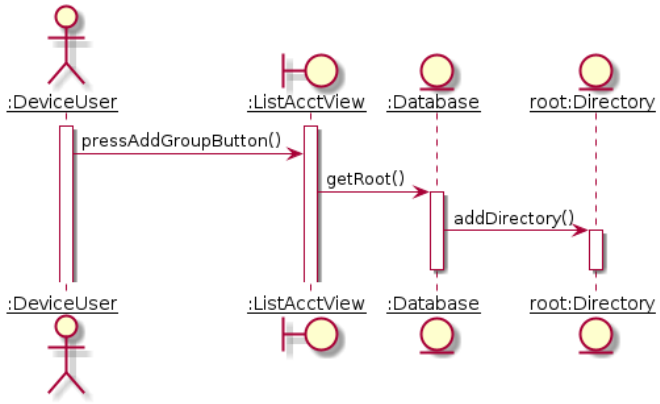


Figure 19: Copy Password to MobileDevice Clipboard

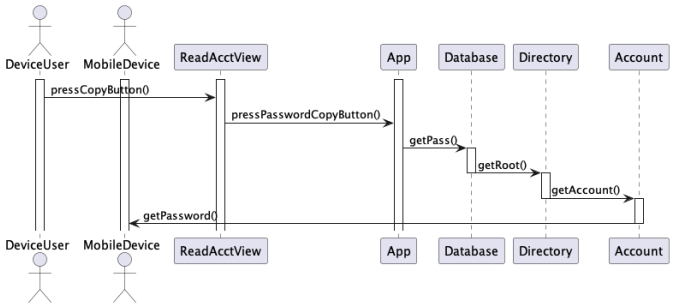


Figure 20: Modify Account Details

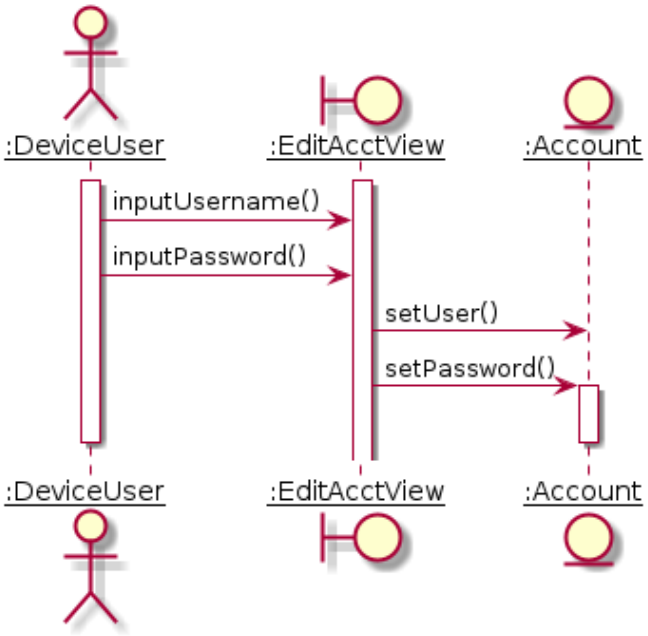
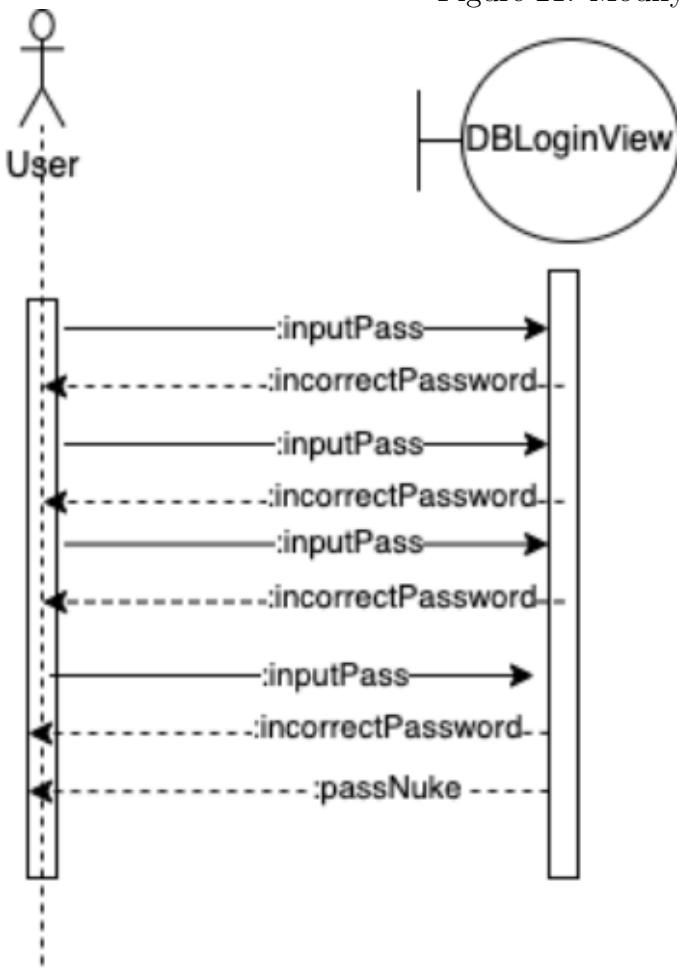


Figure 21: Modify Account Details



6 Glossary

- PassUSB: the project's USB dongle solution that emulates a USB keyboard
- Password manager: a computer program that generates, stores and retrieves passswords for its users
- HID (Human Interface Device): a computer device that facilitates communications between a computer user and a computer
- App: a computer application
- Pairing: the process of recognition and acknowledgement between the mobile device and USB dongle

7 Approval Page

8 References

- https://haveibeenpwned.com
- https://en.wikipedia.org/wiki/List_of_data_breaches
- https://www.okta.com/identity-101/password-entropy
- https://ssd.eff.org/en/module/seven-steps-digital-security
- https://www.geeksforgeeks.org/brute-force-attack/
- https://www.geeksforgeeks.org/understanding-rainbow-table-attack/

9 Appendix

9.1 Project schedule

PassUSB — Gantt Chart Template © 2006-2018 by Vertex42.com — Learn about the Pro version >

Bonobo CO.

Project Start Date: 2/7/2022 (Monday) — Display Week: 1 — Project Lead: Gentman Tan

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Requirement Elicitation**						
0.1	Client meeting	Rey/GT	Mon 2/07/22	Tue 2/08/22	2	0%	2
0.1	Case Analysis	Rey/GT	Wed 2/09/22	Thu 2/10/22	2	0%	2
0.1	Case Analysis	Rey/GT	Fri 2/11/22	Sat 2/12/22	2	0%	1
0.1	User Analysis	Rey/GT	Sun 2/13/22	Mon 2/14/22	2	0%	1
0.0.1	Requirement	Rey/GT	Tue 2/15/22	Wed 2/16/22	2	0%	2
0.0.1	nonfunctional?	Rey/GT	Thu 2/17/22	Fri 2/18/22	2	0%	2
0.1	functional?	Rey/GT	Sat 2/19/22	Sun 2/20/22	2	0%	0
0.1	Prototype	Rey/GT	Mon 2/21/22	Tue 2/22/22	2	0%	2
0.1	Video Mockup Demo	Rey/GT	Wed 2/23/22	Thu 2/24/22	2	0%	2
1	**Design**						
0.1	Sketch User Interface System	Kaiya	Sat 2/26/22	Mon 2/28/22	3	0%	1
0.1	Design Use Cases	Kaiya	Tue 3/01/22	Wed 3/02/22	2	0%	2
0.1	Draw Use Case Diagrams	Kaiya	Thu 3/03/22	Thu 3/03/22	1	0%	1
0.1	Create Sequence Diagram	Kaiya	Fri 3/04/22	Sat 3/05/22	2	0%	1
1	**System Development**						
0.1	Application Login Development	Chidie	Sat 3/05/22	Tue 3/08/22	4	0%	2
0.1	Password generating backend	Chidie	Tue 3/08/22	Thu 3/10/22	3	0%	3
0.1	Information storage and encryption	Chidie	Thu 3/10/22	Sat 3/12/22	3	0%	2
0.1	UI/UX development for dongle	Chidie	Sat 3/12/22	Thu 3/17/22	6	0%	4
0.1	Backend finalization	Chidie	Thu 3/17/22	Fri 3/18/22	2	0%	2
1	**Testing**						
0.1	Build Application	Sonali	Sat 3/19/22	Sat 3/19/22	1	0%	0
0.1	Install Application on test devices	Sonali	Sun 3/20/22	Sun 3/20/22	1	0%	0
0.1	Test iOS Deivces	Sonali	Mon 3/21/22	Mon 3/21/22	1	0%	1
0.1	Test Andoird devices	Sonali	Tue 3/22/22	Tue 3/22/22	1	0%	1
0.1	Test PasUSB dongle	Sonali	Wed 3/23/22	Wed 3/23/22	1	0%	1

9.2 User Interface Designs

9.3 Diary of meeting and tasks

Monday the 31st from 6:15 to 7:20:

- Our group met today for about an hour to discuss project ideas. We picked a project and completed the project proposal.

Wednesday the 9th from 6:15 to 7:00:

- Our group met today for about an hour to complete the in-class assignment.

Monday the 14th from 6:15 to 7:15:

- Our group met today to discuss the Group Project Part I: Analysis. We assigned roles and discussed how to go forward with the execution of the rest of the project.

Wednesday the 16th from 6:15 to 7:00:

- Our group met today to complete the third in-class activity.

Monday the 21st from 5:10 to 6:15:

- Our group met today in class and worked on our use cases for our project.

Monday 28th of March from 6:10 to 6:30

- We discussed Analysis 2 and assigned tasks

Wednesday the 30th of March from 5:45 to 6:15:

- We completed the in-class activity

Wednesday the 6th of April from 5:50 to 7:30:

- We completed the in-class activity

Wednesday the 13th of April from 5:50 to 7:00:

- We worked on the diagrams for the project