Password Safe
With WebDAV Sync

by
Jake Schuenke
Matt Spurr
Tyler Shelton
Kyle Bippus

Contents

Executive Summary

The goal of this project is to create a “password safe” application. This application will allow users to generate and store passwords with varying levels of strength in a secure location. The user will need only a single password to access their safe, and then they can view any of their previously stored passwords, along with the corresponding username. This product will allow for quick and easy lookup of all stored

passwords, as well as directly linking a password to the appropriate site. Passwords can be stored either on a WebDAV server or the Omni Group server.

In analyzing the problem four main needs of the system were identified: the code must be open source, you must be able to generate strong passwords, you have to be able to sync data to a WebDAV server, and there should be a link between the password and websites. These needs map to nine features that are completed in the application.

After the analysis of the problem, how to install the application unto an iPhone is shown. This is a step by step guide that will show you how to install our application onto your phone from source code. After, a user guide that details all of the information contained on each screen is shown. This section also has a guide that shows how to perform common actions in the application. Following this, the maintenance guide will present common errors and bugs known in the app, and how to fix them.

Finally, the document shows the design of the application. This section starts at a high level overview of the system and give a very technical description of the major objects in the system. This allows developers a reference of how any way classes are the way they are.

Analysis of Problem

The goal of this project is to solve a simple problem. In today's day and age, accounts are required for everything. Each of these accounts has its own separate information: a username and a password. These passwords need to be unique and difficult to hack, so users do not lose sensitive information. The "password safe" application will address this issue by providing users with an easy and secure way to generate and store passwords for each of their various accounts.

The password safe will generate passwords with varying levels of strength. Users can set the strength of the password by choosing whether or not to include numbers, letters, or special characters in the password. The application will also store these passwords in a secure location, either on Omni Group's server or a WebDAV server set up by the client. Each of these passwords can be linked to a username and an account such as a website. In addition, users will be able to create and store notes that contain text. This will allow them to keep track of any additional important information they desire.

User and Stakeholder Description

The users of the system fall into three distinct categories. Each of these user types has differing needs and use cases for the application. It is our desire to meet these wants in developing the application.

Nontechnical user profile and environment

The nontechnical user is a typical person with average skills that will use an app developed for the Mac or iOS markets. They know enough to download and install an app, and are most likely to use multiple different websites that keep track of passwords. They are not skilled with setting up their own WebDAV server so they are likely to use OmniGroup's base sync-server. The nontechnical user will want an easy to use and flexible system with a low learning curve and quick functionality.

The environment for a nontechnical user will be the basic graphic user interface found on a Mac computer or an iOS device. The system will require a data or an internet connection. This group is defined as mainly themselves, but technical and developing users have the same use environment.

Technical user profile and environment

The technical user will be a user that has the same basic needs and uses as a nontechnical user, but will use the capability to set up and use his/her own personal WebDAV server. They have pretty good technical skills with good knowledge of setting up a server, and using apps. They share the same end goals as the nontechnical users.

The technical user's environment will still be a Mac or iOS device to run the app. However, this type of user will develop their own server for whatever they choose. They must have a setup guide for the server so it can easily be synced with the app.

Open source developer profile and environment

The open source developer will have very high end technical knowledge. This project will be an open source, so this user will help develop the code and submit new features, or use this as part of a bigger system. They have a fully functional knowledge of the code base and all of the API calls and other types of things in the system.

The technical user has a very specific environment. While they can use the program their main case is the development and advancement of the system. They will develop the code on a Mac as the code will be written in Objective C and will be written for Mac devices.

User Needs

ID	Need	Effects	Risk	Stability	Possible Solution
N1	Ability to generate strong and unique passwords	All users	Low	High	Software will use a set of options like: lower case only, numbers, symbols, and short words to generate a password of a set length
N2	Host the code to allow it to be open source	Open Source Developers	Low	High	Host the code on a public GitHub repository
N3	Allow the user to choose between a private WebDAV server or Omni Sync Server	Technical users & Open Source Developers	High	Low	Set up a basic system through abstraction and good API design that will allow the user to easily set up the system with a different backend system
N4	Link generated passwords to usernames and websites	All Users	Med	Med	Allow for a table structure that will link usernames, passwords, and the websites they belong too.

Features

These are the criteria that will be used to evaluate each feature.

<table>
<tr>
<td>Priority
Indicates the importance of the feature</td>
<td>
<table>
<tr><td>Critical</td><td>Needed for the project to be a success</td></tr>
<tr><td>Important</td><td>Significantly contributes to the quality of the project</td></tr>
<tr><td>Useful</td><td>Not necessary, but a possible bonus feature</td></tr>
</table>
</td>
</tr>
<tr>
<td>Effort
Indicates the effort required to implement the feature</td>
<td>
<table>
<tr><td>High</td><td>Requires a lot of effort</td></tr>
<tr><td>Medium</td><td>Moderate effort required</td></tr>
<tr><td>Low</td><td>Little effort required</td></tr>
</table>
</td>
</tr>
<tr>
<td>Risk
Indicates the likelihood that the feature will cause problems</td>
<td>
<table>
<tr><td>High</td><td>Likely to cause problems</td></tr>
<tr><td>Medium</td><td>Could cause problems</td></tr>
<tr><td>Low</td><td>Unlikely to cause problems</td></tr>
</table>
</td>
</tr>
<tr>
<td>Stability
Indicates the likelihood that the feature will change</td>
<td>
<table>
<tr><td>High</td><td>Unlikely to change</td></tr>
<tr><td>Medium</td><td>Could change</td></tr>
<tr><td>Low</td><td>Likely to have change in specifications</td></tr>
</table>
</td>
</tr>
</table>

ID	Feature	Status	Priority	Effort	Risk	Stability	Reason
F1	Password generator	Approved	Critical	Med	Med	High	The main purpose of the project is to store passwords, so without being able to generate them it will obviously be a failure.
F2	Allow user to set standards for password generation	Approved	Important	Med	Low	Med	It is important for the user experience to allow customization of how passwords are generated so that they can be memorable or very secure.
F3	Store on OmniServer	Approved	Important	Med	Med	High	This should be the default server for non-technical users, so it should be easy for them to use.
F4	Store on chosen WebDAV server	Approved	Important	High	High	High	This will give advanced users the opportunity to store their data wherever they want.
F5	Database links username, password, and website	Approved	Critical	Low	Med	High	These pieces of information need to be linked, because on their own they have no use.
F6	Prompt user with login screen	Approved	Useful	High	Low	Med	Since the application stores sensitive information, we should validate the user's credentials before displaying anything.

Feature Mapping

	N1	N2	N3	N4
F1	X			
F2	X			
F3			X	
F4			X	
F5				X
F6	X			

Use Cases and Test Cases

UC1: Set up Master Password

Description: The user sets up a master password to open the application.
Actors: User
Main Flow:
1. The system prompts for username and master password.
2. The user enters username.
3. The user enters password.
4. The user verifies password. (c)
5. The system sets password.

Alternate Flows:
a. The user can quit the process at any time.
b. The app may shut down unexpectedly (user shut off phone, phone lost battery and died, etc), causing this process to cancel at any stage.
c. The user verifies the password incorrectly
Pre-Conditions: The system is installed
Post-Conditions: The user has a username and password to log into the application
Test Cases:

ID	Description	Conditions	Sample Input	Expected Output
1.1	The user correctly enters a username and password and verifies the password. (Main flow)	The system is installed.	"Username" "Password" "Password"	The user now has an account that can be logged into.
1.2	The user incorrectly verifies the password. (Alternate flow c)	The system is installed.	"Username" "Password" "PAssword"	The user is shown an error message and asked to reenter the password.
1.3	The user correctly enters information, but then cancels the process instead of submitting. (Alternate flow a)	The system is installed.	"Username" "Password" "Password"	No account is created.

UC2: Set Password Standards

Description: The user can choose how strong their password will be by determining what kind of characters to use and password length.
Actors: User
Main Flow:
1. The user navigates to the "Generate Password" screen. (a)
2. The user selects a length for the password. (b)
3. The user selects which type of characters and how many of each to include (lowercase letters, uppercase letters, numbers, or special characters). (b)
4. The user clicks the "Generate Password" button.

5. A password matching the given constraints appears in the box at the bottom of the screen.

Alternate Flows:

a. The device could fail to load the "Generate Password" screen. User is returned to main screen.

b. The process can be canceled at this point by the user or unexpected device shutdown.

c. The app may shut down unexpectedly (user shut off phone, phone lost battery and died, etc.), causing this process to cancel at any stage.

Pre-Conditions: The user needs the app installed on a device and an account to save the password with.

Post-Conditions: The user is on the password creation screen and a password matching the given standards is generated

Test Cases:

ID	Description	Conditions	Sample Input	Expected Output
2.1	The user changes the amount of uppercase letters (Main flow)	The system is installed. All other parameters are left as the default	Uppercase: 5 Special: 0 Numbers: 0 Length: 10	Password with at least as many of each type of character
2.2	The user changes the amount of special characters (Main flow)	The system is installed. All other parameters are left as the default	Uppercase: 0 Special: 3 Numbers: 0 Length: 10	Password with at least as many of each type of character
2.3	The user changes the amount of numbers (Main flow)	The system is installed. All other parameters are left as the default	Uppercase: 0 Special: 0 Numbers: 4 Length: 10	Password with at least as many of each type of character
2.4	The user changes the length of the password (Main flow)	The system is installed. All other parameters are left as the default	Uppercase: 0 Special: 0 Numbers: 0 Length: 12	Password with at least as many of each type of character
2.5	The user changes all parameters. (Main flow)	The system is installed	Uppercase: 2 Special: 5 Numbers: 3 Length: 15	Password with at least as many of each type of character
2.6	The user changes parameters but then exits (Alternate flow b)	The system is installed.	Uppercase: 2 Special: 5 Numbers: 3 Length: 15	Nothing; navigated to the last screen the user was on

UC3: Set up connection to WebDAV server

Description: The user can set up a WebDAV server to connect to instead of the OmniSync server.
Actors: Advanced User
Main Flow:
1. The user navigates to the "Network Settings" screen.
2. The user enters information corresponding to the desired server to sync to. (b)
3. Application connects to the server. (a)

Alternate Flows:
a. The app may shut down unexpectedly (user shut off phone, phone lost battery and died, etc), causing this process to cancel at any stage.
b. The user enters invalid information
Pre-Conditions: User must have a WebDAV server to sync with.
Post-Conditions: Application now syncs with WebDAV servers.
Test Cases:

ID	Description	Conditions	Sample Input	Expected Output
3.1	The user enters the correct information to a WebDAV server (Main flow)	The system is installed. The user has navigated to the "Settings" screen The WebDav server is properly set up and is currently setup to connect to devices	Address: sync.omni.com /spurrme Username: spurrme Password: P@ssw0rd@1	The user connects to the server and the data syncs with a the server
3.2	The user enters the valid address for the server but not the correct login information (Alternate flow b)	The system is installed. The user has navigated to the "Settings" screen The WebDav server is properly set up and is currently setup to connect to devices	Address: sync.omni.com /spurrme Username: spurrme Password: P@ssw0r21	The user does not connect and is prompted with a connect error screen

UC4: Set Notes

Description: The user can store any notes that they wish.

Actors: User

Main Flow:

1. The user navigates to “Notes” screen. (a)
2. The user clicks “Add/Change Note” button.
3. The user is directed to the “Enter Note” screen. (b)
4. The user types a note.
5. The user clicks the “Finished” button. (c)
6. The note is saved.
7. The user is returned to the “Notes” screen.

Alternate Flows:

a. The user may click the “Cancel” button at any step prior to step 5. User will be returned to the “Notes” screen and no data will be saved.

b. This process may be canceled at any time by unexpected device shutdown. Data will not be saved unless the process has already completed step 6.

Pre-Conditions: The user must have the app installed

Post-Conditions: The note must be saved. The user is returned to “Notes” screen.

Test Cases:

ID	Description	Conditions	Sample Input	Expected Output
4.1	The user creates a note (Main flow)	The system is installed. Currently on the “Enter Note” screen	Text: This is a test note	The note is saved and the user is redirected to the notes screen
4.2	The user cancels the note he is creating. (Alternate flow a)	The system is installed. Currently on the “Enter Note” screen	Text: This is a test note	The note is discarded and memory used is wiped
4.3	The application crashes or the phone is turned off while user is creating a note (Alternate flow b)	The system is installed. Currently on the “Enter Note” screen	Text: This is a test note	The application will not save the note

UC5: Password Lookup

Description: The user looks up passwords and their related accounts

Actors: User

Main Flow:

1. The user navigates to Password screen
2. The user selects desired account
3. The system displays information for that account (b)

Alternate Flows:

a. The user can quit the process at any time

b. The app may shut down unexpectedly (user shut off phone, phone lost battery died, etc), causing this process to cancel at any stage.

Pre-Conditions: The system is installed.
The user has an account.

Post-Conditions: Password is displayed along with username

UC6: Add Password

Description: The user sets a password and username for a new account

Actors: User

Main Flow:

1. The user navigates to Generate Password screen (a)
2. The user sets standards for password generation
3. The system generates password
4. The user clicks button to advance to account creation screen (b)
5. The system associates generated password with new account
6. The user enters username
7. The user clicks “Create” button
8. The system stores information (c)

Alternate Flows:

a. The user can quit the process at any time

b. The device may fail to load account creation page. User is returned to main screen.

c. The device may fail to create new account. User is notified and returned to main screen.

Pre-Conditions: The system is installed
The user has an account

Post-Conditions: A password is created for a new account

Test Cases:

ID	Description	Conditions	Sample Input	Expected Output
6.1	The user changes one of the settings for the password (Main flow)	The system is installed. All other settings retain their original values	User moves one of the sliders to a desired setting	That setting is saved and any password generated will obey that setting
6.2	The user clicks the “Generate” button (Main flow)	The system is installed. The password must meet all specified criteria	The user selects password settings and clicks the “Generate” button	A password is displayed to the user that meets all the selected criteria
6.3	The user clicks the “Generate Password and Create Account” button (Main flow)	The system is installed. The password must meet all specified criteria	The user selects password settings and clicks the “Generate Password and Create Account” button	The user is directed to the account creation page. The generated password is automatically placed in “Password” field
6.4	The user enters information but then exits (Alternate flow a)	The system is installed.	User moves sliders to desired settings	Nothing; navigated to the last screen the user was on

UC7: Automatic Sync

Description: User's data is synced at application startup
Actors: User
Main Flow:
1. The user enters Master Password at Login Screen.
2. The system checks the information for correctness. (a)
3. The system then automatically syncs the data with the server.

Alternate Flow:
a. The system checks the information and it is incorrect. The system then displays the Log in screen with a message.
b. The app may shut down unexpectedly (user shut off phone, phone lost battery and died, etc), causing this process to cancel at any stage.
Pre-Conditions: The user has setup passwords, syncing to server, and has enabled Automatic Data Sync
Post-Conditions: The user is at the Main Menu and data has been synced. Device is in the "ground truth" state.
Test Cases:

ID	Description	Conditions	Sample Input	Expected Output
7.1	Data is synced when the application starts (Main flow)	The system is installed. The device must be able to sync with the server. The "Sync Automatically" setting must be active.	User starts the application and enters their master password	All passwords and data that are allowed to sync are synced with the server

Supplementary Specifications and Test Cases

This section details requirements of the system that are not specified through a use case. There is also the methodology behind testing each of the specifications.

SS1: Application is intuitive to use

Source: Users
Stimulus: User opens the application for the first time
Artifact: The system
Environment: During normal operation
Response: The User uses the app for the first time after reading the manual.
Response Measure: The User will judge that the base features of the application are intuitive and easy to use from application launch. The system is intuitive enough for the user to be able to understand how to use it with little experience. Additionally, the user manual explains the system effectively.
Test Case:

ID	Description	Conditions	Sample Input	Expected Output
1.1	The user tests out the application for the first time	The user is shown a prototype	User testing	User thinks that the application is easy to use and has useful features

SS2: New user must be able to generate first new password within 2 minutes from launch

Source: Users
Stimulus: User opens the new password screen
Artifact: The system
Environment: During normal operation
Response: The User creates a new password.
Response Measure: The system will be designed to be responsive and easy to use, so the user can create a password in under 2 minutes from system launch.
Test Case:

ID	Description	Conditions	Sample Input	Expected Output
2.1	The user creates a new password	The system is installed.	User follows basic flow of Use Case 8	A new password is stored, and it took less than 2 minutes

SS3: New user must be able to generate first new note within 2 minutes from launch

Source: Users
Stimulus: User opens the new note screen
Artifact: The system
Environment: During normal operation
Response: The User creates a new note
Response Measure: The system will be designed to be responsive and easy to use, so the user can create a 4 sentence note in under 2 minutes from system launch.

Test Case:

ID	Description	Conditions	Sample Input	Expected Output
3.1	The user creates a note	The system is installed.	User follows basic flow of Use Case 6	A new note is stored, and it took less than 5 minutes.

SS4: New user must be able to view all created passwords within 5 seconds of launch

Source: Users
Stimulus: User opens the Password screen
Artifact: The system
Environment: During normal operation
Response: The User views a password
Response Measure: The system will be designed to be responsive and easy to use, so the user can view a password in under 5 seconds from system launch.
Test Case:

ID	Description	Conditions	Sample Input	Expected Output
4.1	The user views a password	The system is installed.	User follows basic flow of Use Case 7	User sees note in under 5 seconds.

SS5: Application must run at 60 fps

Source: Internal to the system
Stimulus: User navigates screens
Artifact: The system
Environment: During normal operation
Response: The system handles the users input and responds accordingly.
Response Measure: Any supported device running the app should be capable of fluid screen navigation at sixty frames per second. Any input given by the user that moves to another screen or opens a dialog box should run at a minimum of sixty frames per second.
Test Case:

ID	Description	Conditions	Sample Input	Expected Output
5.1	The user uses the system	The system is installed.	The user navigates through screens	The navigations operate at sixty frames per second.

SS6: All generation of passwords must be in a consistent time

Source: Internal to the system
Stimulus: User clicks "Generate Password"
Artifact: The system
Environment: During normal operation
Response: The system generates a password based on the given constraints and displays it to the user
Response Measure: The time taken to complete each generation should be consistent. The average run should take a hundredth of a second. The worst case run should take no longer than a quarter of a second with an animation to show wait time.
Test Case:

ID	Description	Conditions	Sample Input	Expected Output
6.1	The user generates a password	The system is installed.	User changes standards and clicks button to generate password	The average time should be less than a hundredth of a second

SS7: Syncing time with scale linearly with the amount of information stored

Source: Internal to the system
Stimulus: User syncs to Omni Sync or a WebDAV server
Artifact: The system
Environment: During normal operation
Response: The system updates passwords on the server and the device to the most recent version
Response Measure: Syncing data with the server should be a consistently fast process. However, the time taken to sync is dependent on the server cannot be directly specified. To facilitate the syncing process, Password Safe will use coding standards that minimize server interaction. The time taken to sync should scale linearly with the number of passwords (and corresponding username and/or note) that have to be synced. This number is generated by discovering all passwords that have changed since the last sync of the file. To take internet connectivity out of the problem a user will be able to sync a password in a tenth of a second on a Local Area Network.
Test Case:

ID	Description	Conditions	Sample Input	Expected Output
7.1	Data syncs with the server	The system is installed. The device must be able to sync with the server	User closes application	Data synced linearly based on size

Installation Guide

Note: This set of instructions assumes that you have the latest version of Xcode installed, have an Apple Developer Account set up, and certificates are installed on both your Mac computer and your iOS device.

1. Open the project in Xcode.
2. Make sure you have a provisioning profile.
    1. Go to developer.apple.com, click Member Center, and then log in.
    2. Click "Certificates, Identifiers, and Profiles".
    3. In the left panel, click “Provisioning Profiles”.
    4. Find your profile (check that the expiration date is one year from when you signed up as a developer) and download it.
    5. Once you've downloaded it, find the file and double-click it to open it.
    6. If you've downloaded the correct profile, Xcode will tell you it has been verified (Figure 1)

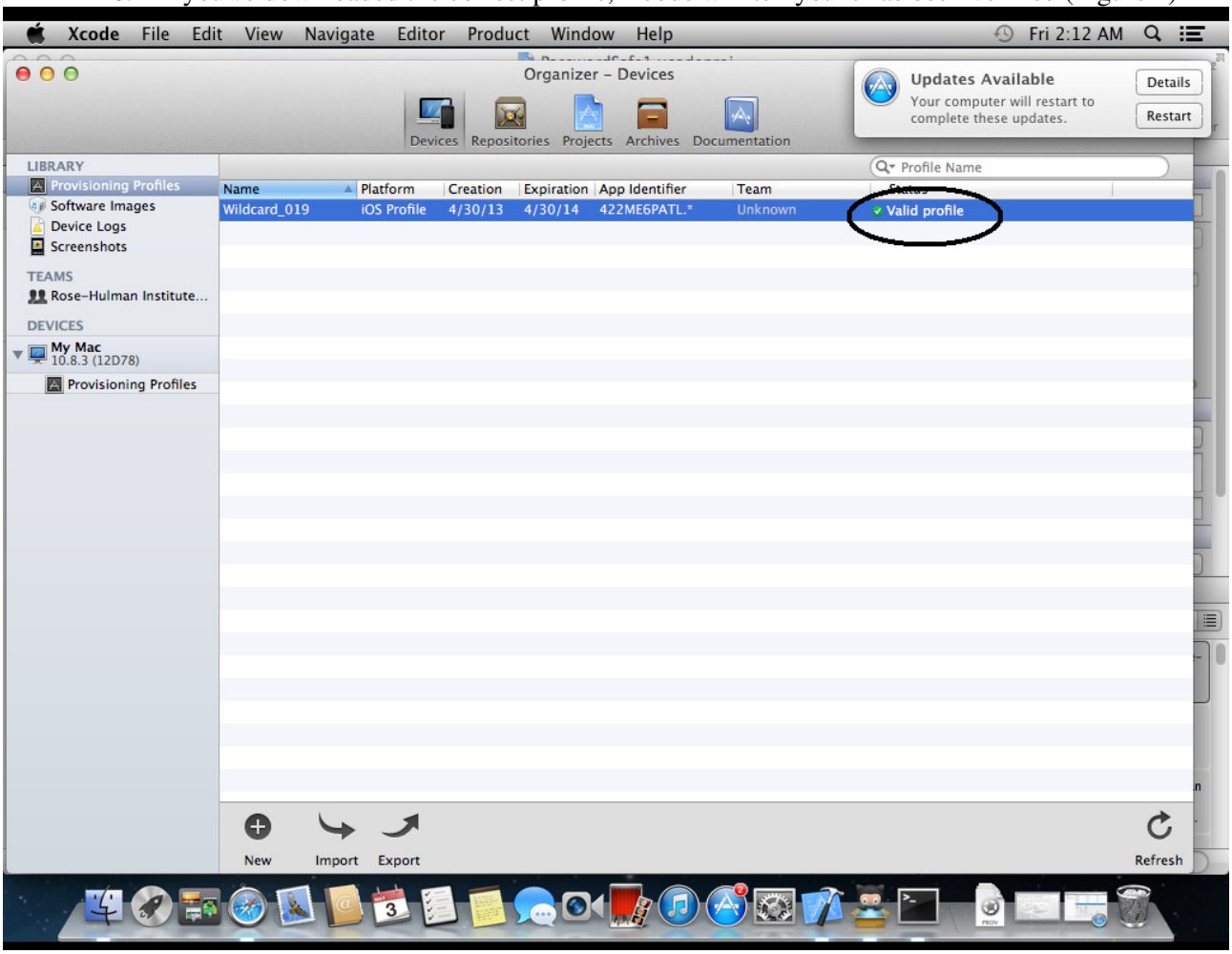


Figure 1: If you downloaded the correct provisioning profile, you should see “Valid profile” as it is circled above.

3. Make sure the target device for building is "iOS Device" (Figure 2).
    1. Look for the button to the right of "Stop." Click it and select "iOS Device".

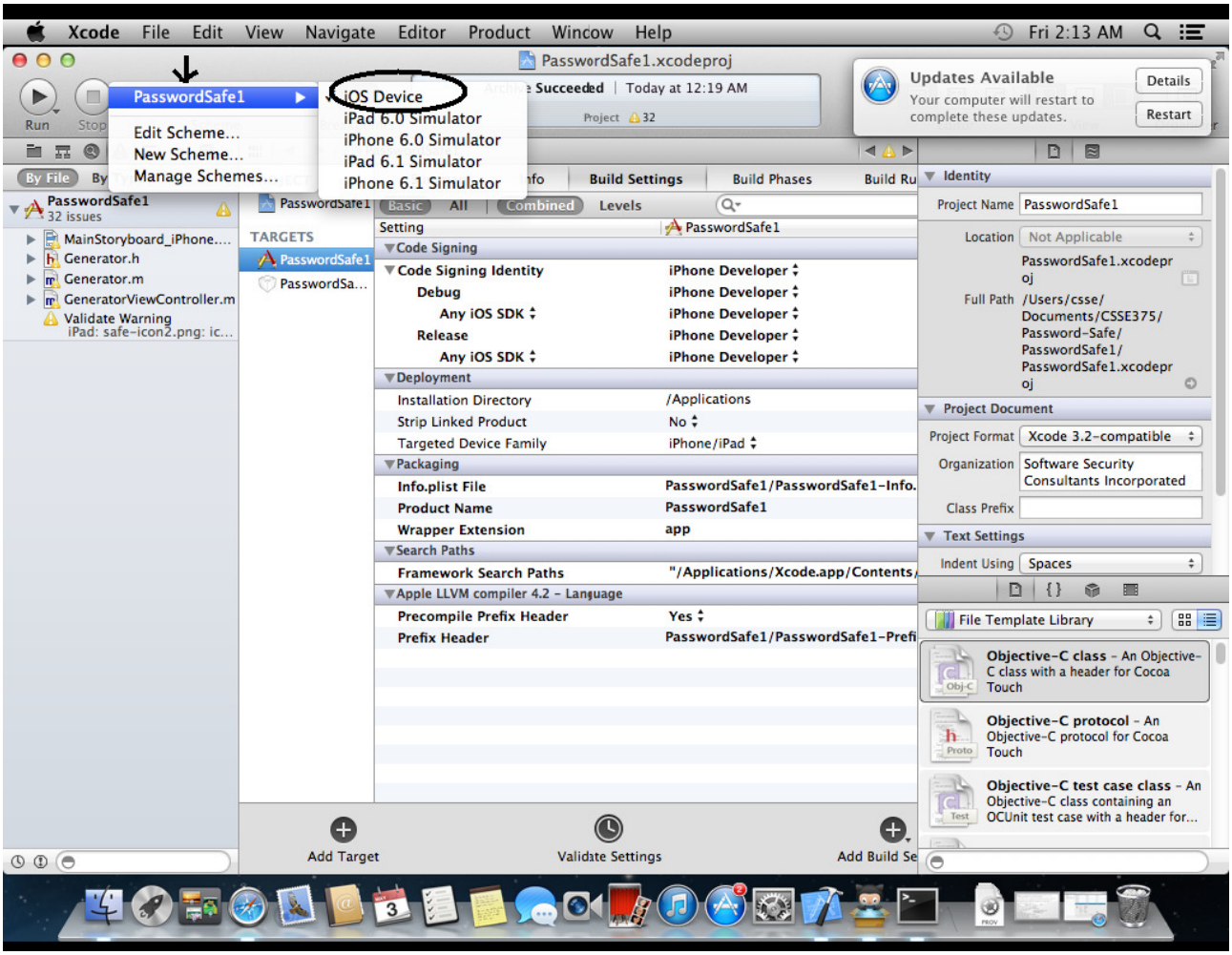


Figure 2: Select the target by clicking the button designated by the arrow above, then select "iOS Device."

4. Press "Cmd + B" to build the project.
5. Go to Menu -> Product -> Clean (Figure 3).
6. Go to Menu -> Product -> Archive (Figure 3).

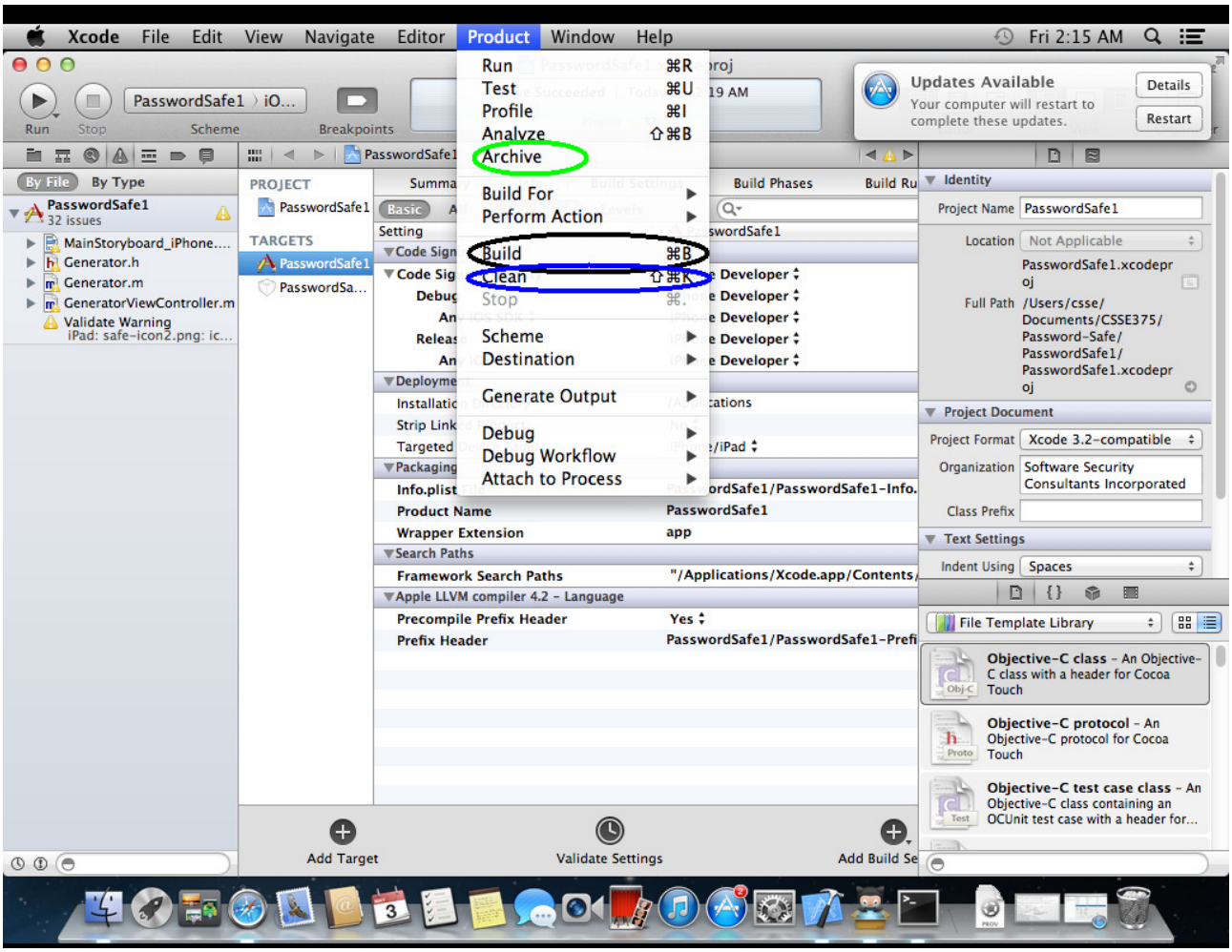


Figure 3: Locations for "Build", "Clean", and "Archive" under Product in the top menu.

7. When the overlay window pops up, click "Distribute..." on the right.
8. Select the "Save for Enterprise or Ad-Hoc Deployment" radio button.
9. Click "Next" twice.
10. In the box that pops up, type in a name next to "Save As", choose a location next to "Where", then click "Save."
11. Download iTunes if you do not have it.
    1. Go to www.itunes.com.
    2. Click the "Download iTunes" button.
    3. Click the "Download Now" button on the left and save where you want.
12. Double click the .ipa Application file. It will appear under the Apps category in iTunes (Figure 4).

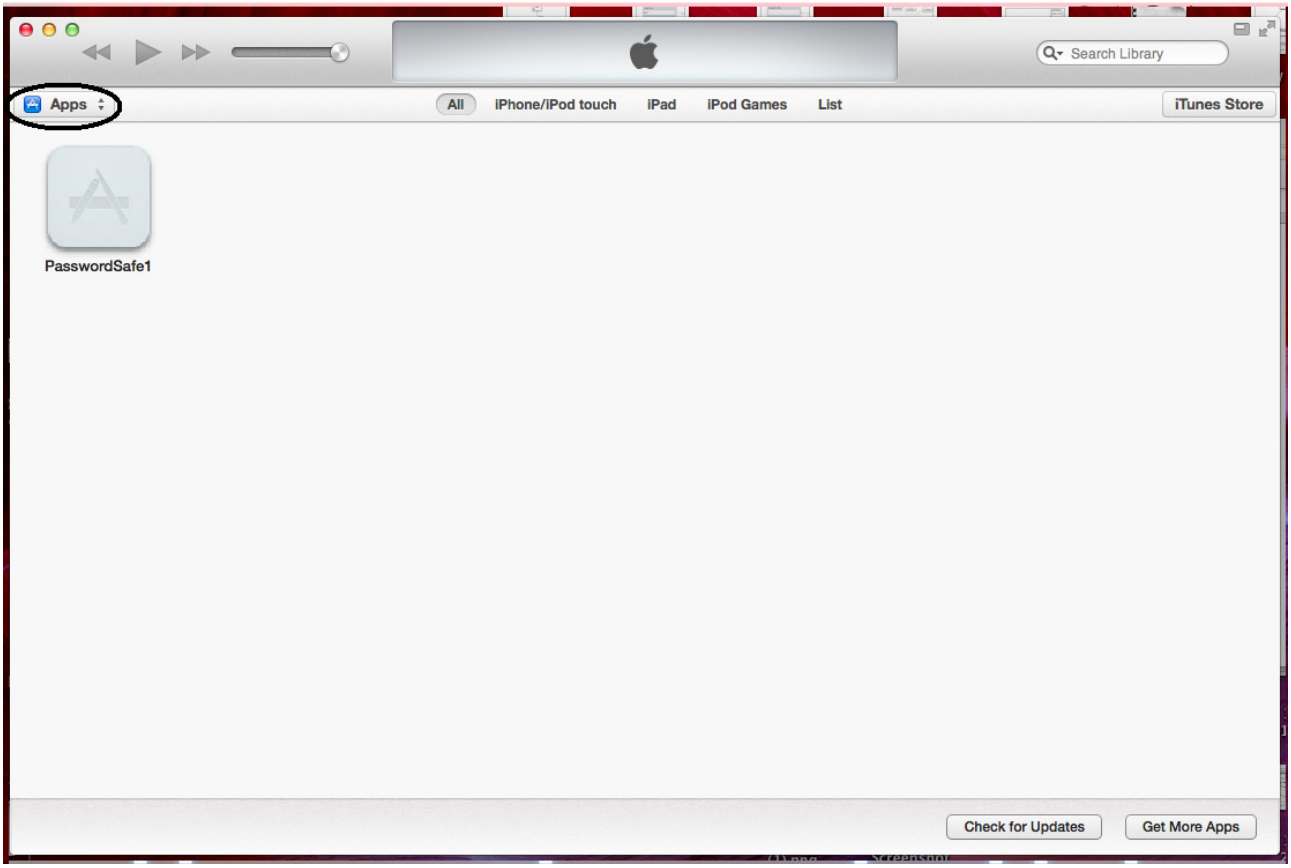


Figure 4: Double-clicking the .ipa file will launch iTunes, and you can view the app in the Apps category.

13. Connect your iOS device and let it sync.
14. If iTunes does not change its window to view your iOS device, click the "iPod" or "iPad" button at the top right, then click the "Apps" tab at the top (Figure 5).

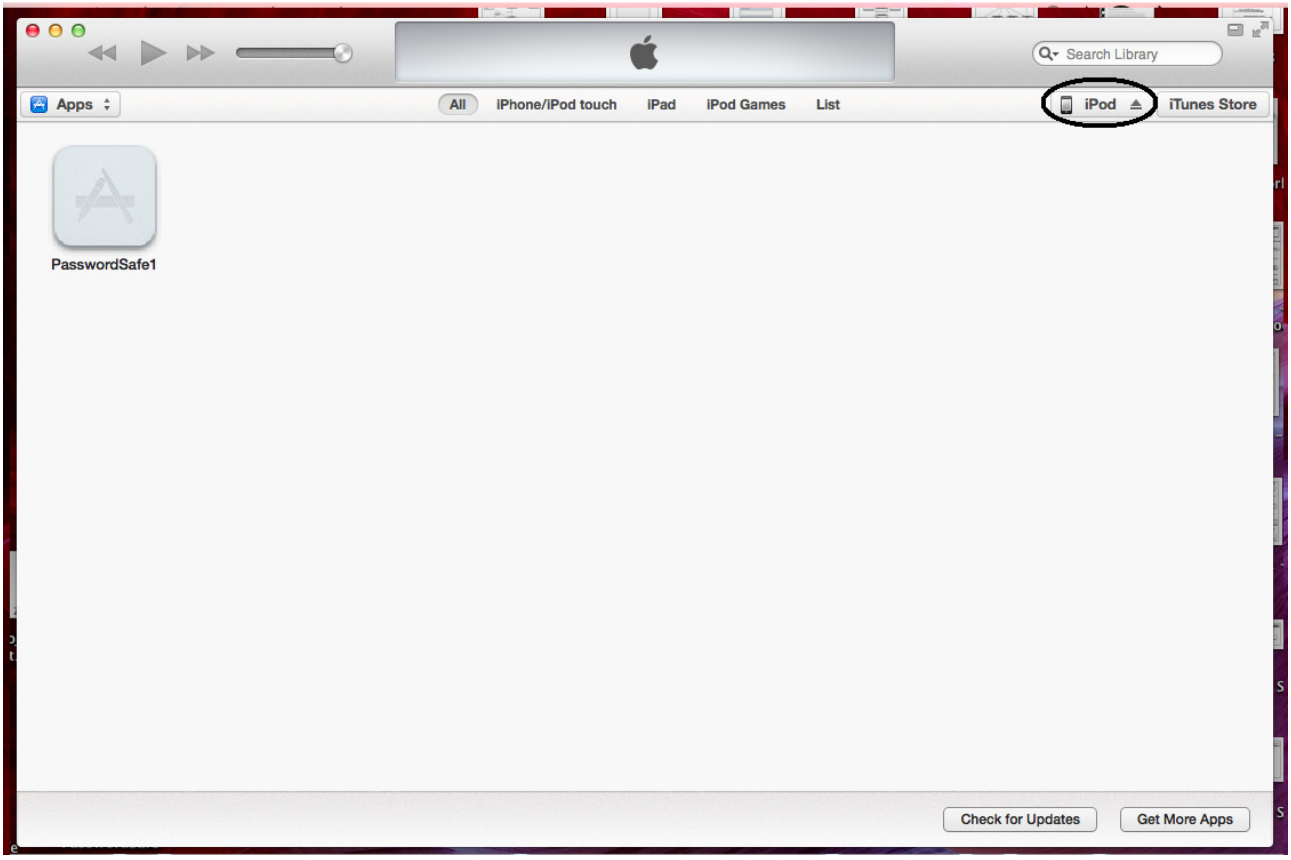


Figure 5: When your iOS device is connected, click the button that is circled above.

15. Click the "Install" button next to the PasswordSafe1 field, then click the "Apply" button that appears on the bottom (Figure 6).
16. When finished, click the Eject icon at the top left, next to the name of your device.

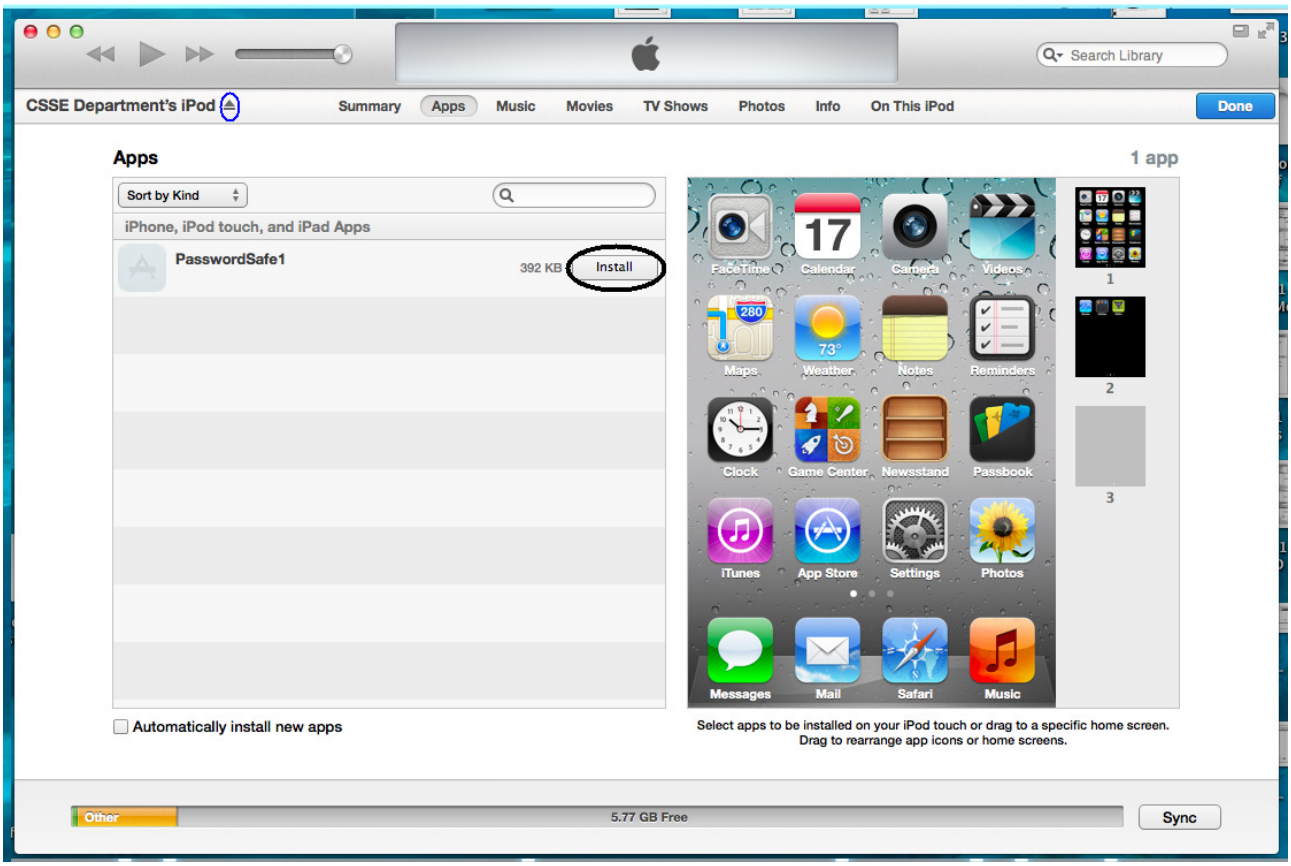


Figure 6: Click the "Install" button circled in black, then click "Apply" in the bottom right. When done, click the button circled in blue to eject the device.

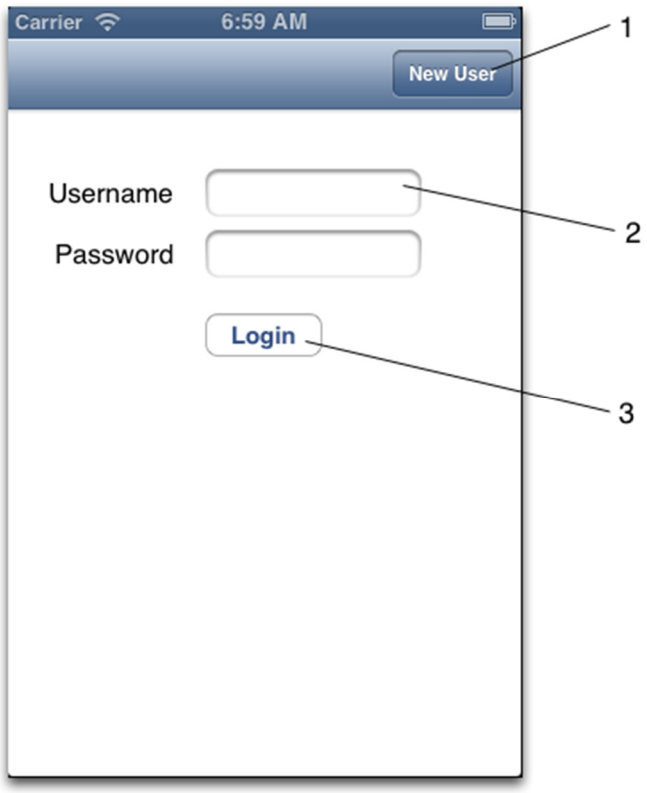
User Guide

This section provides a detailed guide on how to use the application. It will provide a screenshot of every screen on the application, along with a listing of all the buttons and fields on the screen. It will then take you through common actions to run on every screen.

Explanation of Screens

Login Screen

This is the main screen of the application that allows users to log into the application.



Feature Listings

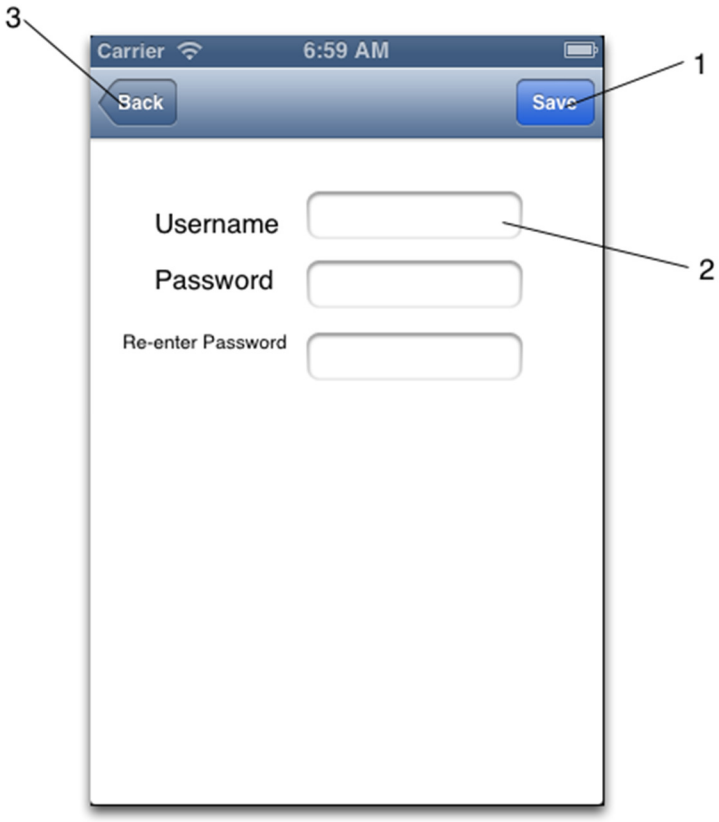
1. New user button
2. Text entry fields
3. Login button

Log in

Enter your username and password into the text field. After this is finished press the log in button. This will either pop up a message is something is wrong, or take you to the notes screen.

New User Screen

This is the screen to add a new user to the application.



Feature Listings

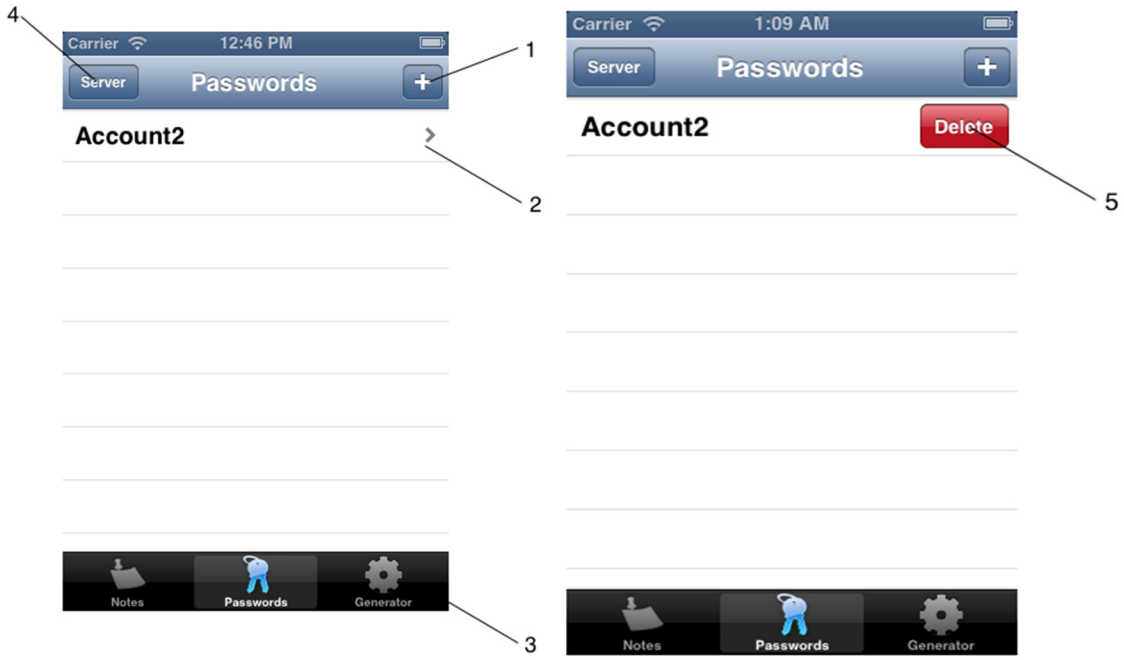
1. Save Button
2. Text entry fields
3. Back button

Add New User

Fill in the username field; once completed, type the password in and confirm it (2), upon completion press the save button (1). If there is a problem there will be a pop-up message. If there is no pop-up message, press the back button (3).

Password Screen

This screen is used to view all of the created passwords on a high level.



Feature Listings

1. Add Password Button
2. Go to Edit Password Screen for that particular password
3. Tab Bar
4. Edit server setting button
5. Delete password button

Viewing an individual password

You can view and edit an individual password's information by tapping on that password (2). This will take you to the "Edit Password Screen".

Deleting a password

You can choose to delete a password by swiping to the right on that password's cell. This will display the delete password button (5). If you then tap on that button, the password will be deleted.

Creating a new password

You can create a new password by tapping on the create button (1). This will take you to the "Create Password screen".

Other actions that can be performed

You can edit the information for the server while on this screen by tapping the edit server button (4). This will change what information is in this table. Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

New Password Screen

This screen is used to create a new password.

Feature Listings

1. Save Button
2. Text entry fields
3. Tab Bar
4. Go back to password screen

Creating a new password

The first step is to fill in the text entry fields (2) with the information you want saved. You do not have to have every field filled in to save a password. Once complete tap on the Save Button (1). This will take you back to the "Passwords Screen". If at any point you wish to go back just tap the Go back to password screen button (4). This will return you to the "Passwords Screen".

Other actions that can be performed

Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

Edit password screen

This screen allows you to view and change information about a password.

Feature Listings

1. Save Button
2. Text entry fields
3. Tab Bar
4. Go back to password screen

Edit a password

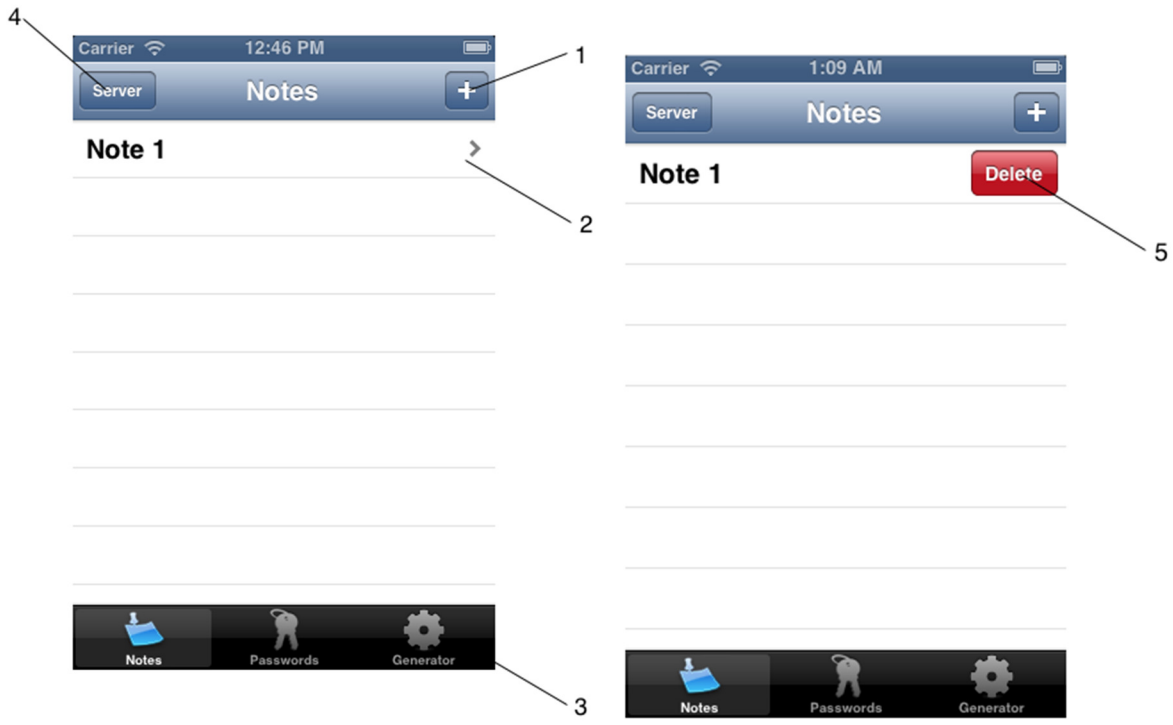
You can make changes to any of the text fields by simply tapping the field and changing the information as desired. If you made changes and want them saved just tap on the save button (1). This will save the changes and return you to the "Passwords Screen". If you would like to go back to the "Passwords screen" simply tap the go back to password screen button (4). This will return you to the "Passwords screen".

Other actions that can be performed

Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

Notes Screen

This screen is used to view all of the created notes on a high level.



Feature Listings

1. Add Note Button
2. Go to Edit Note Screen for that particular Note
3. Tab Bar
4. Edit server setting button
5. Delete Note button

Viewing an individual Note

You can view and edit an individual Note's information by tapping on that Note (2). This will take you to the "Edit Note Screen".

Deleting a Note

You can choose to delete a Note by swiping to the right on that Note's cell. This will display the delete Note button (5). If you then tap on that button, the Note will be deleted.

Creating a new Note

You can create a new Note by tapping on the create button (1). This will take you to the "Create Note screen".

Other actions that can be performed

You can edit the information for the server while on this screen by tapping the edit server button (4). This will change what information is in this table. Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

New Note Screen

This screen is used to create a new note.

Feature Listings

1. Save Button
2. Text entry fields
3. Tab Bar
4. Go back to Note screen

Creating a new Note

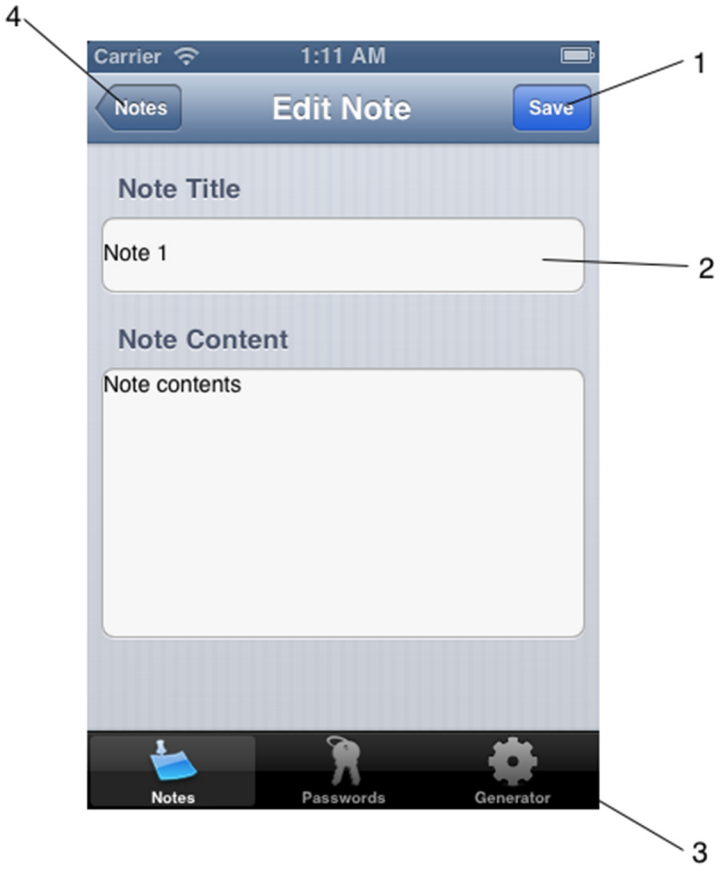
The first step is to fill in the text entry fields (2) with the information you want saved. You do not have to have every field filled in to save a Note. Once complete tap on the Save Button (1). This will take you back to the "Notes Screen". If at any point you wish to go back just tap the Go back to Note screen button (4). This will return you to the "Notes Screen".

Other actions that can be performed

Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

Edit Note Screen

This screen is used to edit and view an already created note.



Feature Listings

1. Save Button
2. Text entry fields
3. Tab Bar
4. Go back to Note screen

Edit a Note

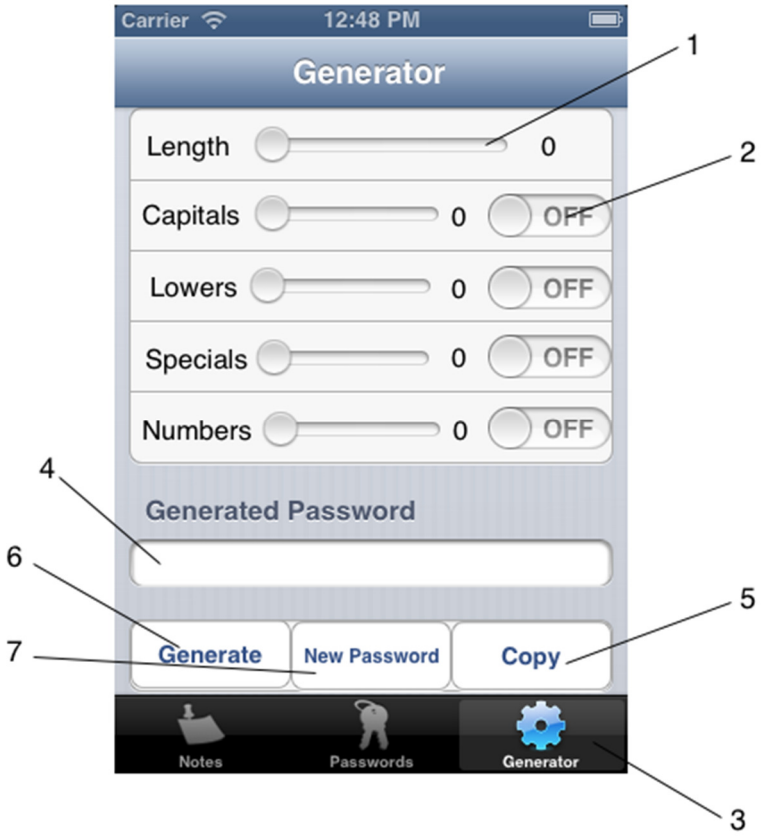
You can make changes to any of the text fields by simply tapping the field and changing the information as desired. If you made changes and want them saved just tap on the save button (1). This will save the changes and return you to the "Notes Screen". If you would like to go back to the "Notes screen" simply tap the go back to Note screen button (4). This will return you to the "Notes screen".

Other actions that can be performed

Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

Generator Screen

This screen is used to generate strong and random passwords. This is done through setting different fields to minimum values.



Feature Listings

1. Length Slider
2. Field Sliders and Switches.
3. Tab Bar
4. Generated Password Field
5. Copy Button
6. Generate Button
7. Create New Password Button

Generating a Password

The first step in generating a password is setting the length slider (1). You can slide this to any value less than 64. The next step is to turn any of the fields on. If you want a password with only capital letters, you must first turn the capital switch (2) on. A slider field slider value (2) of 0 means that you do not car how many of that type of character are in the password.

Once you have your desired combination of length and fields you tap the generate button (6), and the generated password will be displayed in the generated password field (4). By tapping on that field you

can edit the generated password. Also, you can regenerate a password as many times as you would like, you do not have to reset the fields.

Copy the generated password to the clipboard

Once you have a generated password, you just have to tap the Copy button (5) to copy the generated password to the global clipboard.

Creating a new password object with the generated password

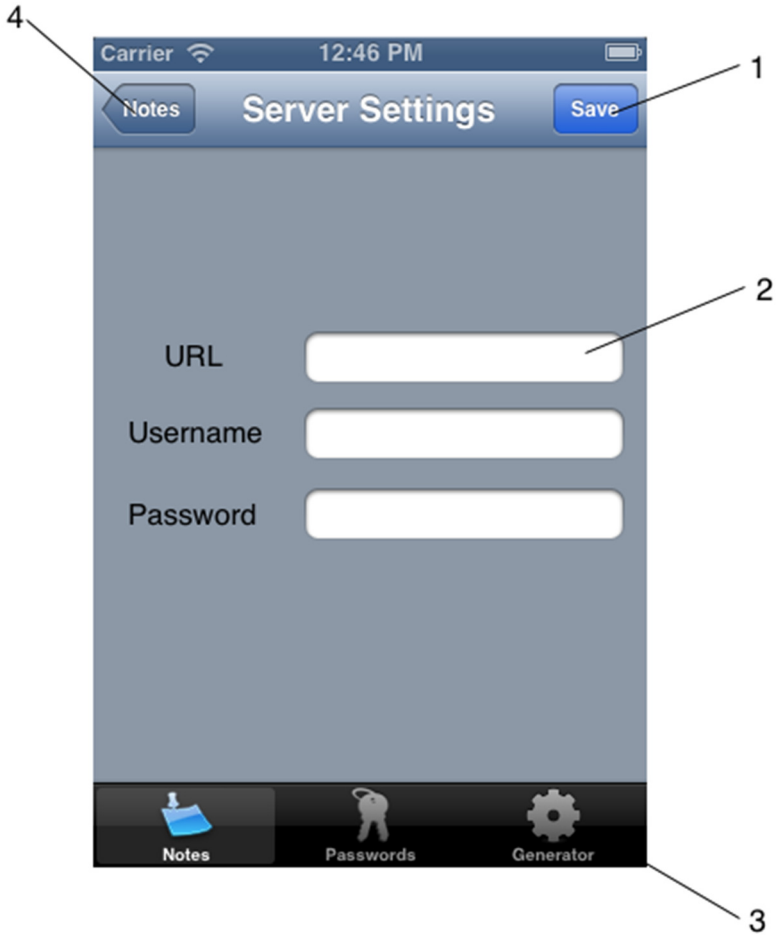
Once you have generated password, if you want to save that generated password to the application, just tap the New password button (7). This will take you to the "New Password" screen and allow you to setup all the information associated with the password.

Other actions that can be performed

Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

Server Settings Screen

This screen is used to enter the information needed to connect to a WebDAV server.



Feature Listings

1. Save Button
2. Entry Field
3. Tab Bar
4. Back Button

Connecting to a WebDAV Server

First enter the URL of your WebDAV server in the URL Field (3). Next, enter your username and password in the remaining fields. Once the information is entered, press the save button in the top right hand corner of the screen (1).

Other actions that can be performed

To return to the previous screen, press the button in the top left corner of the screen with the name of the previous screen (4). Additionally, the Tab at the bottom of screen (3) can be used to jump to the Notes, Password, or Generator Screens.

An explanation of WebDAV URL formatting

	Base URL	Extended	Location of application	Application folder
URL Structure	http://<host>:<port>	/<crx-webapp-path>	/repository	/<workspace>
Example	http://localhost:7402	/crx	/repository	/crx.default
Description	Host and port, on which CRX runs	Path for the CRX repository webapp	Path to which WebDAV servlet is mapped	Name of the workspace mapped throught this path.

Full URL from example:

http://localhost:7402/crx/repository/crx.default

Common Actions Performed

Logging in for the first time

If you do not have a user account:

1. Tap the “New User” button in the top right (see below).

You'll be taken to a new screen (see below). From there:

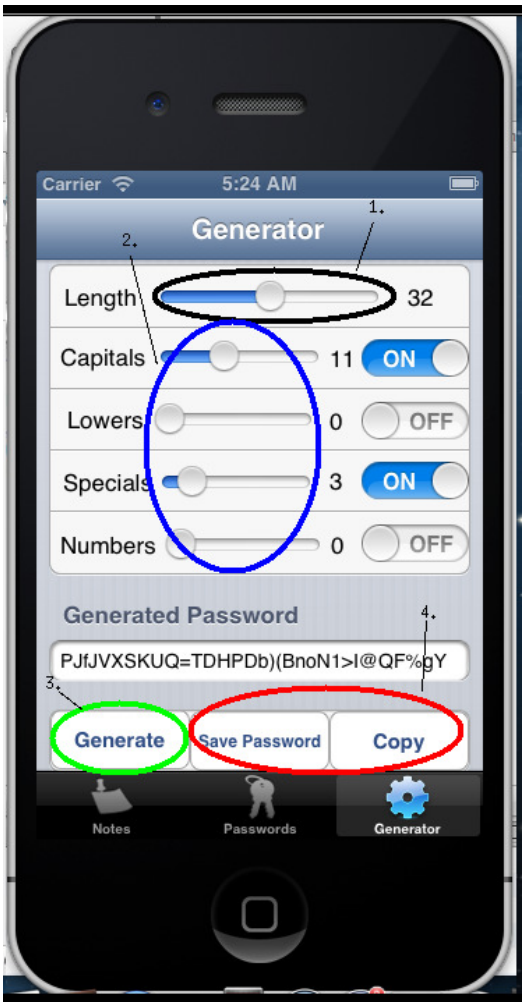
1. Enter a new username.
2. Enter a new password. You'll need to re-enter it exactly.
3. Tap the “Save” button.

Setting up Server information

1. Once you are logged in, tap the “Server” button in the top-left corner of the screen.
2. Type in the URL of the server, as well as the Username and Password for the server.
3. Tap the “Save” button in the top-right corner of the screen.

Generate a Password

1. In the Generator screen, adjust the Length slider to set the length of your new password. You may have up to 64 characters for the password.
2. If you wish to specify a specific number of characters that are Capital letters, Lowercase letters, Special characters (such as #, %, *, $, etc), or Numbers, switch the corresponding type to "ON" and use the slider to set the length. The generator will choose how many characters of each type that is set to "OFF" to fill in the password.
3. Tap the "Generate" button to generate your new password.
4. If you wish to save your password, tap the "Save Password" button. If you wish to copy your password to paste in another application, tap the "Copy" button.



Creating a Note

1. In the Notes screen, tap the "+" button in the top-right corner of the screen.
2. Type a new name for the note in the "Note Title" text box.
3. Type the content of the new note in the "Note Content" text box.
4. When finished, tap the "Save" button in the top-right corner of the screen.

Maintenance Guide

Development Environment

The PasswordSafe project was developed in Xcode 4.5.2. GitHub was used for version control. The repository is located at https://github.com/SoftwareSecurityConsultantsIncorporated/Password-Safe.

Build Instructions

If you do not have the project, download it from https://github.com/SoftwareSecurityConsultantsIncorporated/Password-Safe. Once you have the project, double-click PasswordSafe.xcodeproj in the Finder to open the project in Xcode. In Xcode, select iPhone Simulator as the SDK and then press the Build button to run the application in the simulator.

Planned Changes

- Develop a version that runs on Mac OS X
- Implement a syncing algorithm that only sends changed information to and from the server
- Additional data storage options, such as credit card information or contact information
- Security questions

Troubleshooting

Common Problem	Potential Solution
Blank screen on launching application	The application may be having problems connecting to the server. Check the server settings to make sure that they are valid.
Syncing is not working as you think it should	If you are running the application in the simulator, you need to exit the application on the simulator by pressing the iPhone home button. The local data gets saved to the file when the application is exited, which does not register when you just quit the simulator.

Design of the Application

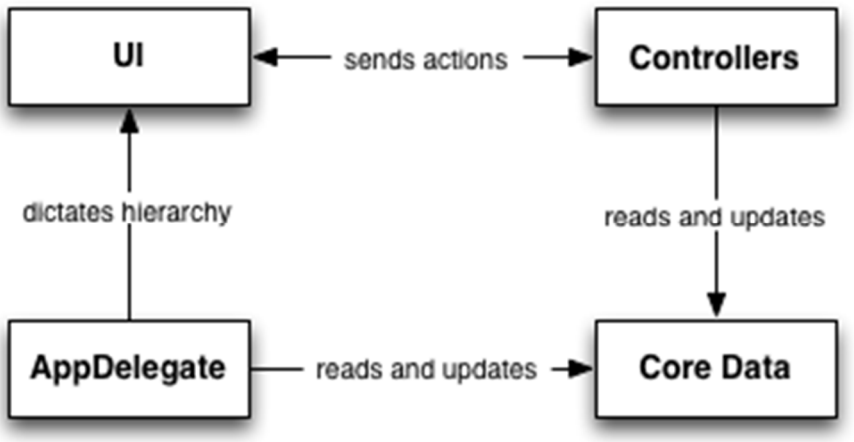
The design of the application follows a model view controller pattern. This design pattern is the standard for most of Apple's products.

Design Class Diagram

This section contains a number of design class diagrams that present the layout of the application in different ways. Each diagram should allow for a more clear understanding of how the system works together.

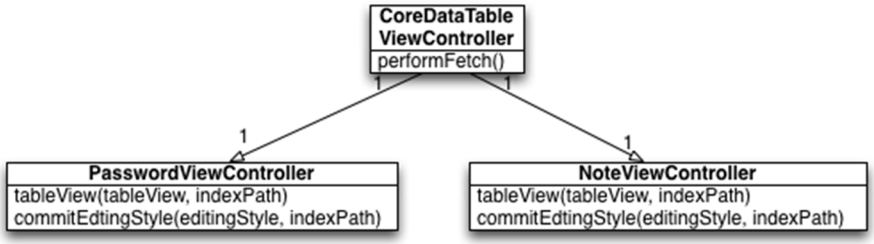
GUI, App Delegate, and Controller Relations

The GUI in this application is created using a storyboard so overall there is minimal code seen from the GUI. The GUI talks to the controllers directly through mapped actions. These actions are handled by the controllers. The controllers also talk to the AppDelegate class to establish connections to other controllers. Finally, if needed, controllers will make changes to Core Data.



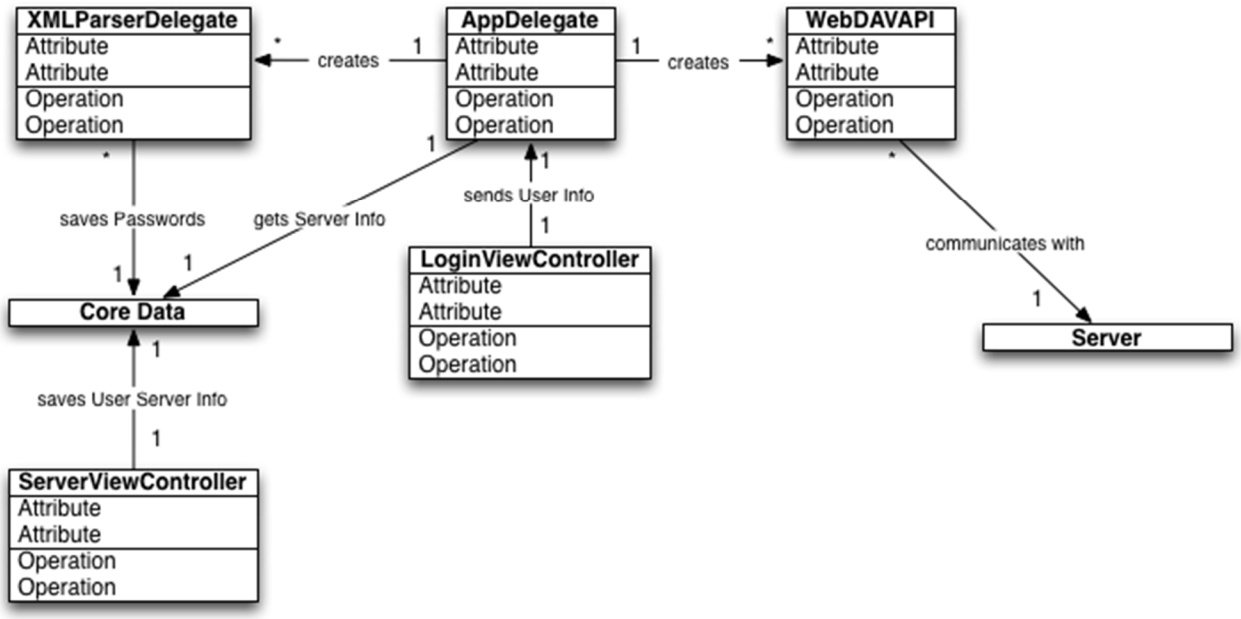
Core Data Table

Both PasswordViewController and NoteViewController are subclasses to the CoreDataTableViewController object. These objects basically rewrite an NSFetchRequest. This allows the table view to easily communicate with Core Data, and still be a Controller.



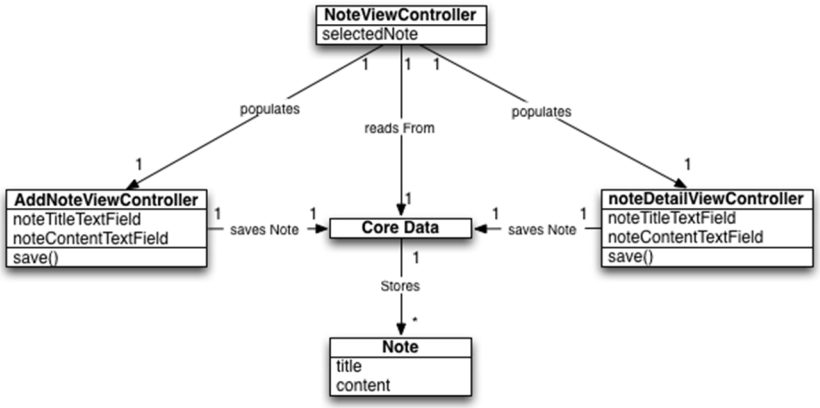
WebDAV API

This DCD shows the relationships between the classes that are affected by syncing. The actual syncing process is explained later in this section, the relationships between the objects is setup to allow for minimal calls to the server to allow for quicker syncing speeds.



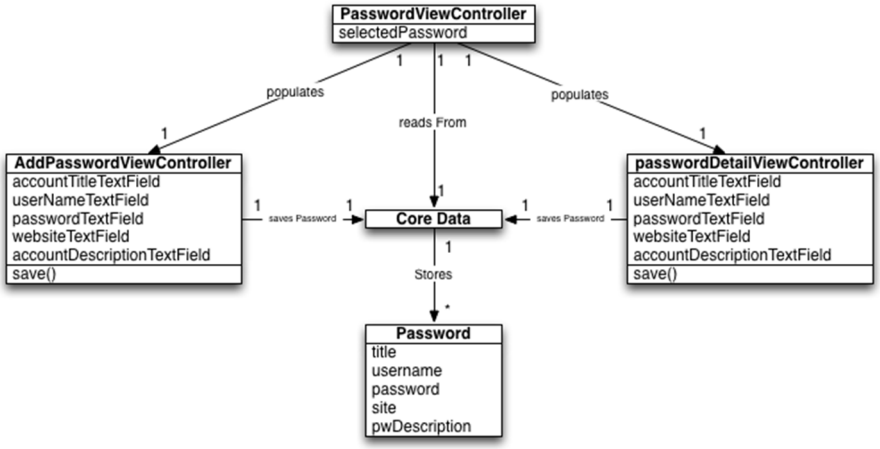
Notes

This DCD shows the relationship between the classes that handle the different CRUD operations of a note.



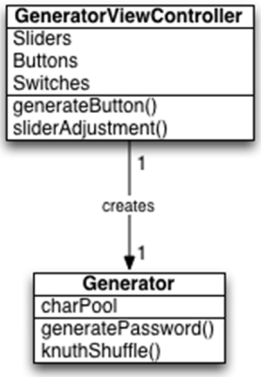
Password

This DCD shows the relationship between the classes that handle the different CRUD operations of a password.



Generator

This DCD shows the relationship between the generator classes. The controller handles all calls to the generate class.



Generator

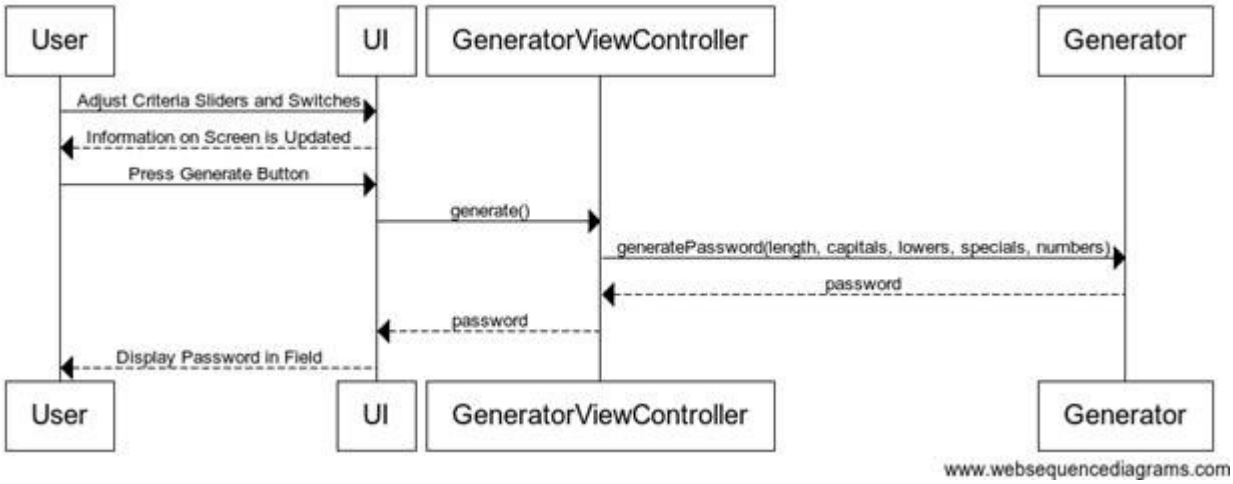
This class stores the fields and methods needed to randomly generate a password given the length of the password, the number of each type of characters, and whether to include each type of character.

The high level behavior is simple. The algorithm will first complete all the requirements given, then will fill the rest of the password with random characters until the length is met. It then randomizes the order of these characters using an algorithm called the Knuth Shuffle.

There were design decisions made when coding the algorithm. First we draw randomly from a single char pool instead of dividing the characters into separate pools based on type. This was done to increase the randomness of the algorithm, although there is a chance that the algorithm never chooses the character it needs and gets stuck in an infinite loop.

Another decision was to include switches related to each type of character on the generator screen instead of just turning them off when set to zero. This was because there was a case where a user would want to include a type of character, but wouldn't have a specific amount they desired. This design covered as many cases as possible and gave the user the most control over the generated format. This did lead to the generate function in the generator class having a large number of parameters, which could be refactored. Also there is a lot of duplicated code in the generator class that could be abstracted out.

Generate Password Sequence Diagram



Core Data Objects

The application uses core data to persist data. Core data generates objects to work with in other classes. Core data objects are essentially RESTful objects for a database. They are created, updated, renamed, and deleted through of method calls. You first create a NSFetchRequest then you set the Entity to that of the object you plan to use, finally you call executefetchrequest() on the NSFetchRequest object. This process writes changes to the core data table specified by the entity.

Note

A note object has title and text fields. The title and text fields are of type NSstring. Core data allows them to be of any length. When working with a noted object you are limited to basic SQL like calls. Notes are used inside of the NoteViewController, AddNoteViewController, and EditNoteViewController classes.

Password

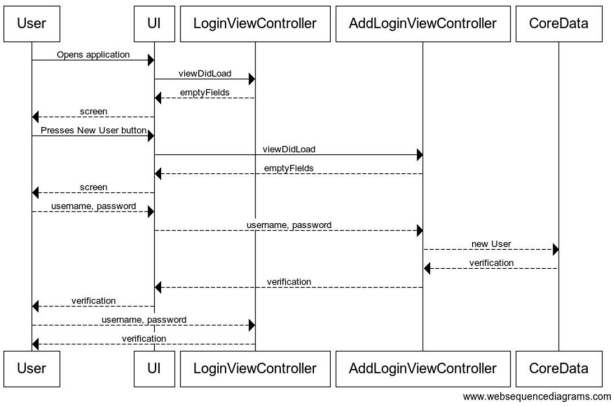
A password object has a title, username, password, website, and description fields that are NSString types. Like a note they are limited to basic CRUD operations. Passwords are used inside of the PasswordViewController, AddPasswordViewController, and EditPasswordViewController classes.

User

The User class allows the storage of user information in Core Data. It has fields to store the user's master account username, master account password, server url, server username, and server password.

When a user first uses the application, he will create a new user account. The AddLoginViewController adds this account information to a new User in Core Data. When the user logs in to the application, the LoginViewController will set the active user in the AppDelegate to that User. This allows the user's information to be accessed by the WebDAVAPI so that it has a url and credentials for uploading and downloading. It also allows the application to update the user's server settings from the ServerViewController. When the server settings are changed, a check is run to see if the settings are valid. If they are not, they will not be saved. Once valid settings have been entered, the user will be able to sync data between his device and server.

Creating a new User Sequence Diagram



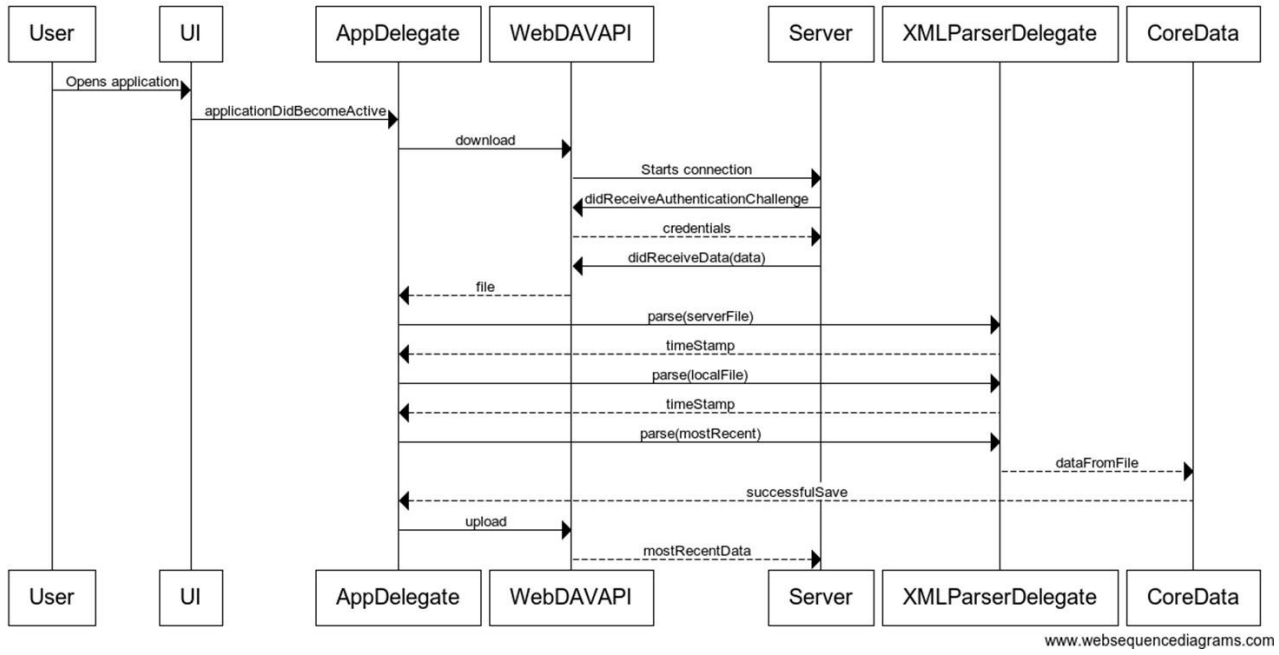
WebDAV Server and API

The WebDAVAPI class is used to manage connections to the server. It contains delegate methods that interact with the server as a connection is underway as well as methods that allow the AppDelegate to begin downloading and uploading.

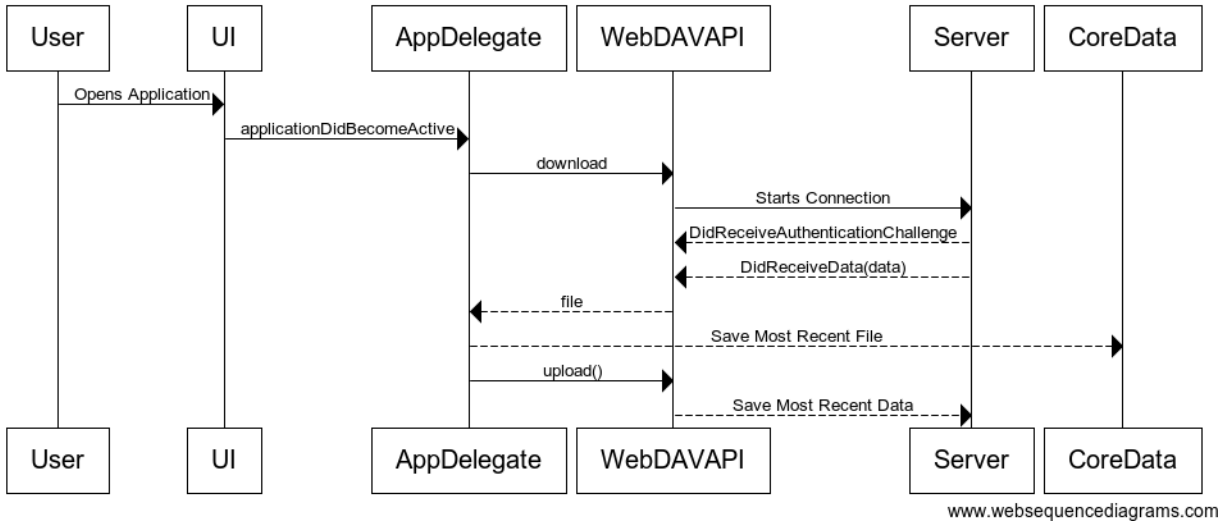
The download and upload methods start a new connection with the server and uses the url, username, and password from the logged in user. The delegate methods then get called on the WebDAVAPI by the server when various events occur. When the server needs credentials, it calls the didReceiveAuthenticationChallenge method, which supplies the user's stored credentials. While downloading, when the server sends data, it calls didReceiveData, which writes that data to a local file. The class is mainly responsible for handling establishing connections with the server and for syncing the data from the server. How these functions actually work are explained in the sequence diagrams later in the section.

The WevDAVAPI was chosen to be initialized in the AppDelegate because it was desired to call download and upload on the API when the user logs in.

Setting up Server Credentials Sequence Diagram



Auto Sync Sequence Diagram



Core Data Object Controller

A Core Data Object controller is a superclass for the Note and Password view controller. This class is from a class at Stanford. This class mostly just copies the code from NSFetchedResultsController's documentation page into a subclass of UITableViewController. This class allows for easy interaction with core data from the view controller.

UI Controllers

The controllers in the application handle all of the action calls from the GUI. They also handle placing information in core data along with creating some other classes. These controllers are fairly large because of our choice to use a storyboard to generate the GUI. This caused the mapping of actions along with other tasks to fall to the controllers to handle. There were significant attempts to cut down on the size of the controllers.

User Interface Design

In prototyping the interface of the application, we came up with three initial designs:

The key concept we wanted to capture was to list the accounts that the application kept track of as well as the sync status of those accounts, and display them at the same time. The first is a simple list view with Auto Sync check boxes. The second is a grid view with ON and OFF switches to change sync statues, and the third is another list view using colors to display sync status.

After collecting user feedback and discussing options with our client, we decided to base our final UI design on the rightmost prototype. This design was the simplest and easiest to read. We also decided to change to a different color scheme to maintain consistency.

Our final design maintained the overall same look-and-feel. Our color scheme is the standard white and blue colors of iOS applications. Most of the buttons are different shades of blue and we list the notes and accounts in a simple list view format. The only time this format is not used is in the password generator, where we use sliders to specify the length and the number of certain types of characters in the new password. One major change in our final design is that we do not display the sync status of the account in the list view format.

The basic hierarchy of the various screens is setup using a storyboard. This includes segues and linking buttons to various function within controllers. This makes it difficult to understand the links between the UI and the controllers as well as making changes. More complex functionality exists within some controllers because of screen transitions and logic that couldn't be done in the storyboard.

Glossary

API - application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other

Data flow diagram - graphical representation of the "flow" of data through an information system, modeling its *process* aspects

Encription - the process of transforming information (referred to as plaintext) using an algorithm (called a cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key

Ground Truth - the state at which a device can be called synched. It is the state where the file on the device and the file on the server match exactly with data that is set to synced.

Synced - when the device is in the “ground truth” state

Use Case - a list of steps, typically defining interactions between a role (known in UML as an "actor") and a system, to achieve a goal

WebDav - an extension of the Hypertext Transfer Protocol (HTTP) that facilitates collaboration between users in editing and managing documents and files stored on World Wide Web servers

References

GitHub site - wwww.github.com
Omni Group site- www.omnigroup.com/
OnePassword site- www.agilebits.com/onepassword
Wallet site- www.moxier.com/wallet
Managing Software Requirements: A Use Case Approach, Second Edition, by Dean Leffingwell and Don Widrig