

UI Software Exploration Project

Maxwell Keleher

Professor Robert Biddle

HCIN 5200

October 31st 2021

Domain and Requirements:

The contemporary internet is rife with accounts for various sites and services. The simplest set of credentials to access a site is a unique username and a secret password. A password manager is a tool that allows users to store their credentials for any of their accounts. However, there is an important tension within authentication: users need to remember the password they have set, but the password should be difficult for someone else to guess. Using your name or birthday is an option that is easy to remember but would be equally easy for someone to guess, a randomized set of letters and numbers would be very difficult to guess but equally difficult to remember. Password managers therefore alleviate the need to memorize many complex passwords and with password generation utilities can encourage users to select unique, secure passwords for all of their accounts.

There are many different iterations on this basic structure but for the purposes of this project I will be focusing on what I consider the most essential features of password managers: password generation and management of stored accounts. Managing the accounts can be further broken down into: creating/editing a set of credentials, exploring the stored accounts, and organizing the stored accounts. There are also many important security considerations when discussing the design and implementation of password managers but for the purposes of this project I will just focus on the interfaces of the manager. I assume that the password managers ensure the appropriate degree of security and that users will access the manager using a master password.

In many available password managers, browsing of stored accounts is largely treated as an unimportant task and focus is placed on prompting users to autofill login forms when they are on the page. In my designs, I put much more emphasis on the actual organizing and browsing of the account credentials. I provide more explanation for these design decisions in the perspective analysis.

A critical aspect of creating a new set of account credentials is choosing an appropriate password. Password generating utilities in password managers often involve merely giving users a long string of random numbers, letters and symbols. This form of password is both difficult to guess and, assuming it is a long string, would be relatively difficult to crack by brute force. Some password managers expand this functionality by giving users control over the variables when creating the passwords, which is the approach I will follow for this project.

Styles:

The first style I chose was direct manipulation. In his discussion of direct manipulation, Shneiderman spends a lot of time highlighting how the style can make interfaces more fun (Shneiderman, 1988, 1997). I hope that this inherent degree of fun will alleviate the frustration that many might feel when interacting with passwords. Moreover, Shneiderman explains that direct manipulation interfaces promote exploration of systems (Shneiderman, 1988, 1997), likely making it appealing to users who have little to no experience using password managers. The responsiveness of the interfaces also makes it easier for users to immediately understand how different interactions affect the results. In the context of generating passwords, this would mean that users would have an easier time understanding how variable changes (length of password or variation of character set) affect the strength of the password. The view that allows users to explore the stored accounts could use a spatial layout similar to the Desktop of most operating systems. This might make the password manager more accessible and familiar to users who have little to no experience using them. Finally, Shneiderman points out that direct manipulation interfaces inspire a sense of confidence amongst users. I believe that generating a sense of confidence and a desire to explore would encourage users to dive deeper into digital security and to take more responsibility for the security of their accounts and devices.

I was motivated to explore the command language style, partially because I was interested in how it contrasts with the direct manipulation style. It is, in a sense, an inversion of direct manipulation; rather than responsive visual interaction it focuses on quickly issuing written commands. Command language more or less struggles where dynamic manipulation thrives but the inverse is also true. In his description of command language interfaces, Shneiderman identifies knowledgeable frequent users as a group who particularly benefit from the style (Shneiderman, 1988). Anyone who works with computers finds themselves needing to work with passwords fairly frequently, so it would not take long for users of password managers to

become knowledgeable. It might therefore benefit users if they are able to quickly and efficiently achieve their goal without fiddling with precise mouse controls. Another important benefit of command language interfaces is related to the fact that these interfaces typically exist on a command line interface such as the terminal. Though command line interfaces lack the immediate responsiveness of a direct manipulation interface, the console will keep a record of the executed commands and the resulting output. This is useful since a user would be able to easily compare the results of different password generating queries or view multiple organizations of their stored accounts at a time.

Design:

Explore

Generate

Password:

xtZxe4ZjzNGPqX4j3MyZ

Password strength:

Length

20

How many capital letters?

1 12

How many numbers?

2 3

How many symbols?

0 0

Figure 1: Direct manipulation password generator

The direct manipulation design has two different screens: the password generator and then the account browser/editor. To generate a password users will have to drag the first slider so that they reach the desired password length. Then the other sliders are used for manipulating the size of the capital letter, number, and symbol character set respectively. These character sets are used to match the requirements given when creating passwords (i.e. passwords must be at least 8 characters long, and contain a minimum of 1 number and capital letter). As the users drag the sliders a random password will appear in the box at the top and the bar along the bottom will fill out to reflect the strength of the chosen password. The icon to the right of the box will copy the generated password to the user's clipboard.

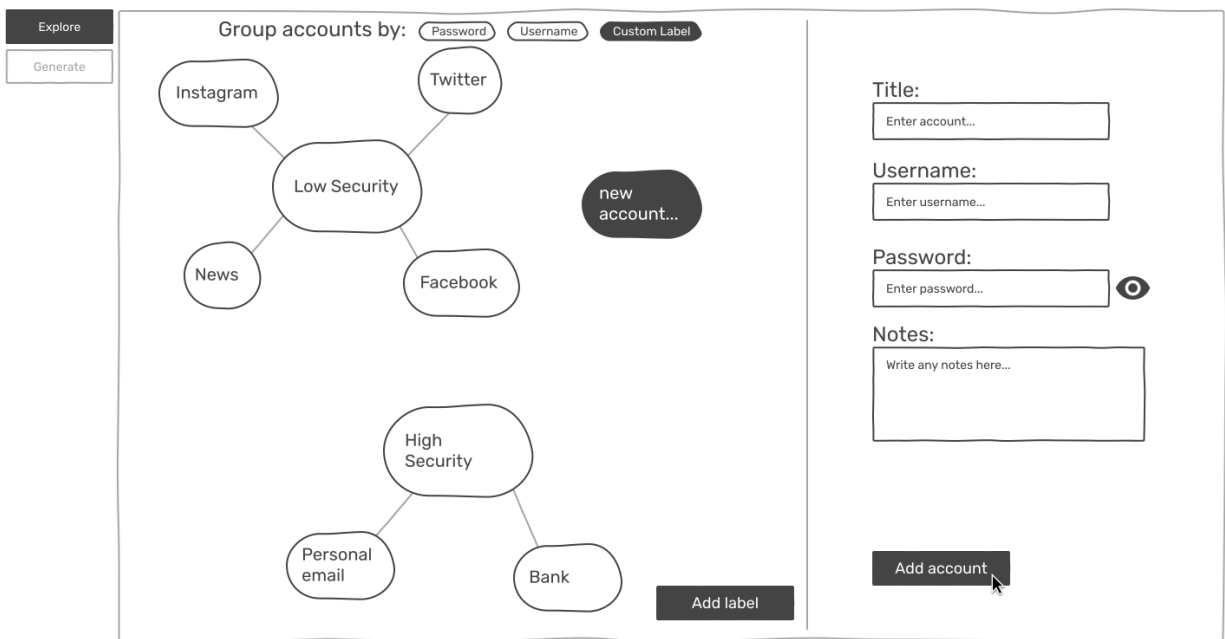


Figure 2: Direct manipulation password explorer custom label view.

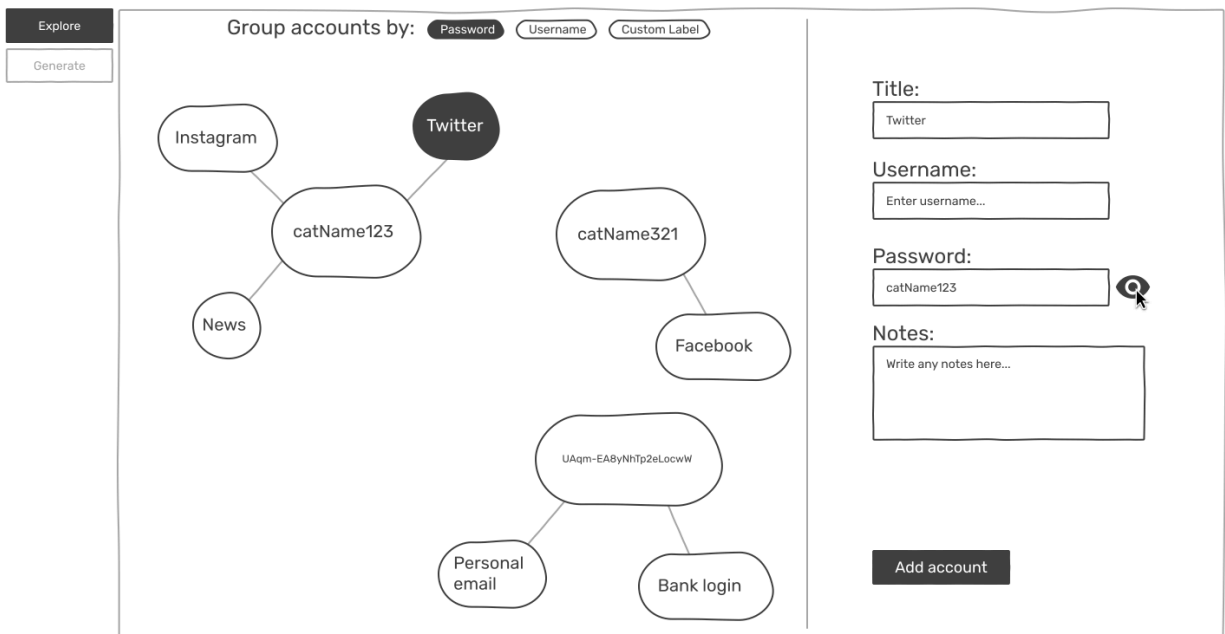


Figure 3: Direct manipulation password explorer shared label view.

The explorer view is where users can create, browse, and edit their account credentials. The left side of the screen is dedicated to a set of editable fields that depict the credentials as well as a descriptive title and any included notes. Navigation of the area would follow the typical behaviour of clicking the background to pan around and scrolling would zoom users in or out. Clicking on one of the account bubbles in the browsing area will populate the credentials into the appropriate field where users can copy or edit the values. There is a button next to the password field that toggles the visibility of the password text. At the bottom of the left side there is a button which will create a new account bubble. Along the top are some thin pill buttons which will change the grouping behaviour. "Password" and "Username" group any accounts who share the given field (Figure 2 shows an example of grouping by shared password). The "Custom Label" option allows users to make their own groupings. In the "Custom Label" view users can click the "Add label" button in the bottom left part of the screen. To group accounts they need simply drag their account near the appropriate label.



```
Command Line

$ gen_pswrd 20 1 2 0

password: DAv&a4*e0rpTcl5jRwa
strength: 10/10

$ gen_pswrd 20 1 2 0

password: V$gwCjGKc)Bmluc0QqRk
strength: 10/10

$ gen_pswrd 20 1 2 -1

password: Gf9Ij90o5fT6xaDtM3ek
strength: 8/10
```

Figure 4: Command language password generator

The command follows the form *gen_pswrd a b c d* where *a* is the desired length of password, *b* is the minimum number of capital letters, *c* is the minimum number of numbers, and *d* is the minimum number of symbols. To balance learnability and functionality, I decided to only permit the definition of the lower bound on the different character sets to reduce the number of parameters in the command. The final command in Figure 4 demonstrates how a value of -1 will prevent a given character set from occurring in the generated password. A user might set these parameters in accordance with the password rules for a website.



```
Command Line

$ set_acc {someTitle, myName, gen_pswrd 20 1 2 0,
"some notes"}
  overwrite existing set y/n?
    title: someTitle
  username: myName
 password: DAv&a4*e0rpTcl5jRwa
  notes:

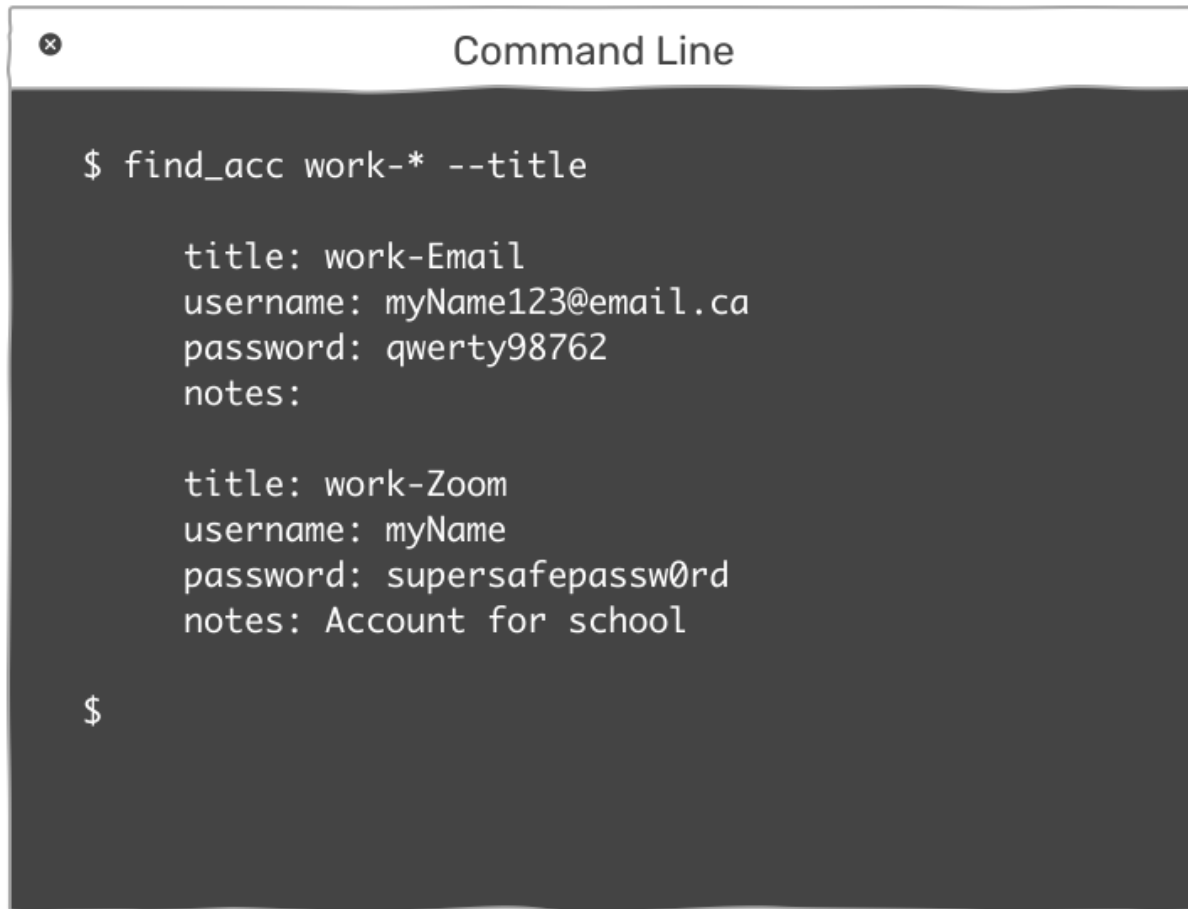
$ y

$
```

Figure 5: Command language credential creation and editing

To create or edit a set of credentials users should execute the following command:
set_acc {a, b, c, d} where *a* is a unique title, *b* is the user name, *c* is the password, and *d* is a string to include as notes. I wrapped the parameters in braces to permit a JSON style parsing of the parameters which would make it easier to drop in the random generation command when updating the credentials. The example also shows how the password generation command might be combined with the account setting command to update a password using a randomly generated one. The quotation marks around the note would also help with parsing complicated strings such as those using whitespace but are unnecessary when using simple strings or when using nested commands. Moreover, I did my best to reduce the amount of commands a user

would need to learn to engage with the interface so I combined the creation and editing commands. Creation of a new account would be very similar to the example in Figure 5 but would not present an overwrite warning if there were no existing credential sets with the given title.



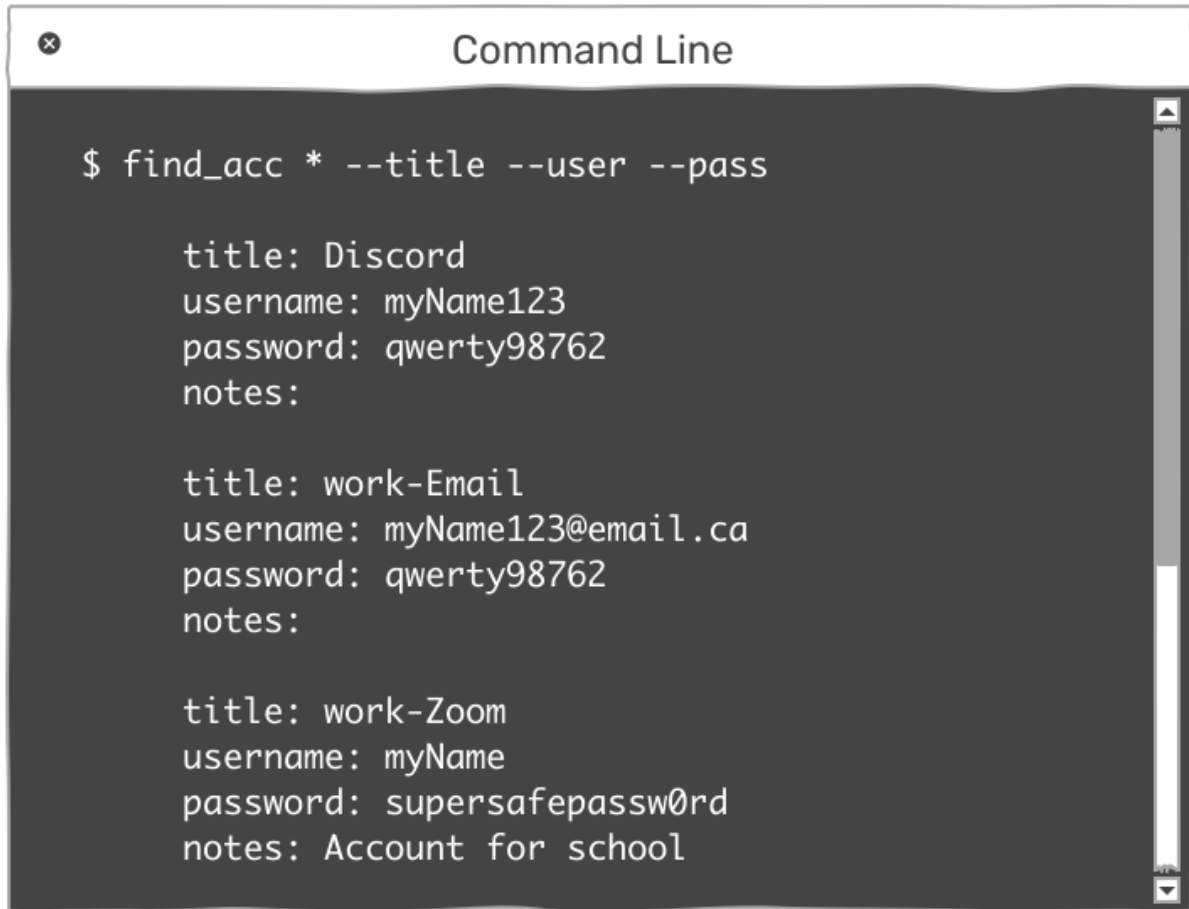
```
$ find_acc work-* --title

title: work-Email
username: myName123@email.ca
password: qwerty98762
notes:

title: work-Zoom
username: myName
password: supersafepassw0rd
notes: Account for school

$
```

Figure 6: Command language searching for titles by prefix



```
$ find_acc * --title --user --pass

title: Discord
username: myName123
password: qwerty98762
notes:

title: work-Email
username: myName123@email.ca
password: qwerty98762
notes:

title: work-Zoom
username: myName
password: supersafepassw0rd
notes: Account for school
```

Figure 7: Command language searching for titles by prefix

In order to browse their accounts users can execute the *find_acc* command. It follows the form *find_acc a --f* where *a* is some regular expression string and *--f* refers to a flag where users choose the field where the search string is being applied. Users can achieve a similar result to the previous direct manipulation group views by searching certain terms and using the appropriate flag. Figure 6 shows a possible use case where the user has added a prefix to their titles to easily find their work related accounts. Figure 7 demonstrates using the wild card to return all of the stored accounts. Regular expressions are incredibly powerful and could be applied to find very specific or general results.

Perspective Analysis:

Brutalism is a design style that has its roots in architecture. In simplest terms, architectural brutalism involves avoiding non-functional, decorative elements (Copeland, n.d.). The goal is to draw focus to and appreciate the physical materials used to construct the buildings (Copeland, n.d.). Though the style is often derided as cold and uninviting, there is an impressive simplicity and efficiency with which it is able to produce functional designs. Beyond architecture, brutalism has made its way onto the internet as a style of website design.



Figure 8: Example of brutalist architecture (Source: https://en.wikipedia.org/wiki/National_Arts_Centre)

Brutalist web design is based upon the same principle of reducing aesthetic clutter and bringing materials to the forefront, but takes a somewhat unintuitive stance regarding what are considered the materials of a site. The guiding principle behind brutalist web design is that the fundamental purpose of a website is to facilitate user interaction with the content of the site (Copeland, n.d.). Although at the deepest level websites are constructed using code, material of most importance to users is the content of the site (Copeland, n.d.).

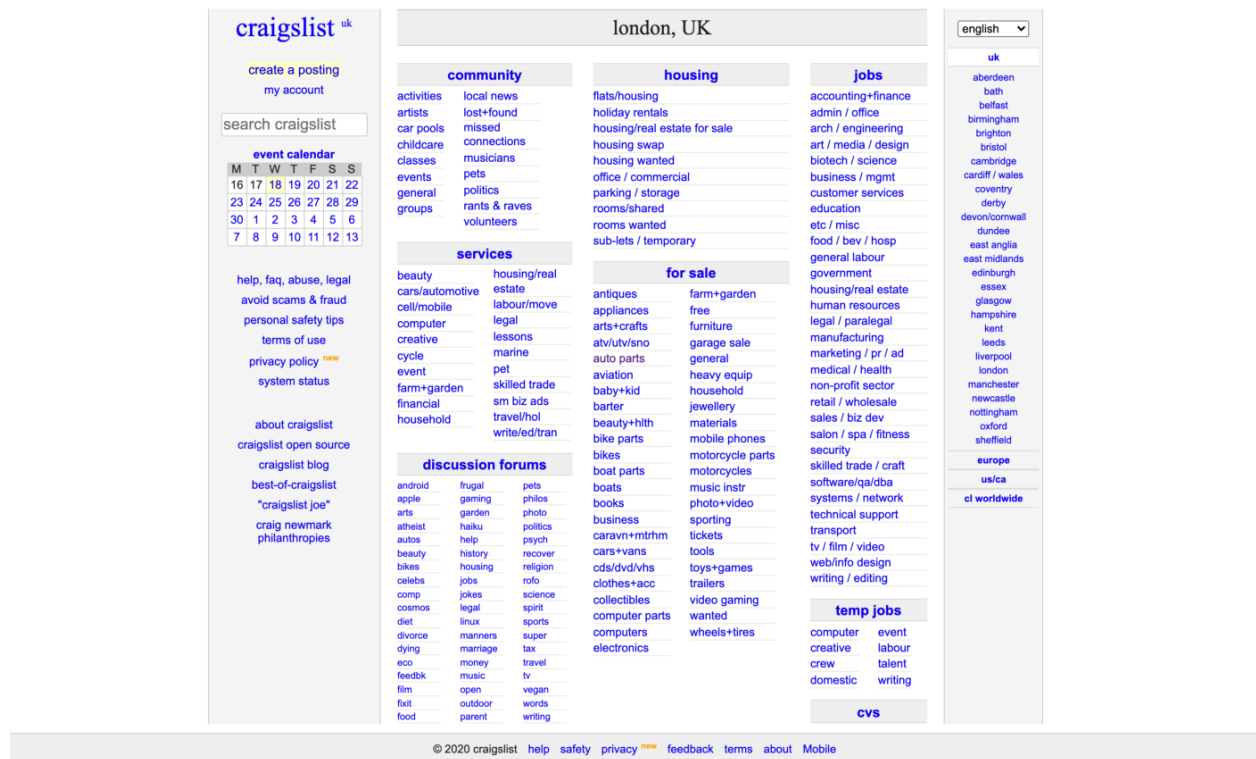


Figure 9: Example of brutalist web design (Source: <https://uxdesign.cc/brutalist-web-design-is-taking-over-the-internet-fee3c66139b5>)

Unfortunately, the stated guidelines of brutalist web design -- “content is readable on all reasonable screens and devices, only hyperlinks and buttons respond to clicks, hyperlinks are underlined and buttons look like buttons, the back button works as expected, view content by scrolling, decoration when needed and no unrelated content, performance is a feature” (Copeland, n.d.) -- are quite dependant on the functionality of a web browser and the hypertext interaction style. Nonetheless, I thought that brutalism would be a particularly relevant perspective for this project since the philosophy of prioritizing the essential functional aspects of an interface seemed to align with the goal of testing the limitations of adhering to a single interface style. Therefore I somewhat translated brutalism so that it could be applied to direct manipulation and command language interfaces outside of a web specific context.

I first set out to recognize the relationship between the users and the “materials” with which they will interact. This is perhaps most easily done by considering the simplest form of

password manager: using pen and paper. Some people do not use digital password managers and opt instead to write down their passwords and the associated account information in a notebook, or on sticky notes. The purpose of these written documents, and therefore password managers, is to make it easier to log into accounts. Moreover, passwords on their own are meaningless unless there is appropriate depiction of their relationship to the username and place that the credentials are used. I would therefore propose that the fundamental relationship in password managers is between the user and their stored account credentials.

The minimum amount of information that should be stored is a title, which is used to label the service or site to which the credentials pertain, the username, the password. For my designs I decided to add a notes field which could permit users to store any extra information pertaining to the account but which is irrelevant to the organization of the accounts. A method of generating passwords seemed like another function worth exploring. Password generation is a utility offered by many digital password managers but not by physical approaches. I felt that it was important to include this functionality while following a brutalist approach since password creation is still part of the relationship people have with their account credentials.

A study from Stobert and Biddle about the relationship between people and their passwords provides some important insights for my brutalist approach. For the purpose of this project, I will focus on the study's insights regarding password reuse since it is pertinent to the relationship between people and their account credentials (Stobert & Biddle, 2014). The study identifies a variety of categories by which users categorize their passwords and accounts (Stobert & Biddle, 2014). These categories included, but are not limited to, the security of the password, frequency of account use, the password rules required by the site, and account similarity (Stobert & Biddle, 2014). Obviously, it would be very complicated to appropriately capture all of the ways in which users organize their credentials so I aim to include versatile organization methods which give users control over how they traverse their stored accounts. Importantly, participants in the study mentioned struggling to keep track of passwords even

when reusing them (Stobert & Biddle, 2014) so reuse strategies may benefit from or complement dedicated password managers rather than simply replace them.

With respect to password generation, the authors of the study highlight the importance of providing password strength meters which I included in both password generators (Stobert & Biddle, 2014). as they help users understand the security impact of changing various parameters. On the note of password generation, I decided not to mandate the use of password generators when creating or updating credentials. To do so would prevent password reuse, so instead random generation exists as an option for those who would prefer it.

A study specifically about password managers found that users were put off, not because of the usability of password managers, but because of a lack of trust and transparency (Alodhyani et al., 2020). Trust is an unfortunately nebulous concept and is therefore difficult to target directly, but transparency is an easier factor to consider in the development of an interface. I hope that the brutalist approach of stripping away aesthetic dressings and focusing on bringing attention to the functional considerations will help users feel that there is a greater degree of transparency in these designs. I think that this focus is most obvious when compared to built in browser password managers. Often, browser based password managers are built around prompting users to create secure passwords during account creation and then automatically filling out sign in pages when they appear. The password manager in this case is acting on behalf of the users and seems to assume that users are largely uninterested in or incompetent with respect to security. Though this could be true for some users, it is impossible to understand the variety of circumstances in which users find themselves so I chose to focus more towards trusting users to use tools in ways which they feel are most appropriate for their lives.

For the direct manipulation interface, I followed the brutalist philosophy by giving users a clear visual way of organizing their accounts. I proposed a mind-map-esque depiction of accounts connected to a shared piece of information. The two standard organization methods I

envisioned were grouping by either shared usernames or passwords. The shared username organization method is meant to help users keep track of how their identity is shared across the internet. For some sites, the only piece of identifying information might be your username but if that username is also being used on less anonymous sites someone could piece together your identity on both sites. Sharing a public identity across accounts may also be a desired behavior. For example, you might have an account for a multiplayer video game and want your friends on that game to be able to easily find you on some sort of voice chat app. The shared password view would be useful for people who heavily engage in password reuse. It would allow them to navigate their accounts by the shared passwords. Users would also be able to gain a quick look at how common a given password is across their accounts and perhaps realize that they have reused a password more often than they are comfortable with. I also propose a most customizable option wherein users can create their own shared nodes onto which they can attach their account credentials. This would let users implement their own organization systems which may not be captured by the shared username or shared password visualizations.

The command language interface is much trickier to design with respect to organization since the style is inherently focused on avoiding spatially navigating large sets of information. I have personally found myself very frustrated navigating through file structures when using terminal. In order to fulfill the brutalist philosophy, every command should allow users to interact with their credentials and any commands for navigation would stray from this goal. That being said, I have already mentioned the importance of encouraging users to organize their credentials. I propose that users search their stored credentials using regular expressions (regex) rather than designing specific organization and navigation commands. This would both encourage users to give descriptive titles to their accounts (perhaps including work or school as a prefix to their accounts and using the appropriate regex query to find all the accounts with the given prefix) and easily incorporates the ability for users to view all their accounts with the regex wildcard (*). I provide examples of both functionalities in the design section.

By boiling down password managers to their most essential features pertaining to the storage of accounts, I hopefully encouraged users to consider the ways that they could benefit from even a simple password management system or at the very least how such small changes in their current practices could have meaningful impacts on their security.

For example, the direct manipulation implementation of a password generator can help users construct a more accurate understanding of the security of their passwords. I believe that getting users to engage on one facet of security issues will build their confidence and inspire future learning about more complex topics. Most important from a brutalist perspective, however, is the way that the generator minimizes what the computer does *for* the user and rather aids and encourages the user in executing the task. Alternative password generators would simply spit out a random password and the more complicated versions may give users simple toggle buttons to permit the use of numbers or symbols. I believe that it is in the spirit of the brutalist philosophy to bring the mechanisms for creating complex passwords to the users' attention.

Discussion and Comparison:

I have already included some discussion of my designs in the previous section so I will include some ideas that were unrelated to the application of my chosen perspective. The following discussion is related to the strengths and weaknesses of the styles both broadly and specifically in the context of password manager interfaces.

Reflecting specifically on the command language interface, I am worried that there is not a clear way to benefit from macros. One of the primary strengths of a command language interface is the way that the commands can be quickly repeated and chained together. When deciding on the interface styles, I recalled Shneiderman's claim that frequent users benefit from command line interfaces. It is true that people may need to log into several accounts in a given day (especially when they are unable to remain logged in to a particular site) and felt that command language could flex its strength within this context. However, I now realize that the typical use case for password managers may not offer many opportunities for command language to shine. Consider for a moment a traditional instance of the notebook approach: being unable to recall the credentials needed to log in, finding the notebook, finding the set of credentials needed to access the site, and then inputting them. If you were to use the command language interface, the experience would be similar although you could more easily find the credentials if you know what to include in a search term. This approach could be much faster than the direct manipulation approach of visually scanning and spatially navigating the accounts but it is a one-off task. Though you could perhaps chain together different search terms to narrow down the results, direct manipulation seems to be a more appropriate choice for exploring information.

Due to the focus on spatial navigation, the direct manipulation interface would also present inefficiencies but the style is more interested in encouraging exploration and could display additional accounts of interest if they share particular information. On the topic of

exploration, I think that the presentation of information in the direct manipulation interface is particularly appropriate given that it requires the least memorization of information. The direct manipulation interface could be criticised for prolonging the ostensibly simple task of finding credentials, but I would stress the importance of appreciating the style for the under explored, if somewhat niche, benefit of avoiding recall based search. Imagine returning to a site that you have not visited in years and therefore remember neither your username, password, or even whether you have an account. As an example, Google allows you to recover your username with phone number or recovery email but that requires that you had the forethought to provide such information and that you remember what you provided. Using the command language interface, you could merely fetch all of your accounts in an unorganized list but that defeats the purpose of a search function. This leaves the direct manipulation interface with its expressive spatial organization. The intended behaviour for this interface is actual exploration over search which makes it the perfect approach for discovering credentials about which you know very little. In *Designing with the Mind in Mind*, Johnson highlights the fact that people struggle more with recall based tasks than recognition ones (Johnson, 2014). This phenomenon likely contributes to the difficulty of remembering unique passwords. If a password manager is meant to help reduce the need for users to engage in recall based memorization, a command language interface may have an inherent flaw.

However, I think that command languages may win out when comparing the password generation capabilities of the two approaches. The command language password generator is the simpler of the two. A user need only set the size and minimum values for character set occurrence before they can see a password. The users cannot control the upper bound of these values but this prevents an overly long command within which it would be difficult to remember what each parameter referred to. The direct manipulation interface is able to give users control over both the minimum and maximum but this may bring its own consequences. Analysis paralysis refers to the phenomenon of taking too long to make a decision by over focusing on

the available options (*Do You Have Analysis Paralysis?*, n.d.). In an effort to follow the spirit of direct manipulation and dynamic queries, I opted to encourage experimentation with the parameters by setting a minimum and maximum bound. There is no refresh button since I felt that the experience of setting fields and pressing a button to generate the passwords would move the interface closer to the command language experience of writing and executing queries without immediate feedback. However, this reliance on a larger set of editable values could cause users to fall into analysis paralysis. The ease with which more complicated information can be interacted with in direct manipulation may lead to bloated interfaces which are actually less approachable than the designer intends.

My primary concern about direct manipulation interfaces pertains to how the playfulness of the style impacts an interface's longevity. Shneiderman frequently mentions how exciting and enticing users find direct manipulation interfaces (Shneiderman, 1988, 1997) but I worry that this is tied to the novelty of the interface. Computers with graphical user interfaces are much more common today than in the 1980s, when Shneiderman defined direct manipulation (Sherugar & Budiu, 2016). Most modern operating systems use direct manipulation as a means of interacting with the files on the computer. However, I doubt many users today would describe organizing the files on their system as a fun task. There may be a time limit on how long direct manipulation remains fun and, once you reach it, direct manipulation can begin to feel frustrating. While studying computer science for my bachelor's degree, I had many conversations with peers who had switched to interacting with their files via command line interfaces. They praised the interfaces for speed and laughed at the laborious process of using graphical user interfaces based on direct manipulation. However, every time I tried to learn the command line interfaces for myself I was appalled by all the upfront learning that was required. Though I would not describe my interactions with file browsers as "fun", I still appreciate the immediate feedback afforded by direct manipulation. Rather than focus on the "fun-ness" of

direct manipulation it is important to leverage its other strengths to provide lasting usability benefits.

Looking back on my 2 designs I cannot help but feel as if there is some sort of in between approach that would benefit from the strengths of each style but overcome their weaknesses. As a whole, the direct manipulation interface seems easier to pick up with little to no explanation. After showing it to my friends and family, it seems to offer a unique and intuitive means of organizing credentials. However, the longer I look at it the more vividly I can imagine people reaching a limit to how long they are willing to engage with a tool designed to appeal to non-expert users. The command language interface seems to have the opposite problem. The commands are quick to write out and a user can quickly pull up their credentials with a single command. The password generator itself is elegantly simple and perhaps could be automated so that a user's passwords never get too old. However, anytime I've shown the interfaces to someone, they need much more explanation to understand how to achieve even simple tasks.

In the end, adhering so strictly to a particular feeling for an interface is sure to limit both its utility to users and its ability to appeal to large audiences. Nonetheless, it is important to engage in these sorts of exploration exercises to push up against boundaries of different design styles and philosophies as well as recognize styles that may have gone overlooked in domains. Without forcing yourself to find creative uses of a particular interaction style, you may miss out on some of the most interesting applications of its strengths and may never realize its most devastating weaknesses.

The direct manipulation interface seems to triumph when exploring the stored account and the command language interface seems more adept at providing a simple, reusable password generator. When deciding on an interaction style, I will be sure to tailor my selection to each particular task rather than force all the functionality to fit under a single style.

References:

- Alodhyani, F., Theodorakopoulos, G., & Reinecke, P. (2020). Password Managers—It's All about Trust and Transparency. *Future Internet*, 12(11), 189.
<https://doi.org/10.3390/fi12110189>
- Copeland, D. (n.d.). *Brutalist Web Design*. Retrieved October 31, 2021, from <https://brutalist-web.design>
- Do You Have Analysis Paralysis? | Psychology Today Canada*. (n.d.). Retrieved October 31, 2021, from <https://www.psychologytoday.com/ca/blog/fixing-families/201904/do-you-have-analysis-paralysis>
- Johnson, J. (2014). Our Attention is Limited; Our Memory is Imperfect. In *Designing with the Mind in Mind* (pp. 87–105). Elsevier. <https://doi.org/10.1016/B978-0-12-407914-4.00007-5>
- Sherugar, S., & Budiu, R. (2016, August 21). *Direct Manipulation: Definition*. Nielsen Norman Group. <https://www.nngroup.com/articles/direct-manipulation/>
- Shneiderman, B. (1988). We can design better user interfaces: A review of human-computer interaction styles. *Ergonomics*, 31(5), 699–710.
<https://doi.org/10.1080/00140138808966713>
- Shneiderman, B. (1997). Direct manipulation for comprehensible, predictable and controllable user interfaces. *Proceedings of the 2nd International Conference on Intelligent User Interfaces - IUI '97*, 33–39. <https://doi.org/10.1145/238218.238281>
- Stobert, E., & Biddle, R. (2014). *The Password Life Cycle: User Behaviour in Managing Passwords*. 243–255.
<https://www.usenix.org/conference/soups2014/proceedings/presentation/stobert>